



---

# Eldo User's Manual

Software Version 6.6\_1 Release 2005.3

---

**Copyright © Mentor Graphics Corporation 2005  
All rights reserved.**

This document contains information that is proprietary to Mentor Graphics Corporation. The original recipient of this document may duplicate this document in whole or in part for internal business purposes only, provided that this entire notice appears in all copies. In duplicating any part of this document, the recipient agrees to make every reasonable effort to prevent the unauthorized use and distribution of the proprietary information.

This document is for information and instruction purposes. Mentor Graphics reserves the right to make changes in specifications and other information contained in this publication without prior notice, and the reader should, in all cases, consult Mentor Graphics to determine whether any changes have been made.

The terms and conditions governing the sale and licensing of Mentor Graphics products are set forth in written agreements between Mentor Graphics and its customers. No representation or other affirmation of fact contained in this publication shall be deemed to be a warranty or give rise to any liability of Mentor Graphics whatsoever.

**MENTOR GRAPHICS MAKES NO WARRANTY OF ANY KIND WITH REGARD TO THIS MATERIAL INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.**

MENTOR GRAPHICS SHALL NOT BE LIABLE FOR ANY INCIDENTAL, INDIRECT, SPECIAL, OR CONSEQUENTIAL DAMAGES WHATSOEVER (INCLUDING BUT NOT LIMITED TO LOST PROFITS) ARISING OUT OF OR RELATED TO THIS PUBLICATION OR THE INFORMATION CONTAINED IN IT, EVEN IF MENTOR GRAPHICS CORPORATION HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

#### **RESTRICTED RIGHTS LEGEND 03/97**

U.S. Government Restricted Rights. The SOFTWARE and documentation have been developed entirely at private expense and are commercial computer software provided with restricted rights. Use, duplication or disclosure by the U.S. Government or a U.S. Government subcontractor is subject to the restrictions set forth in the license agreement provided with the software pursuant to DFARS 227.7202-3(a) or as set forth in subparagraph (c)(1) and (2) of the Commercial Computer Software - Restricted Rights clause at FAR 52.227-19, as applicable.

**Contractor/manufacturer is:**

Mentor Graphics Corporation

8005 S.W. Boeckman Road, Wilsonville, Oregon 97070-7777.

Telephone: 503.685.7000

Toll-Free Telephone: 800.592.2210

Website: [www.mentor.com](http://www.mentor.com)

SupportNet: [www.mentor.com/supportnet](http://www.mentor.com/supportnet)

Contact Your Technical Writer: [www.mentor.com/supportnet/documentation/reply\\_form.cfm](http://www.mentor.com/supportnet/documentation/reply_form.cfm)

**TRADEMARKS:** The trademarks, logos and service marks ("Marks") used herein are the property of Mentor Graphics Corporation or other third parties. No one is permitted to use these Marks without the prior written consent of Mentor Graphics or the respective third-party owner. The use herein of a third-party Mark is not an attempt to indicate Mentor Graphics as a source of a product, but is intended to indicate a product from, or associated with, a particular third party. A current list of Mentor Graphics' trademarks may be viewed at: [www.mentor.com/terms\\_conditions/trademarks.cfm](http://www.mentor.com/terms_conditions/trademarks.cfm).

# Table of Contents

---

<b>Chapter 1</b>	
<b>Introduction to Eldo .....</b>	<b>1-1</b>
Introduction .....	1-1
Overview .....	1-1
Eldo Input and Output Files.....	1-2
Getting Started .....	1-4
How to Run Eldo .....	1-4
Schematic Example.....	1-4
Associated Netlist .....	1-5
Running a Simulation .....	1-6
<b>Chapter 2</b>	
<b>Running Eldo.....</b>	<b>2-1</b>
Running Eldo from the Command Line .....	2-1
Command Line Help.....	2-10
Running the STMicroelectronics Version of Eldo .....	2-10
Running the Motorola Version of Eldo .....	2-10
Eldo Initialization File .....	2-11
<b>Chapter 3</b>	
<b>Eldo Control Language .....</b>	<b>3-1</b>
Introduction .....	3-1
Overview of the .cir File Structure .....	3-1
General Aspects of the Language Syntax .....	3-2
Documentation Conventions.....	3-2
First Line.....	3-2
Continuation Lines .....	3-2
Comment Lines.....	3-2
Component Names .....	3-3
Parameter Names .....	3-3
String Parameters .....	3-3
Reserved Keywords .....	3-3
Node Names .....	3-4
Values .....	3-4
Model Names .....	3-5
Scale Factors .....	3-5
Directives .....	3-5
Arithmetic Functions .....	3-7
Operators .....	3-10
Operator Precedence .....	3-10
Arithmetic Operators .....	3-11
Boolean Operators .....	3-11

Bitwise Operators . . . . .	3-11
Expressions . . . . .	3-12
Simulation Counters . . . . .	3-13
Node & Element Information . . . . .	3-13
Grounded Capacitors Information . . . . .	3-14
Matrix Information . . . . .	3-14
Newton Block Information . . . . .	3-14
Convergence Information . . . . .	3-14
Temperature Handling . . . . .	3-15
Devices . . . . .	3-17
Sources . . . . .	3-20
Macromodels . . . . .	3-24
Analog . . . . .	3-24
Digital . . . . .	3-26
Mixed . . . . .	3-27
Magnetic . . . . .	3-27
Switched Capacitor . . . . .	3-28
Commands . . . . .	3-29
<b>Chapter 4 . . . . .</b>	
<b>Device Models . . . . .</b>	<b>4-1</b>
Introduction . . . . .	4-1
Resistor . . . . .	4-3
Resistor Model Syntax . . . . .	4-8
Capacitor . . . . .	4-13
Capacitor Model Syntax . . . . .	4-16
Inductor . . . . .	4-20
Inductor Model Syntax . . . . .	4-25
Coupled Inductor . . . . .	4-27
RC Wire . . . . .	4-28
RC Wire Model Syntax . . . . .	4-29
Semiconductor Resistor . . . . .	4-34
Semiconductor Resistor Model Syntax . . . . .	4-34
Semiconductor Resistor Model Equations . . . . .	4-36
Transmission Line . . . . .	4-41
Lossy Transmission Line . . . . .	4-43
Level 1 . . . . .	4-43
Level 2 . . . . .	4-45
Level 3 . . . . .	4-46
Level 4 . . . . .	4-48
Error message treatment . . . . .	4-50
Technical precision . . . . .	4-51
Lossy Transmission Line: W Model . . . . .	4-59
RLGC file syntax . . . . .	4-61
RLGC model syntax . . . . .	4-64
Tabular RLGCmodel syntax . . . . .	4-66
Error message treatment . . . . .	4-69
Lossy Transmission Line: U Model . . . . .	4-70

## Table of Contents

---

Model Syntax . . . . .	4-71
Error message treatment . . . . .	4-75
Microstrip Models. . . . .	4-76
MTEE: Microstrip T Junction. . . . .	4-77
MBEND: Microstrip Bend (Arbitrary Angle, Optimally Mitered) . . . . .	4-80
MBEND2: 90-degree Microstrip Bend (Mitered) . . . . .	4-83
MBEND3: 90-degree Microstrip Bend (Optimally Mitered) . . . . .	4-85
MCORN: 90-degree Microstrip Bend (Unmiterred) . . . . .	4-88
MSTEP: Microstrip Step in Width . . . . .	4-90
VIA2: Cylindrical Via Hole in Microstrip . . . . .	4-92
SBEND: Unmiterred Stripline Bend . . . . .	4-95
STEE: Stripline T Junction . . . . .	4-97
SSTEP: Stripline Step in Width . . . . .	4-99
Junction Diode . . . . .	4-102
Diode Model Syntax . . . . .	4-105
Berkeley Level 1 (Eldo Level 1). . . . .	4-106
Modified Berkeley Level 1 (Eldo Level 2). . . . .	4-106
Fowler-Nordheim Model (Eldo Level 3) . . . . .	4-106
JUNCAP (Eldo Level 8) . . . . .	4-107
JUNCAP2 (Eldo Level 8, DIOLEV=11) . . . . .	4-108
Philips Diode Level 500 (Eldo Level 9) . . . . .	4-108
Diode Level 21 . . . . .	4-108
BJT—Bipolar Junction Transistor . . . . .	4-109
BJT Model Syntax . . . . .	4-111
Automatic Selection of BJT Model Via AREA Specifications . . . . .	4-113
Modified Gummel-Poon Model (Eldo Level 1) . . . . .	4-114
Philips Mextram 503.2 Model (Eldo Level 4) . . . . .	4-114
Improved Berkeley Model (Eldo Level 5) . . . . .	4-114
VBIC v1.2 Model (Eldo Level 8) . . . . .	4-115
VBIC v1.1.5 Model (Eldo Level 8) . . . . .	4-116
HICUM Model (Eldo Level 9) . . . . .	4-116
Philips Mextram 504 Model (Eldo Level 22) . . . . .	4-117
Philips Modella Model (Eldo Level 23) . . . . .	4-117
HICUM Level0 Model (Eldo Level 24) . . . . .	4-119
JFET—Junction Field Effect Transistor . . . . .	4-120
JFET Model Syntax . . . . .	4-122
MESFET—Metal Semiconductor Field Effect Transistor . . . . .	4-123
MOSFET . . . . .	4-124
MOSFET Models . . . . .	4-127
Noise in MOSFETs. . . . .	4-130
Selection of MOSFET Models via W/L Specifications (Binning) . . . . .	4-131
MOS Parasitics Common Approach . . . . .	4-132
Berkeley SPICE Models . . . . .	4-139
MERCKEL MOSFET Models . . . . .	4-140
Berkeley SPICE BSIM1 Model (Eldo Level 8) . . . . .	4-142
Berkeley SPICE BSIM2 Model (Eldo Level 11) . . . . .	4-143
Modified Berkeley Level 2 (Eldo Level 12) . . . . .	4-144
Modified Berkeley Level 3 (Eldo Level 13) . . . . .	4-144
Modified Lattin-Jenkins-Grove Model (Eldo Level 16) . . . . .	4-144

Enhanced Berkeley Level 2 Model (Eldo Level 17) . . . . .	4-145
EKV MOS Model (Eldo Level 44 or EKV) . . . . .	4-145
Berkeley SPICE BSIM3v2 Model (Eldo Level 47) . . . . .	4-146
Berkeley SPICE BSIM3v3 Model (Eldo Level 53) . . . . .	4-147
Motorola SSIM Model (Eldo Level 54 or SSIM) . . . . .	4-151
Berkeley SPICE BSIM3SOI v1.3 Model (Eldo Level 55) . . . . .	4-151
Berkeley SPICE BSIM3SOI v2.x and v3.x Model (Eldo Level 56) . . . . .	4-154
Philips MOS 9 Model (Eldo Level 59 or MOSP9) . . . . .	4-157
Berkeley SPICE BSIM4 Model (Eldo Level 60) . . . . .	4-158
TFT Polysilicon Model (Eldo Level 62) . . . . .	4-159
Philips MOS Model 11 Level 1101 (Eldo Level 63) . . . . .	4-160
TFT Amorphous-Si Model (Eldo Level 64) . . . . .	4-161
Philips MOS Model 11 Level 1100 (Eldo Level 65) . . . . .	4-162
HiSIM Model (Eldo Level 66) . . . . .	4-162
SP Model (Eldo Level 67) . . . . .	4-163
Berkeley SPICE BSIM5 Model (Eldo Level 68) . . . . .	4-164
Philips MOS Model 11 Level 1102 (Eldo Level 69) . . . . .	4-164
Philips PSP Model (Eldo Level 70) . . . . .	4-165
BTA HVMOS Model (Eldo Level 101) . . . . .	4-166
S-Domain Filter . . . . .	4-167
Z-Domain Filter . . . . .	4-169
Subcircuit Instance . . . . .	4-171

<b>Chapter 5 . . . . .</b>	
<b>Sources . . . . .</b>	<b>5-1</b>
Introduction . . . . .	5-1
Independent Sources . . . . .	5-1
Linear Dependent Sources . . . . .	5-1
Non-linear Dependent Sources . . . . .	5-2
S, Y, Z Parameter Extraction . . . . .	5-3
Independent Voltage Source . . . . .	5-4
Independent Current Source . . . . .	5-11
Amplitude Modulation Function . . . . .	5-17
Exponential Function . . . . .	5-19
Noise Function . . . . .	5-21
Pattern Function . . . . .	5-23
Pulse Function . . . . .	5-25
Piece Wise Linear Function . . . . .	5-27
Single Frequency FM Function . . . . .	5-32
Sine Function . . . . .	5-33
Trapezoidal Pulse With Bit Pattern Function . . . . .	5-35
Exponential Pulse With Bit Pattern Function . . . . .	5-37
Voltage Controlled Voltage Source . . . . .	5-39
Current Controlled Current Source . . . . .	5-46
Voltage Controlled Current Source . . . . .	5-50
Current Controlled Voltage Source . . . . .	5-57
S, Y, Z Parameter Extraction . . . . .	5-61

---

## Table of Contents

---

<b>Chapter 6</b>	.....	
<b>Analog Macromodels</b>	.....	<b>6-1</b>
Eldo Analog Macromodels	.....	6-1
General Notes on the Use of FAS Macromodels	.....	6-2
Comparator	.....	6-3
Op-amp (Linear)	.....	6-5
Op-amp (Linear 1-pole)	.....	6-7
Application Area	.....	6-9
Op-amp (Linear 2-pole)	.....	6-11
Delay	.....	6-14
Saturating Resistor	.....	6-15
Voltage Limiter	.....	6-17
Voltage Controlled Switch	.....	6-19
Current Controlled Switch	.....	6-21
Triangular to Sine Wave Converter	.....	6-24
Staircase Waveform Generator	.....	6-26
Sawtooth Waveform Generator	.....	6-28
Triangular Waveform Generator	.....	6-30
Amplitude Modulator	.....	6-32
Pulse Amplitude Modulator	.....	6-34
Sample & Hold	.....	6-36
Track & Hold	.....	6-38
Pulse Width Modulator	.....	6-40
Voltage Controlled Oscillator	.....	6-43
Peak Detector	.....	6-45
Level Detector	.....	6-48
Logarithmic Amplifier	.....	6-50
Anti-logarithmic Amplifier	.....	6-52
Differentiator	.....	6-54
Integrator	.....	6-56
Adder, Subtractor, Multiplier & Divider	.....	6-58
<b>Chapter 7</b>	.....	
<b>Digital Macromodels</b>	.....	<b>7-1</b>
Eldo Digital Macromodels	.....	7-1
Digital Model Definition	.....	7-3
Delay	.....	7-6
Inverter	.....	7-7
Exclusive-OR Gate	.....	7-8
2-Input Digital Gates	.....	7-9
3-Input Digital Gates	.....	7-11
Multiple Input Digital Gates	.....	7-12
Mixed Signal Macromodels	.....	7-13
Analog to Digital Converter	.....	7-14
Digital to Analog Converter	.....	7-16

<b>Chapter 8</b>		
<b>Magnetic Macromodels .....</b>		<b>8-1</b>
Eldo Magnetic Macromodels.....		8-1
Transformer Winding .....		8-2
Non-linear Magnetic Core 1 .....		8-3
Non-linear Magnetic Core 2 .....		8-6
Linear Magnetic Core .....		8-9
Magnetic Air Gap .....		8-10
Transformer (Variable # of Windings).....		8-11
Ideal Transformer .....		8-13
<b>Chapter 9</b>		
<b>Switched Capacitor Macromodels .....</b>		<b>9-1</b>
Introduction .....		9-1
Switch Level Representation .....		9-1
Macromodels .....		9-1
Operational Amplifier.....		9-3
Switch .....		9-8
Z-domain Representation.....		9-11
General Notes on the Use of Macromodels .....		9-11
Ideal Operational Amplifier .....		9-12
SC Integrators & LDI's .....		9-13
Inverting Switched Capacitor .....		9-15
Non-inverting Switched Capacitor .....		9-17
Parallel Switched Capacitor .....		9-19
Serial Switched Capacitor.....		9-21
Serial-parallel Switched Capacitor .....		9-23
Bi-linear Switched Capacitor .....		9-26
Unswitched Capacitor.....		9-28
Applications .....		9-30
Non-inverting Integrator .....		9-31
LDI Phase Control Non-inverting Integrator .....		9-32
Euler Forward Integrator .....		9-33
LDI Phase Control Euler Forward Integrator .....		9-34
Euler Backward Integrator .....		9-35
LDI Phase Control Euler Backward Integrator.....		9-36
Tutorial—SC Low Pass Filter .....		9-36
<b>Chapter 10</b>		
<b>Simulator Commands.....</b>		<b>10-1</b>
Introduction .....		10-1
Command Line Help.....		10-2
.A2D .....		10-4
.AC .....		10-12
.ADDLIB .....		10-18
.AGE .....		10-20
.AGEMODEL.....		10-21
.ALTER .....		10-22

## Table of Contents

---

.BIND .....	10-26
.CALL_TCL .....	10-28
.CHECKBUS .....	10-30
.CHECKSOA .....	10-33
.CHRENT .....	10-35
.CHRSIM .....	10-37
.COMCHAR .....	10-39
.CONNECT .....	10-40
.CONSO .....	10-41
.CORREL .....	10-42
.D2A .....	10-44
.DATA .....	10-51
.DC .....	10-54
.DCMISMATCH .....	10-61
.DEFAULT .....	10-65
.DEFMAC .....	10-66
.DEFMOD .....	10-68
.DEFPLOTDIG .....	10-69
.DEFWAVE .....	10-70
.DEL .....	10-74
.DISCARD .....	10-75
.DISFLAT .....	10-76
.DISTRIB .....	10-77
.DSP .....	10-78
.DSPF_INCLUDE .....	10-79
.DSPMOD .....	10-85
.END .....	10-89
.ENDL .....	10-90
.ENDS .....	10-91
.EQUIV .....	10-92
.EXTMOD .....	10-94
.EXTRACT .....	10-95
.FFILE .....	10-115
.FORCE .....	10-118
.FOUR .....	10-119
.FUNC .....	10-120
.GLOBAL .....	10-121
.GUESS .....	10-122
.HIER .....	10-123
.IC .....	10-125
.IGNORE_DSPF_ON_NODE .....	10-127
.INCLUDE .....	10-128
.INIT .....	10-130
.IPROBE .....	10-131
.LIB .....	10-133
.LOAD .....	10-139
.LOOP .....	10-140
.LOTGROUP .....	10-142
.LSTB .....	10-144

---

## Table of Contents

---

.MALIAS .....	10-145
.MAP_DSPF_NODE_NAME .....	10-146
.MC .....	10-147
.MCMOD .....	10-153
.MEAS .....	10-155
.MODDUP .....	10-159
.MODEL .....	10-160
.MODLOGIC .....	10-164
.MONITOR .....	10-165
.MPRUN .....	10-166
.MSELECT .....	10-173
.NET .....	10-176
.NEWPAGE .....	10-177
.NOCOM .....	10-178
.NODESET .....	10-179
.NOISE .....	10-180
.NOISETRAN .....	10-182
.NOTRC .....	10-185
.NWBLOCK .....	10-186
.OP .....	10-187
.OPTFOUR .....	10-192
.OPTIMIZE .....	10-197
.OPTION .....	10-198
.OPTNOISE .....	10-199
.OPTPWL .....	10-203
.OPTWIND .....	10-204
.PARAM .....	10-205
.PART .....	10-214
.PLOT .....	10-216
.PLOTBUS .....	10-250
.PRINTBUS .....	10-252
.PRINT .....	10-254
.PRINTFILE .....	10-256
.PROBE .....	10-258
.PROTECT .....	10-267
.PZ .....	10-268
.RAMP .....	10-269
.RESTART .....	10-271
.SAVE .....	10-274
.SCALE .....	10-277
.SENS .....	10-278
.SENPARAM .....	10-280
.SETBUS .....	10-282
.SETKEY .....	10-283
.SETSOA .....	10-284
.SIGBUS .....	10-291
.SINUS .....	10-294
.SNF .....	10-295
.SOLVE .....	10-296

## Table of Contents

---

.STEP .....	10-298
.SUBCKT .....	10-306
.SUBDUP .....	10-312
.TABLE .....	10-313
.TEMP .....	10-314
.TF .....	10-315
.TITLE .....	10-316
.TOPCELL .....	10-317
.TRAN .....	10-318
.TSAVE .....	10-323
.TVINCLUDE .....	10-325
.UNPROTECT .....	10-327
.USE .....	10-328
.USEKEY .....	10-330
.USE_TCL .....	10-331
.WCASE .....	10-332
.WIDTH .....	10-335
<b>Chapter 11 .....</b>	<b>11-1</b>
<b>Simulator and Control Options .....</b>	
Introduction .....	11-1
.OPTION .....	11-2
Cadence Compatibility Options .....	11-6
Precise Compatibility Options .....	11-6
SPICE Compatibility Options .....	11-6
Simulator Compatibility Options .....	11-6
Netlist Parser Control Options .....	11-7
Simulation Speed, Accuracy & Efficiency Options .....	11-13
Miscellaneous Simulation Control Options .....	11-24
Model Control Options .....	11-30
RC Reduction Options .....	11-40
Noise Analysis Options .....	11-41
Simulation Display Control Options .....	11-42
Simulation Output Control Options .....	11-44
Optimizer Output Control Options .....	11-51
File Generation Options .....	11-52
Mathematical Algorithm Options .....	11-55
Mixed-Mode Options .....	11-57
Other Options .....	11-59
<b>Chapter 12 .....</b>	<b>12-1</b>
<b>Compatibility Options .....</b>	
Introduction .....	12-1
Devices .....	12-2
Commands .....	12-4
Options .....	12-5
Netlist .....	12-7
Arithmetic Functions and Operators .....	12-8

<b>Chapter 13</b>		
<b>TIs spice Compatibility .....</b>		<b>13-1</b>
Introduction .....	13-1	
Devices .....	13-1	
Commands .....	13-3	
Netlist .....	13-8	
Functions and Sources .....	13-9	
Options .....	13-9	
Eldo Extensions for TIs spice Compatibility .....	13-9	
<b>Chapter 14</b>		
<b>Transient Noise Analysis .....</b>		<b>14-1</b>
Introduction to Transient Noise Analysis .....	14-1	
.NOISETRAN Command .....	14-3	
Example 1—High-rate Particle Detector .....	14-4	
Description of the Particle Detector Behavior .....	14-4	
Analysis of the Noise Performance .....	14-5	
Netlist for Example 1 .....	14-8	
Example 2—Switched Capacitor Filter .....	14-9	
Characterization of the Amplifier .....	14-10	
Netlist Used for the Amplifier Simulations .....	14-12	
Switch Noise Model .....	14-14	
Simulation Results of the Complete Circuit .....	14-14	
Netlist for Example 2 .....	14-16	
<b>Chapter 15</b>		
<b>Working with S, Y, Z Parameters .....</b>		<b>15-1</b>
Introduction .....	15-1	
Simulation Setup for S, Y, Z Parameter Extraction .....	15-1	
S, Y, Z Parameter Extraction .....	15-2	
For S Parameter Extraction .....	15-2	
For Mixed Mode S Parameter Extraction .....	15-2	
For Y Parameter Extraction .....	15-4	
For Z Parameter Extraction .....	15-4	
Matrix Parameter Extraction .....	15-4	
For G Parameter Extraction .....	15-4	
For H Parameter Extraction .....	15-5	
For T Parameter Extraction .....	15-5	
For A Parameter Extraction .....	15-5	
Output File Specification .....	15-5	
Transient Simulation of Circuits Characterized in the Frequency Domain .....	15-7	
Introduction .....	15-7	
Technical Background .....	15-8	
Implementation Issues .....	15-11	
Applications .....	15-12	
Functionality .....	15-13	
Instantiating a Block Defined by S-Parameters .....	15-14	
Touchstone, Data Format .....	15-18	

## Table of Contents

---

Mixed Mode S Parameter Extraction . . . . .	15-19
<b>Chapter 16</b>	
<b>Speed and Accuracy . . . . .</b>	<b>16-1</b>
Introduction . . . . .	16-1
Speed and Accuracy in Eldo . . . . .	16-2
Integration methods . . . . .	16-2
Time Step Control . . . . .	16-4
Newton Iterations Accuracy Control . . . . .	16-9
Global Tuning of the Accuracy—EPS . . . . .	16-10
Global Tuning of the Accuracy—TUNING . . . . .	16-11
IEM algorithm . . . . .	16-12
OSR algorithm . . . . .	16-13
Simulation of Large Circuits . . . . .	16-16
Tips for troubleshooting and/or improving convergence and performance . . . . .	16-19
Operating Point Calculation . . . . .	16-19
.JC V<NN>=VALUE . . . . .	16-19
.NODESET V<NN>=VALUE . . . . .	16-19
.GUESS V<NN>=VALUE . . . . .	16-19
.RAMP . . . . .	16-20
.OPTION VMIN, VMAX . . . . .	16-20
.SAVE, .USE & .RESTART . . . . .	16-20
.SAVE DC in Combination with RESTART . . . . .	16-22
.SAVE DC in Combination with USE . . . . .	16-22
.SAVE END in Combination with RESTART . . . . .	16-23
More Sophisticated SAVE & RESTART Procedures . . . . .	16-23
Aborting simulation and saving current state . . . . .	16-24
<b>Chapter 17</b>	
<b>Integral Equation Method (IEM) . . . . .</b>	<b>17-1</b>
Introduction . . . . .	17-1
Application Domains . . . . .	17-1
Circuit Elements Supported—Limitations . . . . .	17-1
Use of IEM, Tolerance Parameters . . . . .	17-2
Efficient Usage of IEM . . . . .	17-3
Examples for IEM . . . . .	17-4
Introduction . . . . .	17-4
Stability . . . . .	17-5
Example 1—bjtinv . . . . .	17-5
Example 2—ivdd . . . . .	17-7
Accuracy . . . . .	17-10
Example 3—oscil . . . . .	17-10
Example 4—moy1 . . . . .	17-13
Speed . . . . .	17-15
Example 5—ladder . . . . .	17-15
Example 6—opamp_5pin . . . . .	17-17

<b>Chapter 18</b>		
<b>Pole-Zero Post-processor . . . . .</b>		<b>18-1</b>
Introduction . . . . .		18-1
Running the Dialog for Pole-Zero Analysis . . . . .		18-2
Frequency Limit . . . . .		18-3
Pole-Zero Cancellation by Threshold . . . . .		18-3
Hand Selection . . . . .		18-3
Select Index . . . . .		18-4
FNS Model . . . . .		18-4
<b>Chapter 19</b>		
<b>Monte Carlo Analysis . . . . .</b>		<b>19-1</b>
Introduction . . . . .		19-1
Usage . . . . .		19-1
Define the .MC Command . . . . .		19-1
Assign Nominal Parameter Values . . . . .		19-3
LOT/DEV Correlation . . . . .		19-4
Define MC Parameters . . . . .		19-5
Examples . . . . .		19-8
<b>Chapter 20</b>		
<b>Optimizer in Eldo . . . . .</b>		<b>20-1</b>
Introduction . . . . .		20-1
Overview . . . . .		20-1
Performing an Optimization . . . . .		20-2
Designing a Low Noise Amplifier Example . . . . .		20-5
Robust Optimization Using Corners . . . . .		20-8
Basic Notions and Definitions in Optimization . . . . .		20-11
Known Limitations . . . . .		20-21
Modeling Capabilities in Eldo . . . . .		20-23
Optimization Variables Specification . . . . .		20-23
Problem Statement and Measurements Specification . . . . .		20-28
Objective Specification . . . . .		20-29
Inequality Constraints Specification . . . . .		20-31
Equality Constraints Specification . . . . .		20-32
Data and Curve Fitting . . . . .		20-33
Interpolated Measurements . . . . .		20-37
Data Driven Analyses . . . . .		20-37
The Optimization Command . . . . .		20-38
Command Syntax . . . . .		20-38
Dichotomy and Pass/Fail Methods . . . . .		20-40
Command Syntax . . . . .		20-40
Set-up Time Computation . . . . .		20-41
Exploiting Output Results . . . . .		20-42
Optimizer Output Control Options . . . . .		20-45
References . . . . .		20-46

---

## Table of Contents

---

<b>Chapter 21</b>		
<b>Reliability Simulation.....</b>		<b>21-1</b>
Introduction .....	21-1	
Running Reliability Simulation in Eldo.....	21-1	
Reliability Commands .....	21-1	
.AGE .....	21-2	
.AGEMODEL .....	21-5	
.SETKEY .....	21-6	
.USEKEY .....	21-7	
Monitoring STRESS Values.....	21-7	
Reliability Simulation Limitations .....	21-8	
<b>Chapter 22</b>		
<b>Post-Processing Library.....</b>		<b>22-1</b>
Introduction .....	22-1	
Registering User Defined Functions inside Eldo.....	22-1	
.USE_TCL Command—Use Tcl File.....	22-1	
Macro-like Usage of UDF .....	22-1	
Keyword Parameters .....	22-2	
Making Specific Calls to UDF inside Eldo .....	22-3	
.CALL_TCL Command—Call Tcl Function .....	22-3	
Working with Waveforms inside UDF .....	22-4	
Waveforms in Arguments .....	22-4	
Waveforms in Expressions .....	22-4	
evalExpr Command .....	22-5	
defineVec Command .....	22-6	
Built-in Waveform Functions .....	22-6	
RF Functions .....	22-16	
Built-in DSP Functions .....	22-18	
Accessing Waves inside an External Database .....	22-26	
Miscellaneous Commands .....	22-28	
Examples .....	22-29	
<b>Chapter 23</b>		
<b>IBIS Models support in Eldo .....</b>		<b>23-1</b>
Introduction .....	23-1	
General Syntax.....	23-1	
Detailed Syntax .....	23-6	
Output Buffer .....	23-6	
Output ECL Buffer .....	23-8	
Input Buffers .....	23-9	
input_ecl .....	23-10	
IO Buffer .....	23-10	
IO ECL Buffer .....	23-10	
tristate Buffer .....	23-10	
tristate ECL Buffer .....	23-11	

<b>Chapter 24</b>	
<b>Tutorials.....</b>	<b>24-1</b>
Introduction .....	24-1
Tutorial #1—Parallel LCR Circuit .....	24-2
Netlist Explanation .....	24-3
Simulation Results .....	24-4
Tutorial #2—4th Order Butterworth Filter .....	24-5
Netlist Explanation .....	24-5
Simulation Results—1 .....	24-7
Simulation Results—2 .....	24-8
Tutorial #3—Band Pass Filter .....	24-9
Netlist Explanation .....	24-10
Simulation Results .....	24-12
Tutorial #4—Low Pass Filter .....	24-13
Netlist Explanation .....	24-14
Simulation Results .....	24-15
Tutorial #5—Colpitts Oscillator .....	24-16
Netlist Explanation .....	24-17
Simulation Results .....	24-18
Tutorial #6—High Voltage Cascade .....	24-19
Netlist Explanation .....	24-20
Simulation Results—1 .....	24-21
Simulation Results—2 .....	24-22
Tutorial #7—Non-inverting Amplifier .....	24-23
Netlist Explanation .....	24-24
Simulation Results .....	24-25
Tutorial #8—Bipolar Amplifier.....	24-26
Netlist Explanation .....	24-27
Simulation Results .....	24-28
Tutorial #9—SC Low Pass Filter.....	24-29
Simulation Results—1 .....	24-30
Simulation Results—2 .....	24-31
<b>Appendix A</b>	
<b>Error Messages .....</b>	<b>A-1</b>
Error Message Classification .....	A-1
Warnings.....	A-1
Syntax Errors .....	A-1
Effects .....	A-1
Error Messages .....	A-2
Global Errors .....	A-2
Errors Related to Nodes .....	A-5
Errors Related to Objects .....	A-6
Errors Related to Commands .....	A-13
Errors Related to Models .....	A-20
Errors Related to AMODELS.....	A-21
Errors Related to Subcircuits .....	A-22
Miscellaneous Errors .....	A-22

## Table of Contents

---

Warning Messages . . . . .	A-26
Global Warnings . . . . .	A-26
Warnings Related to Nodes . . . . .	A-28
Warnings Related to Objects . . . . .	A-30
Warnings Related to Commands . . . . .	A-32
Warnings Related to Models . . . . .	A-38
Warnings Related to Subcircuits . . . . .	A-39
Miscellaneous Warnings . . . . .	A-40
<b>Appendix B</b>	
<b>Troubleshooting . . . . .</b>	<b>B-1</b>
Introduction . . . . .	B-1
Common Netlist Errors . . . . .	B-1
<b>Appendix C</b>	
<b>Examples . . . . .</b>	<b>C-1</b>
Introduction . . . . .	C-1
Example#1—SC Schmitt Trigger . . . . .	C-2
Simulation Results . . . . .	C-3
Example#2—4-bit Adder . . . . .	C-3
Simulation Results . . . . .	C-6
Example#3—CMOS Op-amp (Open Loop) . . . . .	C-7
Simulation Results . . . . .	C-9
Example#4—CMOS Op-amp (Closed Loop) . . . . .	C-9
Simulation Results . . . . .	C-11
Example#5—5th Order Elliptic SC Low Pass Filter . . . . .	C-11
Simulation Results . . . . .	C-14
Example#6—Charge Control in MOS 4 & 6 . . . . .	C-14
Simulation Results . . . . .	C-16
Example#7—Active RC Band Pass Filter . . . . .	C-16
Simulation Results . . . . .	C-19
Example#8—2nd Order Delta Sigma Modulator . . . . .	C-19
Simulation Results . . . . .	C-22
Example#9—Operational Amplifier . . . . .	C-22
Simulation Results—1 . . . . .	C-23
Simulation Results—2 (Zoom) . . . . .	C-24
<b>Appendix D</b>	
<b>Eldo Interactive Mode . . . . .</b>	<b>D-1</b>
Introduction . . . . .	D-1
Shared Commands . . . . .	D-2
To Read Information . . . . .	D-3
PRINT <expression> . . . . .	D-3
DISPLAY E string[*] . . . . .	D-4
To Reset Several Features . . . . .	D-4
DC Control Options . . . . .	D-4
Accuracy Control Options . . . . .	D-4
Time-step Control Options . . . . .	D-5

Change Features .....	D-5
SET .....	D-5
DELETE .....	D-6
DISABLE .....	D-6
ENABLE .....	D-6
CHMOD .....	D-7
FORCE .....	D-7
HIGH .....	D-7
LOW .....	D-7
PWL .....	D-7
RELEASE .....	D-7
TRANSITION .....	D-8
SETBUS .....	D-8
BUS .....	D-9
To Control Execution .....	D-9
LOAD <filename> .....	D-9
SAVESIM <filename> .....	D-9
STOP IF <condition> .....	D-10
STOP SIMU .....	D-10
RUN .....	D-10
NEXT [SIMU] .....	D-10
CONT .....	D-11
QUIT .....	D-11
CONNECT XELGA .....	D-11
VIEW plot_name .....	
UNVIEW plot_name .....	D-11
<b>Appendix E</b>	
<b>Eldo Utilities .....</b>	<b>E-1</b>
Utility to Convert .chi to .cir .....	E-1
Eldo Encryption .....	E-2
<b>Appendix F</b>	
<b>Spectre to Eldo Converter .....</b>	<b>F-1</b>
Prerequisites .....	F-1
Supported Features .....	F-1
Netlist Conversion .....	F-2
Supported Features .....	F-2
Limitations .....	F-3
Usage .....	F-4
<b>Appendix G</b>	
<b>EZwave and Xelga Differences .....</b>	<b>G-1</b>
Introduction .....	G-1
Xelga .....	G-1
EZwave .....	G-2
Xelga-EZwave cross references .....	G-5

## Table of Contents

---

<b>Appendix H</b>		
<b>STMicroelectronics Models .....</b>		<b>H-1</b>
Introduction .....		H-1
How to Invoke Eldo-ST .....		H-1
What Does it Change? .....		H-1
Licensing.....		H-2
STMicroelectronics Version of Eldo.....		H-2
Junction Diode Models .....		H-2
MOSFET Models .....		H-3
Bipolar Junction Transistor Model.....		H-3

## Index

## End-User License Agreement

# List of Figures

---

Figure 1-1. Eldo Input & Output Files.....	1-2
Figure 1-2. Cascaded Inverter Circuit .....	1-4
Figure 1-3. Inverter Subcircuit.....	1-5
Figure 1-4. EZwave output (.wdb).....	1-7
Figure 4-1. Coupled Inductor.....	4-27
Figure 4-2. Cross-Sectional View of Diffused Resistor .....	4-35
Figure 4-3. Microstrip Line Structure .....	4-54
Figure 4-4. Covered Pair Microstrip Line Structure .....	4-54
Figure 4-5. stripline Structure .....	4-54
Figure 4-6. Microstrip T Junction .....	4-77
Figure 4-7. Microstrip T Junction Symbol .....	4-77
Figure 4-8. Equivalent circuit Microstrip T Junction .....	4-78
Figure 4-9. Microstrip Bend (Arbitrary Angle, Optimally Mitered).....	4-80
Figure 4-10. Microstrip Bend (Arbitrary Angle, Optimally Mitered) Symbol .....	4-80
Figure 4-11. Equivalent circuit Microstrip Bend .....	
(Arbitrary Angle, Optimally Mitered).....	4-81
Figure 4-12. 90-degree Microstrip Bend (Mitered).....	4-83
Figure 4-13. 90-degree Microstrip Bend (Mitered) Symbol .....	4-83
Figure 4-14. Equivalent circuit MBEND2.....	4-84
Figure 4-15. 90-degree Microstrip Bend (Optimally Mitered) .....	4-85
Figure 4-16. 90-degree Microstrip Bend (Optimally Mitered) Symbol .....	4-85
Figure 4-17. Equivalent Circuit MBEND3 .....	4-87
Figure 4-18. 90-degree Microstrip Bend (Unmitedered).....	4-88
Figure 4-19. 90-degree Microstrip Bend (Unmitedered) Symbol .....	4-88
Figure 4-20. Equivalent circuit Microstrip corner.....	4-89
Figure 4-21. Microstrip Step in Width.....	4-90
Figure 4-22. Microstrip Step in Width Symbol .....	4-90
Figure 4-23. Equivalent circuit of a microstrip step in width .....	4-91
Figure 4-24. Cylindrical Via Hole in Microstrip .....	4-92
Figure 4-25. Cylindrical Via Hole in Microstrip Symbol .....	4-93
Figure 4-26. Equivalent circuit Cylindrical Via Hole in Microstrip .....	4-94
Figure 4-27. Unmitedered Stripline Bend .....	4-95
Figure 4-28. Unmitedered Stripline Bend Symbol .....	4-95
Figure 4-29. Equivalent circuit of an unmitedered stripline bend.....	4-96
Figure 4-30. Stripline T Junction.....	4-97
Figure 4-31. Stripline T Junction Symbol .....	4-97
Figure 4-32. Stripline Step in Width .....	4-99
Figure 4-33. Stripline Step in Width Symbol .....	4-99
Figure 4-34. Equivalent circuit for a Stripline Step in Width .....	4-100
Figure 5-1. AM Function Example .....	5-18

## List of Figures

---

Figure 5-2. Exponential Function . . . . .	5-20
Figure 5-3. Noise Function . . . . .	5-22
Figure 5-4. Pattern Function . . . . .	5-24
Figure 5-5. Pulse Function . . . . .	5-25
Figure 5-6. Voltage Source Characteristics . . . . .	5-26
Figure 5-7. Piece Wise Linear Function . . . . .	5-28
Figure 5-8. Piece Wise Linear Function Example 1 . . . . .	5-29
Figure 5-9. Piece Wise Linear Function Example 2 . . . . .	5-30
Figure 5-10. Single Frequency FM Function. . . . .	5-32
Figure 5-11. Sine Function . . . . .	5-33
Figure 5-12. 1st Sine Function Example . . . . .	5-34
Figure 5-13. 2nd Sine Function Example . . . . .	5-34
Figure 5-14. Trapezoidal Pulse with Bit Pattern . . . . .	5-36
Figure 5-15. Exponential Pulse with Bit Pattern . . . . .	5-38
Figure 6-1. Comparator Macromodel . . . . .	6-3
Figure 6-2. Op-amp (Linear) Macromodel . . . . .	6-5
Figure 6-3. Op-amp (Linear) Model Characteristics . . . . .	6-6
Figure 6-4. Op-amp (Linear 1-pole) Macromodel. . . . .	6-7
Figure 6-5. Op-amp (Linear 1-pole) Model Characteristics . . . . .	6-8
Figure 6-6. Op-amp (Linear 1-pole) Application Area . . . . .	6-9
Figure 6-7. Op-amp (Linear 1-pole) Frequency Response . . . . .	6-10
Figure 6-8. Op-amp (Linear 2-pole) Macromodel. . . . .	6-11
Figure 6-9. Op-amp (Linear 2-pole) Model Characteristic . . . . .	6-12
Figure 6-10. Delay Macromodel . . . . .	6-14
Figure 6-11. Saturating Resistor Macromodel. . . . .	6-15
Figure 6-12. Saturating Resistor Model Characteristics . . . . .	6-16
Figure 6-13. Voltage Limiter Macromodel . . . . .	6-17
Figure 6-14. Voltage Limiter Model Characteristics. . . . .	6-18
Figure 6-15. Voltage Controlled Switch Macromodel . . . . .	6-19
Figure 6-16. Current Controlled Switch Macromodel. . . . .	6-21
Figure 6-17. Triangular to Sine Wave Converter Macromodel. . . . .	6-24
Figure 6-18. Staircase Waveform Generator Macromodel . . . . .	6-26
Figure 6-19. Staircase Waveform Generator Model Characteristics. . . . .	6-27
Figure 6-20. Sawtooth Waveform Generator Macromodel. . . . .	6-28
Figure 6-21. Sawtooth Waveform Generator Model Characteristics . . . . .	6-28
Figure 6-22. Triangular Waveform Generator Macromodel . . . . .	6-30
Figure 6-23. Triangular Waveform Generator Model Characteristics. . . . .	6-31
Figure 6-24. Amplitude Modulator Macromodel . . . . .	6-32
Figure 6-25. Pulse Amplitude Modulator Macromodel. . . . .	6-34
Figure 6-26. Sample & Hold Macromodel . . . . .	6-36
Figure 6-27. Sample & Hold Model Characteristics . . . . .	6-37
Figure 6-28. Track & Hold Macromodel. . . . .	6-38
Figure 6-29. Track & Hold Model Characteristics . . . . .	6-39
Figure 6-30. Pulse Width Modulator Macromodel . . . . .	6-40
Figure 6-31. Pulse Width Modulator Model Characteristics. . . . .	6-41

Figure 6-32. Pulse Width Modulator Duty Cycle . . . . .	6-42
Figure 6-33. Voltage Controlled Oscillator Macromodel . . . . .	6-43
Figure 6-34. Peak Detector Macromodel. . . . .	6-45
Figure 6-35. Peak Detector Simulation Results. . . . .	6-47
Figure 6-36. Level Detector Macromodel . . . . .	6-48
Figure 6-37. Logarithmic Amplifier Macromodel. . . . .	6-50
Figure 6-38. Anti-logarithmic Amplifier Macromodel . . . . .	6-52
Figure 6-39. Differentiator Macromodel . . . . .	6-54
Figure 6-40. Integrator Macromodel . . . . .	6-56
Figure 6-41. Adder, Subtractor, Multiplier & Divider Macromodels . . . . .	6-58
Figure 7-1. Digital Model Parameter Thresholds . . . . .	7-4
Figure 7-2. Delay Macromodel . . . . .	7-6
Figure 7-3. Inverter Macromodel. . . . .	7-7
Figure 7-4. Analog to Digital Converter Macromodel . . . . .	7-14
Figure 7-5. Digital to Analog Converter Macromodel . . . . .	7-16
Figure 8-1. Transformer Winding Macromodel . . . . .	8-2
Figure 8-2. Non-linear Magnetic Core 1 Macromodel . . . . .	8-3
Figure 8-3. Symmetric B-H loops with Different Amplitudes . . . . .	8-5
Figure 8-4. Asymmetric Minor Loops. . . . .	8-5
Figure 8-5. Non-linear Magnetic Core 2 Macromodel . . . . .	8-6
Figure 8-6. Symmetric B-H loops with Different Amplitudes . . . . .	8-8
Figure 8-7. Asymmetric Minor Loops. . . . .	8-8
Figure 8-8. Linear Magnetic Core Macromodel . . . . .	8-9
Figure 8-9. Magnetic Air Gap Macromodel . . . . .	8-10
Figure 8-10. Transformer (Variable # of Windings) Macromodel . . . . .	8-11
Figure 8-11. Ideal Transformer Macromodel . . . . .	8-13
Figure 9-1. Operational Amplifier Macromodel . . . . .	9-3
Figure 9-2. Cascaded Amplifier Macromodel . . . . .	9-4
Figure 9-3. Equivalent Circuit (Single-stage Amplifier) . . . . .	9-5
Figure 9-4. Equivalent Circuit (Two-stage Amplifier) . . . . .	9-5
Figure 9-5. Switch Macromodel . . . . .	9-8
Figure 9-6. Closed Switch Equivalent Circuit . . . . .	9-9
Figure 9-7. Open Switch Equivalent Circuit . . . . .	9-9
Figure 9-8. NMOS Switch. . . . .	9-9
Figure 9-9. PMOS Switch . . . . .	9-10
Figure 9-10. Ideal Operational Amplifier Macromodel. . . . .	9-12
Figure 9-11. SC Integrators & LDI's. . . . .	9-13
Figure 9-12. Inverting Switched Capacitor Macromodel . . . . .	9-15
Figure 9-13. Non-inverting Switched Capacitor Macromodel . . . . .	9-17
Figure 9-14. Parallel Switched Capacitor Macromodel. . . . .	9-19
Figure 9-15. Serial Switched Capacitor . . . . .	9-21
Figure 9-16. Serial-parallel Switched Capacitor Macromodel . . . . .	9-23
Figure 9-17. Bi-linear Switched Capacitor Macromodel. . . . .	9-26
Figure 9-18. Unswitched Capacitor Macromodel . . . . .	9-28
Figure 9-19. Non-inverting Integrator . . . . .	9-31

## List of Figures

---

Figure 9-20. LDI Phase Control Non-inverting Integrator Macromodel . . . . .	9-32
Figure 9-21. Euler Forward Integrator Macromodel . . . . .	9-33
Figure 9-22. LDI Phase Control Euler Forward Integrator Macromodel . . . . .	9-34
Figure 9-23. Euler Backward Integrator Macromodel. . . . .	9-35
Figure 9-24. LDI Phase Control Euler Backward Integrator Macromodel . . . . .	9-36
Figure 10-1. Piece Wise Linear Example 1 . . . . .	10-36
Figure 10-2. Piece Wise Linear Example 2 . . . . .	10-36
Figure 10-3. PWL, PWL_CTE and PWL_LIN Functions. . . . .	10-73
Figure 10-4. Smith chart plot using STOSMITH/ZTOSMITH/YTOSMITH functions	10-249
Figure 14-1. Circuit Components with their Added Noise Sources . . . . .	14-2
Figure 14-2. High-rate Particle Detector Circuit . . . . .	14-4
Figure 14-3. Simulation Results—Input & Output Signals. . . . .	14-5
Figure 14-4. Noise at Amplifier O/P . . . . .	14-6
Figure 14-5. Noise at Analog Memory O/P. . . . .	14-7
Figure 14-6. Switched Capacitor Filter Circuit Schematic . . . . .	14-9
Figure 14-7. Amplifier Schematic . . . . .	14-10
Figure 14-8. AC & Noise Simulation Results of Amplifier . . . . .	14-11
Figure 14-9. AC & Noise Simulation of Amplifier Macromodel . . . . .	14-12
Figure 14-10. Simulation Results of the Filter . . . . .	14-15
Figure 14-11. Frequency Response of the Filter . . . . .	14-16
Figure 17-1. Example 1—bjtinv . . . . .	17-6
Figure 17-2. Example 2—ivdd . . . . .	17-9
Figure 17-3. Example 3—oscil . . . . .	17-13
Figure 17-4. Example 4—moy1 . . . . .	17-15
Figure 17-5. Example 5—ladder . . . . .	17-17
Figure 17-6. Example 6—opamp_5pin . . . . .	17-20
Figure 19-1. Monte Carlo Analysis Example . . . . .	19-9
Figure 20-1. Operating diagram of the Optimization process . . . . .	20-2
Figure 20-2. Examples of feasible regions . . . . .	20-13
Figure 20-3. Illustration of the constraints $c_l^{(i)} \leq c_I^{(i)}(x) \leq c_u^{(i)}$ . . . . .	20-14
Figure 20-4. Different types of minima . . . . .	20-16
Figure 20-5. The ad-hoc method of alternating variables . . . . .	20-18
Figure 20-6. Discretization of final parameters . . . . .	20-26
Figure 20-7. Deviation between the model and the extracted measures . . . . .	20-33
Figure 23-1. Subcircuit for the nodal representation of an IBIS driver. . . . .	23-7
Figure 23-2. Splitting of C_comp in case when all 4 keywords C_comp_pu, C_comp_pd, C_comp_gc, C_comp_pc are specified . . . . .	23-8
Figure 23-3. Subcircuit for the nodal representation of an IBIS load . . . . .	23-9
Figure 24-1. Parallel LCR Circuit . . . . .	24-2
Figure 24-2. Tutorial #1—Simulation Results . . . . .	24-4
Figure 24-3. 4th Order Butterworth Filter . . . . .	24-5
Figure 24-4. Tutorial #2—Simulation Results—1 . . . . .	24-7
Figure 24-5. Tutorial #2—Simulation Results—2 . . . . .	24-8
Figure 24-6. Band Pass Filter . . . . .	24-9
Figure 24-7. Tutorial #3—Simulation Results . . . . .	24-12

Figure 24-8. Low Pass Filter . . . . .	24-13
Figure 24-9. Tutorial #4—Simulation Results . . . . .	24-15
Figure 24-10. Colpitts Oscillator . . . . .	24-16
Figure 24-11. Tutorial #5—Simulation Results . . . . .	24-18
Figure 24-12. High Voltage Cascade. . . . .	24-19
Figure 24-13. Tutorial #6—Simulation Results—1 . . . . .	24-21
Figure 24-14. Tutorial #6—Simulation Results—2 . . . . .	24-22
Figure 24-15. Non-inverting Amplifier . . . . .	24-23
Figure 24-16. Tutorial #7—Simulation Results . . . . .	24-25
Figure 24-17. Bipolar Amplifier . . . . .	24-26
Figure 24-18. Tutorial #8—Simulation Results . . . . .	24-28
Figure 24-19. Tutorial #9—Simulation Results—1 . . . . .	24-30
Figure 24-20. Tutorial #9—Simulation Results—2 . . . . .	24-31
Figure C-1. Example#1—Simulation Results . . . . .	C-3
Figure C-2. Example#2—Simulation Results—1 . . . . .	C-6
Figure C-3. Example#2—Simulation Results—2 . . . . .	C-7
Figure C-4. Example#3—Simulation Results . . . . .	C-9
Figure C-5. Example#4—Simulation Results . . . . .	C-11
Figure C-6. Example#5—Simulation Results . . . . .	C-14
Figure C-7. Example#6—Simulation Results . . . . .	C-16
Figure C-8. Example#7—Simulation Results . . . . .	C-19
Figure C-9. Example#8—Simulation Results . . . . .	C-22
Figure C-10. Example#9—Simulation Results—1 . . . . .	C-23
Figure C-11. Example#9—Simulation Results—2 (Zoom) . . . . .	C-24

# List of Tables

---

Table 3-1. Reserved Keywords not available in .PARAM .....	3-3
Table 3-2. Arithmetic Functions & Operators .....	3-7
Table 3-3. Operator Precedence .....	3-10
Table 3-4. Boolean Operators .....	3-11
Table 3-5. Bitwise Operators .....	3-11
Table 4-1. Eldo Device Models .....	4-1
Table 4-2. Resistor Model—Level 1 Parameters .....	4-8
Table 4-3. Resistor Model—Level 2 Parameters .....	4-11
Table 4-4. Resistor Model—Level 4 Parameters .....	4-11
Table 4-5. Capacitor Model Parameters .....	4-17
Table 4-6. Inductor Model Parameters .....	4-25
Table 4-7. RC Wire Model Parameters .....	4-30
Table 4-8. Semiconductor Resistor Model Parameters .....	4-35
Table 4-9. LDTL Level 3 Parameter Combinations .....	4-53
Table 4-10. RLGC Separator Characters .....	4-63
Table 4-11. Lossy Transmission Line: U Model Parameter Combinations .....	4-74
Table 4-12. Microstrip T Junction Parameters .....	4-77
Table 4-13. Microstrip Bend (Arbitrary Angle, Optimally Mitered) Parameters .....	4-80
Table 4-14. 90-degree Microstrip Bend (Mitered) Parameters .....	4-83
Table 4-15. 90-degree Microstrip Bend (Optimally Mitered) Parameters .....	4-86
Table 4-16. 90-degree Microstrip Bend (Unmiterred) Parameters .....	4-88
Table 4-17. Microstrip Step in Width Parameters .....	4-91
Table 4-18. Cylindrical Via Hole in Microstrip Parameters .....	4-93
Table 4-19. Unmiterred stripline Bend Parameters .....	4-95
Table 4-20. stripline T Junction Parameters .....	4-98
Table 4-21. stripline Step in Width Parameters .....	4-100
Table 4-22. Diode Models .....	4-105
Table 4-23. Diode Models with -compat .....	4-106
Table 4-24. BJT Models .....	4-112
Table 4-25. BJT Models with -compat .....	4-113
Table 4-26. VBIC Version Selection .....	4-115
Table 4-27. HICUM Version Selection .....	4-116
Table 4-28. JFET Models .....	4-122
Table 4-29. MESFET Models .....	4-123
Table 4-30. MOSFET Models .....	4-127
Table 4-31. MOS Levels with -compat .....	4-129
Table 4-32. MOS Noise Models .....	4-130
Table 4-33. MOS Noise Models .....	4-131
Table 4-34. MOSFET Common Parameters .....	4-133

Table 4-35. RLEV Default Values .....	4-137
Table 4-36. ALEV Default Values .....	4-137
Table 4-37. DIOLEV Default Values .....	4-137
Table 4-38. DCAPLEV Default Values .....	4-138
Table 4-39. Merckel-Spice .....	4-141
Table 4-40. EKV MOS Models .....	4-145
Table 4-41. BSIM3v3 Models .....	4-147
Table 4-42. Berkeley BSIM3v3 States .....	4-151
Table 4-43. BSIM3SOI Version Selection .....	4-154
Table 4-44. SOIMOD Selection .....	4-154
Table 4-45. Philips MOS9 Version Selection .....	4-158
Table 4-46. Berkeley BSIM4 Version Selection .....	4-159
Table 4-47. Philips MOS Model 11 Version Selection .....	4-160
Table 4-48. HiSIM Version Selection .....	4-163
Table 6-1. Comparator Parameters .....	6-4
Table 6-2. Op-amp (Linear) Model Parameters .....	6-5
Table 6-3. Op-amp (Linear 1-pole) Model Parameters .....	6-7
Table 6-4. Op-amp (Linear 2-pole) Model Parameters .....	6-11
Table 6-5. Saturating Resistor Model Parameters .....	6-15
Table 6-6. Voltage Limiter Model Parameters .....	6-17
Table 6-7. Voltage Controlled Switch Model Parameters .....	6-19
Table 6-8. Current Controlled Switch Model Parameters .....	6-21
Table 6-9. Triangular to Sine Wave Converter Model Parameters .....	6-24
Table 6-10. Staircase Waveform Generator Model Parameters .....	6-26
Table 6-11. Sawtooth Waveform Generator Model Parameters .....	6-28
Table 6-12. Triangular Waveform Generator Model Parameters .....	6-30
Table 6-13. Amplitude Modulator Model Parameters .....	6-32
Table 6-14. Pulse Amplitude Modulator Model Parameters .....	6-34
Table 6-15. Sample & Hold Model Parameters .....	6-36
Table 6-16. Track & Hold Model Parameters .....	6-38
Table 6-17. Pulse Width Modulator Model Parameters .....	6-40
Table 6-18. Voltage Controlled Oscillator Model Parameters .....	6-43
Table 6-19. Peak Detector Model Parameters .....	6-45
Table 6-20. Level Detector Model Parameters .....	6-48
Table 6-21. Logarithmic Amplifier Model Parameters .....	6-50
Table 6-22. Anti-logarithmic Amplifier Model Parameters .....	6-52
Table 6-23. Differentiator Model Parameters .....	6-54
Table 6-24. Integrator Model Parameters .....	6-56
Table 6-25. Adder, Subtractor, Multiplier & Divider Model Parameters .....	6-58
Table 7-1. 2-Input Digital Gate Types .....	7-9
Table 7-2. 3-Input Digital Gate Types .....	7-11
Table 7-3. Multiple Input Digital Gate Types .....	7-12
Table 8-1. Transformer Winding Model Parameters .....	8-2
Table 8-2. Non-linear Magnetic Core 1 Model Parameters .....	8-3
Table 8-3. Non-linear Magnetic Core 2 .....	8-6

## List of Tables

---

Table 8-4. Linear Magnetic Core Model Parameters .....	8-9
Table 8-5. Magnetic Air Gap Model Parameters .....	8-10
Table 8-6. Transformer Model Parameters .....	8-11
Table 9-1. Operational Amplifier Model Parameters .....	9-6
Table 9-2. Switch Model Parameters .....	9-10
Table 9-3. Inverting Switched Capacitor Model Parameters .....	9-15
Table 9-4. Non-inverting Switched Capacitor Model Parameters .....	9-17
Table 9-5. Parallel Switched Capacitor .....	9-19
Table 9-6. Serial Switched Capacitor Model Parameters .....	9-21
Table 9-7. Serial-parallel Switched Capacitor Model Parameters .....	9-24
Table 9-8. Bi-linear Switched Capacitor Model Parameters .....	9-26
Table 9-9. Unswitched Capacitor Model Parameters .....	9-28
Table 10-1. Eldo Commands .....	10-1
Table 10-2. .A2D - Global Parameters .....	10-5
Table 10-3. .D2A Global Parameters .....	10-44
Table 10-4. Extract Function Parameters .....	10-99
Table 10-5. Transient Extraction Language Functions .....	10-100
Table 10-6. General Extraction Language Functions .....	10-101
Table 10-7. XYCOND Arithmetic Expressions .....	10-110
Table 10-8. LSTB Output Formats .....	10-144
Table 10-9. Model Types .....	10-160
Table 10-10. Operating Point—optyp values .....	10-189
Table 10-11. Operating Point—optyp values .....	
Dynamic Part for Charge Control Model .....	10-190
Table 10-12. Spice3e Capacitance / Eldo Capacitance .....	10-191
Table 10-13. Default Values for Time Window Parameters .....	10-192
Table 10-14. Reserved Keywords not available in .PARAM .....	10-206
Table 10-15. Two-port Parameter Keywords .....	10-223
Table 10-16. MOSFET Plotting and Printing .....	10-226
Table 10-17. BJT Plotting and Printing .....	10-229
Table 10-18. JFET Plotting and Printing .....	10-231
Table 10-19. Diode Plotting and Printing .....	10-232
Table 10-20. Common Plotting and Printing .....	10-233
Table 10-21. Element Output .....	10-237
Table 10-22. Two-port Parameters .....	10-242
Table 10-23. Special Characters .....	10-264
Table 11-1. Simulator Compatibility Options .....	11-2
Table 11-2. Netlist Parser Control Options .....	11-2
Table 11-3. Simulation Speed, Accuracy & Efficiency Options .....	11-2
Table 11-4. Miscellaneous Simulation Control Options .....	11-3
Table 11-5. Model Control Options .....	11-3
Table 11-6. RC Reduction Options .....	11-4
Table 11-7. Noise Analysis Options .....	11-4
Table 11-8. Simulation Display Control Options .....	11-4
Table 11-9. Simulation Output Control Options .....	11-4

Table 11-10. Optimizer Output Control Options .....	11-5
Table 11-11. File Generation Options .....	11-5
Table 11-12. Mathematical Algorithm Options .....	11-5
Table 11-13. Mixed-Mode Options .....	11-5
Table 11-14. Other Options .....	11-5
Table 12-1. MOS Levels with -compat .....	12-2
Table 12-2. BJT Models with -compat .....	12-2
Table 12-3. Diode Models with -compat .....	12-3
Table 12-4. Resistor Level with -compat .....	12-3
Table 13-1. MOS Levels in TIspice Compatibility Mode .....	13-1
Table 13-2. BJT Models in TIspice Compatibility Mode .....	13-1
Table 13-3. Supported Keywords for Voltage on a Device .....	13-5
Table 15-1. Default Rule .....	15-3
Table 16-1. Global Tuning of Accuracy .....	16-10
Table 22-1. Equivalence between C and PPL syntax .....	22-4
Table 22-2. Built-in Waveform Functions .....	22-6
Table 22-3. Built-in DSP functions .....	22-18
Table 22-4. Post-Processing Library Examples .....	22-29
Table 23-1. Corner Selection .....	23-4
Table 24-1. Tutorials .....	24-1
Table A-1. Global Errors .....	A-2
Table A-2. Errors Related to Nodes .....	A-5
Table A-3. Errors Related to Objects .....	A-6
Table A-4. Errors Related to Commands .....	A-13
Table A-5. Errors Related to Models .....	A-20
Table A-6. Errors Related to AMODELS .....	A-21
Table A-7. Errors Related to Subcircuits .....	A-22
Table A-8. Miscellaneous Errors .....	A-22
Table A-9. Global Warnings .....	A-26
Table A-10. Warnings Related to Nodes .....	A-28
Table A-11. Warnings Related to Objects .....	A-30
Table A-12. Warnings Related to Commands .....	A-32
Table A-13. Warnings Related to Models .....	A-38
Table A-14. Warnings Related to Subcircuits .....	A-39
Table A-15. Miscellaneous Warnings .....	A-40
Table A-16. Miscellaneous Warnings .....	A-41
Table B-1. Reserved Keywords not available in .PARAM .....	B-2
Table C-1. Eldo Examples .....	C-1
Table G-1. Xelga-EZwave cross references .....	G-5
Table H-1. Eldo/Eldo-ST Model Levels .....	H-1
Table H-2. STMicroelectronics Model Selection .....	H-2
Table H-3. STMicroelectronics Junction Diode Model Selection .....	H-2
Table H-4. STMicroelectronics MOSFET Model Selection .....	H-3
Table H-5. STMicroelectronics BJT Model Selection .....	H-3

# Chapter 1

## Introduction to Eldo

---

### Introduction

The Eldo™ analog simulator is the core component of a comprehensive suite of analog and mixed-signal simulation tools. Eldo offers a unique partitioning scheme allowing the use of different algorithms on differing portions of design. It allows the user a flexible control of simulation accuracy using a wide range of device model libraries, and gives a high accuracy yield in combination with high speed and high performance.

### Overview

The following is a list of the major product features of Eldo:

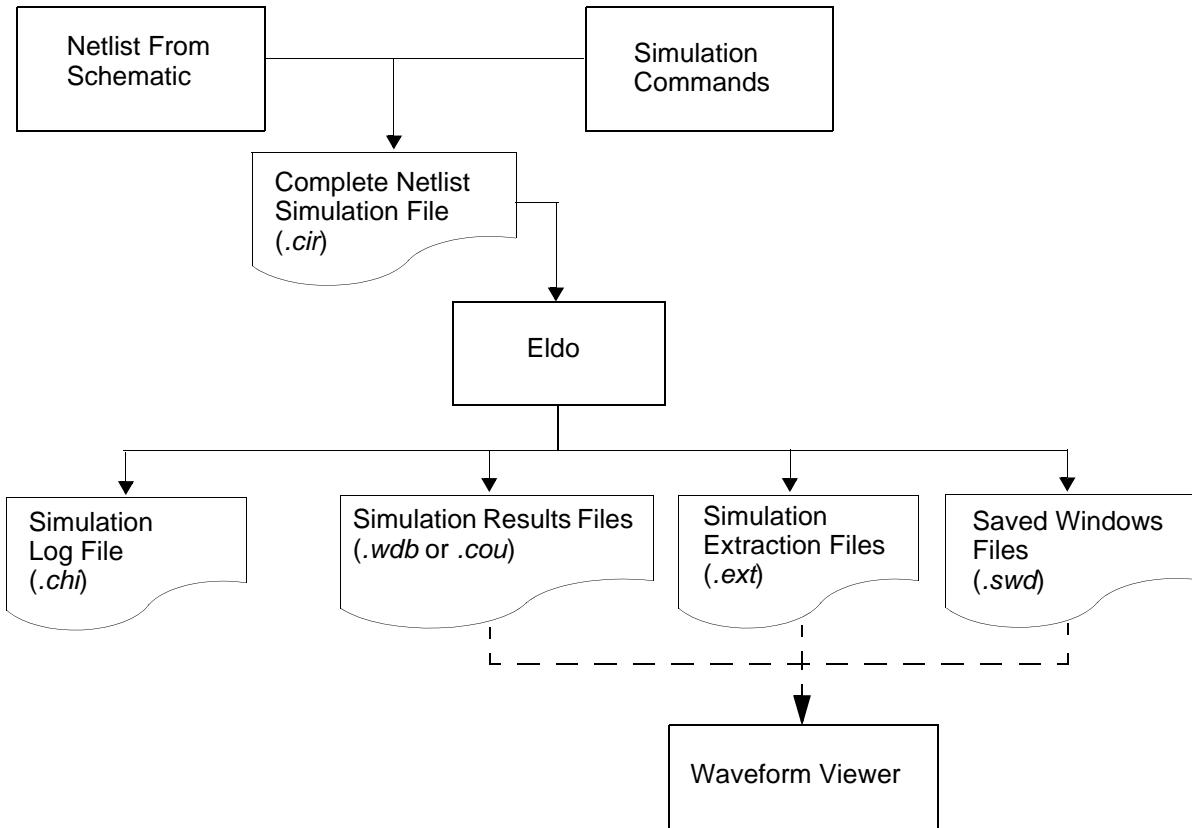
- Eldo is the core technology allowing to address RF simulation (Eldo RF) and mixed-signal (ADVance MS and ADVance MS Mach)
- Simulation of very large circuits (up to around 300,000 transistors) in time and frequency domains
- 3× to 10× gain in simulation speed over other commercial SPICE simulators, while maintaining same accuracy
- Three complementary transient simulation algorithms (OSR, Newton, IEM)
- Flexible user control of simulation accuracy
- Unique transient noise algorithm
- Advanced analysis options such as pole-zero, enhanced Monte-Carlo analysis
- S and Z-domain generalized transfer functions
- Reliability simulation
- Extensive device model libraries including leading MOS, bipolar and MESFET transistor models such as the BSIM3v3.x, BSIM 4, MM9, Mextram and HICUM
- IBIS (I/O Buffer Information Specification) model support
- Integration into Mentor Graphics IC flow, consisting of Design Architect IC for schematic capture, IC station for the layout side, and Calibre/Calibre xRC for DRC/LVS and extraction. This flow provides a complete, front-to-back design and verification environment for analog, mixed-signal and RF.

- Integration into Cadence's Analog Artist environment (Artist Link)

## Eldo Input and Output Files

The following flowchart shows the input files that must be provided for an Eldo simulation run and the output files that Eldo produces:

**Figure 1-1. Eldo Input & Output Files**



The figure above shows the main files used by Eldo. A brief description of each is given below:

*<file>.cir*

The main Eldo control file, containing circuit netlist, stimulus and simulation control commands. This file is SPICE compatible, the Eldo control language being a superset of the Berkeley SPICE syntax.

*<file>.chi*

SPICE compatible output log file containing ASCII data, including results and error messages.

*<file>.wdb*

A binary output file for mixed-signal JWDB format files. Viewed with the EZwave waveform viewer. The resulting output file is smaller than files based on the cou format files.

<file>.swd	A saved windows file used by the EZwave waveform viewer. This file contains information on waveforms and their display and cursor settings, window format settings and complex waveform transition settings.
<file>.cou	A binary output file containing Eldo analog simulation results data. A special interface is provided to access this data from your own post-processor software if required. MGC post-processors also read and write to this file. Please refer to <i>Eldo cou Library User's Manual</i> for more details.
<file>.ext	A file containing extraction or waveform information, created when using a <b>.EXTRACT</b> command in the netlist. This file will not always be output, it depends on the type of simulation and the specification of the <b>.EXTRACT</b> command.

Other files not shown in the figure are:

<file>.meas	A file containing extraction or waveform information when the commands <b>.EXTRACT</b> or <b>.PLOT W(XX)</b> are present in an input netlist. This type of file is also generated when functions are used in <b>.DEFWAVE</b> commands. In most cases, the result of such functions are known only at the end of the simulation, so the waves issued from a <b>.DEFWAVE</b> can not be plotted in the standard <b>.cou</b> file, but only in a specific file generated at the end of the simulation. This file has the <b>.meas</b> extension. This kind of file is empty if you don't use <b>.PLOT</b> or <b>.PRINT</b> commands in the netlist.
<file>.aex	A file containing extraction information, created when <b>.OPTION AEX</b> is used in conjunction with a <b>.EXTRACT</b> command.
<file>.pz	The output file used by the Pole Zero post-processor.
<file>.spi3	SPICE3 compatible output file.
<file>.wsf	Cadence compatible output file.

The **.pz**, **.spi3**, **.wsf**, **.wdb** and **.isdb** files are only produced by Eldo when the user has set appropriate options in the input file.



For more details, refer to [Simulator Commands](#).

---

# Getting Started

## How to Run Eldo

To run an Eldo simulation, a *.cir* control file must be supplied to the simulator. As can be seen in the previous example, this file must include the following:

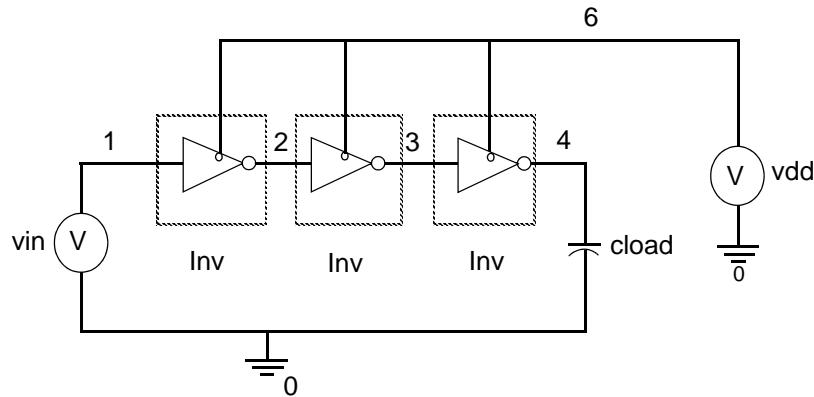
- Circuit connectivity, i.e. a netlist.
- Model parameter values defining the specific device models to be used.
- Electrical stimuli (sources).
- Simulation options and commands.

Input and output formats are compatible with Berkeley SPICE 2G6, however Eldo provides additional features not implemented in SPICE.

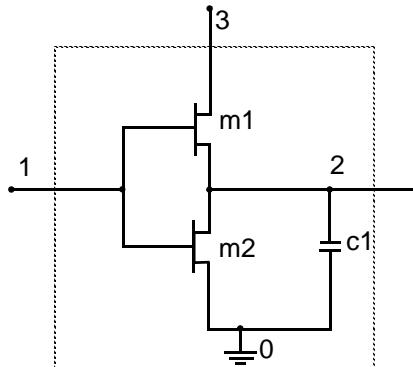
## Schematic Example

This example consists of a simple cascade of three inverters. The figures below show the circuit diagram for the cascade together with the inverter subcircuit. In order to create the Eldo netlist, node names must be assigned to the circuit. The complete netlist is shown on the following page.

**Figure 1-2. Cascaded Inverter Circuit**



**Figure 1-3. Inverter Subcircuit**



## Associated Netlist

The *.cir* control file for Eldo can be generated using a basic text editor, or alternatively a schematic editor that is capable of generating Spice-like format.



For further information, please refer to [Eldo Control Language](#).

## Sample circuit .cir control file

```
* MOS model definitions
.model m1 nmos level=3 vto=1v uo=550 vmax=2.0e5
+ cgdo=0.4p cgbo=2.0e-10 cgso=4.0e-11 cjsw=10.e-9
+ mjsw=0.3 tox=1.0e-7 nsub=1.0e16 nfs=1.5e10
+ xj=0.5u ld=0.5u pb=0.75 delta=0.9 eta=0.95
+ kappa=0.45 gamma=0.37
.model p1 pmos level=3 vto=-1v uo=230 vmax=1.9e5
+ cgdo=0.4p cgbo=2.0e-10 cgso=4.0e-11 cjsw=10.e-9
+ mjsw=0.3 tox=1.0e-7 nsub=1.0e16 nfs=1.5e10
+ xj=0.5u ld=0.5u pb=0.75 delta=0.9 eta=0.95
+ kappa=0.45 gamma=0.37
* Subcircuit definition
.subckt inv 1 2 3
m2 2 1 0 0 m1 w=10u l=4u ad=100p pd=40u as=100p
m1 2 1 3 3 p1 w=15u l=4u ad=100p pd=40u as=100p
c1 2 0 0.5p
.ends inv
* Subcircuit calls
x1 1 2 6 inv
x2 2 3 6 inv
x3 3 4 6 inv
cload 4 0 1p
* Electrical source definitions
vdd 6 0 5v
vin 1 0 pulse(0 5 10e-9 5e-9 5e-9 30e-9 50e-9)
```

```
* Simulation options & commands
.tran 0.5n 100n uic
.ic v(1)=0
.plot tran v(1) v(2) v(3) v(4)
.print tran v(1) v(2) v(3) v(4)
.option eps=0.5e-3 tnom=50 list node
.end
```

---

 Please see the [Documentation Conventions](#) for a detailed description on the meanings of the different fonts, brackets etc. used throughout this manual.

---

## Running a Simulation

To run a simulation from the command line (see “[Running Eldo](#)” on page 2-1 for a full list of options) use the following command:

```
eldo cir_file_name.cir
```

After the simulation has been completed, Eldo writes simulator information to the `.chi` file. This file will contain details of the simulation including any warning or error messages, which may have been encountered during simulation. A binary `.wdb` file is also generated by default as an output of simulation. The user can view the results written to the `.wdb` file, with the EZwave viewer. The `.wdb` file can be opened in the EZwave viewer the using the following command:

```
ezwave cir_file_name.wdb
```

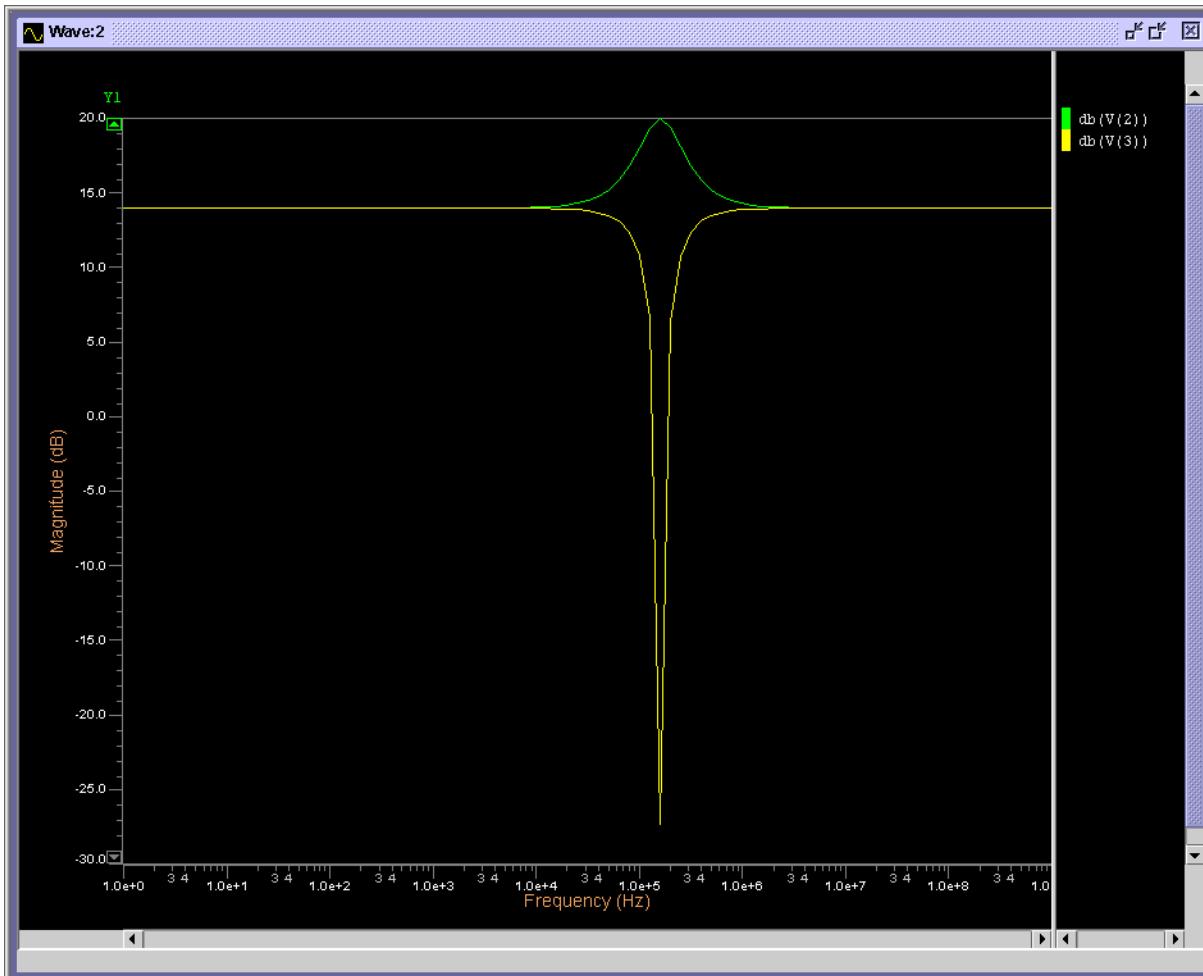
---

 For further information regarding the use of EZwave, please refer to the *EZwave User’s Manual*.

---

See [Figure 1-4](#) for an example binary output file (`.wdb`) viewed with EZwave.

Figure 1-4. EZwave output (.wdb)





# Chapter 2

## Running Eldo

---

### Running Eldo from the Command Line

When invoking Eldo at the command line, the following command line flags can be used:

```
eldo  cir_filename
      [-b] [-l log_filename] [-rel release_number] [-queue]
      [-lib object_library_dir_name] [-searchpath path1:path2:path3]
      [-o output_filename] [-wB cou_filename] [-outpath output_dir_name]
      [-i] [-crypt] [-oldcrypt] [-clean] [-float] [-cou47]
      [-noascii] [-nochi] [-nogwl] [-gwl gwl_lib_name] [-cou] [-noconf]
      [-ezwave [-isaving[val]] [-jwdb_threshold[val]] [-wdb_config swd_file]
      [-wdl_timeout]] [-spiout] [-ri] [-dbp] [-m53] [-E] [-EE] [-define macro]
      [-tuning [FAST|STANDARD|ACCURATE|VHIGH|BACKANNOTATE]] [-h hostname]
      [-libinc] [-couext] [-nocouext] [-extract] [-inter] [-eil file]
      [-part_ui] [-savetime time] [-m subckt_name] [-64b]
      [-restart ["[save_file] [file=wfile]"] [-probeop2]
      [-compat] [-compmod] [-compnet] [-noinit] [-nojwdb]
      [-jwdb_nocomplex] [-jwdb_norffolder] [-jwdb_servermode]
      [-out] [-outname] [-alter alter_name | alter_index] [-silent] [-verbose]
      [-mthread] [-cntthread] [-usethread]
      [-help commands|devices|sources|manual]
```

It is also possible to create a file named *eldo.ini* which will be interpreted and loaded at the very beginning of each simulation. “Loading *eldo.ini*” is displayed whenever a valid *eldo.ini* file is found. See “[Eldo Initialization File](#)” on page 2-11 for further details.

All command line flags are optional, except for the *cir\_filename* which is mandatory. The function of these command line flags is as follows:

- **cir\_filename**  
Name of the *.cir* control file to be simulated. Default extension is *.cir*.
- **-b**  
Runs the simulation in the background. If not specified, the simulation is interactive.
- **-l**  
Log file name for a background simulation (-b). If not specified, the log file is set to *eldo\_<PID>.log*, where *PID* is the process identifier.
- **-rel**  
Software release number. If not specified, the current release is used. Specified on its own no simulation is performed.

- **-queue**  
If there is no license available for Eldo, the job will be queued. Used for batch runs.
- **-lib**  
Name of the directory containing Eldo object library files for dynamic linking. This option is typically only used for Eldo-UDM/GUDM/UDRM analog behavioral modeling. Eldo will search for the specified directory and then search for Eldo library files (*libeldo*\*) inside this directory, if either of these are not found Eldo will issue a warning. If not specified, the directory is set to: *\$MGC\_AMS\_HOME/eldo/\$eldover/\$AMS\_MACHINE*
- **-searchpath path\_list**  
Libraries and include files are searched inside the specified path when not found. There is no limit on the number of search paths that can be used. This has the same effect as **SEARCH=path1** option, see [page 11-60](#).
- **-o**  
Output *.chi* file name. If not specified, the *.chi* file is set to *cir\_filename.chi*. *output\_filename* has to be the full pathname of the ASCII output file. If *output\_filename* does not contain any ‘/’ character (i.e. if *output\_filename* is just a filename with no path), then any specified *output\_dir\_name* string will be used to create *<output\_dir\_name>/<output\_filename>*.
- **-wB**  
Output *.cou* file name. *cou\_filename* has to be the full pathname of the binary cou output file. If *cou\_filename* does not contain any ‘/’ character (i.e. if *cou\_filename* is just a filename with no path), then any specified *output\_dir\_name* string will be used to create *<output\_dir\_name>/<cou\_filename>*. This flag should not be used with JWDB output.
- **-outpath**  
Directory in which all output files are created. If not specified, output files are created in the same directory as the *.cir* input file. The final character of *output\_dir\_name* must be a forward slash ‘/’. If omitted, this will be added automatically.
- **-i**  
Input *.cir* file name. The *.cir* input file is mandatory and so this flag identifier is optional. The last parameter in the command line is taken to be the *cir\_filename* by default.
- **-crypt**  
Enables the use of ELIB-PLUS encrypted libraries.
- **-oldcrypt**  
Specify this flag to read libraries which have been encrypted using a previous version of Eldo. The Eldo encryption key changed in v6.1. By default, Eldo expects a new version of encrypted libraries.

- `-clean`

Remove all old intermediate files for the simulation *filename* specified. Useful for cleaning up old intermediate simulation files left around after a crashed simulation.

- `-float`

Forces Eldo to create *.cou* / *.wdb* files containing FLOAT rather than DOUBLE values, which results in saving 50% of the disk space. This is useful when it is known that the output file will be very large. However, for FFT, floating numbers are sometimes inappropriate as they cause a loss in accuracy. Note this flag was previously named `-couf`.

- `-cou47`

Forces Eldo to create a different *.cou* file format (4.7) which is used by Xelga. This format allows faster loading of a large database with many curves and/or several simulations from which only a few number of curves should be loaded.



Please refer to the *Eldo cou Library User's Manual* for further information.

- `-noascii`

Prevents plot/print information from being produced in the ASCII output (*.chi* file), see also **NOASCII** option on [page 11-44](#).

- `-nochi`

Prevents ASCII output from being produced (no *.chi* file is produced).

- `-nogwl`

All output files are generated except waveform files (*.cou*, *.wdb*, *.tr0...*)

- `-cou`

Generate output in binary Cou format file, see also **COU** option on [page 11-52](#). This can also be specified using the invoke command `-gwl cou`. When JWDB output is not disabled, the *.cou* database will only contain real and imaginary parts of complex waveforms. Phase, magnitude and dB formats can be built dynamically from real and imaginary waveforms by Xelga, but only real and imaginary parts of complex waveforms will be displayed in Xelga when opening the *.cou* file, whatever the format requested in the netlist. To avoid this limitation, use `-jwdb_nocomplex` or `-nojwdb`.

- `-gwl gwl_lib_name`

Forces Eldo to generate output in the required format (JWDB, etc.).

`-gwl jwdb`

Generate JWDB format files (extension *.wdb*). This is always generated by default, see also **JWDB** option on [page 11-53](#).

`-gwl cou`

Generate binary Cou format file, see also **COU** option on [page 11-52](#). When JWDB output is not disabled, the *.cou* database will only contain real and imaginary parts of

complex waveforms. Phase, magnitude and dB formats can be built dynamically from real and imaginary waveforms by Xelga, but only real and imaginary parts of complex waveforms will be displayed in Xelga when opening the .cou file, whatever the format requested in the netlist. To avoid this limitation, use  
`-jwdb_nocomplex` or `-nojwdb`.

`-gwl csdf`

Generate CSDF format file. CSDF is the Common Simulation Data Format, see also [CSDF option on page 11-52](#).

`-gwl isdb`

Generate ISDB format for SimWave, see also [ISDB option on page 11-53](#).

`-gwl psf`

Generate binary Cadence format (used with Artist Link), see also [PSF option on page 11-54](#).

`-gwl psfascii`

Generate Cadence format in ASCII, see also [PSFASCII option on page 11-54](#).

`-gwl psfop`

Generate output for OP data in binary Cadence format (used with Artist Link). Can be used if PSF files are required for back-annotation of the OP results.

`-gwl psfasciio`

Generate output for OP data in ASCII Cadence format (used with Artist Link). Can be used if PSF files are required for back-annotation of the OP results.

- **`-ezwave`**

Displays Eldo simulation results in the EZwave waveform viewer while the simulation is running (marching waveforms). This option automatically enables option `-gwl jwdb` if required. Eldo will plot the waveforms as defined in the netlist. When the simulation is complete, Eldo exits but EZwave remains displayed.

- **`-isaving [val]`**

Specifies the “spill” threshold. Loads Eldo in incremental saving mode. Incremental saving allows the user to save the waveform data to a .wdb file when the Joint Wave DataBase (JWDB) reaches the threshold specified (in bytes). The default value is 100 MB.

- **`-jwdb_threshold [val]`**

Loads Eldo in incremental saving mode. Incremental saving allows the user to save the waveform data to a .wdb file when the Joint Wave DataBase (JWDB) reaches the threshold specified (in Megabytes). The default value is 100 MB.

---

**Note**

---

The two command line flags above are identical except for the default unit.

---

- `-wdb_config <swd_filename>`

The `.swd` (saved window database) file specified in this argument may be the result from a previous EZwave session. The page composition of the netlist will be replaced by the one specified in the `.swd` file. Waveforms resulting from expressions or measurements will be “replayed” using new waves.

- `-wdl_timeout time`

Instructs Eldo to wait the specified `time` seconds for a response from the JWDB server. By default, the timeout is 20 seconds.

- `-noconf`

When this option is used at Eldo invocation, and then the `^C` command is issued, then Eldo exits immediately without asking whether a save file must be created. Also, it will not ask whether or not to re-enter FAS debugger, if necessary. This option is used in the Analog Artist integration.

- `-spfout`

Causes the netlist to be printed without any line numbers in the output file.

- `-ri`

Forces AC output to be (**REAL**, **IMAG**) by default.

- `-dbp`

Forces AC output to be (**DB**, **PHASE**) by default.

---

**Note**

The command `VDB(x)` will always dump `VDB(x)`. The two flags above (`-ri` and `-dbp`) apply to `.PROBE AC V(X)` for instance.

---

- `-m53`

Specifies the ruling surrounding the M factor should be as it was for versions of Eldo before and including v5.3, see also **M53** option on [page 11-28](#).

- `-E`

Allows extended use of pre-processor commands to define macros and replace them inside the netlist. Only the main netlist is sent to the C pre-processor.

- `-EE`

Allows extended use of pre-processor commands to define macros and replace them inside the netlist. The main netlist and all include files are sent to the C pre-processor.

- `-define macro`

Define a macro at invoke time. Allows the user to define variables used by the pre-processor.



Further information on -E/-EE/-define options can be found on [page 3-5](#).

- **-tuning** [ FAST | STANDARD | ACCURATE | VHIGH | BACKANNOTATE ]  
Selects the default mode of operation of Eldo as regards to precision and speed, see also **TUNING** option on [page 11-22](#).
- **-h**  
Run the Eldo simulation on the machine (CPU) specified by `hostname`.
- **-libinc**  
Specifies that the full contents of every library are read by Eldo in one pass upon completion of reading the input file. This was the default mechanism in the v5.8 version of Eldo. All the libraries (**.LIB**) are included without filtering the objects (model, card, or subcircuit) that are not used in the specific netlist. See also **LIBINC** option on [page 11-16](#).
- **-couext**  
Merge extractions. Forces Eldo to dump **.EXTRACT/.MEAS** information into the `.cou` file rather than in the `.ext` file. This only works if there are no **.ALTER** commands present.
- **-nocouext**  
Forces Eldo to dump **.EXTRACT/.MEAS** information into separate files `.cou` and `.ext` files rather than merged.
- **-extract filename**  
Activates extraction mode (perform measurements using waves stored in `.cou` or `.wdb` files).  
Eldo only performs the **.EXTRACT** commands, without performing the simulation(s).  
Eldo will decorrelate the *simulation* from the *extraction*. `filename` is the name of an Xelga `.cou` or EZwave `.wdb` output file created from a previous simulation using the **.PLOT/.PROBE** command. The extractions in the netlist will be solved using the corresponding waves contained in this file. This functionality can also be specified with the netlist command **.EXTMOD**, see [page 10-94](#).

---

**Note**

Multiple levels of extraction are not supported with the **-extract** flag.

---

- **-inter**  
Used to invoke Eldo in the interactive mode. Commands are sent interactively instead of sending the commands in the netlist.
- **-eil file**  
Used to invoke Eldo in the interactive mode. Commands will be read from the specified file at invocation.



Further information on the interactive mode can be found in the [Eldo Interactive Mode](#) appendix.

- `-part_ui`  
Flag for Eldo Mach to open the netlist partitioning browser GUI. Please refer to [page 2-5](#) of the *Eldo Mach User's Manual* for further information.
- `-savetime time`  
Creates a `.sav` file which saves the context of a simulation every `time` hour of CPU time used in a transient analysis. If the system crashes during a run, the simulation may be restarted from the last saving time. See the `.SAVE` command on [page 10-274](#) for further information.
- `-m subckt_name`  
Macro simulation. Specifies that all the devices at the top of the circuit (except voltage and current sources) are ignored, and the subcircuit specified `subckt_name` becomes the top-level of the circuit. This can be useful for subcircuit testing.
- `-64b`  
Runs the simulation in 64-bit mode (available for Sun, HP and Linux platforms that have had 64-bit OS installed). This enables simulation of circuits which would require more than 2GB of memory, and which would therefore not work on 32-bit machines.

---

**Note**

Only a subset of Eldo/Eldo RF is ported. The following features are not supported:  
ESim/SimPilot interface, HDL-A, Output file format other than `.cou/.wdb` file, Eldo Mach, HVMOS model, IBIS models, and SSIM model.

- `-restart ["[save_file] [file=wfile]" ]`  
Restart a simulation run with information previously saved using the `.SAVE` command. The quotes are mandatory if more than one argument is specified. See the `.RESTART` command on [page 10-271](#) for further information.
- `-probeop2`  
Eldo will generate a `circuit.opx`, where `x` is the index of the run, in which DCOP information will be dumped. This file is an ASCII file which is read by the DA-IC environment. The content of this file is similar to that created when `.OPTION PROBEOP` is used, but there is some additional information which is dumped: temperature, node information, and now current out of X leaf-instances (leaf-cells are instances which do not call any other X instances).
- `-compat`  
For simulator compatibility the `-compat` runtime flag can be specified, or alternatively the `.OPTION COMPAT` command. When Eldo is invoked with this argument, the main

effect is that it accepts some extensive syntax used in other languages. Full details are provided where applicable throughout this manual. For more information and a full list of the effects this flag/command option has, see “[Compatibility Options](#)” on page 12-1.

Flag `-compat` is equivalent to setting both `-compmod` and `-compnet` flags shown below:

- `-compmod`

Triggers only the automatic conversion of models (can alternatively be set with `.OPTION COMPMOD`).

- `-compnet`

Causes the netlist to be interpreted as compatible format, but the models themselves are treated as Eldo Spice models (can alternatively be set with `.OPTION COMPNET`).



For more information and a full list of the effects the `COMPxxx` flags/command options have, see “[Compatibility Options](#)” on page 12-1.

---

- `-noinit`

Disable loading the `eldo.ini` file.

- `-nojwdb`

Disable the creation of `.wdb` file (can alternatively be set with `.OPTION NOJWDB`).

- `-jwdb_nocomplex`

Disable the generation of complex waves (can alternatively be set with `.OPTION NOWAVECOMPLEX`).

- `-jwdb_norffolder`

Disable storing RF results in separate folders within the JWDB database. Instead results will be added to the AC and TRAN folders.

- `-jwdb_servermode`

Used to specify the JWDB server launched by Eldo can be re-used by other simulations.

Useful data is stored in a file pointed to by the environment variable

`AMS_WDBSERVER_INFO`, its default is `$HOME/.ezwave/jwdbserver.info`.

- `-out`

Specifies the file name for all simulation output files. If not specified, files will use the default of `cir_filename`. A path can be defined as long as the directory exists. By default, the files are created in the same directory as the `.cir` file. It can be used when `.cou` output is specified.

- `-outname`

Specifies the file name for JWDB simulation output files (`.swd` and `.wdb`). If not specified, files will use the default of `cir_filename`. A path can be defined as long as the directory

exists. By default, the files are created in the same directory as the *.cir* file. It can be used when *.wdb* output is specified.

- **-alter**

Allows running one specific **.ALTER** section by specifying it's name or index, without first running the main netlist. Syntax:

```
-alter alter_name | alter_index
```

The *.chi* file and std output indicate that Eldo is only running a specific **.ALTER** due to this command line flag. Specifying an alter\_index of 0 means the normal run without any alters; everything in the netlist is simulated except the modified re-run alter statements.

- **-silent**

Disables all displays to the standard output.

- **-verbose**

Forces Eldo to display more detailed reporting with some information messages in the standard output terminal. See also **VERBOSE** option on [page 11-43](#). Eldo will print hints about syntax which is valid but ignored if the appropriate analysis is not found in the netlist. For example:

```
Warning 10001: No optimization command has been found in the netlist.
```

As a consequence:

- 1) .option OPSELD0\_NETLIST is ignored
- 2) .PARAMOPT are interpreted like .PARAM using the initial value, or ignored if no initial value has been specified.
- 3) GOAL=MINIMIZE is ignored on measurement FOO
- 4) GOAL is ignored on measurement FOO2
- 5) UBOUND is ignored on measurement FOO3
- 6) GOAL=MAXIMIZE is ignored on measurement FOO4

```
Warning 10002: COMMAND .MC has not been found in the netlist.
```

As a consequence:

- 1) .option DISPLAY\_CARLO is ignored

- **-mthread**

Activates multi-threading for a single DC or TRAN simulation. Eldo will share computer resources on a multi-processor machine. Eldo will make use of all the possible CPUs on the machine. See also **MTHREAD** option on [page 11-8](#).

Eldo will then use these processors as much as possible. It will share the work between the different CPUs in order to speed-up simulation. Note that the CPUs should not already be in use, otherwise simulation will be slower.

Statistics, generated at the end of simulation, show how many CPUs have been used for the current simulation. This number will also be printed out at the beginning of the TRAN simulation.

Multi-threading is not available on Linux 64-bit machines.

- **-cntthread**  
Checks how many CPUs Eldo can access on a multi-processor machine. Eldo will share computer resources for multi-threading a single DC or TRAN simulation. See also **CNTTHREAD** option on [page 11-8](#).
- **-usethread val**  
Activates multi-threading for a single DC or TRAN simulation. Eldo will share computer resources on a multi-processor machine. Eldo will make use of the number of CPUs as specified with this flag. The number specified can exceed the number of CPUs available, but this is not recommended. See also **USETHREAD** option on [page 11-13](#).

## Command Line Help

A simple online help can be accessed from the command line with:

```
eldo -help [commands|devices|sources|manual]
commands   Simulator commands
devices    Device models
sources    Sources and Macromodels
manual     Full Eldo User's Manual
```

Each of the first three help options will open a link document in Acrobat Reader, which will then allow you to select the command, device model, source or macromodel you require information on.

Entering `eldo -help` without any option will display the list of available topics.

This flag can be specified without the `cir_filename` which is usually mandatory.

## Running the STMicroelectronics Version of Eldo

The STMicroelectronics version of Eldo can be run from the command line using the following additional command line flag:

```
eldo -stver ... cir_filename
```



Further information can be found in the “[STMicroelectronics Models](#)” on page H-1.

---

## Running the Motorola Version of Eldo

The Motorola (SSIM model) version of Eldo can be run from the command line using the following additional command line flag:

```
eldo -ssim ... cir_filename
```



Further information can be found in the “[Motorola SSIM Model \(Eldo Level 54 or SSIM\)](#)” on page 4-151.

## Eldo Initialization File

An Eldo system initialization file named *eldo.ini* can be created. This file will be interpreted and loaded at the very beginning of each simulation. “Loading *eldo.ini*” is displayed whenever a valid *eldo.ini* file is found. Specifying the *-noinit* argument will disable the loading of the *eldo.ini* file.

This initialization file can be used to specify some configuration options always included in the *.cir* file.

The search order is:

- path specified by environment variable \$ELDO\_INI\_FILE\_PATH
- current directory
- \$HOME directory

This file is separated into blocks which are specified using a tag (no mandatory order):

- environment variables definition (after tag [env])
- arguments for command line (after tag [argu])
- netlist commands (after tag [eldo])

A typical *eldo.ini* file may look like:

```
# This line is a comment
[env]
# There must be no blanks between variable name, equal sign
# and variable value
OPTION_DIR=.
MODEL_DIR=../models
LIB_DIR=../libs

[argu]
-outpath $OPTION_DIR/results
-gwl jwdb
-compat

[eldo]
.option noascii notrc
.include $OPTION_DIR/options.inc
.option post probe
```



# Chapter 3

## Eldo Control Language

---

### Introduction

The Eldo *.cir* control file contains all the information necessary to run an Eldo simulation. The Eldo Control Language is used to specify all circuit descriptions and simulation commands in the *.cir* file. The Eldo Control Language is a superset of the standard Berkeley SPICE 2G6 language. Standard Berkeley SPICE control files will thus be accepted by Eldo. However, Eldo provides additional features not available in SPICE. This chapter provides an overview of the *.cir* file structure and general aspects of the language syntax. The following chapters then provide full definitions of the Eldo control language syntax for device, source and macromodel instantiations, and all of the command set.

### Overview of the *.cir* File Structure

#### Example

```
my_example_circuit --- As in SPICE, the first line is treated as comment.

* Model definitions
.model m1 nmos level=3 vto=1v ! Comments following Eldo statements
* start with "!"
+ uo=550 vmax=2.0e5 cgdo=0.4p ! Note the continuation line starting
* with "+"
* Blank lines can be used to make files
* more readable
* Subcircuit definitions
.subckt inv 1 2 3
m2 2 1 0 0 m1 w=10u l=4u ad=100p pd=40u as=100p
m1 2 1 3 3 p1 w=15u l=4u ad=100p pd=40u as=100p
c1 2 0 0.5p
.ends inv
* Subcircuit calls
x1 1 2 6 inv
cload 4 0 1p
* Electrical source definitions
vdd 6 0 5v
vin 1 0 pulse (0 5 10e-9 5e-9 5e-9 30e-9 50e-9)
* Simulation options & commands
.tran 0.5n 100n uic
.ic v(1)=0
.plot tran v(1) v(2) v(3) v(4)
.print tran v(1) v(2) v(3) v(4)
.option eps=0.5e-3 tnom=50 list node
.end
```

## General Aspects of the Language Syntax

The following sections in this chapter provide a summary of the Eldo language syntax.

The later chapters in this manual contain the complete descriptions of all the devices, sources, macromodels and commands listed in the following pages. This chapter is designed as a quick reference to the syntax for these and is of use to the more experienced user.

## Documentation Conventions



See the [Documentation Conventions](#) for a detailed description of the meanings of the different fonts, brackets, etc. used throughout this manual.

---

### First Line

The first line is format free and reserved for the circuit title. This line is mandatory and serves as the heading on graphical results output.

### Continuation Lines

The length of one input line is limited to 2000 characters. A line may be continued by using the + character at the beginning of the new line. In arithmetic expressions, this leading + sign will be ignored arithmetically and treated as a continuation. Two + signs may be placed together to both continue the line, *and* perform the addition operation.

### Comment Lines

A new comment line must begin with the \* character. If the comment follows an Eldo statement on the same line, it must begin with the ! character. The ! character must be preceded by a white space. Otherwise, the ! character will be considered as a valid character that can, for example, be used in node names.

```
* <comment line>
<Eldo statement> ! <inline comment>
```

A set of comment lines can also be grouped together into a block as shown below:

```
#com
A block of
comment
#endcom
```

Note that if this method is used  
the \* character need not be present

## Component Names

Component names start with the component reserved characters and continue with an arbitrary sequence of alphanumeric characters, including %, \$, #, \_. Component names cannot be broken at the end of a line.

## Parameter Names

Parameter names may contain an arbitrary sequence of alphanumeric characters, including %, \$, #, \_. Parameter names cannot be broken at the end of a line. Parameter names should not contain boolean operators. Such a name can be quite ambiguous.

## String Parameters

Eldo accepts quoted character strings as parameter values. These string values may be used for model names and filenames. To use a string as a parameter, enclose the string with double quotes, for example:

```
.param TT1="ResMod"
```

To keep the case of the string enclose the string with single quotes first and then enclose with double quotes, for example:

```
.param TT2=' 'PwlModFile.src' '
```

The value of the string is retrieved simply by specifying the dollar sign (\$) and parentheses (). Examples:

```
.param MOD="Pmos1"
m1 d g s b $(MOD) w=lu l=lu

.param STIMFILE=' 'Stim.txt'
v1 1 0 pwl file=$(STIMFILE) R
```

## Reserved Keywords

The following keywords are special in that they may appear in expressions. However, they may not be specified in a **.PARAM** command if an RF analysis is specified in the netlist.

**Table 3-1. Reserved Keywords not available in .PARAM**

AMNOISE	BFACTOR	BOPT	FREQ	GA_mag	GA_dB
GAC	GAM_mag	GAM_dB	GAMMA_OPT	GAMMA_OPT_MAG	GASM_mag
GASM_dB	GAUM_mag	GAUM_dB	GOPT	GP_mag	GP_dB
GPC	INOISE	KFACTOR	LSC	MUFACTOR	NFMIN_mag

**Table 3-1. Reserved Keywords not available in .PARAM**

NFMIN_dB	ONOISE	PHI_OPT	PHNOISE	POWER	RNEQ
SCALE	SNF_mag	SNF_dB	SSC	TEMP	TGP_mag
TGP_dB	TIME	TNOM <sup>a</sup>	XAXIS		

a. TNOM may be specified as a parameter in a .PARAM command when .OPTION DEFPTNOM is set. The temperature value used by the Eldo model evaluator is always that which is set with .OPTION TNOM=val.

If an RF analysis is specified in the netlist, and if any .PARAM is named with one of these keywords, it will be rejected. For example, the following statement will generate an error:

```
.PARAM SCALE=VAL
```

## Node Names

Node names may contain an arbitrary sequence of alphanumeric characters including !, \$, #, \_\_, [ , ], <, >. Node names cannot be broken at the end of a line. If the first character of a node name is numeric then it is forbidden for an alphabetic character to follow in the same name: all characters must then be numeric. Numeric characters can, however, follow an alphabetic character in the same node name.

- |       |                    |
|-------|--------------------|
| 1TOTO | Illegal node name. |
| 123   | Legal node name.   |
| TOTO1 | Legal node name.   |

## Node Names Used Inside Subcircuits

If we wish to access nodes from a higher level of hierarchy than that in which they are defined, it may be done as shown in the following example:

```
X27.X113.N3
```

Legal node name.

The node N3 is located within a subcircuit X113 which, in turn, is located inside another subcircuit X27.



For more information about the usage of nodes inside subcircuits, please refer to “.SUBCKT” on page 10-306.

---

## Values

Values are always handled as real numbers. They may be specified in exponential notation or with scale factors.

## Model Names

Model names *CANNOT* start with a numeric. This causes compilation of the netlist to be broken, giving an error message.

## Scale Factors

For scaling, you can choose between the exponential notation, or one of the following:

$A=1.0\times10^{-18}$   $F=1.0\times10^{-15}$   $P=1.0\times10^{-12}$   $N=1.0\times10^{-9}$   $U=1.0\times10^{-6}$   $M=1.0\times10^{-3}$   
 $K=1.0\times10^3$   $MEG=1.0\times10^6$   $G=1.0\times10^9$   $T=1.0\times10^{12}$  dB (for decibels)

## Notes

- Letters which are not scale factors are ignored if they immediately follow a number. Hence 10, 10V and 10Hz all represent the same number, 10. However, 10A will be interpreted as 1.0e-17, because of the Ato scaling factor.
- Letters immediately following a scale factor are ignored. Thus M, MA, MSEC, and MMHOS all represent the same scale factor, M.
- The scale factor M represents  $1\times10^{-3}$  or ‘milli’ units.  $1\times10^6$  or ‘mega’ units are specified using the MEG scale factor. This is commonly confused in SPICE syntax.
- Scale factors are not cumulative. KK is not MEG, but K, since the second letter is ignored.
- M.K.S. units are used throughout the netlist.

## Directives

### Directives interpreted by the Eldo parser (default)

By default, (without the **-E** or **-EE** flag) Eldo understands the following simple pre-processor commands: #if, #define <name>, #ifdef <name>, #ifndef <name> #else, #endif.

These can be used to select a part of the netlist, for example:

```
#define NO_RESISTOR
...
#ifndef NO_RESISTOR
R1 A B 1k
#endif
...
```

A *define* can also be specified at invoke time with:

```
eldo <arguments> -define <name>
```

For example, **-define foo** on the command line is equivalent to `#define foo` in the netlist. For both methods, the netlist statement `#ifdef foo` will be true.

The `#if` statement can be used for making complex conditional statements using logical operators: OR `||`, AND `&&`; and comparison operators: not equal to `!=`, equal to `==`. The `#if` directive can also be used in conjunction with the defined function i.e. if a `#define <name>` is active, `defined(<name>)` returns 1 whatever value is assigned to `<name>`.

## Example

```
V1 1 0 1
#define U 0
#if (defined(U) || defined(A))
R1 1 0 1
#else
R1 1 0 2
#endif
.end
```

Because U is defined in the above example, `defined(U)` will return 1. The `#if` statement is true and R1 will be set to 1. When the `#if` statement is substituted with: `#if (defined(U) && defined(A))`, the `#if` statement will no longer be true because A is not defined. R1 would therefore be set to 2.

## Directives interpreted using the C pre-processor (-E/-EE arguments)

For an extended use of pre-processor commands to define macros and replace them inside the netlist (see the example below), the **-E** or **-EE** flag needs to be specified. These are not the Eldo default because the parsing of large circuits may be significantly slower.

The **-E** flag forces Eldo to provide only the main netlist to the C pre-processor, whereas **-EE** ensures that all include files will be pre-processed before parsing.

The `#include` directive can only be used when the **-E/-EE** flags are specified.

Note: The C pre-processor analyses files independently. Therefore `#define` statements are only known to the file they are defined in.

The **-define** flag can also be specified on the command line, and it will have the same effect as without the **-E/-EE** flags. To use `#define` in Eldo in the same way you define macros in the C language, the syntax is as follows:

```
#define macro(args) expression
```

When the macro is encountered in the netlist, it is replaced by the expression. Arguments of the macro will be replaced by the literals in the macro call. For example:

```
#define sat_margin(device) abs(vds(device)-vdss(device))
```

```
.extract sat_margin(XM0.M1)
```

will be replaced by:

```
.extract abs(vds(XM0.M1)-vdss(XM0.M1))
```

#### **Note**

 Because -E/-EE uses the C pre-processor, you must ensure that there is a carriage return proceeding the directive in the file to avoid problems. Uppercase function names are not accepted. Using comments defined with #com and #endcom are not compatible with -E/-EE.

## Arithmetic Functions

A set of arithmetic functions may be used in Eldo for the calculation of device parameters, model parameters, new waves etc. These are listed below:

**Table 3-2. Arithmetic Functions & Operators**

Function	Returns
<b>SQRT(VAL)</b>	Square root of VAL
<b>LOG(VAL)</b>	Neperian logarithm of VAL
<b>LOG10(VAL)</b>	Decimal logarithm of VAL
<b>DB(VAL)</b>	Value in dBs of VAL ( $20 \times \log_{10}(VAL)$ )
<b>EXP(VAL)</b>	Exponent of VAL
<b>COS(VAL)</b>	Cosine of VAL, where VAL is defined in radians
<b>SIN(VAL)</b>	Sine of VAL, where VAL is defined in radians
<b>TAN(VAL)</b>	Tangent of VAL, where VAL is defined in radians
<b>ACOS(VAL)</b>	Arc cosine of VAL
<b>ASIN(VAL)</b>	Arc sine of VAL
<b>ATAN(VAL)</b>	Arc tangent of VAL
<b>COSH(VAL)</b>	Hyperbolic cosine of VAL
<b>SINH(VAL)</b>	Hyperbolic sine of VAL
<b>TANH(VAL)</b>	Hyperbolic tangent of VAL
<b>SGN(VAL)</b>	Returns +1 if VAL>0, 0 if VAL=0, -1 if VAL<0
<b>SIGN(VAL)</b>	Returns +1 if VAL is positive or null, -1 otherwise
<b>SIGN(VAL1, VAL2)</b>	Returns the <b>ABS(VAL1)*SGN(VAL2)</b>

**Table 3-2. Arithmetic Functions & Operators**

Function	Returns
<b>PWR(VAL1, VAL2)</b>	Returns the absolute value of VAL1, raised to the power of VAL2, with the sign of VAL1
<b>POW(VAL1, VAL2)</b>	Returns the value of VAL1 to the power of the integer part of VAL2
<b>ABS(VAL)</b>	Absolute value of VAL
<b>INT(VAL)</b>	Integer value of VAL (equivalent to <b>TRUNC</b> )
<b>TRUNC(VAL)</b>	Truncated value of VAL (Integer part of real value)
<b>ROUND(VAL)</b>	Rounded, to the nearest integer, value of VAL
<b>CEIL(VAL)</b>	Ceiling rounding function, returns the smallest integer value not less than VAL. This is known as rounding up. For example ceil(1.25) returns 2.0 and ceil(-1.25) return -1.0.
<b>FLOOR(VAL)</b>	Floor rounding function, returns the largest integer value not greater than VAL. This is known as rounding down. For example floor(1.25) returns 1.0 and floor(-1.25) return -2.0.
<b>MIN(VAL1, ..., VALn)</b> <b>DMIN(VAL1, ..., VALn)</b>	Returns the minimum of VAL1 to VALn. There is no limit to the number of values that can be specified
<b>MAX(VAL1, ..., VALn)</b> <b>DMAX(VAL1, ..., VALn)</b>	Returns the maximum of VAL1 to VALn. There is no limit to the number of values that can be specified
<b>DERIV(VAL)</b>	Returns the derivative of VAL
<b>REAL()</b>	Returns the real part of a complex number
<b>IMAG()</b>	Returns the imaginary part of a complex number
<b>MAGNITUDE()</b>	Returns the magnitude of a complex number
<b>CONJ()</b>	Returns the conjugate of a complex number
<b>COMPLEX(a, b)</b>	Returns a complex number using ‘a’ as the real part and ‘b’ as the imaginary part
<b>STOSMITH(val)</b> <b>YTOSMITH(val)</b> <b>ZTOSMITH(val)</b>	Returns a normalized value of a complex quantity. These functions can be used to convert complex functions (S/Y/Z parameters generated by a <b>.STEP</b> analysis) so they can be correctly plotted in a Smith chart; see <a href="#">page 10-248</a> for an example
<b>DDT(VAL)</b>	Returns the derivative of VAL
<b>IDT(VAL)</b>	Returns the integral of VAL
<b>LIMIT(a, b, c)</b>	Returns b if a < b, returns c if a > c, returns a otherwise
<b>BITOF(a, b)</b>	Returns “1” if bit b of the integer value of parameter a is a “1”. Returns “0” if bit b of the integer value of parameter a is a “0”

**Table 3-2. Arithmetic Functions & Operators**

Function	Returns
<b>PWL</b> (xvalue, interp, x1, y1, ... xn, yn)	Returns the equivalent output value at the input value <code>xvalue</code> , <code>interp=0 1</code> specifies whether the <code>y</code> value is interpolated linearly (1) or not (0). <code>xn</code> and <code>yn</code> are used to calculate the equivalent output value

## Notes

- Where `VAL` is written above, it normally means a numeric value, but in certain cases, the functions may also be applied to waves.
- The **MAX** and **MIN** functions are automatically converted into **DMAX** and **DMIN** functions when necessary, e.g.  
`R1 1 2 {min(2,1,5)}` is equivalent to:  
`R1 1 2 {dmin(2,1,5)}` which is also equivalent to:  
`R1 1 2 1`  
There is no limit to the number of arguments that can be used in **MAX**, **MIN**, **DMAX** and **DMIN** arguments.
- If three arguments are specified for **MIN** (i.e. `MIN(a,b,c)`), this represents the **MIN** of waveform `a` in the time window `[b, c]`. Use **DMIN** to return the minimum of `a`, `b` and `c` as computed at each time step.
- DDT(VAL)** and **IDT(VAL)** can *only* be used on E & G elements, see the corresponding descriptions for the “[Voltage Controlled Voltage Source](#)” on page 5-39 and “[Voltage Controlled Current Source](#)” on page 5-50. Expressions associated with R/L/C devices do not support DDT/IDT operators.  
The **DDT** operator differs from the **DERIV** operator in the sense that **DDT** utilizes the integration scheme used by Eldo, while the **DERIV** operator exclusively uses the Backward-Euler algorithm.
- Nested **COMPLEX()** statements are forbidden, as well as using complex quantities for the real or imaginary part. Corresponding errors are:

```
ERROR 3040: Nested complex(,) functions is not allowed.
ERROR 3041: Complex quantities can not be used inside complex(,) function.
```

## -compat flag

When the `-compat` flag is active, the following arithmetic function/operator rules apply:

$\log(x) = \text{sign}(x) * \log(\text{abs}(x))$   
 $\log10(x) = \text{sign}(x) * \log10(\text{abs}(x))$   
 $\text{db}(x) = \text{sign}(x) * 20.0 * \log10(\text{abs}(x))$   
 $\text{sqrt}(x)$  is  $-\sqrt{\text{abs}(x)}$  if  $x$  is negative.  
 $x^{**n}$  is computed as  $x^{**n}$  if  $x$  is positive,  $-(\text{abs}(x)^{**n})$  if  $x$  is negative, and 0 if  $x$  is 0.

The power operator (^) has highest precedence (same as standard Eldo); prior to v6.3\_2 it had lower precedence in -compat mode than the multiplication and division operators.

---

**Note**

In Eldo standard mode: sqrt(x) returns an error if x is negative.

$x^{**n}$  is computed as  $\exp(n*\log(x))$  if x is strictly positive, 0 otherwise.

---

## Operators

### Operator Precedence

The order of precedence and associativity of operators in Eldo affect the evaluation of expressions. For example, in the expression  $a=2+b*3$ , which happens first, the addition or the multiplication? Expressions with higher-precedence operators are evaluated first.

Table 3-3 summarizes the precedence and associativity (the order in which the operands are evaluated) of Eldo operators, listing them in order of precedence from highest to lowest. Where several operators appear together, they have equal precedence and are evaluated according to their associativity.

**Table 3-3. Operator Precedence**

Operator	Description	Associativity
()	function call	left-to-right
! -	logical NOT, unary negation	right-to-left
** ^	power, power (synonym)	left-to-right
* /	multiply, divide	left-to-right
+ -	add, subtract	left-to-right
<< >>	bitwise left shift, bitwise right shift	left-to-right
< <= > >=	less than, less than or equal, greater than, greater than or equal	left-to-right
== !=	equal, not equal	left-to-right
&	bitwise AND	left-to-right
	bitwise OR	left-to-right
&&	logical AND	left-to-right
	logical OR	left-to-right

## Arithmetic Operators

The arithmetic operators available are `+`, `-`, `*`, `/` and `^` (or `**`) for power.

---

### Note



The power operator (`^`) has the highest precedence, e.g.

`1+4^2` gives the result: 17

`2*3^2` gives the result: 18

---

## Boolean Operators

The following boolean expressions/operators are available:

**Table 3-4. Boolean Operators**

Operator	Meaning
<code>!=</code>	not equal to
<code>==</code>	equal to
<code>&lt;</code>	less than
<code>&lt;=</code>	less than or equal to
<code>&gt;</code>	greater than
<code>&gt;=</code>	greater than or equal to
<code>  </code>	OR operator
<code>&amp;&amp;</code>	AND operator

## Bitwise Operators

The following Bitwise operators are available:

**Table 3-5. Bitwise Operators**

Operator	Meaning
<code>&amp;</code>	Bitwise AND operator
<code> </code>	Bitwise OR operator
<code>&lt;&lt;</code>	Bitwise shift left operator
<code>&gt;&gt;</code>	Bitwise shift right operator

## Expressions

Expressions can be used in a netlist with certain restrictions. Numerical expressions must be contained within braces {}, single quotes '' or parentheses () whereas string expressions should be contained in double quotes ". Mathematical grouping within expressions must be done using normal brackets (). Constants and parameters may be used in expressions, together with the built-in functions and operators described above. Expressions may be used in the following situations:

- Parameters in the calculation of MOS geometries and R, C and L values.
- Parameter values in the .MODEL command.
- Time point values in the signal descriptions PULSE, PWL, SFFM, SIN and EXP.
- Parameters values in the .SIGBUS command.
- Voltage and current source values.
- S and Z transform (FNS and FNZ) devices.
- .PARAM, .EXTRACT and .DEFWAVE commands.
- E and G sources described by functions or tables.
- R, C and L devices described by functions.

---

 Some parameters may appear in expressions but will cause an error if used in a .PARAM command. For more information on these see “[Reserved Keywords](#)” on page 3-3.

---

## Examples

```
r1 1 2 {3.0*p1-4k}
.model nn nmos vt0={p2-p2/2.0}
e1 1 2 value={15v*sqrt(v(3,2))}
.defwave pow=v(a)*i(b)
.param x1={2*sqrt(a)}
```

### -compat flag

In -compat mode, double quotes are considered as single quotes. (In standard Eldo mode, double quotes are used to specify a parameter string.)

## Conditional Evaluation of Expressions

Parameters or source values can be evaluated in expressions containing conditional statements.

## Syntax

```
VALIF(CONDITION,expression1,expression2)
EVAL(CONDITION?expression1:expression2)
```

If CONDITION is TRUE, then **VALIF** (or **EVAL**) returns expression1 else it returns expression2. The keyword **VALIF** (or **EVAL**) can be used in any expression.

## Example

```
.param p1 = 1.0
.param p2 = 2.0
.param p3 = valif(p1>p2,p1+1.5,p2+1.5)
```

Here, P3 will be assigned the value 3.5.

### Note



The **EVAL** syntax is closer to 'C' language, and may be more convenient for some users.

---

## Simulation Counters

After a simulation has been completed, Eldo writes simulation information, in tabular form, to the ASCII output (*.chi*) file. The following information is output:

## Node & Element Information

Information concerning circuit nodes and elements is written to the *.chi* file in the following format:

NUNODS	NCNODS	NUMNOD	NUMEL	DIODES	BJT	JFET	MOSFET
16	16	16	22	0	0	0	19

where the parameters have the following definitions:

NUNODS	Number of nodes before subcircuit expansion.
NCNODS	Number of nodes after subcircuit expansion.
NUMNOD	Total number of nodes including those created by parasitic resistances.
NUMEL	Total number of elements contained in the circuit.
DIODES	Number of diode elements contained in the circuit.
BJT	Number of BJT elements contained in the circuit.
JFET	Number of JFET elements contained in the circuit.
MOSFET	Number of MOSFET transistor elements contained in the circuit.

## Grounded Capacitors Information

Information concerning grounded capacitors is written to the *.chi* file in the following format:

```
NUMGC
1
```

where the parameters have the following definitions:

NUMGC      Number of grounded capacitors not taken into account by NUMEL.

## Matrix Information

When Eldo creates a matrix, the following information is written to the *.chi* file:

```
NSTOP      NTERM      PERSPA
13         122        7.219e+01
```

where the parameters have the following definitions:

NSTOP      Number of lines in the matrix.

NTERM      Number of terms in the matrix.

PERSPA      Sparsity coefficient in percent (%).

## Newton Block Information

When Eldo creates a number of Newton blocks, the following information is written to the *.chi* file:

```
NBLOCKS     NODEBLK     MAXSIZE     MINSIZE
```

where the parameters have the following definitions:

NBLOCKS      Total number of Newton blocks created.

NODEBLK      Number of nodes contained in each Newton block.

MAXSIZE      Size of the biggest Newton block.

MINSIZE      Size of the smallest Newton block.

## Convergence Information

Information concerning circuit nodes and elements is written to the *.chi* file in the following format:

```
NUMTTP      NUMRTP      LTERTP      INWCALL      ITERNW      MEMSIZE
80          15           2           243         2.000e+00    581896
```

where the parameters have the following definitions:

NUMTTP	Number of steps accepted by the simulator and sent to the binary output (.wdw/.cou) file.			
NUMRTP	Number of steps rejected due to the truncation error being too large.			
LTERTP	Number of time steps rejected due to LTE.			
INWCALL	Total number of iterations or Newton calls needed to solve the Newton blocks.			
ITERNW	Total number of Newton calls for .OP, .DC and .AC analyses and is the average number of Newton calls needed to achieve convergence for a .TRAN analysis.			
MEMSIZE	Memory size allocated to the circuit by Eldo.			
NDEVCALL	NKIRCH	NMAXCALL	ITERM	LATENCY
16038	0	9	1.00e+00	5.208e+00%

where the parameters have the following definitions:

NDEVCALL	Number of device calls.
NKIRCH	Number of calls or iterations needed to solve Kirchoff's Law (OSR only).
NMAXCALL	Maximum number of calls needed to solve a time or DC point.
ITERM	Average number of OSR loops.
LATENCY	Percentage of latency in the circuit.

## Temperature Handling

Eldo allows temperature handling using the commands .TEMP, TNOM, TMOD and T and allows formulation of temperature dependent functions using the variable TEMPER (or TEMP). These commands and functions are briefly described below:

The TNOM function from the .OPTION command is used to set the nominal simulation temperature, i.e. the temperature at which parameter calculations are made. Default is 27 °C.

---

### Note



TNOM may appear in expressions.

TNOM is a reserved keyword, however it may be specified as a parameter in a .PARAM command when .OPTION DEFPTNOM is set. The temperature value used by the Eldo model evaluator is always that which is set with .OPTION TNOM=val. See options DEFPTNOM on [page 11-25](#) and TNOM [page 11-30](#).

---

The **.TEMP** command is used to execute several successive simulations at various temperatures. See “**.TEMP**” on page 10-314.

The **TMOD** parameter (in certain models) is used to set the model temperature. The value of this parameter overrides the **.TEMP** command above. The **T** parameter (in certain devices) is used to set the temperature of an individual instance of a device or model. This parameter overrides the **TMOD** command above.



Please refer to the [Device Models](#) chapter.

---

To summarize, the order of priority of the above temperature related commands and parameters is **T**, then **TMOD** and then **.TEMP**, with decreasing priority, i.e. **T** has the highest priority.

**TEMPER** is a variable returned by the simulator which gives the value of the current simulation temperature and may be used in subsequent calculations. This variable will be the present simulation temperature resulting from either a **.TEMP** command, a **.DC TEMP** sweep or, if neither are specified, the value of **TNOM** given in the **.OPTION** command. The **TEMPER** variable may be used in the formulation of temperature dependent expressions. Any expressions containing the **TEMPER** variable will be automatically reevaluated in the case of a change in this temperature.

---

**Note**

---



The **TEMP** variable is synonymous with the **TEMPER** variable. Both refer to the temperature of the circuit.

---

## Example

The **TEMPER** variable may be used in conjunction with **VALUE={EXPR}** in resistors, capacitors and inductors to specify devices whose values vary with temperature.



Refer to these components in the [Device Models](#) chapter.

---

```
Cvariable 3 7 VALUE={C0*(1+0.002*(TEMPER^2))}
```

This specifies a capacitor **Cvariable** connected between nodes 3 and 7 and its value defined as the nominal capacitance **C0** multiplied by  $(1 + 0.002 \times \text{TEMPER}^2)$ . The **TEMPER** variable may also be used in expressions for model parameters.

# Devices

## Resistor

```
Rxx N1 N2 [MOD[EL]=MNAME] [VAL] [[TC1=]T1] [[TC2=]T2] [[TC3=]T3]
+ [AC=VAL|{EXPR}] [T[EMP]=VAL] [DTEMP=VAL] [M=VAL] [L=VAL] [W=VAL]
+ [KEEPRMIN] [NONOISE] [KF=VAL] [AF=VAL] [WEEXP=VAL] [LEEXP=VAL]
+ [FEXP=VAL] [FMIN=VAL] [FMAX=VAL] [NBF=VAL]
Rxx N1 N2 [MOD[EL]=MNAME] VALUE={EXPR} [RESTORE_CAUSALITY=val]
+ [[TC1=]T1] [[TC2=]T2] [[TC3=]T3] [AC=VAL] [T[EMP]=VAL] [DTEMP=VAL]
+ [M=VAL] [KEEPRMIN] [NONOISE] [KF=VAL] [AF=VAL]
+ [WEEXP=VAL] [LEEXP=VAL] [FEXP=VAL] [FMIN=VAL] [FMAX=VAL]
+ [NBF=VAL] [FIT=VAL] [CFMAX=VAL] [CDELF=VAL]
Rxx N1 N2 [[TC1=]T1] [[TC2=]T2] [[TC3=]T3] [AC=VAL] [T[EMP]=VAL]
+ [DTEMP=VAL] [M=VAL] [KF=VAL] [AF=VAL] [WEEXP=VAL] [LEEXP=VAL]
+ [FEXP=VAL] [FMIN=VAL] [FMAX=VAL] [NBF=VAL]
+ TABLE EXPR [KEEPRMIN] [NONOISE]
Rxx NP NN POLY VAL {COEF} [TC1=T1] [TC2=T2] [TC3=T3]
```

## Capacitor

```
Cxx NP NN [MOD[EL]=MNAME] [DCCUT] [VAL] [M=VAL] [L=VAL] [W=VAL]
+ [T[EMP]=VAL] [DTEMP=VAL] [TC1=T1] [TC2=T2] [TC3=T3] [IC=VAL]
Cxx NP NN POLY VAL {COEF} [TC1=T1] [TC2=T2] [TC3=T3] [M=VAL]
+ [CTYPE=VAL] [IC=VAL]
Cxx NP NN [VALUE=]{EXPR} [RESTORE_CAUSALITY=val]
+ [TC1=T1] [TC2=T2] [TC3=T3] [CTYPE=VAL]
```

## Inductor

```
Lxx NP NN [MOD[EL]=MNAME] [DCFEED] [VAL] [M=VAL1] [T[EMP]=VAL] [DTEMP=VAL]
+ [IC=VAL3] [TC1=T1] [TC2=T2] [TC3=T3] [R=VAL4]
Lxx NP NN POLY VAL {LN} [IC=VAL] [R=VAL] [TC1=T1] [TC2=T2] [TC3=T3]
Lxx NP NN [VALUE=]{EXPR} [RESTORE_CAUSALITY=val] [R=VAL|R VALUE=EXPR|R
+ TABLE {fval rval}) [TC1=T1] [TC2=T2] [TC3=T3]
```

## Coupled Inductor

Kxx Lyy Lzz KVAL

## RC Wire

```
Rxx N1 N2 MNAME [[R=]VAL] [TC1=VAL] [TC2=VAL] [C=VAL] [CRATIO=VAL]
+ [L=VAL] [W=VAL] [M=VAL] [T[EMP]=VAL] [DTEMP=VAL] [SCALE=VAL]
```

## Semiconductor Resistor

```
Pxx N1 N2 NS MNAME [R=VAL] [L=VAL] [CL=VAL] [W=VAL] [CW=VAL] [AREA=VAL]
```

## Transmission Line

```
Txx NAP NAN NBP NBN [Z0=VAL1] TD=VAL2
Txx NAP NAN NBP NBN [Z0=VAL1] F=VAL3 [NL=VAL4]
```

## Lossy Transmission Line

```
Yxx LDTL [PIN:] P1...PN [REFin] PN+1...P2N REfout  
+ [PARAM:] [LEVEL=val] [LENGTH=val] [M=val] [SAVEFIT=val]
```

## Lossy Transmission Line: W Model

```
Wxx N=nb_line  
+ P1...PN PGNDin PN+1...P2N PGNDout  
+ RLGCfile=file_name L=length [FP=val]  
+ [MULTIDEBYE=val] [SAVEFIT=val] [COMPAT=val] [FGD=val]
```

## Lossy Transmission Line: U Model

```
Uxx P1...PN PGNDin PN+1...P2N PGNDout UNAME L=length [SAVEFIT=val]
```

## MTEE: Microstrip T Junction

```
Yxx MTEE P1 P2 P3 P4 P5 P6 PARAM: [W1=val] [W2=val] [W3=val]  
+ [T=val] [Er=val] [H=val]
```

## MBEND: Microstrip Bend (Arbitrary Angle, Optimally Mitered)

```
Yxx MBEND P1 P2 P3 P4 PARAM: [W=val] [H=val] [Er=val] [T=val]  
+ [RHO=val] [TAND=val] [M=val] [ANGLE=val]
```

## MBEND2: 90-degree Microstrip Bend (Mitered)

```
Yxx MBEND2 P1 P2 P3 P4 PARAM: [H=val] [W=val] [Er=val]
```

## MBEND3: 90-degree Microstrip Bend (Optimally Mitered)

```
Yxx MBEND3 P1 P2 P3 P4 PARAM: [W=val] [H=val] [Er=val] [T=val]  
+ [RHO=val] [TAND=val]
```

## MCORN: 90-degree Microstrip Bend (Unmitered)

```
Yxx MCORN P1 P2 P3 P4 PARAM: [W=val] [H=val] [Er=val]
```

## MSTEP: Microstrip Step in Width

```
Yxx MSTEP P1 P2 P3 P4 PARAM: [W1=val] [W2=val] [ER=val]  
+ [H=val] [F=val] [ASYMMETRICAL=val] [T=val]
```

## VIA2: Cylindrical Via Hole in Microstrip

```
Yxx VIA2 P1 P2 PARAM: [H=val] [R=val] [COND=val] [T=val] [F=val]
```

## SBEND: Unmitered Stripline Bend

```
Yxx SBEND P1 P2 P3 P4 PARAM: [W=val] [B=val] [ER=val] [T=val]  
+ [ANGLE=val] [F=val]
```

## STEE: Stripline T Junction

```
Yxx STEE P1 P2 P3 P4 P5 PARAM: [W1=val] [W2=val] [W3=val]
+ [B=val] [ER=val] [T=val] [F=val]
```

## SSTEP: Stripline Step in Width

```
Yxx SSTEP P1 P2 P3 P4 PARAM: [W1=val] [W2=val] [B=val] [T=val]
+ [ER=val] [F=val]
```

## Junction Diode

```
Dxx NP NN [NM] MNAME [[AREA=]AREA_VAL] [PERI=PERIVAL] [PGATE=PGATE_VAL]
+ [T[EMP]=VAL] [DTEMP=VAL] [M=VAL] [OFF=0|1] [NOISE=0|1] [NONOISE]
Dxx NP NN [NM] MNAME [FMIN=VAL] [FMAX=VAL] [NBF=VAL]
```

## BJT—Bipolar Junction Transistor

```
Qxx NC NB NE [NS] [TH] MNAME [[AREA=]AREA_VAL] [AREAB=AREA_VAL]
+ [AREAC=AREA_VAL] [T[EMP]=VAL] [DTEMP=VAL] [M=VAL] [OFF=0|1]
+ [NOISE=0|1] [NONOISE]
Qxx NC NB NE [NS] MNAME [FMIN=VAL] [FMAX=VAL] [NBF=VAL]
```

## JFET—Junction Field Effect Transistor

```
Jxx ND NG NS MNAME [[AREA=]AREA_VAL] [L=VAL] [W=VAL]
+ [T[EMP]=VAL] [DTEMP=VAL] [OFF] [NONOISE]
Jxx ND NG NS MNAME [FMIN=VAL] [FMAX=VAL] [NBF=VAL]
```

## MESFET—Metal Semiconductor Field Effect Transistor

```
Jxx ND NG NS MNAME [AREA] [L=VAL] [W=VAL] [T[EMP]=VAL] [DTEMP=VAL]
+ [OFF] [NONOISE]
Jxx ND NG NS MNAME [FMIN=VAL] [FMAX=VAL] [NBF=VAL]
```

## MOSFET

```
Mxx ND NG NS [NB] [{NN}] [MOD[EL]=]MNAME [[L=]VAL] [[W=]VAL]
+ [AD=VAL] [AS=VAL] [PD=VAL] [PS=VAL] [GEO=VAL] [NRD=VAL]
+ [NRS=VAL] [M=VAL] [RDC=VAL] [RSC=VAL] [T[EMP]=VAL] [DTEMP=VAL] [NONOISE]
Mxx ND NG NS [NB] [{NN}] [MOD[EL]=]MNAME [W=VAL] [L=VAL]
+ [FMIN=VAL] [FMAX=VAL] [NBF=VAL]
```

## S-Domain Filter

```
FNSxx IN OUT [RIN=val] [ROUT=val] NN {NN}, DN {DN}
```

## Z-Domain Filter

```
FNZxx IN OUT FREQ=VAL [RIN=val] [ROUT=val] NN {NN}, DN {DN}
```

## Subcircuit Instance

```
XXX NN {NN} NAME [PAR=VAL] [PAR={EXPR}] [M=VAL] [TEMP=VAL]
+ [(SWITCH|ANALOG|OSR|DIGITAL)] [NONOISE|NOISE=0]
```

# Sources

## Independent Voltage Source

```
Vxx NP NN [[DC] DCVAL] [AC [ACMAG [ACPHASE]]]
+ [TIME_DEPENDENT_FUNCTION1] [TC1=val] [TC2=val]
+ [RPORT=val [NONOISE]] [RPORT_TC1=val] [RPORT_TC2=val]
+ [IPORT=val] [CPORt=val] [LPORT=val] [MODE=keyword] [NOISETEMP=val]
Vxx NP NN [[DC] DCVAL] [AC [ACMAG [ACPHASE]]]
+ [TIME_DEPENDENT_FUNCTION1] [TC1=val] [TC2=val]
+ ZPORT_FILE=string [IPORT=val] [CPORt=val] [LPORT=val] [MODE=keyword]
+ [NOISETEMP=val]
Vxx NP NN [[DC] DCVAL] [AC [ACMAG [ACPHASE]]]
+ [TC1=val] [TC2=val] [RPORT=val [NONOISE]]
+ [RPORT_TC1=val] [RPORT_TC2=val] [IPORT=val] [CPORt=val]
+ [LPORT=val] [MODE=keyword] [NOISETEMP=val] NOISE [THN=VAL] [FLN=VAL]
+ [ALPHA=VAL] [FC=VAL] [N=VAL] [FMIN=VAL] [FMAX=VAL]
+ [NBF=VAL]
Vxx NP NN [[DC] DCVAL] [AC [ACMAG [ACPHASE]]]
+ [TC1=val] [TC2=val] ZPORT_FILE=string [IPORT=val] [CPORt=val]
+ [LPORT=val] [MODE=keyword] [NOISETEMP=val] NOISE [THN=VAL] [FLN=VAL]
+ [ALPHA=VAL] [FC=VAL] [N=VAL] [FMIN=VAL] [FMAX=VAL]
+ [NBF=VAL]
Vxx NP NN [[DC] DCVAL] [AC [ACMAG [ACPHASE]]]
+ [TC1=val] [TC2=val] [RPORT=val [NONOISE]]
+ [RPORT_TC1=val] [RPORT_TC2=val] [IPORT=val] [CPORt=val]
+ [LPORT=val] [MODE=keyword] [NOISETEMP=val] NOISE TABLE
+ [[INTERP=]DEC|OCT|LIN|LOG] [DB|MA]
+ (f1 val1) (f2 val2) ...
Vxx NP NN [[DC] DCVAL] [AC [ACMAG [ACPHASE]]]
+ [TC1=val] [TC2=val] ZPORT_FILE=string [IPORT=val] [CPORt=val]
+ [LPORT=val] [MODE=keyword] [NOISETEMP=val] NOISE TABLE
+ [[INTERP=]DEC|OCT|LIN|LOG] [DB|MA]
+ (f1 val1) (f2 val2) ...
Vxx NP NN [[DC] DCVAL] [AC [ACMAG [ACPHASE]]]
+ [TC1=val] [TC2=val] [RPORT=val [NONOISE]]
+ [RPORT_TC1=val] [RPORT_TC2=val] [IPORT=val] [CPORt=val]
+ [LPORT=val] [MODE=keyword] [NOISETEMP=val] FOUR
+ fund1 [fund2 [fund3]] MA|RI|DB|PMA|PDB|PDBM
+ (int_val1 [,int_val2 [,int_val3]]) real_val1 real_val2
+ {(int_val1 [,int_val2 [,int_val3]]) real_val1 real_val2}
Vxx NP NN [[DC] DCVAL] [AC [ACMAG [ACPHASE]]]
+ [TC1=val] [TC2=val] ZPORT_FILE=string
+ [IPORT=val] [CPORt=val] [LPORT=val] [MODE=keyword] [NOISETEMP=val] FOUR
+ fund1 [fund2 [fund3]] MA|RI|DB|PMA|PDB|PDBM
+ (int_val1 [,int_val2 [,int_val3]]) real_val1 real_val2
+ {(int_val1 [,int_val2 [,int_val3]]) real_val1 real_val2}
```

---

1. Refer to the EXP, PATTERN, PULSE, PWL, SFFM and SIN source functions.

## Independent Current Source

```
Ixx NP NN [[DC] DCVAL] [AC [ACMAG [ACPHASE]]]
+ [TIME_DEPENDENT_FUNCTION1] [TC1=val] [TC2=val]
+ [REPORT=val [NONOISE]] [RPORT_TC1=val] [RPORT_TC2=val]
+ [IPORT=val] [CPORT=val] [LPORT=val] [NOISETEMP=val]
Ixx NP NN [[DC] DCVAL] [AC [ACMAG [ACPHASE]]]
+ [TIME_DEPENDENT_FUNCTION1] [TC1=val] [TC2=val]
+ ZPORT_FILE=string [IPORT=val] [CPORT=val] [LPORT=val] [MODE=keyword]
+ [NOISETEMP=val]
Ixx NP NN [[DC] DCVAL] [AC [ACMAG [ACPHASE]]]
+ [TC1=val] [TC2=val] [RPORT=val [NONOISE]]
+ [RPORT_TC1=val] [RPORT_TC2=val] [IPORT=val] [CPORT=val]
+ [LPORT=val] [NOISETEMP=val] NOISE [THN=VAL] [FLN=VAL]
+ [ALPHA=VAL] [FC=VAL] [N=VAL] [FMIN=VAL] [FMAX=VAL]
+ [NBF=VAL]
Ixx NP NN [[DC] DCVAL] [AC [ACMAG [ACPHASE]]]
+ [TC1=val] [TC2=val] [IPORT=val] [CPORT=val]
+ ZPORT_FILE=string [LPORT=val] [MODE=keyword] [NOISETEMP=val]
+ NOISE [THN=VAL] [FLN=VAL] [ALPHA=VAL] [FC=VAL] [N=VAL] [FMIN=VAL]
+ [FMAX=VAL] [NBF=VAL]
Ixx NP NN [[DC] DCVAL] [AC [ACMAG [ACPHASE]]]
+ [TC1=val] [TC2=val] [RPORT=val [NONOISE]]
+ [RPORT_TC1=val] [RPORT_TC2=val] [IPORT=val] [CPORT=val]
+ [LPORT=val] [NOISETEMP=val] NOISE TABLE
+ [[INTERP=]DEC|OCT|LIN|LOG] (f1 val1) (f2 val2) ...
Ixx NP NN [[DC] DCVAL] [AC [ACMAG [ACPHASE]]]
+ [TC1=val] [TC2=val] ZPORT_FILE=string [IPORT=val] [CPORT=val]
+ [LPORT=val] [MODE=keyword] [NOISETEMP=val] NOISE TABLE
+ [[INTERP=]DEC|OCT|LIN|LOG] (f1 val1) (f2 val2) ...
Ixx NP NN [[DC] DCVAL] [AC [ACMAG [ACPHASE]]]
+ [TC1=val] [TC2=val] [RPORT=val [NONOISE]]
+ [RPORT_TC1=val] [RPORT_TC2=val] [IPORT=val] [CPORT=val]
+ [LPORT=val] [NOISETEMP=val] FOUR fund1 [fund2 [fund3]]
+ MA|RI|DB|PMA|PDB|PDBM
+ (int_val1 [,int_val2 [,int_val3]]) real_val1 real_val2
+ {(int_val1 [,int_val2 [,int_val3]]) real_val1 real_val2}
Ixx NP NN [[DC] DCVAL] [AC [ACMAG [ACPHASE]]]
+ [TC1=val] [TC2=val] ZPORT_FILE=string [IPORT=val] [CPORT=val]
+ [LPORT=val] [MODE=keyword] [NOISETEMP=val] FOUR fund1 [fund2 [fund3]]
+ MA|RI|DB|PMA|PDB|PDBM
+ (int_val1 [,int_val2 [,int_val3]]) real_val1 real_val2
+ {(int_val1 [,int_val2 [,int_val3]]) real_val1 real_val2}
```

## Amplitude Modulation Function

**AM** (AMPLITUDE OFFSET FM FC TD)

## Exponential Function

**EXP** (V1 V2 [TD1 [TAU1 [TD2 [TAU2]]]])

---

1. Refer to the **EXP**, **PULSE**, **PWL**, **SFFM** and **SIN** source functions.

## Noise Function

```
NOISE THN FLN ALPHA [FC N] [FMIN] [FMAX] [NBF]
```

## Pattern Function

```
PATTERN VHI VLO TDELAY TRISE TFALL TSAMPLE BITS R
```

## Pulse Function

```
PULSE (V0 V1 [TD [TR [TF [PW [PER]]]]])
```

## Piece Wise Linear Function

```
PWL (TN VN {TN VN} [TD=val] [R=val] [SHIFT=val] [R] [SCALE=val]
+ [STRETCH=val])
PWL (FILE=<pwl_file> [TD=val] [R=val] [SHIFT=val] [R] [SCALE=val]
+ [STRETCH=val])
PWL (FILE=<pwl_file> [COL=val] [ISTEP=val] [ISTART=val] [ISTOP=val]
+ [TD=val] [R=val] [SHIFT=val] [R] [SCALE=val] [STRETCH=val])
```

## Single Frequency FM Function

```
SFFM (SO SA [FC [MDI [FS]])
```

## Sine Function

```
SIN (VO VA [FR [TD [THETA [PHASE]]]])
```

## Trapezoidal Pulse With Bit Pattern Function

```
PBIT V0 V1 TD TD01 TR01 TD10 TF10 BITTIME {PATTERN} [R]
```

## Exponential Pulse With Bit Pattern Function

```
EBIT V0 V1 TD TD01 TAU01 TD10 TAU10 BITTIME {PATTERN} [R]
```

## Voltage Controlled Voltage Source

```
EXX NP NN [VCVS] NCP NCN VAL [MIN=VAL] [MAX=VAL]
+ [TC1=VAL] [TC2=VAL] [SCALE=VAL] [ABS=VAL]
EXX NP NN [VCVS] NCP NCN VAL0 {VALn} [MIN=VAL] [MAX=VAL]
+ [TC1=VAL] [TC2=VAL] [SCALE=VAL] [ABS=VAL]
EXX NP NN [VCVS] POLY(ND) PCP PCN {PCP PCN} PN {PN}
+ [MIN=VAL] [MAX=VAL] [TC1=VAL] [TC2=VAL] [SCALE=VAL] [ABS=VAL]
EXX NP NN PWL(1) NCP NCN PWL_LIST [DELTA=val]
+ [TC1=VAL] [TC2=VAL] [SCALE=VAL] [ABS=VAL]
EXX NP NN NAND(ND) | AND(ND) | OR(ND) | NOR(ND) PCP PCN {PCP PCN}
+ PWL_LIST [TC1=VAL] [TC2=VAL] [SCALE=VAL] [ABS=VAL]
EXX NP NN DELAY NCP NCN [TD=val] [ABS=VAL]
EXX NP NN VALUE={EXPR} [MIN=VAL] [MAX=VAL]
+ [TC1=VAL] [TC2=VAL] [SCALE=VAL] [ABS=VAL]
EXX NP NN [MIN=VAL] [MAX=VAL] [TC1=VAL] [TC2=VAL] [SCALE=VAL]
+ TABLE EXPR=(XN YN) {(XN YN)} [ABS=VAL]
```

```

Exx NP NN INTEGRATION|DERIVATION NCP NCN VAL
+ [MIN=VAL] [MAX=VAL] [TC1=VAL] [TC2=VAL] [SCALE=VAL] [ABS=VAL]
Exx NP NN FNS NCP NCN n0 n1 ... nm, p0 p1 ... pn
Exx NP NN PZ NCP NCN a zr1 zil ... zrm zim, b pr1 pil ... prn pin
Exx NP NN FREQ NCP NCN f0 a0 ph0 f1 a1 ph1... fn an phn
+ [RESTORE_CAUSALITY=val]
Exx NP NN TRANS[FORMER] NCP NCN VAL [MIN=VAL] [MAX=VAL]
+ [TC1=VAL] [TC2=VAL] [SCALE=VAL] [ABS=VAL]

```

## Current Controlled Current Source

```

Fxx NP NN [CCCS] VN VAL [MIN=VAL] [MAX=VAL]
+ [TC1=VAL] [TC2=VAL] [SCALE=VAL] [ABS=VAL]
Fxx NP NN [CCCS] POLY(N) VN {VN} PN {PN}
+ [MIN=VAL] [MAX=VAL] [TC1=VAL] [TC2=VAL] [SCALE=VAL] [ABS=VAL]
Fxx NP NN PWL(1) VN PWL_LIST [DELTA=val]
+ [TC1=VAL] [TC2=VAL] [SCALE=VAL] [ABS=VAL]
Fxx NP NN NAND(ND)|AND(ND)|OR(ND)|NOR(ND) VN {VN}
+ PWL_LIST [TC1=VAL] [TC2=VAL] [SCALE=VAL] [ABS=VAL]
Fxx NP NN DELAY VN [TD=val] [ABS=VAL]
Fxx NP NN INTEGRATION|DERIVATION VN VAL [MIN=VAL] [MAX=VAL]
+ [TC1=VAL] [TC2=VAL] [SCALE=VAL] [ABS=VAL]

```

## Voltage Controlled Current Source

```

Gxx NP NN [VCR|VCCAP|VCCS]] NCP NCN VAL [MIN=VAL] [MAX=VAL]
+ [TC1=VAL] [TC2=VAL] [SCALE=VAL] [ABS=VAL]
Gxx NP NN [VCR|VCCAP|VCCS] POLY(ND) PCP PCN {PCP PCN} PN {PN}
+ [MIN=VAL] [MAX=VAL] [TC1=VAL] [TC2=VAL] [SCALE=VAL] [ABS=VAL]
Gxx NP NN VCR [PWL(1)|NPWL(1)|PPWL(1)] NCP NCN PWL_LIST
+ [DELTA=val] [MIN=VAL] [MAX=VAL] [TC1=VAL] [TC2=VAL]
+ [SCALE=VAL] [ABS=VAL]
Gxx NP NN NAND(ND)|AND(ND)|OR(ND)|NOR(ND) PCP PCN {PCP PCN}
+ PWL_LIST [TC1=VAL] [TC2=VAL] [SCALE=VAL] [ABS=VAL]
Gxx NP NN DELAY NCP NCN [TD=val] [ABS=VAL]
Gxx NP NN VALUE={EXPR} [MIN=VAL] [MAX=VAL]
+ [TC1=VAL] [TC2=VAL] [SCALE=VAL] [ABS=VAL]
Gxx NP NN [VCR|VCCAP|VCCS] [MIN=VAL] [MAX=VAL] [TC1=VAL] [TC2=VAL]
+ [SCALE=VAL] [ABS=VAL] TABLE EXPR=(XN YN) {(XN YN)}
Gxx NP NN INTEGRATION|DERIVATION NCP NCN VAL
+ [MIN=VAL] [MAX=VAL] [TC1=VAL] [TC2=VAL] [SCALE=VAL] [ABS=VAL]
Gxx NP NN FREQ NCP NCN f0 a0 ph0 f1 a1 ph1... fn an phn
+ [RESTORE_CAUSALITY=val]

```

## Current Controlled Voltage Source

```

Hxx NP NN [CCVS] VN VAL [MIN=VAL] [MAX=VAL]
+ [TC1=VAL] [TC2=VAL] [SCALE=VAL] [ABS=VAL]
Hxx NP NN [CCVS] POLY(N) VN {VN} PN {PN}
+ [MIN=VAL] [MAX=VAL] [TC1=VAL] [TC2=VAL] [SCALE=VAL] [ABS=VAL]
Hxx NP NN PWL(1) VN PWL_LIST [DELTA=val]
+ [TC1=VAL] [TC2=VAL] [SCALE=VAL] [ABS=VAL]
Hxx NP NN NAND(ND)|AND(ND)|OR(ND)|NOR(ND) VN {VN}
+ PWL_LIST [TC1=VAL] [TC2=VAL] [SCALE=VAL] [ABS=VAL]
Hxx NP NN DELAY VN [TD=val] [ABS=VAL]

```

```
Hxx NP NN INTEGRATION|DERIVATION VN VAL [MIN=VAL] [MAX=VAL]
+ [TC1=VAL] [TC2=VAL] [SCALE=VAL] [ABS=VAL]
```

## S, Y, Z Parameter Extraction

```
Vyy NP NN IPORT=VAL [RPORT=VAL] [CPORT=VAL] [LPORT=VAL] [MODE=KEYWORD]
Vyy NP NN IPORT=VAL ZPORT_FILE=string [CPORT=VAL] [LPORT=VAL]
+ [MODE=KEYWORD]
Iyy NP NN IPORT=VAL [RPORT=VAL] [CPORT=VAL] [LPORT=VAL] [MODE=KEYWORD]
Iyy NP NN IPORT=VAL ZPORT_FILE=string [CPORT=VAL] [LPORT=VAL]
+ [MODE=KEYWORD]
```

# Macromodels

## Analog

### Comparator

```
COMPxx INP INN OUT [MNAME] [VHI=VAL1] [VLO=VAL2]
+ [VOFF=VAL3] [VDEF=VAL4] [TCOM=VAL5] [TPD=VAL6]
COMPDxx INP INN OUTP OUTN [MNAME] [VHI=VAL1] [VLO=VAL2]
+ [VOFF=VAL3] [VDEF=VAL4] [TCOM=VAL5] [TPD=VAL6]
```

### Op-amp (Linear)

```
Yxx OPAMP0 [PIN:] INP INN OUT AGND
+ [PARAM: PAR=VAL {PAR=VAL}] [MODEL: MNAME]
Yxx OPAMP0D [PIN:] INP INN OUTN OUTP AGND
+ [PARAM: PAR=VAL {PAR=VAL}] [MODEL: MNAME]
```

### Op-amp (Linear 1-pole)

```
Yxx OPAMP1 [PIN:] INP INN OUT AGND
+ [PARAM: PAR=VAL {PAR=VAL}] [MODEL: MNAME]
Yxx OPAMP1D [PIN:] INP INN OUTN OUTP AGND
+ [PARAM: PAR=VAL {PAR=VAL}] [MODEL: MNAME]
```

### Op-amp (Linear 2-pole)

```
Yxx OPAMP2 [PIN:] INP INN OUT AGND
+ [PARAM: PAR=VAL {PAR=VAL}] [MODEL: MNAME]
Yxx OPAMP2D [PIN:] INP INN OUTN OUTP AGND
+ [PARAM: PAR=VAL {PAR=VAL}] [MODEL: MNAME]
```

## Delay

```
DELxx IN OUT VAL
```

## Saturating Resistor

```
Yxx SATR [PIN:] IN OUT [PARAM: PAR=VAL {PAR=VAL}] [MODEL: MNAME]
```

## Voltage Limiter

```
YXX SATV [PIN:] INP INN OUTP OUTN  
+ [PARAM: PAR=VAL {PAR=VAL}] [MODEL: MNAME]
```

## Voltage Controlled Switch

```
YXX VSWITCH [PIN:] NP NN CP CN [PARAM: PAR=VAL {PAR=VAL}] [MODEL: MNAME]
```

## Current Controlled Switch

```
YXX CSWITCH [PIN:] NP NN IC: VNAME  
+ [PARAM: PAR=VAL {PAR=VAL}] [MODEL: MNAME]
```

## Triangular to Sine Wave Converter

```
YXX TRI2SIN [PIN:] INP INN OUTP OUTN  
+ [PARAM: PAR=VAL {PAR=VAL}] [MODEL: MNAME]
```

## Staircase Waveform Generator

```
YXX STAIRGEN [PIN:] NP NN [PARAM: PAR=VAL {PAR=VAL}] [MODEL: MNAME]
```

## Sawtooth Waveform Generator

```
YXX SAWGEN [PIN:] NP NN [PARAM: PAR=VAL {PAR=VAL}] [MODEL: MNAME]
```

## Triangular Waveform Generator

```
YXX TRIGEN [PIN:] NP NN [PARAM: PAR=VAL {PAR=VAL}] [MODEL: MNAME]
```

## Amplitude Modulator

```
YXX AMM [PIN:] INP INN OUTP OUTN [PARAM: PAR=VAL {PAR=VAL}] [MODEL: MNAME]
```

## Pulse Amplitude Modulator

```
YXX PAM [PIN:] INP INN OUTP OUTN [PARAM: PAR=VAL {PAR=VAL}] [MODEL: MNAME]
```

## Sample & Hold

```
YXX SA_HO [PIN:] INP INN OUTP OUTN  
+ [PARAM: PAR=VAL {PAR=VAL}] [MODEL: MNAME]
```

## Track & Hold

```
YXX TR_HO [PIN:] INP INN OUTP OUTN CRT  
+ [PARAM: PAR=VAL {PAR=VAL}] [MODEL: MNAME]
```

## Pulse Width Modulator

```
YXX PWM [PIN:] CTRP CTRN OUTP OUTN  
+ [PARAM: PAR=VAL {PAR=VAL}] [MODEL: MNAME]
```

## Voltage Controlled Oscillator

```
YXX VCO [PIN:] INP INN OUTP OUTN  
+ [PARAM: PAR=VAL {PAR=VAL}] [MODEL: MNAME]
```

## Peak Detector

```
YXX PEAK_D [PIN:] INP INN OUTP OUTN CRT  
+ [PARAM: PAR=VAL {PAR=VAL}] [MODEL: MNAME]
```

## Level Detector

```
YXX LEV_D [PIN:] INP INN OUTP OUTN  
+ [PARAM: PAR=VAL {PAR=VAL}] [MODEL: MNAME]  
YXX LEV_D [PIN:] INP INN OUTP OUTN REF  
+ [PARAM: PAR=VAL {PAR=VAL}] [MODEL: MNAME]
```

## Logarithmic Amplifier

```
YXX LOGAMP [PIN:] IN OUT [PARAM: PAR=VAL {PAR=VAL}] [MODEL: MNAME]
```

## Anti-logarithmic Amplifier

```
YXX EXPAMP [PIN:] IN OUT [PARAM: PAR=VAL {PAR=VAL}] [MODEL: MNAME]
```

## Differentiator

```
YXX DIFF [PIN:] IN OUT [PARAM: PAR=VAL {PAR=VAL}] [MODEL: MNAME]
```

## Integrator

```
YXX INTEG [PIN:] IN OUT [PARAM: PAR=VAL {PAR=VAL}] [MODEL: MNAME]
```

## Adder, Subtractor, Multiplier & Divider

```
YXX ADD [PIN:] IN1 IN2 OUT [PARAM: PAR=VAL {PAR=VAL}] [MODEL: MNAME]  
YXX SUB [PIN:] IN1 IN2 OUT [PARAM: PAR=VAL {PAR=VAL}] [MODEL: MNAME]  
YXX MULT [PIN:] IN1 IN2 OUT [PARAM: PAR=VAL {PAR=VAL}] [MODEL: MNAME]  
YXX DIV [PIN:] IN1 IN2 OUT [PARAM: PAR=VAL {PAR=VAL}] [MODEL: MNAME]
```

## Digital

### Digital Model Definition

```
.MODEL MNAME LOGIC [VHI=VAL1] [VLO=VAL2] [VTH=VAL3]  
+ [VTI=VAL4] [VTLO=VAL5] [TPD=VAL6] [TPDUP=VAL7]  
+ [TPDOWN=VAL8] [CIN=VAL9] [DRV1=VAL10] [DRVH=VAL11]
```

## Delay

```
DELxx IN OUT VAL
```

## Inverter

```
INVXX IN OUT [REF1 REF2] [MNAME] [PAR=VAL]
```

## Exclusive-OR Gate

```
XORXX IN1 IN2 OUT [REF1 REF2] [MNAME] [PAR=VAL]
```

## 2-Input Digital Gates

```
<dgate><xxx> IN1 IN2 OUT [REF1 REF2] [MNAME] [PAR=VAL]
Nand NANDXX IN1 IN2 OUT [REF1 REF2] [MNAME] [PAR=VAL]
And ANDXX IN1 IN2 OUT [REF1 REF2] [MNAME] [PAR=VAL]
Nor NORXX IN1 IN2 OUT [REF1 REF2] [MNAME] [PAR=VAL]
Or ORXX IN1 IN2 OUT [REF1 REF2] [MNAME] [PAR=VAL]
Xor XORXX IN1 IN2 OUT [REF1 REF2] [MNAME] [PAR=VAL]
```

## 3-Input Digital Gates

```
<dgate><xxx> IN1 IN2 IN3 OUT [REF1 REF2] [MNAME] [PAR=VAL]
Nand NAND3XX IN1 IN2 IN3 OUT [REF1 REF2] [MNAME] [PAR=VAL]
And AND3XX IN1 IN2 IN3 OUT [REF1 REF2] [MNAME] [PAR=VAL]
Nor NOR3XX IN1 IN2 IN3 OUT [REF1 REF2] [MNAME] [PAR=VAL]
Or OR3XX IN1 IN2 IN3 OUT [REF1 REF2] [MNAME] [PAR=VAL]
```

## Multiple Input Digital Gates

```
<dgate><xx> IN1 IN2... {INX} OUT [REF1 REF2] [MNAME] [PAR=VAL]
Nand NAND#XX IN1 IN2...{INX} OUT [REF1 REF2] [MNAME] [PAR=VAL]
And AND#XX IN1 IN2...{INX} OUT [REF1 REF2] [MNAME] [PAR=VAL]
Nor NOR#XX IN1 IN2...{INX} OUT [REF1 REF2] [MNAME] [PAR=VAL]
Or OR#XX IN1 IN2...{INX} OUT [REF1 REF2] [MNAME] [PAR=VAL]
```

## Mixed

### Analog to Digital Converter

```
ADCXX CLK IN OUTSB{OUTSB} [EDGE=VAL1] [VTH=VAL2] [VHI=VAL3]
+ [VLO=VAL4] [VINF=VAL5] [VSUP=VAL6] [TCOM=VAL7] [TPD=VAL8]
```

### Digital to Analog Converter

```
DACXX CLK INSB{INSB} OUT [EDGE=VAL1] [VTH=VAL2]
+ [VTIN=VAL3] [VHI=VAL4] [VLO=VAL5] [TPD=VAL6] [SL=VAL7]
```

## Magnetic

### Transformer Winding

```
YXX WINDING [PIN:] E1 E2 M1 M2 [PARAM: PAR=VAL {PAR=VAL}] [MODEL: MNAME]
```

## Non-linear Magnetic Core 1

```
YXX NLCORE1 [PIN:] MP MN [PARAM: PAR=VAL {PAR=VAL}] [MODEL: MNAME]
```

## Non-linear Magnetic Core 2

```
YXX NLCORE2 [PIN:] MP MN [PARAM: PAR=VAL {PAR=VAL}] [MODEL: MNAME]
```

## Linear Magnetic Core

```
YXX LINCORE [PIN:] MP MN [PARAM: PAR=VAL {PAR=VAL}] [MODEL: MNAME]
```

## Magnetic Air Gap

```
YXX AIRGAP [PIN:] MP MN [PARAM: PAR=VAL {PAR=VAL}] [MODEL: MNAME]
```

## Transformer (Variable # of Windings)

```
YXX LVTRANS [PIN:] P1P P1N P2P P2N {PNP PNN}  
+ [PARAM: PAR=VAL {PAR=VAL}] [MODEL: MNAME]
```

## Ideal Transformer

```
YXX JTRAN N1 N2 N3 N4 [PARAM: A=VAL]
```

## Switched Capacitor

## Operational Amplifier

```
OPAXX INP INN OUTP OUTN [MNAME] [LEVEL=VAL1] [VOFF=VAL2]  
+ [SL=VAL3] [CIN=VAL4] [RS=VAL5] [VSAT=VAL6] [VSATN=VAL7] [GAIN=VAL8]  
+ [FC=VAL9] [FNDP=VAL10] [IMAX=VAL11] [CMRR=VAL12]
```

## Switch

```
SXX NC N1 N2 [MNAME] [RON [CREC]]
```

## Ideal Operational Amplifier

```
YXX SC_IDEAL [PIN:] INP INN OUT [PARAM: M=val]
```

## Inverting Switched Capacitor

```
YXX SC_I [PIN:] P1 P2 N2 N1 [PARAM: PAR=VAL {PAR=VAL}] [MODEL: MNAME]
```

## Non-inverting Switched Capacitor

```
YXX SC_N [PIN:] P1 P2 N1 N2 [PARAM: PAR=VAL {PAR=VAL}] [MODEL: MNAME]
```

## Parallel Switched Capacitor

```
YXX SC_P [PIN:] P1 P2 N [PARAM: PAR=VAL {PAR=VAL}] [MODEL: MNAME]
```

## Serial Switched Capacitor

```
YXX SC_S1 [PIN:] IN OUT [PARAM: PAR=VAL {PAR=VAL}] [MODEL: MNAME]  
YXX SC_S2 [PIN:] IN OUT [PARAM: PAR=VAL {PAR=VAL}] [MODEL: MNAME]
```

## Serial-parallel Switched Capacitor

```
YXX SC_SP1 [PIN:] IN OUT REF [PARAM: PAR=VAL {PAR=VAL}] [MODEL: MNAME]  
YXX SC_SP2 [PIN:] IN OUT REF [PARAM: PAR=VAL {PAR=VAL}] [MODEL: MNAME]
```

## Bi-linear Switched Capacitor

```
YXX SC_B [PIN:] IN OUT [PARAM: PAR=VAL {PAR=VAL}] [MODEL: MNAME]
```

## Unswitched Capacitor

```
YXX SC_U [PIN:] IN OUT [PARAM: PAR=VAL {PAR=VAL}] [MODEL: MNAME]
```

# Commands

## Analog-to-Digital Converter

```
.A2D SIM=simulator eldo_node_name  
+ [digital_node_name] [MOD=model_name] [parameters_list]
```

## AC Analysis

```
.AC TYPE nb fstart fstop [SWEEP DATA=dataname] [UIC] [MONTE=val]  
.AC TYPE nb fstart fstop [SWEEP parameter_name TYPE nb start stop]  
+ [UIC] [MONTE=val]  
.AC TYPE nb fstart fstop [SWEEP parameter_name  
+ start stop incr] [UIC] [MONTE=val]  
.AC DATA=dataname [SWEEP DATA=dataname] [UIC] [MONTE=val]  
.AC DATA=dataname [SWEEP parameter_name TYPE  
+ nb start stop] [UIC] [MONTE=val]  
.AC DATA=dataname [SWEEP parameter_name start stop incr]  
+ [UIC] [MONTE=val]  
.AC LIST {list_of_frequency_points} [SWEEP DATA=dataname]  
+ [UIC] [MONTE=val]  
.AC LIST {list_of_frequency_points}  
+ [SWEEP parameter_name TYPE nb start stop] [UIC] [MONTE=val]  
.AC LIST {list_of_frequency_points}  
+ [SWEEP parameter_name start stop incr] [UIC] [MONTE=val]  
.AC ADAPTIVE tolerance_value fstart fstop
```

## Insert a Model or Subcircuit File

```
.ADDLIB N DIR_NAME
```

## Age Analysis

```
.AGE [TAGE=value] [NBRUN[S]=value]
+ [LIN[={YES|ON|1}|{NO|OFF|0}]] [LOG[={YES|ON|1}|{NO|OFF|0}]]
+ [TSTART=value] [TSTOP=value]
+ [MODE=AGEsim | AGEload | AGESave] [AGELIB=file_name]
+ [AGEALL[={YES|ON|1}|{NO|OFF|0}]]
+ [ASCII[={YES|ON|1}|{NO|OFF|0}]]
+ [COMPUTE_LAST[={YES|ON|1}|{NO|OFF|0}]]
+ [PLOT={FRESH_FINAL|ALL}]
+ [TUNIT=year|month|day|hour|min]
+ [AGEDSIM={YES|ON|1}|{NO|OFF|0}]
+ [LOGMODE_MINEXP=<value>]
```

## Reliability Model Parameter Declaration

```
.AGEMODEL MODEL=model_name [parameter=value]
```

## Generalized Re-run Facility

```
.ALTER [LABEL]
[ELEMENT]
[SUBCKT]
[COMMAND]
[COMMENT]
.ALTER | .END
```

## Configure Spice Descriptions

```
.BIND inst=inst_name | from_part to_part [mapping=assoc_file_name]
.BIND inst=inst_name from_part to_part [mapping=assoc_file_name]
```

## Call Tcl Function

```
.CALL_TCL [TRAN|AC|DC|...]
+ WHERE=START|START_OF_RUN|END_OF_RUN|END
+ [PLOT=[YES|NO|0|1]] [LABEL=alias_name] tcl_function_call
```

## Check Bus Values

```
.CHECKBUS BNAME [VTH[1]=VAL1] [VTH2=VAL2]
+ [BASE=DEC|OCT|BIN|HEX] [LOCK=1] TN VAL {TN VAL} [REPORTX=0|1]
.CHECKBUS BNAME [VTH[1]=VAL1 [VTH2=VAL2]] [TSAMPLE=VAL] [TDELAY=VAL]
+ [BASE=DEC|OCT|BIN|HEX] [LOCK=1] PATTERN BITS {BITS} [REPORTX=0|1]
```

## Check Safe Operating Area Limits

```
.CHECKSOA [TRAN] [TSTART=val1 [TSTOP=val2]] [AUTOSTOP]
+ [NOMERGE] [NOLIB] [FILE=file_name] [NOXWINDOW]
+ [SUBCKT={list_of_subckt_instances}] [RUNTMSG]
```

## Piece Wise Linear Source

```
.CHRENT NODE TN VN {TN VN} [P|F]
.CHRENT NODE (TN VN {TN VN}) FACTN {(TN VN {TN VN}) FACTN} [P|F]
```

## Input from a Prior Simulation

```
.CHRSIM IN OUT FILE [TSTART=V1] [TSTEP=V2] [BP=0|1|2]
+ [ZOOMTIME] [FORMAT=WDB|JWDB] [RUN=val]
```

## Change Comment Character

```
.COMCHAR char
```

## Connect Two Nodes

```
.CONNECT N1 N2
```

## Current Used by a Circuit

```
.CONSO VN {VN}
```

## Correlation between Parameters

```
.CORREL [PARAM=]param_list cc=VAL
.CORREL DEV[ICE]=device_list PARAM=param_list cc=VAL
```

## Digital-to-Analog Converter

```
.D2A [SIM=simulator] eldo_node_name
+ [digital_node_name] [MOD=model_name] [parameters_list]
```

## Parameter Sweep

```
.DATA dataname parameter_list
+ val_list1
+ val_list2
+ ...
[.ENDDATA]
.DATA dataname MER[GE] | LAM[INATED]
+ file=filename1 param=column ... param=column
+ file=filename2 param=column ...
+ ...
+ [out=outfile]
[.ENDDATA]
```

## DC Analysis

```
.DC
.DC CNAM [L|W] [TYPE nb] START STOP INCR [SWEEP DATA=dataname] [MONTE=val]
.DC CNAM [L|W] [TYPE nb] START STOP INCR
+ [SWEEP parameter_name TYPE nb start stop] [MONTE=val]
.DC CNAM [L|W] [TYPE nb] START STOP INCR
+ [SWEEP parameter_name start stop incr] [MONTE=val]
```

```
.DC SNAM [TYPE nb] START STOP INCR [SNAM2 START2 STOP2 INCR2]
+ [SWEEP DATA=dataname] [MONTE=val]
.DC SNAM [TYPE nb] START STOP INCR [SNAM2 START2 STOP2 INCR2]
+ [SWEEP parameter_name TYPE nb start stop] [MONTE=val]
.DC SNAM [TYPE nb] START STOP INCR [SNAM2 START2 STOP2 INCR2]
+ [SWEEP parameter_name start stop] incr [MONTE=val]
.DC TEMP START STOP INCR [SWEEP DATA=dataname] [MONTE=val]
.DC TEMP START STOP INCR [SWEEP parameter_name TYPE nb start stop]
+ [MONTE=val]
.DC TEMP START STOP INCR [SWEEP parameter_name start stop incr]
+ [MONTE=val]
.DC PARAM PARAM_NAME START STOP INCR [SWEEP DATA=dataname] [MONTE=val]
.DC PARAM PARAM_NAME START STOP INCR [SWEEP parameter_name
+ TYPE nb start stop] [MONTE=val]
.DC PARAM PARAM_NAME START STOP INCR [SWEEP parameter_name
+ start stop incr] [MONTE=val]
.DC DATA=dataname [SWEEP DATA=dataname] [MONTE=val]
.DC DATA=dataname [SWEEP parameter_name TYPE nb start stop] [MONTE=val]
.DC DATA=dataname [SWEEP parameter_name start stop incr] [MONTE=val]
```

## DC Mismatch Analysis

```
.DCMISMATCH V(net_name[, net_name2])
+ [SORT_REL=value] [SORT_ABS=value] [SORT_NBMAX=value] [NSIGMA=value]
.DCMISMATCH I(vsrc_name)
+ [SORT_REL=value] [SORT_ABS=value] [SORT_NBMAX=value] [NSIGMA=value]
```

## Set Default Conditions

```
.DE[FAULT] TYPE VALUE
.DE[FAULT] TYPE {KEYWORD [VALUE]}
```

## Macro Definition

```
.DEFMAC MAC_NAME(ARG{, ARG})=EXPRESSION
```

## Model Name Mapping

```
.DEFMOD alias_model_name actual_model_name
```

## Plotting an Analog Signal as a Digital Bus

```
.DEFPLOTDIG [VTH[1]=VAL1 [VTH2=VAL2]]
```

## Waveform Definition

```
.DEFWAVE [SWEEP] [ANALYSIS] WAVE_NAME=WAVE_EXPR
```

## Remove library name

```
.DEL LIB LIB_NAME
```

## Ignore Instances or Subckt Definitions

```
.DISCARD INST | SUBCKT=(NAME, {NAME})
```

## Disable Flat Netlist Mode

```
.DISFLAT
```

## User Defined Distributions (Monte Carlo)

```
.DISTRIB DIST_NAME (DEV1 PROB1) [{(DEVn PROBn)}]
```

## DSP (Digital Signal Processing) Computation

```
.DSP LABEL=label_name MODEL=model_name waveform_name
```

## PSD (Power Spectral Density) Computation

```
.DSPMOD DSP=CORRELO|PERIODO LABEL=label_name
+ [TSTART=val] [TSTOP=val] [FS=val] [NBPT=val]
+ [PADDING=val] [WINDOW=name] [ALPHA=val] [BETA=val]
+ [NORMALIZED=val] [INTERPOLATE=val] [DISPLAY_INPUT=val]
+ [FNORMAL=val] [FMIN=val] [FMAX=val]
+ [NAUTO=val] [NCORR=val] [NPSD=val] [NSECT=val]
```

## Histogram Computation

```
.DSPMOD DSP=HISTOGRAM LABEL=label_name NBINTERVAL=val
+ [XSTART=val XSTOP=val SAMPLE=YES|NO FS=val]
```

## Load DSPF File

```
.DSPF_INCLUDE [FILE=]DSPF_FILENAME [INST={list_of_subckt_inst}]
+ [LEVEL=C|RC|RCC] [DEV=DSPF|SCH[EMATIC]] [RMINVAL=val] [CMINVAL=val]
+ [CCMINVAL=val] [ADDXNET] [ADDX]
.DSPF_INCLUDE [FILE=]DSPF_FILENAME [LEVEL=C|RC|RCC]
+ [DEV=DSPF|SCH[EMATIC]] DEDICATEDX=subckt_name [RMINVAL=val]
+ [CMINVAL=val] [CCMINVAL=val] [ADDXNET] [ADDX]
```

## End Eldo Netlist

```
.END
```

## End Eldo Library Variant Description

```
.ENDL
```

## End Eldo Subcircuit Description

```
.ENDS
```

## Replace Node Name for Display

```
.EQUIV new_name=netlist_name
```

## Extract Mode

```
.EXTMOD [FILE=filename]
```

## Extract Waveform Characteristics

```
.EXTRACT [EXTRACT_INFO] [LABEL=NAME] [FILE=FNAME] [VECT]
+ [CATVECT] $MACRO|FUNCTION [OPTIMIZER_INFO] [MC_INFO]
```

## S, Y, Z Parameter Output File Specification

```
.FFILE S|Y|Z|G|H|T|A [SINGLELINE] FILENAME [HZ|KHZ|MHZ|GHZ] [RI|MA|DB]
```

## Initial Transient Analysis Conditions

```
.FORCE [NODE] {node_name value}
+ [IND|OBJ] {object_name value}
```

## FFT Select Waveform

```
.FOUR LABEL = label_name waveform_name
```

## User Defined Function

```
.FUNC P(a,b,...) EXPR
```

## Global Node Allocation

```
.GLOBAL NN {NN}
```

## Initial DC Analysis Conditions

```
.GUESS V(NN)=VAL [SUBCKT=subckt_name] {V(NN)=VAL [SUBCKT=subckt_name]}
```

## Changing the Hierarchy Separator

```
.HIER . | / | <char>
```

## Initial Transient Analysis Conditions

```
.IC V(NN)=VAL [SUBCKT=subckt_name] {V(NN)=VAL [SUBCKT=subckt_name]}
```

## Ignore DSPF on Specified Node

```
.IGNORE_DSPF_ON_NODE NODE
```

## Include a File in an Input Netlist

```
.INC [LUDE] FNAME
```

## Initial Digital Circuit Conditions

```
.INIT NODE [DC=VAL] TI TS VALI {TI TS VALI}
```

## Insert Circuit Information from a Library File

```
.LIB [KEY=KNAME] FNAME [LIBTYPE]  
.LIB LIBTYPE
```

## Use Previously Simulated Results

```
.LOAD [FILE=]filename
```

## Insert a Feedback Loop

```
.LOOP INPUT OUTPUT [R|C|I|V VALUE] [DISCONNECT=DEV_NAME] [KEEPINPUT]
```

## Share Distributions

```
.LOTGROUP group_name[/distrib_type]=val[%]
```

## Loop Stability Analysis

```
.LSTB SOURCE_NAME
```

## Map Eldo Node to DSPF Node

```
.MAP_DSPF_NODE_NAME LOGICAL=ELDONAME DSPF=NEWNAME
```

## Monte Carlo Analysis

```
.MC RUNNO [OUTER] [OV] [SEED=integer_value] [NONOM] [ALL]  
+ [VARY=LOT|DEV] [IRUN=val] [NBBINS=val] [ORDMCS] [MCLIMIT]  
+ [PRINT_EXTRACT=NOMINAL|ALL|run_number]
```

## LOT & DEV Variation Specification on Model Parameters (Monte Carlo)

```
.MCMOD MNAME [(list_of_instances)] PAR LOT|DEV=VAL {PAR LOT|DEV=VAL}  
.MCMOD MNAME PAR LOTGROUP=my_lot_group
```

## Measure Waveform Characteristics

```
.MEAS TRAN|AC|DC [VECT] [CATVECT] label_name TRIG trig_spec TARG targ_spec  
.MEAS TRAN|AC|DC [VECT] [CATVECT] label_name WHEN when_spec AT val  
.MEAS TRAN|AC|DC [VECT] [CATVECT] label_name FIND wave WHEN when_spec  
.MEAS TRAN|AC|DC [VECT] [CATVECT] label_name FIND wave AT val  
.MEAS TRAN|AC|DC [VECT] [CATVECT] label_name FIND W('wave')  
+ WHEN when_spec [FROM=val] [TO=val]
```

```
.MEAS TRAN|AC|DC [VECT] [CATVECT] label_name DERIVATIVE wave
+ WHEN when_spec
.MEAS TRAN|AC|DC [VECT] [CATVECT] label_name DERIVATIVE wave AT val
.MEAS TRAN|AC|DC [VECT] [CATVECT] label_name meas_k wave
+ [FROM=val] [TO=val]
.MEAS TRAN|AC|DC [VECT] [CATVECT] label_name PARAM='expression'
```

## Aspire/SimPilot Command

```
.MODDUP device_name [... device_name]
.MODDUP element_name
```

## Device Model Description

```
.MODEL MNAME TYPE [PAR=VAL]
.MODEL LIB FILENAME MODNAME [LIBTYPE]
```

## Digital Model Definition

```
.MODLOGIC MNAME [VHI=VAL1] [VLO=VAL2] [VTH=VAL3] [VTHI=VAL4]
+ [VTLO=VAL5] [TPD=VAL6] [TPDUP=VAL7] [TPDOWN=VAL8]
+ [CIN=VAL9] [DRVl=VAL10] [DRVH=VAL11]
```

## Monitor Simulation Steps

```
.MONITOR ANALYSIS [=] [modulo]
```

## Multi-Processor Simulation

```
.MPRUN [ALL|HOST={host[(nbjobs)]}|FILE=filename] [NBLICENSES=val]
+ [MAX_NBJOBS=val] [CLEAN=YES|NO] [QUEUE=YES|NO] [SETENV=YES|NO]
+ [VIEW_COMMAND=YES|NO] [CHECK_DELAY=val] [INIT_FILE=filename]
+ [DEFAULT_INIT=YES|NO] [CD_WORKDIR=YES|NO] [LOGFILE=YES|NO]
+ [SHELL_SYNTAX=(source_cmd, setenv_cmd, setenv_sep, init_anacad_ext)]
+ [USE_LOCAL_HOST=YES|NO] [FLAT=YES|NO]
+ [FILE_PREFIX=(name1,name2,...,nameX)]
.MPRUN DISPATCHER= [LSF |
+ (dispatcher_name, install_check_cmd, submission_cmd)
+ [DISPATCHER_OPTIONS=(options)]
+ [NBLICENSES=val] [MAX_NBJOBS=val] [CLEAN=YES|NO]
+ [QUEUE] [SETENV] [VIEW_COMMAND] [CHECK_DELAY=val]
+ [INIT_FILE=filename [DEFAULT_INIT=YES|NO]]
+ [USE_LOCAL_HOST=YES|NO] [FLAT=YES|NO]
+ [FILE_PREFIX=(name1,name2,...,nameX)]
.MPRUN
+ DISPATCHER_TEMPLATE=command_line
+ [USE_LOCAL_HOST=YES|NO] [FLAT=YES|NO]
+ [FILE_PREFIX=(name1,name2,...,nameX)]
```

## Network Analysis

- 2-port network
  - .NET output input RIN=val ROUT=val
- 1-port network

```
.NET input RIN=val
```

## Control Page Layout

```
.NEWPAGE
```

## Suppress Comment Lines from Output File

```
.NOCOM
```

## DC Analysis Conditions

```
.NODESET V(NN)=VAL [SUBCKT=subckt_name] {V(NN)=VAL [SUBCKT=subckt_name]}
```

## Noise Analysis

```
.NOISE OUTV INSRC NUMS (X<subckt_instance_name>)
```

## Transient Noise Analysis

```
.NOISETRAN FMIN=VAL FMAX=VAL NBRUN=VAL [NBF=VAL] [AMP=VAL]  
+ [SEED=VAL] [NOMOD=VAL] [NONOM] [TSTART=VAL] [TSTOP=VAL]  
+ [MRUN] [ALL] [NBBINS=VAL] [FMIN_FLICKER=VAL]
```

## Suppress Netlist from an Output File

```
.NOTRC
```

## Partition Netlist into Newton Blocks

```
.NWBLOCK [RELTOL=value] [VNTOL=value] list_of_nodes
```

## DC Operating Point Calculation

```
.OP [[KEYWORD] T1 {[KEYWORD] TN}]  
.OP TIME=VAL|END [STEP=VAL] [TEMP=VAL]  
.OP DC=VAL [DC2=VAL] [STEP=VAL] [TEMP=VAL]
```

## FFT Post-processor Options

```
.OPTFOUR [TSTART=VAL|EXPR] [TSTOP=VAL|EXPR] [NBPT=VAL] [FS=VAL]  
+ [INTERPOLATE=0|1|2|3] [NOROUNDING[=1]] [WINDOW=name] [ALPHA=VAL]  
+ [BETA=VAL] [FMIN=VAL] [FMAX=VAL] [FNORMAL=freq] [PADDING=1|2|3]  
+ [NORMALIZED=0|1] [DISPLAY_INPUT=0|1]
```

## Optimization

```
.OPTIMIZE [qualifier=value {, qualifier=value }]  
+ [PARAM=list_of_parameters | *] [RESULTS=list_of_targets | *]
```

## Simulator Configuration

```
.OPT[ION] OPTION[=VAL] {OPTION[=VAL]}
```

## AC Noise Analysis

```
.OPTNOISE [ALL ON|OFF] [<CLASS> ON|OFF]
+ [R ON|OFF|<max>] [OUTSOURCE ON|OFF] [NSWEIGHT <FILENAME>]
+ [SORT D|V|TD|TV [SN <n>|SV <value>]] [NBW <FMIN> <FMAX>]
```

## Accuracy by Time Window

```
.OPTPWL PARAM=((TIME1,VALUE1) (TIME2,VALUE2)...)
+ PARAM=((TIME1,VALUE1)...)
```

## Accuracy by Time Window

```
.OPTWIND PARAM=(TIME1,TIME2,VALUE1)... (TIME1N,TIME2N,VALUEN)
```

## Global Declarations

```
.PARAM PAR=VAL {PAR=VAL}
.PARAM PAR=EXPR {PAR=EXPR}
.PARAM PAR="NAME"
.PARAM PAR(a,b)=EXPR
.PARAM PAR=VAL | PAR=EXPR
+ LOT|DEV[/GAUSS|/UNIFORM|/USERDIST]=VAL|(dtype,-3sig,+3sig
+ [,bi,-dz,+dz [,off,sv [,scale]])
.PARAM PAR=VAL LOTGROUP=my_lot_group
.PARAM PAR=MC_DISTRIBUTION
.PARAM PAR=VAL DEVX=VAL
```

## Plotting of Simulation Results

```
.PLOT [ANALYSIS] OVN [(LOW, HIGH)] [(VERSUS)]
+ {OVN [(LOW, HIGH)]} [UNIT=NAME] [(SCATTERED)] [STEP=value]
.PLOT AC|FSST S(i, j) [(SMITH[, zref])] [(POLAR)]
.PLOT FOUR FOURxx(label_name) [(SPECTRAL)]
.PLOT DSP DSPxx(label_name)
.PLOT EXTRACT [MEAS | SWEEP]
.PLOT [CONTOUR] MEAS(meas_name_x) MEAS(meas_name_y) [(SCATTERED)]
+ [(SMITH[, zref])] [(POLAR)]
.PLOT [ANALYSIS] TWO_PORT_PARAM [(SMITH[, zref])] [(POLAR)]
```

## Plotting of Bus Signals

```
.PLOTBUS BNAME [VTH[1]=VAL1 [VTH2=VAL2]]
.PLOTBUS BNAME[MSB:LSB] | BNAME<MSB:LSB> | BNAMEMSB:LSB
```

## Printing of Results

```
.PRINT [ANALYSIS] [syn_name=]OVN
```

## Printing of Bus Signals

```
.PRINTBUS BNAME [VTH1=VAL1 [VTH2=VAL2]]  
.PRINTBUS BNAME[MSB:LSB] | BNAME<MSB:LSB> | BNAMEMSB:LSB
```

## Print Tabular Output File

```
.PRINTFILE [ANALYSIS] OVN FILE=filename [STEP=value]
```

## Output Shortform

```
.PROBE [ANALYSIS] [ALL|I|IX|ISUB|PORT|PRINT|SG|SPARAM|S|Q|V|VN|VTOP|  
+ VX|VXN|W|WTOP] [MASK[=]mask_name] [PRINT] [STEP=val]  
.PROBE [ANALYSIS] [MASK[=]mask_name] [alias_name=] OVN [PRINT] [STEP=val]
```

## Netlist Protection

```
.PROTECT
```

## Pole-Zero Analysis

```
.PZ OV
```

## Automatic Ramping

```
.RAMP DC VAL [SIMPLIFY]  
.RAMP TRAN T1 T2
```

## Restart Simulation

```
.RESTART FNAME [FILE=WNAME]  
.RESTART ["fileBasename"] [NEWEST|LONGEST|TIME=VALUE] [FILE=WNAME]
```

## Save Simulation Run

```
.SAVE [[FILE=]FNAME] DC|END|TIME=VAL1 [REPEAT] [ALT|SEQ]  
+ [TEMP=VAL2] [STEP=VAL3] [TYPE=NODESET|IC] [LEVEL=ALL|TOP] [CARLO=index]
```

## Automatic Scaling of Active Devices

```
.SC[ALE] ELTYPE KEYWORD VALUE [KEYWORD VALUE ...]  
+ [ELEMENTS ALL | EXCEPT] [ELNAME1 ELNAME2 ...] [(ELNAME1 ELNAME2)]  
.SC[ALE] ELTYPE KEYWORD VALUE [KEYWORD VALUE ...]  
+ MODELS MODNAME1 [MODNAME2 ...]  
.SC[ALE] MDTYPE KEYWORD VALUE [KEYWORD VALUE ....]  
+ [MODELS ALL | EXCEPT] [MODNAME1 MODNAME2] [(MODNAME1 MODNAME2...)]  
.SC[ALE] P FACTOR=VALUE [SUBCKT=SUBNAME]  
+ [PARAMS ALL | EXCEPT] [PARAM1 PARAM2 ...] [(PARAM11 PARAM22)]
```

## Sensitivity Analysis

```
.SENS OVN {OVN}
```

## Sensitivity Analysis

```
.SENSPARAM sub[ckt]=subckt_name param=parameter_list
+ [var[iation]=value] [inst[ance]=instance_list]
+ [sort=inc[reasing] | dec[reasing] | alpha[betical]]
+ [sort_nbmax=value] [sort_abs=value | sort_rel=value]
```

## Create Bus

```
.SETBUS BNAME PN {PN}
```

## Set Reliability Model Key (Password)

```
.SETKEY [MODEL=model_name] KEY=key_value
```

## Set Safe Operating Area

```
.SETSOA [LABEL=<STRING>] E {EXPRESSION=(MIN,MAX[,XAXIS])}
.SETSOA [LABEL=<STRING>] E
+ IF(EXPR) THEN({PARAM=(MIN,MAX[,XAXIS]))})
+ ELSE({PARAM=(MIN,MAX[,XAXIS]))}) ENDIF
.SETSOA [LABEL=<STRING>] D DNAME [SUBCKT=subckt_list|INST=inst_list]
+ {PARAM=(MIN,MAX[,XAXIS])}
.SETSOA [LABEL=<STRING>] D DNAME [SUBCKT=subckt_list|INST=inst_list]
+ IF(EXPR) THEN({PARAM=(MIN, MAX[, XAXIS]))})
+ ELSE({PARAM=(MIN, MAX[, XAXIS]))}) ENDIF
.SETSOA [LABEL=<STRING>] M MNAME [SUBCKT=subckt_list|INST=inst_list]
+ {PARAM=(MIN,MAX[,XAXIS])}
.SETSOA [LABEL=<STRING>] M MNAME [SUBCKT=subckt_list|INST=inst_list]
+ IF(EXPR) THEN({PARAM=(MIN, MAX[, XAXIS]))})
+ ELSE({PARAM=(MIN, MAX[, XAXIS]))}) ENDIF
```

## Set Bus Signal

```
.SIGBUS BNAME | BNAMEMSB:LSB | BNAME[MSB:LSB] | BNAME<MSB:LSB>
+ [VHI=VAL1] [VLO=VAL2] [TFALL=VAL3] [TRISE=VAL4]
+ [BASE=OCTAL|DEC|BIN|HEXA] TN VAL {TN VAL} [P] [SIGNED=NONE|1COMP|2COMP]
.SIGBUS BNAME | BNAMEMSB:LSB | BNAME[MSB:LSB] | BNAME<MSB:LSB>
+ [VHI=VAL1] [VLO=VAL2] [TFALL=VAL3] [TRISE=VAL4]
+ [THOLD=VAL5] [TDELAY=VAL6] [BASE=OCTAL|DEC|BIN|HEXA]
+ PATTERN ${PAT} ${$(PAT)} | VAL {VAL} [Z] [SIGNED=NONE|1COMP|2COMP]
```

## Sinusoidal Voltage Source

```
.SINUS NODE VO VA FR [TD [THETA]]
```

## Spot Noise Figure

```
.SNF INPUT=(LIST_OF_DEVICES) OUTPUT=(LIST_OF_DEVICES)
+ [INPUT_TEMP=VAL] [NOISETEMP=VAL]
```

## Sizing Facility

```
.SOLVE PARAM param_name MIN MAX expr=expr [TOL=VAL]
+ [RELTOL=VAL] [GRID=VAL]
.SOLVE obj_name [W|L] MIN MAX expr=expr [TOL=VAL]
+ [RELTOL=VAL] [GRID=VAL]
.SOLVE CNAME [W|L] MIN MAX OPSIZE [TOL=VAL]
```

## Parameter Sweep

```
.STEP TEMP|DIPOLE INCR_SPEC
.STEP MOS W|L INCR_SPEC
.STEP MNAME PARAM_NAME INCR_SPEC
.STEP PARAM PAR_NAME INCR_SPEC {PAR_NAME INCR_SPEC}
.STEP PARAM PARAM_NAME INCR_SPEC, {[VALSTART] VALSTOP VALUE}
.STEP ITEM INCR_SPEC {ITEM2 BOUND}
.STEP (ITEM1,ITEM2,...,ITEMn)
+ LIST | = (VALi1, VALi2... VALin)... (VALj1, VALj2... VALjn)
```

## Subcircuit Definition

```
.SUBCKT NAME NN{NN}[(SWITCH|ANALOG|OSR|DIGITAL)]
+ [(NONOISE)] [(INLINE)] [PARAM: PAR=VAL {PAR=VAL}]
...
<CIRCUIT_COMPONENTS>
...
.ENDS [NAME]
.SUBCKT LIB FNAME SNAME [LIBTYPE]
```

## Subcircuit Duplicate Parameters

```
.SUBDUP SUBCKT_NAME
```

## Value Tables

```
.TABLE NAME AC|DC|TRAN (X1 Y1) {(XN YN)}
```

## Set Circuit Temperature

```
.TEMP TS {TS}
```

## Transfer Function

```
.TF OV IN
```

## Set Title of Binary Output File

```
.TITLE name
```

## Select the TOP Cell Subcircuit

```
.TOPCELL [=] <SUBCKT_NAME>
```

## Transient Analysis

```
.TRAN TPRINT TSTOP [TSTART [HMAX]] [SWEEP DATA=dataname] [UIC] [MONTE=val]
.TRAN TPRINT TSTOP [TSTART [HMAX]] [SWEEP parameter_name
+ TYPE nb start stop] [UIC] [MONTE=val]
.TRAN TPRINT TSTOP [TSTART [HMAX]] [SWEEP parameter_name start stop incr]
+ [UIC] [MONTE=val]
.TRAN INCRn Tn [{INCRn Tn}] [TSTART=val]
+ [SWEEP DATA=dataname] [UIC] [MONTE=val]
.TRAN INCRn Tn [{INCRn Tn}] [TSTART=val] [SWEEP parameter_name
+ TYPE nb start stop] [UIC] [MONTE=val]
.TRAN INCRn Tn [{INCRn Tn}] [TSTART=val] [SWEEP parameter_name
+ start stop incr] [UIC] [MONTE=val]
.TRAN DATA=dataname [SWEEP DATA=dataname] [UIC] [MONTE=val]
.TRAN DATA=dataname [SWEEP parameter_name TYPE nb start stop]
+ [UIC] [MONTE=val]
.TRAN DATA=dataname [SWEEP parameter_name start stop incr]
+ [UIC] [MONTE=val]
```

## Save Simulation Run at Multiple Time Points

```
.TSAVE [REPLACE | NOREPLACE] TIME=VALUE [FILE="FILEBASENAME"]
```

## Test Vector Files

```
.TVINCLUDE [FILE=]FILENAME [COMP=ON|OFF] [ERRNODE[=YES | NO]]
```

## Netlist Protection

```
.UNPROTECT
```

## Use Previously Simulated Results

```
.USE FILE_NAME [NODESET | IC | GUESS | OVERWRITE_INPUT]
```

## Use Reliability Model Key (Password)

```
.USEKEY [MODEL=model_name] KEY=key_value
```

## Use Tcl File

```
.USE_TCL FILENAME
```

## Worst Case Analysis

```
.WCASE DC | AC | TRAN [OUTPUT=MIN | MAX | BOTH] [VARY=LOT | DEV | BOTH] [TOL=VAL]
+ [ALL]
```

## Set Printer Paper Width

```
.WIDTH OUT=80 | 132
```

# Chapter 4

## Device Models

### Introduction

The following table summarizes the Device Models available.

**Table 4-1. Eldo Device Models**

Resistor	Capacitor	Inductor	Coupled Inductor
RC Wire	Semiconductor Resistor		
Transmission Line	Lossy Transmission Line	Lossy Transmission Line: W Model	Lossy Transmission Line: U Model
Microstrip Models			
Junction Diodes	Berkeley Level 1 (Eldo Level 1)	Modified Berkeley Level 1 (Eldo Level 2)	Fowler-Nordheim Model (Eldo Level 3)
	JUNCAP (Eldo Level 8)	Philips Diode Level 500 (Eldo Level 9)	Diode Level 21
	JUNCAP2 (Eldo Level 8, DIOLEV=11)		
BJT—Bipolar Junction Transistors	Modified Gummel-Poon Model (Eldo Level 1)	Philips Mextram 503.2 Model (Eldo Level 4)	Improved Berkeley Model (Eldo Level 5)
	VBIC v1.2 Model (Eldo Level 8)	VBIC v1.1.5 Model (Eldo Level 8)	HICUM Model (Eldo Level 9)
	Philips Mextram 504 Model (Eldo Level 22)	Philips Modella Model (Eldo Level 23)	HICUM Level0 Model (Eldo Level 24)
JFET—Junction Field Effect Transistor			
MESFET—Metal Semiconductor Field Effect Transistor			
MOSFETs	Berkeley SPICE Models	MERCKEL MOSFET Models	Berkeley SPICE BSIM1 Model (Eldo Level 8)
	Berkeley SPICE BSIM2 Model (Eldo Level 11)	Modified Berkeley Level 2 (Eldo Level 12)	Modified Berkeley Level 3 (Eldo Level 13)
	Modified Lattin-Jenkins-Grove Model (Eldo Level 16)	Enhanced Berkeley Level 2 Model (Eldo Level 17)	EKV MOS Model (Eldo Level 44 or EKV)

**Table 4-1. Eldo Device Models**

MOSFETs (cont.)	Berkeley SPICE BSIM3v2 Model (Eldo Level 47)	Berkeley SPICE BSIM3v3 Model (Eldo Level 53)	Motorola SSIM Model (Eldo Level 54 or SSIM)
	Berkeley SPICE BSIM3SOI v1.3 Model (Eldo Level 55)	Berkeley SPICE BSIM3SOI v2.x and v3.x Model (Eldo Level 56)	Philips MOS 9 Model (Eldo Level 59 or MOSP9)
	Berkeley SPICE BSIM4 Model (Eldo Level 60)	TFT Polysilicon Model (Eldo Level 62)	Philips MOS Model 11 Level 1101 (Eldo Level 63)
	TFT Amorphous-Si Model (Eldo Level 64)	Philips MOS Model 11 Level 1100 (Eldo Level 65)	HiSIM Model (Eldo Level 66)
	SP Model (Eldo Level 67)	Berkeley SPICE BSIM5 Model (Eldo Level 68)	Philips MOS Model 11 Level 1102 (Eldo Level 69)
	Philips PSP Model (Eldo Level 70)	BTA HVMOS Model (Eldo Level 101)	
S-Domain Filter	Z-Domain Filter	Subcircuit Instance	

## Resistor

```
Rxx N1 N2 [MOD[EL]=MNAME] [VAL] [[TC1=]T1] [[TC2=]T2] [[TC3=]T3]
+ [AC=VAL|{EXPR}] [T[EMP]=VAL] [DTEMP=VAL] [M=VAL] [L=VAL] [W=VAL]
+ [KEEPRMIN] [NONOISE] [KF=VAL] [AF=VAL] [WEEXP=VAL] [LEEXP=VAL]
+ [FEXP=VAL] [FMIN=VAL] [FMAX=VAL] [NBF=VAL]
Rxx N1 N2 [MOD[EL]=MNAME] VALUE={EXPR} [RESTORE_CAUSALITY=val]
+ [[TC1=]T1] [[TC2=]T2] [[TC3=]T3] [AC=VAL] [T[EMP]=VAL] [DTEMP=VAL]
+ [M=VAL] [KEEPRMIN] [NONOISE] [KF=VAL] [AF=VAL]
+ [WEEXP=VAL] [LEEXP=VAL] [FEXP=VAL] [FMIN=VAL] [FMAX=VAL]
+ [NBF=VAL] [FIT=VAL] [CFMAX=VAL] [CDEL=VAL]
Rxx N1 N2 [[TC1=]T1] [[TC2=]T2] [[TC3=]T3] [AC=VAL] [T[EMP]=VAL]
+ [DTEMP=VAL] [M=VAL] [KF=VAL] [AF=VAL] [WEEXP=VAL] [LEEXP=VAL]
+ [FEXP=VAL] [FMIN=VAL] [FMAX=VAL] [NBF=VAL]
+ TABLE EXPR [KEEPRMIN] [NONOISE]
Rxx NP NN POLY VAL {COEF} [TC1=T1] [TC2=T2] [TC3=T3]
```

### Parameters

- **XX**  
Resistor name.
- **N1, N2**  
Names of the resistor nodes.
- **VAL**  
Value of resistor in  $\Omega$  at nominal temperature. This value can be assigned directly or via the **.PARAM** command. Optional if a resistor model is used. For more information on **.PARAM**, see [page 10-205](#) of this manual.

#### Note



If a parameter name is required to be specified as the 4th token of the syntax, enclose it in braces {}, single quotes '', or use the syntax `r=param_name` to distinguish it from the model name.

- **MOD[EL]=MNAME**  
Model name.
- **TC1, TC2, TC3**  
First, Second, and Third order temperature coefficients. Default values are zero. The temperature coefficients can be expressed simply as values, without the `TC1=` for example. `TC3` can be specified even if `TC1/TC2` are not: they would default to zero.
- **POLY**  
Keyword to identify the resistor as non-linear polynomial.

**Note**

 Instead of using the **POLY** keyword, you can also specify parameters **VC1=value** and **VC2=value**. **VC1** is equivalent to the first parameter of the polynomial array, **VC2** is equivalent to the second parameter of the polynomial array.

---

- **COEF**

Polynomial coefficients used to calculate the voltage dependency of the resistor. The value of the resistor is computed as:

$$\text{Resistor Value} = \text{VAL} + R_1 \times V + R_2 \times V^2 + \dots + R_n \times V^n$$

where **V** is the voltage across the resistor.

- **AC=VAL**

Specifies the resistor value, in  $\Omega$ , which is used in AC analysis only.

- **AC={EXPR}**

Enables the instantiation of a resistor with a functional expression to be simulated in AC analysis.

- **VALUE={EXPR}**

Keyword indicating that the resistor has a functional description. This enables the instantiation of a frequency-dependent resistor to be simulated in all analysis modes (AC, Transient, SST and MODSST). The expression can make use of the **FREQ** keyword to specify frequency. A typical application is to model skin-effect, with **R** varying according to **SQRT(FREQ)**.

**Note**

 If **VALUE={EXPR}** is frequency based then the allowed syntax is as follows:

**Rxx NP NN VALUE={EXPR} [TC1=T1] [TC2=T2] [TC3=T3]**

**VALUE={EXPR}** and **AC={EXPR}** correspond to two different syntaxes, both accepted by Eldo, but they cannot be specified together.

**VALUE={EXPR}** can be used together with **AC=VAL** only when **VALUE={EXPR}** is *not* frequency based.

If the value of the resistor is defined by a **VALUE** expression, the actual value will be recalculated at each timestep during transient analysis. However, if an AC analysis is performed, the value is calculated only once in the DC analysis, and then considered to be static.

If the keyword **VALUE** is omitted, e.g. **rxx n1 n2 {expr}**, then it is assumed that it is possible to evaluate the expression before runtime, i.e. that the necessary parameters are available to the simulator. If this is not the case, errors will occur.

---

- **RESTORE\_CAUSALITY=val**

The device characteristics are defined by an equation, when set to 1 the causality of the equation will be restored (if required) by building the corresponding imaginary part, this

will produce results with an even higher level of accuracy. When set to 0 (default) the causality of the equation will not be restored.

- **T [ EMP ]=VAL**

Sets temperature for the individual device, in degrees Celsius. Default nominal temperature=27°C.

- **DTEMP=VAL**

Temperature difference between the device and the rest of the circuit, in degrees Celsius. Default value is 0.0.

**Note**

 **TEMP** and **DTEMP** are mutually exclusive. If both are specified, the last one is utilized.

- **M=VAL**

Device multiplier, simulating the effect of multiple devices in parallel: effective resistance value is divided by **M**. Default is 1.

The device is first evaluated without the **M** factor, and at the very end of the device computation, all scaling quantities are multiplied / divided by **M**. Input values **W** and **L** are not affected. Models are chosen depending on input **W** and **L**, if required. Options **MINL**, **MAXL**, **MINW**, **MAXW**, etc. do not apply either, since they check the input values of **W** and **L**.

**Note**

 Using an M factor value less than 1 could lead to simulating devices that cannot be physically realized.

- **KEEPRMIN**

Any effect of **RMMINRVAL** and **MINRVAL** options will be ignored for the device. This is a way to force Eldo to keep the device under all circumstances unless the device is equal to zero.

- **NONOISE**

Specifies that no noise model will be used for this device when performing noise analysis. Therefore, the device presents no noise contribution to the noise analysis.

- **TABLE**

Keyword indicating that the resistor accepts a table description.

- **KF=VAL**

Flicker noise coefficient. Default is 0.

- **AF=VAL**

Flicker noise exponent. Default is 1.0.

- **WEEXP=VAL**  
Flicker noise exponent. Default is 0.0.
- **LEEXP=VAL**  
Flicker noise exponent. Default is 0.0.
- **FEXP=VAL**  
Flicker noise exponent. Default is 1.0. Flicker noise equation is:

$$S_{id} = K_F \cdot \frac{I^{AF}}{W_{eff}^{WEEXP} \cdot L_{eff}^{LEEXP} \cdot freq^{FEXP}}$$

- **FMIN=VAL**  
Lower limit of the noise frequency band.
- **FMAX=VAL**  
Upper limit of the noise frequency band.

---

**Note**



**FMIN** and **FMAX** define the frequency band of the noise sources. This frequency range may sometimes not correspond to the noise frequency band at the output of the circuit. For instance, the band (**FMIN**, **FMAX**) does not correspond to the output noise frequency band in the case of filters or oscillators and mixers that exhibit frequency conversion. **FMIN** is also used to specify the algorithm used to generate the noise source generated by the resistor. When **FMIN>0** the resistor noise source is generated with sinusoids; when **FMIN=0** it is generated with a continuous spectrum between **FMIN** and **FMAX**.

---

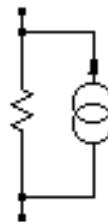
- **NBF=VAL**  
Specifies the number of sinusoidal sources with appropriate amplitude and frequency and with randomly distributed phase from which the noise source is composed. Default value is 50. This parameter has no effect when **FMIN** is set to 0.
- **FIT=VAL**  
Specifies the method used to perform transient analysis. **FIT=1** indicates the fitting method is used, **FIT=0** indicates the convolution method is used. Default value is 1.
- **CFMAX=VAL**  
Specifies the maximum frequency value used in the functional description of the resistor. It is used in transient analysis to perform impulse response using a convolution method. Default is  $1.0 \times 10^9$  Hz. Can only be used when **FIT=0**.
- **CDELF=VAL**  
Specifies the frequency interval to compute the functional description of the resistor. It must have the form  $2^n$  for the convolution computation (automatic check and correction are done if not). Default is  $1.0 \times 10^9 / 1024 = 9.765625 \times 10^5$  Hz. Can only be used when **FIT=0**.

**Note**

The parameters **FIT**, **CFMAX**, **CDELF** should only be used when the resistor is frequency dependent, else they will be ignored.

The user can specify the minimum value that resistors in the netlist can take using the option **RSMALL**. For more information please refer to [page 11-37](#).

## Noise in Resistors



The thermal noise of a resistor is as follows:

$$SI = \frac{4 \times k \times T}{R}$$

## Examples

```
r1 n3 n4 3.3k
```

Specifies a  $3.3\text{k}\Omega$  resistor placed between nodes `n3` and `n4`.

```
r2 n1 n2 rval
.param rval=2k
```

Specifies a resistor named `r2` of value `rval` between nodes `n1` and `n2`. The resistor value is declared globally in the **.PARAM** command.

```
r3 1 2 value={2k*v(3,4)*i(v5)}
```

Specifies a resistor `r3` between nodes `1` and `2` whose value is described by the expression in curly brackets.

```
rg4 4 5 table (v(p3n)) = (0, 1e11) (1v, 1e3)
```

The value of resistor `rg4` is  $1\times10^{11}\Omega$  when  $v(p3n)=0\text{V}$ ,  $1\text{k}\Omega$  when  $v(p3n)=1\text{V}$ . The other values are interpolated.

```
r5 1 2 value={50*sqrt(1+(FREQ/10e6))}
```

Specifies a frequency-dependent resistor `r5` between nodes `1` and `2` whose value is described by the expression in curly brackets.

```
r1 1 2 ac=3 value={2k*v(3,4)*i(v5)}
```

Specifies a resistor **r1** between nodes 1 and 2 whose value is described by the expression in curly brackets. This value is superseded when an AC analysis is performed, where the **ac** value is used instead, in this case  $3\Omega$ .

```
r2 3 4 value={50*sqrt(1+(FREQ/10e6))} tc1=0.001
+ tc2=0.004 tc3=0.003
```

Specifies a frequency dependent resistor **r2** between nodes 3 and 4, whose value is described by the expression in curly brackets, that also has first, second and third order temperature coefficients.

## Resistor Model Syntax

```
.MODEL MNAME R[ES] [ {PAR=VAL} ]
```

---

**Note**



Specifying **SCALM=val** in the **.MODEL** statement in a resistor model overrides the global scaling specified by the **.OPTION SCALM=val** statement.

---

## Level 1

**Table 4-2. Resistor Model—Level 1 Parameters**

Nr.	Name	Description	Default	Units
1	R or SCALE[R]	Resistance multiplier	1	
2	RDEF	Value of resistor. This value has priority over values computed from L and W	0	$\Omega$
3	POLY <sup>a</sup>	Keyword to identify the resistor as non-linear polynomial		
4	POLYV	Used to change the formula used for computing the R value	2	
5	REVSP	Used to change the formula used for computing the R value	0	
6	TC1	First order temperature coefficient	0	$^{\circ}\text{C}^{-1}$
7	TC2	Second order temperature coefficient	0	$^{\circ}\text{C}^{-2}$
8	TC3	Third order temperature coefficient	0	$^{\circ}\text{C}^{-3}$
9	L	Default resistor length	0	m
10	W	Default resistor width	0	m
11	LOT	Correlated device tolerance (Monte Carlo analysis)		

**Table 4-2. Resistor Model—Level 1 Parameters**

Nr.	Name	Description	Default	Units
12	DEV	Uncorrelated device tolerance (Monte Carlo analysis)		
13	NOISE	Noise contribution of resistor <sup>b</sup>	1.0	
14	WEEXP	Flicker noise exponents	0.0	
15	LEEXP		0.0	
16	FEXP		1.0	
17	TNOM	Nominal temperature	27	°C

a. Instead of using the **POLY** keyword, you can also specify parameters **VC1=value** and **VC2=value**. **VC1** is equivalent to the first parameter of the polynomial array, **VC2** is equivalent to the second parameter of the polynomial array.

b. Noise contribution of the resistor will be expressed as  $4 \times KB \times T \times NOISE/R$ . Where **KB** is the Boltzmann coefficient.

The below equation defines resistor value as a function of temperature, where **T** is the operating temperature specified either by the **.TEMP** command, or the **T** parameter. **Tnom** is the nominal temperature for which the resistor has resistance **VAL**. Default value of **Tnom** is 27 °C and is adjustable using **.OPTION TNOM**.

$$VAL(T) = VAL(T_{nom})(1 + TC1(T - T_{nom}) + TC2(T - T_{nom})^2 + TC3(T - T_{nom})^3)$$

The Resistor model has additional levels for the **.MODEL** card.

---

**Note**


---

**TC1, TC2, TC3** and **R** are used by model levels 1, 2, 3 and 4.

---

### POLY usage

The **POLY** keyword identifies the model as a non-linear polynomial. It applies to the resistance/capacitance/inductor model syntax and is accepted on the model card in which the coefficients are used to generate R, C or L values which are dependent on bias. Syntax:

```
.MODEL <model_name> [R|C|IND] POLY <list_of_parameter_values>
```

This polynomial list can be given in addition to existing model parameters, for example:

```
.MODEL <model_name> R TC1=0.1 POLY <list_of_parameter_values>
+ TC2=0.2
```

The bias independent value of the resistor will be computed as if there were no polynomial parameters. Then, the active value of the resistor will be equal to the product of the “bias-independent value” multiplied by the polynomial expression.

Notes:

- **POLY** can be specified in a capacitance instance the same way it is used for resistance.
- Instead of using the **POLY** keyword, you can also specify parameters **VC1=value** and **VC2=value**. **VC1** is equivalent to the first parameter of the polynomial array, **VC2** is equivalent to the second parameter of the polynomial array.

## REVSP and POLYV usage

- Parameter **POLYV** can also be specified on the **.MODEL** card. **POLYV** can take two values: equal to 2 or not equal to 2. It will be used to change the formula which is used for computing the R/L/C value. Assuming the model:

```
.MODEL Foo C POLY P1 P2 P3... Pn  
C pin1 pin2 Foo <cinst>
```

If  $V = V(\text{pin1}) - V(\text{pin2})$ :

then if **POLYV** is 2: the value of C will be computed as:

$$C = \langle cinst \rangle \times (P1 + P2 \times V + P3 \times V^2 + \dots)$$

if **POLYV** is not 2: the value of C will be computed as:

$$C = \langle cinst \rangle \times (1 + P1 \times V + P2 \times V^2 + \dots)$$

P1, P2 represent the polynomial coefficients. Default value for **POLYV** is 2.

- Parameter flag **REVSP** can also take two values: equal to 1 or not equal to 1. **POLYV** and **REVSP** parameters are flags to modify computations. They do not interact directly within the computation. If parameter **REVSP** is set to 1 on the **.MODEL** card, and if **POLYV** is not 2, then the formula for computing the current generated by the resistor is:

$$I = \frac{V}{r_{inst}} \times \left( 1 + \frac{P1 \times V}{2} + \frac{P2 \times V^2}{3} + \dots \right)$$

which leads to a resistor of value:

$$R(V) = \frac{1}{\delta I / \delta V} = \frac{r_{inst}}{(1 + P1 \times V + P2 \times V^2 + \dots)}$$

P1, P2 represent the polynomial coefficients. Default for **REVSP** is 0.

## Example

```
.model rmodel res tc1=0.001 tc2=0.005 tc3=0.008  
...  
r2 n1 n19 rmodel 2.5k
```

Specifies a  $2.5\text{k}\Omega$  resistor placed between nodes **n1** and **n19**. The first order temperature coefficient is 0.001, the second order temperature coefficient is 0.005, and the third order temperature coefficient is 0.008, all being defined using the **.MODEL** command.

## Level 2

This is a private ST model. For a description of equations please contact STMicroelectronics.

This is the default model when the `-stver` flag is specified (or `.option stver`).

Parameters:

**Table 4-3. Resistor Model—Level 2 Parameters**

Nr.	Name	Description	Default	Units
1	RHO	Sheet resistance	0.0	Ω/sq
2	RCON <sup>a</sup>	Resistance of contacts	0	Ω
3	NC1 (NC)	Number of ohmic contacts at node 1		
4	NC2	Number of ohmic contacts at node 2		
5	L	Default resistor length	0	m
6	W	Default resistor width	0	m
7	DL	Delta length	0	m
8	DW	Delta width	0	m

a. RCON is taken into account in the Resistor computation if NC1, NC2 are defined.

## Level 3

This corresponds to the RC Wire model.



Please see the “[RC Wire](#)” on page 4-28.

## Level 4

This is the default model inside Accusim. Parameters:

**Table 4-4. Resistor Model—Level 4 Parameters**

Nr.	Name	Description	Default	Units
1	RSH	Sheet resistance	50.0	Ω/sq
2	L	Default resistor length	0	m
3	W	Default resistor width	0	m
4	LNARROW	Delta length	0	m
5	WNARROW	Delta width	0	m
6	NARROW	Delta	0	m

**Table 4-4. Resistor Model—Level 4 Parameters**

Nr.	Name	Description	Default	Units
7	R	Multiplier (used for Monte Carlo)	1	

The device value is computed as follows:

$$R = RSH \times \frac{L_{eff}}{W_{eff}}$$

$L_{eff} = L - LNARROW$  if **LNARROW** is specified,  
or  $L_{eff} = L - NARROW$  if **NARROW** is specified.

$W_{eff} = W - WNARROW$  if **WNARROW** is specified,  
or  $W_{eff} = W - NARROW$  if **NARROW** is specified.

# Capacitor

```
Cxx NP NN [MOD[EL]=MNAME] [DCCUT] [VAL] [M=VAL] [L=VAL] [W=VAL]
+ [T[EMP]=VAL] [DTEMP=VAL] [TC1=T1] [TC2=T2] [TC3=T3] [IC=VAL]
Cxx NP NN POLY VAL {COEF} [TC1=T1] [TC2=T2] [TC3=T3] [M=VAL]
+ [CTYPE=VAL] [IC=VAL]
Cxx NP NN [VALUE={EXPR}] [RESTORE_CAUSALITY=val]
+ [TC1=T1] [TC2=T2] [TC3=T3] [CTYPE=VAL]
```

## Parameters

- **XX**  
Capacitor name.
- **NP**  
Name of the positive node.
- **NN**  
Name of the negative node.
- **VAL**  
Value of the capacitor in Farads (voltage independent value). This value can be assigned directly or via the **.PARAM** command. Optional if a capacitor model is used. For more information on **.PARAM**, see [page 10-205](#) of this manual.
- **POLY**  
Keyword to identify the capacitor as non-linear polynomial.

---

**Note**

Instead of using the **POLY** keyword, you can also specify parameters **VC1=value** and **VC2=value**. **VC1** is equivalent to the first parameter of the polynomial array, **VC2** is equivalent to the second parameter of the polynomial array.

---

- **VALUE={EXPR}**

Keyword indicating that the capacitor has a functional description. This enables the instantiation of a frequency-dependent capacitor to be simulated in all analysis modes (AC, Transient, SST and MODSST). The expression can make use of the **FREQ** keyword to specify frequency. A typical application is to model skin-effect, with C varying according to **SQRT(FREQ)**.

---

**Note**

If the keyword **VALUE** is omitted, e.g. **cxx n1 n2 {expr}**, then it is assumed that it is possible to evaluate the expression before runtime, i.e. that the necessary parameters are available to the simulator. If this is not the case, errors will occur.

---

- **RESTORE\_CAU<sub>SALITY</sub>=val**

The device characteristics are defined by an equation, when set to 1 the causality of the equation will be restored (if required) by building the corresponding imaginary part, this will produce results with an even higher level of accuracy. When set to 0 (default) the causality of the equation will not be restored.

- **MNAME**

Model name. This is useful in combination with Monte Carlo analysis.

- **DCCUT**

For DC, AC, TRAN, MODSST analyses, the **DCCUT** device is assumed to be a normal capacitance of **VAL** Farads. For **.SST**, **.SSTAC**, **.SSTNOISE** and **.SSTXF** analyses, the **DCCUT** device corresponds to an open circuit in DC and a short circuit for all other frequencies.

- **COEF**

Polynomial coefficients used to calculate the voltage dependency of the capacitance. The value of the capacitor is computed as:

$$\text{Capacitor Value} = \text{VAL} + C_1 \times V + C_2 \times V^2 + \dots + C_n \times V^n$$

where **V** is the voltage across the capacitor.

- **M=VAL**

Multiplier representing the number of parallel devices in the simulation. The total capacitor value will be multiplied by this parameter value. Default value=1.0.

The device is first evaluated without the **M** factor, and at the very end of the device computation, all scaling quantities are multiplied / divided by **M**. Input values **W** and **L** are not affected. Models are chosen depending on input **W** and **L**, if required. Options **MINL**, **MAXL**, **MINW**, **MAXW**, etc. do not apply either, since they check the input values of **W** and **L**.

---

**Note**



Using an M factor value less than 1 could lead to simulating devices that cannot be physically realized.

---

- **L=VAL**

Capacitor length (Effective  $L = L \times \text{SHRINK}$ ), see model parameter list where the shrink factor is listed.

- **W=VAL**

Capacitor width (Effective  $W = W \times \text{SHRINK}$ ), see model parameter list where the shrink factor is listed.

- **T [ EMP ]=VAL**

Sets temperature for the individual device, in degrees Celsius. Default nominal temperature=27°C.

- **DTEMP=VAL**

Temperature difference between the device and the rest of the circuit, in degrees Celsius. Default value is 0.0.

**Note**



**TEMP** and **DTEMP** are mutually exclusive. If both are specified, the last one is utilized.

- **TC1, TC2, TC3**

First, Second, and Third order temperature coefficients. Default values are zero.

- **IC=VAL**

Sets the initial guess for the voltage across the capacitor prior to a transient analysis. To use this option, the **UIC** parameter must also be present in the **.TRAN** statement. For more information on **.TRAN**, see [page 10-318](#) of this manual.

- **CTYPE=VAL**

Determines the calculation mode, for capacitors defined by polynomial or functional expressions. The default value is 0.

When **CTYPE**=0, current out of bias-dependent capacitor is computed as  $I(C) = dQ/dt$ , with Q computed by integrating the equation  $dQ = C \times dV$ . Default.

When **CTYPE**=1, current out of bias-dependent capacitor is also computed as  $I(C) = dQ/dt$ , but with Q computed as  $Q = C \times V$ .

Eldo will check dependencies of the capacitor value. When the value of the capacitor does not depend on the bias of the pins of the device, the **CTYPE**=1 formulation is normally more appropriate. In this case Eldo will behave on the device as if **CTYPE**=1 had been set, unless option **NOAUTOCODE** is set, or unless **CTYPE** is explicitly set on the device,

**Note**



The **IC=VAL** statement cannot be given immediately after a model name, for instance in the following example: **Cxx NP NN MNAME IC=value**, MNAME would not be considered as a model name, but as the value of the capacitor, and Eldo would look for a parameter named MNAME. Ensure any other parameter is inserted preceding the **IC** parameter, for example: **Cxx NP NN MNAME TC1=0 IC=value**

**Note**



If the value of the capacitor is defined by a **POLY** or **VALUE** expression, the actual value will be recalculated at each timestep during transient analysis. If an AC analysis is performed however, the value is calculated only once in the DC analysis, and then considered to be static.

## Examples

```
c1 n3 n4 0.5pf
```

Specifies a 0.5pF capacitor **c1** placed between nodes **n3** and **n4**.

```
c1 n3 n7 poly 5p 0.1p 0.07p 0.004p
```

Specifies a 5pF capacitor **c1** placed between nodes **n3** and **n7**, whose voltage dependency is described by the 3rd order polynomial:

$$\text{value} = 5\text{p} + 0.1\text{p} \times V + 0.07\text{p} \times V^2 + 0.004\text{p} \times V^3$$

where **V** is the voltage across the capacitor.

```
c2 n1 n2 cval
.param cval=0.4p
```

Specifies a capacitor **c2** of value **cval** placed between nodes **n1** and **n2**. The capacitor value is declared globally in the **.PARAM** command.

```
c1 1 2 value={2n*v(3, 4)*i(v5)}
```

Specifies a capacitor **c1** between nodes **1** and **2** whose value is described by the expression in curly brackets.

Dynamic current calculation (**CTYPE**) example:

```
v10 2 0 sin (0 1 100meg)
r10 2 0 1
v1 1 0 1
c1 1 0 value = {1.0e-9 * v(2,0)}
.tran 1n 100n
.print tran i(v1)
.plot tran i(v1)
.end
```

Eldo will issue a warning that **CTYPE=1** is emulated on device C1.

## Capacitor Model Syntax

```
.MODEL MNAME C[AP] [ {PAR=VAL} ]
```

The Eldo capacitor model supports Monte Carlo analysis whereby the tolerance of the capacitor can be defined to vary in a correlated or un-correlated way over a number of simulation runs.

---

### Note

 Specifying **SCALM=val** in the **.MODEL** statement in a capacitor model overrides the global scaling specified by the **.OPTION SCALM=val** statement.

---

**Table 4-5. Capacitor Model Parameters**

Nr.	Name	Description	Default	Units
1	C <sup>a</sup> or SCALE[C]	Capacitance multiplier	1	
2	CDEF	Value of capacitor if value isn't given in instance	0	F
3	LOT	Correlated device tolerance (Monte Carlo analysis)		
4	DEV	Uncorrelated device tolerance (Monte Carlo analysis)		
5	POLY <sup>b</sup>	Keyword to identify the capacitor as non-linear polynomial		
6	POLYV	Used to change the formula used for computing the C value	2	
7	REVSP	Used to change the formula used for computing the C value	0	
8	TC1	First order temperature coefficient	0	°C <sup>-1</sup>
9	TC2	Second order temperature coefficient	0	°C <sup>-2</sup>
10	TC3	Third order temperature coefficient	0	°C <sup>-3</sup>
11	CAPSW	Sidewall fringing capacitance	0	Fm <sup>-1</sup>
12	COX <sup>c</sup>	Bottomwall capacitance	0	Fm <sup>-2</sup>
13	DEL	Small decrement in dimensions of a device structure on both ends of L and W due to process effects (undercutting)	0	
14	DI	Relative dielectric constant	3.9	
15	SHRINK <sup>d</sup>	Shrink factor for physical dimensions	1	
16	THICK	Insulator thickness	0	m
17	L	Default capacitor length	0	m
18	W	Default capacitor width	0	m
19	DTEMP	Temperature difference of capacitor with respect to the rest of the circuit	0	
20	SCALM	Model parameter scaling factor. This overrides the global SCALM value defined using the .OPTION command.	1	
21	TNOM	Nominal temperature	27	°C

a. Is computed as  $L_{\text{eff}} \times W_{\text{eff}} \times C_{\text{oxeff}} + 2 \times (L_{\text{eff}} + W_{\text{eff}}) \times \text{CAPSW}$

- b. Instead of using the **POLY** keyword, you can also specify parameters **VC1=value** and **VC2=value**. **VC1** is equivalent to the first parameter of the polynomial array, **VC2** is equivalent to the second parameter of the polynomial array.
- c.  $C_{oxeff}$  is computed as  $\text{EPSO} \times \text{DI}/\text{TOX}$  if **TOX** is specified,  $\text{EPSO} \times \text{DI}/\text{THICK}$  if **THICK** is specified (**TOX** and **THICK** are synonymous),  $\text{EPSO} \times \text{DI}/0.5\mu$  otherwise.
- d. Effective value of length =  $(L \times \text{SHRINK} - 2 \times \text{DEL} \times \text{SCALEM})$   
Effective value of width =  $(W \times \text{SHRINK} - 2 \times \text{DEL} \times \text{SCALEM})$

If **COX** is not specified, it is computed as:

$$\text{COX} = \frac{\varepsilon_0 \times \text{DI}}{\text{THICK}} \text{ if } \text{DI} \text{ is specified, COX} = \frac{\varepsilon_x}{\text{THICK}} \text{ otherwise,}$$

where  $\varepsilon_0$  is the permittivity of air, and  $\varepsilon_{ox}$  is the permittivity of the oxide.



**Tip:** For detailed information on usage of **POLY**, **REVSP**, and **POLYV** parameters, see [page 4-9](#) of this manual.

---

## Examples

```
*MODEL definition
.model cmodel cap lot=2%
...
*main circuit
c3 1 s cmodel 0.5pf
```

Specifies a 0.5pF capacitor **c3** placed between the nodes **1** and **s**, using the **.MODEL** command to define its parameters for a Monte Carlo analysis.

```
c1 1 2 cmodell tc1=25e-3 m=2 dtemp=10
.model cmodell c cox=1e-12 caps=1e-12 del=0.01
+ l=1 w=1 tc1=50e-6 tc2=0
```

These are valid instance and model cards from which nominal capacitance may be evaluated from a simplified equation (no **SHRINK** or scale factors):

Capacitance =  
 $(L - 2 \times \text{DEL}) \times (W - 2 \times \text{DEL}) \times \text{COX} + 2 \times (L - 2 \times \text{DEL}) \times (W - 2 \times \text{DEL}) \times \text{CAPSW}$

In the following example, the bias independent value of the capacitor will be computed as if there were no **POLY** parameters. Then, the active value of the capacitor will be equal to the product of the “bias-independent-value” multiplied by the polynomial expression.

```
.MODEL model_name C TC1=1 POLY
+ <list_of_parameter_values> TC2=value
```

**POLYV** example:

Assuming the model:

```
.MODEL Foo C POLY P1 P2 P3.. Pn
C pin1 pin2 Foo <cinst>
```

And, for example,  $V = V(pin1) - V(pin2)$ :

- If **POLYV=2**: the value of  $C$  will be computed as:  
$$C = <cinst> * (P1 + P2 * V + P3 * V * V + \dots)$$
- If **POLYV≠2**: the value of  $C$  will be computed as:  
$$C = <cinst> * (1 + P1 * V + P2 * V * V + \dots)$$

## Inductor

```
Lxx NP NN [MOD[EL]=MNAME] [DCFEED] [VAL] [M=VAL1] [T[EMP]=VAL] [DTEMP=VAL]
+ [IC=VAL3] [TC1=T1] [TC2=T2] [TC3=T3] [R=VAL4]
Lxx NP NN POLY VAL {LN} [IC=VAL] [R=VAL] [TC1=T1] [TC2=T2] [TC3=T3]
Lxx NP NN [VALUE={EXPR}] [RESTORE_CAUSALITY=val] [R=VAL|R VALUE=EXPR|R
+ TABLE {fval rval}] [TC1=T1] [TC2=T2] [TC3=T3]
Lxx {port_list} KMATRIX=data_block
Lxx {port_list} RELUCTANCE=({rn,cn,valn}) <options>
Lxx {port_list} RELUCTANCE {FILE="file"} <options>
```

### Parameters

- **XX**  
Inductor name.
- **NP**  
Name of the positive node.
- **NN**  
Name of the negative node.
- **VAL**  
Inductor value in Henrys. This value can be assigned directly or via the **.PARAM** command. Optional if an inductor model is used. For more information on **.PARAM**, see [page 10-205](#) of this manual.
- **POLY**  
Keyword to identify the inductor as non-linear polynomial.

---

#### Note



Instead of using the **POLY** keyword, you can also specify parameters **VC1=value** and **VC2=value**. **VC1** is equivalent to the first parameter of the polynomial array, **VC2** is equivalent to the second parameter of the polynomial array.

---

- **VALUE={EXPR}**

Keyword indicating that the inductor has a functional description. This enables the instantiation of a frequency-dependent inductor to be simulated in all analysis modes (AC, Transient, SST and MODSST). The expression can make use of the **FREQ** keyword to specify frequency. A typical application is to model skin-effect, with L varying according to **SQRT(FREQ)**.

---

#### Note



If the keyword **VALUE** is omitted, e.g. **lxx n1 n2 {expr}**, then it is assumed that it is possible to evaluate the expression before runtime, i.e. that the necessary parameters are available to the simulator. If this is not the case errors will occur.

---

- **RESTORE\_CAU<sub>S</sub>LITY=val**

The device characteristics are defined by an equation, when set to 1 the causality of the equation will be restored (if required) by building the corresponding imaginary part, this will produce results with an even higher level of accuracy. When set to 0 (default) the causality of the equation will not be restored.

- **MNAME**

Model name. This is useful in combination with Monte Carlo analysis. If **MOD=** is specified, Eldo will look for a **.MODEL** card. If not specified, Eldo will look for a parameter named **foo**.

- **DCFEED**

For DC, AC, TRAN, MODSST analyses, the **DCFEED** device is assumed to be a normal inductance of VAL Henrys. For **.SST**, **.SSTAC**, **.SSTNOISE** and **.SSTXF** analyses, the **DCFEED** device corresponds to an short circuit in DC and an open circuit for all other frequencies.

- **LN**

Coefficients of a polynomial used to calculate the inductances. The inductor is expressed as a function of the current **I** across the element. The value of the inductance is computed as:

$$\text{Inductor Value} = \text{VAL} + L_1 \times I + L_2 \times I^2 + \dots + L_n \times I^n$$

where **I** is the current through the inductor.

- **IC=VAL**

Sets the initial ‘guess’ for the current through the inductor prior to a transient analysis. To use this option the **UIC** parameter must also be present in the **.TRAN** statement.

- **M=VAL**

Multiplier representing the number of parallel devices in the simulation. Default value=1.0.

The device is first evaluated without the **M** factor, and at the very end of the device computation, all scaling quantities are multiplied / divided by **M**. Input values **w** and **L** are not affected. Models are chosen depending on input **w** and **L**, if required. Options **MINL**, **MAXL**, **MINW**, **MAXW**, etc. do not apply either, since they check the input values of **w** and **L**.

---

**Note**



Using an M factor value less than 1 could lead to simulating devices that cannot be physically realized.

---

- **T [ EMP ]=VAL**

Sets temperature for the individual device, in degrees Celsius. Default nominal temperature=27°C.

- **DTEMP=VAL**

Temperature difference between the device and the rest of the circuit, in degrees Celsius.  
Default value is 0.0.

**Note**

 **TEMP** and **DTEMP** are mutually exclusive. If both are specified, the last one is utilized.

---

- **TC1, TC2, TC3**

First, Second, and Third order temperature coefficients.

- **R=VAL | R VALUE={EXPR} | R TABLE={fval rval}**

**R** is a resistor that is added in series with inductor **L**. Default value is 0.0. When **R VALUE={EXPR}** is specified, it indicates that the resistor has a functional description, enabling a frequency-dependent resistor. The expression can make use of the **FREQ** keyword to specify frequency. When **R TABLE** is specified, it indicates that the resistor accepts a table description, with pairs of frequency and resistor values specified. This device is supported for all analyses (DC, AC, TRAN, SST, MODSST, etc.).

**Note**

 If the value of the inductor is defined by a **POLY** or **VALUE** expression, the actual value will be recalculated at each timestep during transient analysis. If an AC analysis is performed however, the value is calculated only once in the DC analysis, and then considered to be static.

---

- **KMATRIX=data\_block**

K-element for modeling the parasitic inductive effect of general 3-D interconnects. It is described by a multi-port inductor record and the matrix values are provided through a data block (**.DATA**). Values in the matrix are reluctance values, not inductance values.

Eldo will check that:

- the K-element has an even number of nodes
- the **KMATRIX** has a valid value and a data block exists
- the matrix is positive definite
- all diagonal terms are defined and have positive values
- there is no duplicate entry and no term is defined below the diagonal
- **port\_list**

List of ports for K-element, specified as pairs of electrical node names, each defining an interconnect branch.

- **RELUCTANCE**

Keyword indicates that reluctance (inverse inductance) will be provided. The unit for reluctance is inverse Henry ( $H^{-1}$ ). The presence of this keyword indicates that all tokens between Lname and the RELUCTANCE keyword should be interpreted as node names.

- **RELUCTANCE=( {rn, cn, valn} )**

Inline form of reluctance specification for modeling the parasitic inductive effect of general 3-D interconnects. `rn, cn, valn` is a triplet of values describing the reluctance between two interconnect branches.

`rn` integer value of branch #1; integer refers to the pair of nodes that define branch #1

`cn` integer value of branch #2; integer refers to the pair of nodes that define branch #2

`valn` reluctance value between branch #1 and branch #2 ( $H^{-1}$ )

- **FILE="file"**

External file format of reluctance specification for modeling the parasitic inductive effect of general 3-D interconnects. The data files should contain three columns of data. Each row should contain an (r, c, val) triplet separated by spaces. The r, c, and val values may be expressions surrounded by single quotes. Multiple files may be specified to allow the reluctance data to be spread over several files if necessary. The files should not contain a header row.

- **<options>**

Valid options for reluctance specification include:

`SHORTALL=yes | no`

causes all inductors to be converted to short circuits, and all reluctance matrix values to be ignored.

`IGNORE_COUPLING=yes | no`

causes all off-diagonal terms to be ignored (i.e., set to zero).

## Examples

```
11 n13 n8 5u
```

Specifies a 5 $\mu$ H inductor `11` placed between nodes `n13` and `n8`.

```
11 n10 n5 poly 7u 0.15 0.03
```

Specifies a 7 $\mu$ H inductor `11` placed between nodes `n10` and `n5`, whose current dependency is described by a second order polynomial of the form:

$$\text{value} = 7\mu + 0.15 \times I + 0.003 \times I^2$$

where  $I$  is the current across the inductor.

```
13 n1 n2 lval
.param lval=0.6ph
```

Specifies an inductor named **13** of value **lval** placed between nodes **n1** and **n2**. The inductor value is declared globally in the **.PARAM** command.

```
11 1 2 value={2u*v(3, 4)*i(v5)}
```

Specifies an inductor **11** between nodes **1** and **2** whose value is described by the expression in curly brackets.

K-element example:

```
L_ThreeNets (a,1) (1,2) (2,a_1) (b,4) (4,5) (5,b_1) (c,7) (7,8)
(8,c_1) KMATRIX=StartK
.DATA StartX
+ PORT1 PORT2 VALUE
+ 1      1      103e9
+ 1      4      -34.7e9
+ 1      7      -9.95e9
+ 4      4      114e9
+ 4      7      -34.7e9
+ 7      7      103e9
+ 2      2      103e9
+ 2      5      -34.7e9
+ 2      8      -9.95e9
+ 5      5      114e9
+ 5      8      -34.7e9
+ 8      8      103e9
+ 3      3      103e9
+ 3      6      -34.7e9
+ 3      9      -9.95e9
+ 6      6      114e9
+ 6      9      -34.7
+ 9      9      103e9
```

Another K-element example using the alternate inline syntax:

```
L_ThreeNets a 1 1 2 2 a_1 b 4 4 5 5 b_1 c 7 7 8 8 c_1
+ RELUCTANCE=(
+ 1, 1, 103e9,
+ 1, 4, -34.7e9,
+ 1, 7, -9.95e9,
+ 4, 4, 114e9,
+ 4, 7, -34.7e9,
+ 7, 7, 103e9,
+ 2, 2, 103e9,
+ 2, 5, -34.7e9,
+ 2, 8, -9.95e9,
+ 5, 5, 114e9,
+ 5, 8, -34.7e9,
+ 8, 8, 103e9,
+ 3, 3, 103e9,
+ 3, 6, -34.7e9,
+ 3, 9, -9.95e9,
+ 6, 6, 114e9,
+ 6, 9, -34.7e9,
+ 9, 9, 103e9 )
```

Another K-element example using the alternate external file syntax:

```
L_ThreeNets a 1 1 2 2 a_1 b 4 4 5 5 b_1 c 7 7 8 8 c_1 RELUCTANCE
+ FILE="reluctance.dat"
```

where the file *reluctance.dat* contains the three columns of data.

## Inductor Model Syntax

```
.MODEL MNAME IND [ {PAR=VAL} ]
```

The inductor model provided with Eldo is, as with the capacitor model described before, used in conjunction with a Monte Carlo analysis whereby the tolerance of the inductor can be defined to vary in a correlated or uncorrelated way during a number of simulation runs.

**Table 4-6. Inductor Model Parameters**

Nr.	Name	Description	Default	Units
1	L or SCALEL	Inductance multiplier	1	
2	LOT	Correlated device tolerance (Monte Carlo analysis)		
3	DEV	Uncorrelated device tolerance (Monte Carlo analysis)		
4	POLY <sup>a</sup>	Keyword to identify the inductor as non-linear polynomial		
5	POLYV	Used to change the formula used for computing the L value	2	
6	REVSP	Used to change the formula used for computing the L value	0	
7	TC1	First order temperature coefficient	0	°C <sup>-1</sup>
8	TC2	Second order temperature coefficient	0	°C <sup>-2</sup>
9	LDEF	Inductor value to be provided to the inductor instance if no instance parameters are specified	0	H
10	TNOM	Nominal temperature	27	°C

a. Instead of using the **POLY** keyword, you can also specify parameters **VC1=value** and **VC2=value**. **VC1** is equivalent to the first parameter of the polynomial array, **VC2** is equivalent to the second parameter of the polynomial array.



**Tip:** For detailed information on usage of **POLY**, **REVSP**, and **POLYV** parameters, see [page 4-9](#) of this manual.

## Examples

The following specifies a 0.5 pH inductor placed between the nodes 1 and s, using the **.MODEL** command to define the parameters for a Monte Carlo analysis.

```
*MODEL definition
.model lmodel ind lot=2%
...
*main circuit
11 1 s lmodel 0.5ph
```

In the following, the **scalel** parameter multiplies the inductance value given in the instantiation. Therefore this inductor will have a value of 6 ph.

```
*MODEL definition
.model lmodel ind scalel=3
...
*main circuit
11 1 s lmodel 2ph
```

In the following, the value of L1 will be 1uH. This shows how the inductor **.MODEL** card provides the value to the inductor instance when no instance parameters are specified,

```
.MODEL foo IND LDEF=1u
L1 1 2 MOD=foo
```

## Coupled Inductor

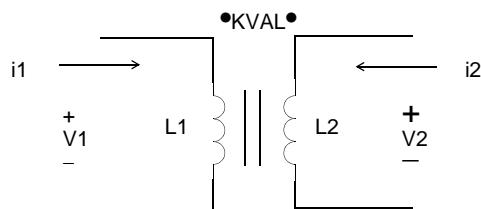
**KXX LYY LZZ KVAL**

### Parameters

- **XX**  
Name of the coupled inductor.
- **YY, ZZ**  
Names of the two inductors building **KXX**.
- **KVAL**  
Coupling coefficient where usual value is  $0 < \text{KVAL} < 1$ . All other values are also accepted by Eldo, however the simulator will give the warning message:

unusual coefficient ... <KVAL>

**Figure 4-1. Coupled Inductor**



### Example

```
17 4 3 0.7m
18 5 9 0.4m
k12 17 18 0.2
```

Specifies the coupling of inductors **17** and **18**, with a coupling coefficient of 0.2. The coupling element, or transformer, is given the symbol **k12**.

## RC Wire

```
RXX N1 N2 MNAME [[R=]VAL] [TC1=VAL] [TC2=VAL] [C=VAL] [CRATIO=VAL]
+ [L=VAL] [W=VAL] [M=VAL] [T[EMP]=VAL] [DTEMP=VAL] [SCALE=VAL]
```

### Parameters

- **xx**  
RC wire name.
- **N1, N2**  
Names of the RC wire nodes.
- **MNAME**  
Model name.
- **[R=]VAL**  
Resistance value in  $\Omega$ . Default is the value of **RES** as specified in the **.MODEL** statement. Specifying **R=** is optional, the value can be specified without it.
- **TC1=VAL**  
First order temperature coefficient of the resistance in  $^{\circ}\text{C}^{-1}$ . Default is the value of **TC1R** as specified in the **.MODEL** statement.
- **TC2=VAL**  
Second order temperature coefficient of the resistance in  $^{\circ}\text{C}^{-2}$ . Default is the value of **TC2R** as specified in the **.MODEL** statement.
- **C=VAL**  
Capacitance connected from node **N2** to BULK in Farads. See the “[RC Wire Model Syntax](#)” on page 4-29. Default is the value of **CAP** as specified in the **.MODEL** statement. The **C** value is split between the two pins of the resistor. The model parameter **CRATIO** controls the splitting.
- **CRATIO=VAL**  
Controls the splitting of the **C** value between the two pins of the resistor. **C $\times$ CRATIO** is the value which will be attached to node **N1**, while **C $\times$ (1.0 – CRATIO)** will be attached to node **N2**. Default is 0.0. This parameter cannot be specified for an instance. It can only be specified on the **.MODEL** statement.
- **L=VAL**  
Resistor length in meters. Default is the value of **L** as specified in the **.MODEL** statement.
- **W=VAL**  
Resistor width in meters. Default is the value of **W** as specified in the **.MODEL** statement.
- **M=VAL**  
Multiplier to simulate parallel resistors. Default is 1.

The device is first evaluated without the **M** factor, and at the very end of the device computation, all scaling quantities are multiplied / divided by **M**. Input values **W** and **L** are not affected. Models are chosen depending on input **W** and **L**, if required. Options **MINL**, **MAXL**, **MINW**, **MAXW**, etc. do not apply either, since they check the input values of **W** and **L**.

---

**Note**

Using an M factor value less than 1 could lead to simulating devices that cannot be physically realized.

---

- **T [EMP ]=VAL**

Sets temperature for the individual device, in degrees Celsius. Default nominal temperature=27°C.

- **DTEMP=VAL**

Temperature difference between the device and the rest of the circuit, in degrees Celsius. Default value is 0.0.

---

**Note**

**TEMP** and **DTEMP** are mutually exclusive. If both are specified, the last one is utilized.

---

- **SCALE=VAL**

Element scale factor for resistance and capacitance. Default is 1.

---

**Note**

In order to select the RC Wire model, the **LEVEL** parameter must be set to **LEVEL=3** in the model **MNAME**.

The model name **MNAME** is mandatory for the RC wire model. At least one parameter must be specified in this model. If this is not the case, Eldo interprets the definition as a normal resistor.

The same model name **MNAME** cannot be used for both a normal resistor model and an RC Wire model.

---

## Example

```
r1 n3 n4 mod1 tc1=0.001
```

Specifies an RC wire placed between nodes **n3** and **n4**. The first order temperature coefficient is defined to be 0.001.

## RC Wire Model Syntax

```
.MODEL MNAME R[ES] [PAR=VAL]
```

The RC wire model can be made to behave like a simple resistor or an elementary transmission line. This is achieved by specifying an optional capacitor from node n2 to BULK or ground. The bulk node functions as a ground plane for the wire capacitance. The wire is described by a drawn length and width. The resistance of the wire is the effective length multiplied by the **RSH** parameter then divided by the effective width.

**Table 4-7. RC Wire Model Parameters**

Nr.	Name	Description	Default	Units
1	LEVEL	Must be set to 3 to select RC WIRE model		
2	RDEF	Value of resistor. This value has priority over values computed from L and W	0	Ω
3	POLY <sup>a</sup>	Keyword to identify the resistor as non-linear polynomial		
4	POLYV	Used to change the formula used for computing the R value	2	
5	REVSP	Used to change the formula used for computing the R value	0	
6	BULK	Default reference node for capacitance	0	
7	CAP	Default capacitance	0	F
8	CAPSW	Sidewall fringing capacitance	0	Fm <sup>-1</sup>
9	COX	Bottomwall capacitance	0	Fm <sup>-2</sup>
10	DI	Relative dielectric constant	0	
11	DLR	Difference between drawn length and actual length	0	m
12	DW	Difference between drawn width and actual width	0	m
13	L	Default length of wire	0	m
14	RES	Default resistance	0	Ω
15	RSH	Sheet resistance/square	0	
16	SHRINK	Shrink factor	1	
17	TC1C	1st order temperature coefficient for capacitance	0	°C <sup>-1</sup>
18	TC2C	2nd order temperature coefficient for capacitance	0	°C <sup>-2</sup>
19	TC1R	1st order temperature coefficient for resistance	0	°C <sup>-1</sup>
20	TC2R	2nd order temperature coefficient for resistance	0	°C <sup>-2</sup>
21	THICK	Dielectric thickness	0	m
22	W	Default width of wire	0	m
23	KF	Flicker noise coefficient	0.0	

**Table 4-7. RC Wire Model Parameters**

Nr.	Name	Description	Default	Units
24	AF	Flicker noise exponents	1.0	
25	WEEXP		0.0	
26	LEEXP		0.0	
27	FEXP		1.0	

a. Instead of using the **POLY** keyword, you can also specify parameters **VC1=value** and **VC2=value**. **VC1** is equivalent to the first parameter of the polynomial array, **VC2** is equivalent to the second parameter of the polynomial array.

## Calculation of Wire Resistance

The element width and length are scaled by the parameters **SCALE** and **SHRINK**. The model width and length are scaled by the parameters **SCALM=x** (.OPTION command) and **SHRINK**. The effective width and length are calculated from the following equations:

$$W_{\text{eff}} = W_{\text{scaled}} - (2 \times DW_{\text{eff}})$$

$$L_{\text{eff}} = L_{\text{scaled}} - (2 \times DLR_{\text{eff}})$$

where:

$$DW_{\text{eff}} = DW \times SCALM$$

$$DLR_{\text{eff}} = DLR \times SCALM$$

$$L_{\text{scaled}} = L \times SHRINK \times SCALM$$

$$W_{\text{scaled}} = W \times SHRINK \times SCALM$$

If element resistance **R** is specified:

$$R_{\text{eff}} = \frac{R \times \text{SCALE (element)}}{M}$$

Otherwise, if  $(W_{\text{eff}} \times L_{\text{eff}} \times RSH)$  is greater than zero, then:

$$R_{\text{eff}} = \frac{L_{\text{eff}} \times RSH \times \text{SCALE (element)}}{M \times W_{\text{eff}}}$$

If  $(W_{\text{eff}} \times L_{\text{eff}} \times RSH)$  is zero, then:

$$R_{\text{eff}} = \frac{\text{RES} \times \text{SCALE (element)}}{M}$$

If **AC** resistance is specified in the element, then:

$$R_{\text{ACeff}} = \frac{AC \times \text{SCALE (element)}}{M}$$

Otherwise, if **RAC** is specified in the model, **RAC** is used:

$$RAC_{eff} = \frac{RAC \times SCALE(element)}{M}$$

If neither are specified, it defaults to:

$$RAC_{eff} = Reff$$

If the resistance is less than **resmin**, it is reset to **resmin** and a warning message is issued.

$$resmin = \frac{1}{GMAX \times 1000 \times M}$$

---

**Note**

 **REVSP** and **POLYV** usage:

If parameter **REVSP** is set to 1 on the **.MODEL** card, and if **POLYV** is not 2, then the formula for computing the current generated by the Resistor is:

$$I = V / <rinst> * (1 + P1/2 + P2/3 * V * V ...)$$

which leads to a Resistor of value:

$$R(V) = 1/(dI/dV) = <rinst> / (1 + P1.V + P2.V*V)$$

---

## Calculation of Wire Capacitance

The effective length is the scaled drawn length less  $2 \times DW_{eff}$ .  $L_{eff}$  represents the effective length of the resistor from physical edge to physical edge.  $DW_{eff}$  is the distance from the drawn edge of the resistor to the physical edge of the resistor. The effective width is the same as the width used in the resistor calculation.

$$L_{eff} = L_{scaled} - (2 \times DW_{eff})$$

$$W_{eff} = W_{scaled} - (2 \times DW_{eff})$$

If the element capacitance **C** is specified:

$$CAP_{eff} = C \times SCALE \times M$$

Otherwise, the capacitance is selected from  $L_{eff}$ ,  $W_{eff}$  and **COX**.

$$AP_{eff} = M \times SCALE \times [L_{eff} \times W_{eff} \times COX + 2.0 \times (W_{eff} + L_{eff}) \times CAPSW]$$

If **COX** is not specified, but the model parameter **THICK** is not zero, then:

$$COX = \frac{DI \times \epsilon_0}{THICK}$$

if **DI** is not zero or:

$$COX = \frac{\epsilon_{ox}}{THICK}$$

if  $DI = 0$ , where:

$$\epsilon_0 = 8.8542149 \times 10^{-12} \frac{F}{meter}$$

$$\epsilon_{ox} = 3.453148 \times 10^{-11} \frac{F}{meter}$$

If only model capacitance **CAP** is specified:

$$CAP_{eff} = CAP \times SCALE \times M$$

## Example

```
.model rmodel r cap=0.5p tc1c=0.01 tc2c=0.05
...
r2 n1 n2 r rmodel tc1=0.05
```

Specifies an RC wire placed between nodes n1 and n2 of model type rmodel. Electrical parameters of the model are defined via the **.MODEL** command.

## Semiconductor Resistor

**Pxx N1 N2 NS MNAME [R=VAL] [L=VAL] [CL=VAL] [W=VAL] [CW=VAL] [AREA=VAL]**

### Parameters

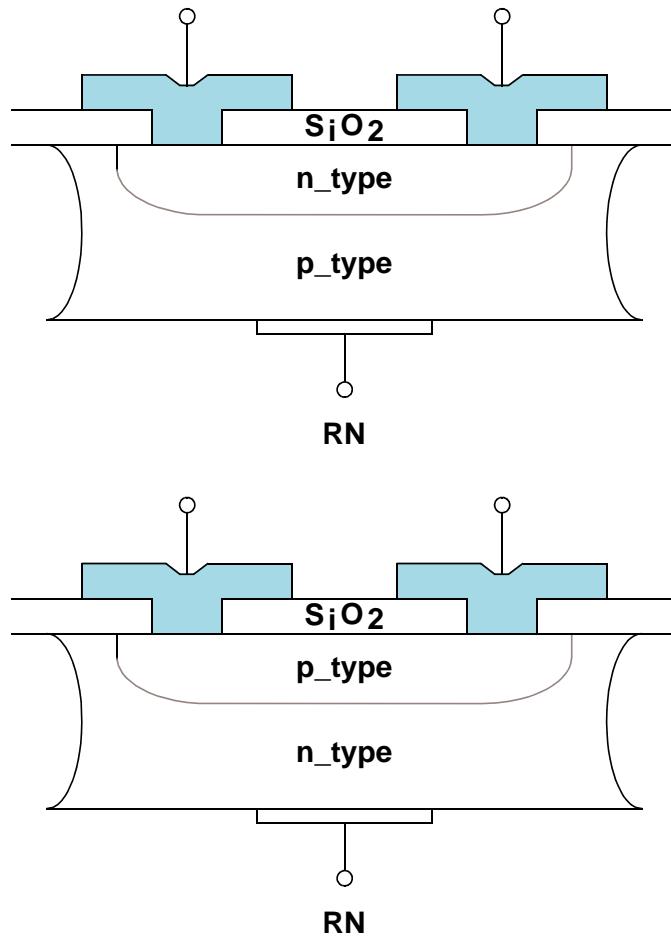
- **xx**  
Semiconductor resistor name.
- **N1, N2, NS**  
Names of the semiconductor resistor nodes.
- **MNAME**  
Model name.
- **R=VAL**  
Specifies the resistance. (If **R=0**, or if **R** is not specified, then automatic calculations are used.)
- **L=VAL**  
Specifies the resistor length. Default is  $10\mu\text{m}$ .
- **CL=VAL**  
Specifies the contact offset length. Default is  $2\mu\text{m}$ .
- **W=VAL**  
Specifies the resistor width. Default is  $2\mu\text{m}$  (**DEFW** from **.MODEL** statement).
- **CW=VAL**  
Specifies the contact offset width. Default is  $0\mu\text{m}$ .
- **AREA=VAL**  
Specifies the area of the resistor.

## Semiconductor Resistor Model Syntax

**.MODEL MNAME TYPE [ PNAME=PVAL ]**

- **TYPE**  
This must be either **RN** or **RP**.

The figure below illustrates the cross-sectional view of a diffused semiconductor resistor.

**Figure 4-2. Cross-Sectional View of Diffused Resistor****Table 4-8. Semiconductor Resistor Model Parameters**

Nr.	Name	Description	Default	Units
1	TC1	First temperature coefficient of bulk	0.0	$\Omega^{\circ}\text{C}^{-1}$
2	TC2	Second temperature coefficient of bulk	0.0	$\Omega^{\circ}\text{C}^{-1}$
3	CTC1	First temperature coefficient of contact hole	0.0	$\Omega^{\circ}\text{C}^{-1}$
4	CTC2	Second temperature coefficient of contact hole	0.0	$\Omega^{\circ}\text{C}^{-1}$
5	RSH	Sheet resistance of bulk	10	$\Omega/\text{sq}$
6	CRSH	Sheet resistance of contact hole	10	$\Omega/\text{sq}$
7	A1	Contact coefficient 1	1.0	
8	B1	Contact coefficient 2	1.0	
9	DEFW	Default width	$2 \times 10^{-6}$	m

**Table 4-8. Semiconductor Resistor Model Parameters**

Nr.	Name	Description	Default	Units
10	NARROW	Narrowing due to side etching	0.0	m
11	IS	Saturation current	$1 \times 10^{-14}$	A
12	RS	Ohmic resistance	0.0	$\Omega$
13	M	Grading coefficient	0.5	
14	N	Emission coefficient	1.0	
15	TT	Transition time	0.0	s
16	CJO	Zero bias junction capacitance	0.0	F
17	PB	Junction built-in potential	1.0	V
18	EG	Energy gap	1.1	eV
19	XTI (PT)	Saturation current temperature exponent	3.0	
20	KF	Flicker noise coefficient	0.0	
21	AF	Flicker noise exponent	1.0	
22	FC	Forward bias non-ideal junction capacitance coefficient	0.5	
23	BV	Reverse breakdown voltage	•	V
24	IBV	Current at <b>BV</b>	$1 \times 10^{-3}$	A

## Semiconductor Resistor Model Equations

### Resistance R calculation

$$R = R_{BLK} + 2RC$$

$$R = RB0 + RB1 + 2RC = R(TNOM)$$

### Contact Resistance RC

$$RC = CRSH \cdot \frac{A_1}{W^{B_1}} = RC(TNOM)$$

### Bulk Resistance

$$R_{BLK} = RB0 + RB1$$

- Resistance for bulk part

$$RB0 = \frac{L - 2CL - \text{NARROW}}{W - \text{NARROW} + 2CW} \cdot RSH = RB0(TNOM)$$

- Resistance near the contact

$$RB1 = \frac{2CL - \text{NARROW}}{W - \text{NARROW} + 2CW} \cdot CRS = RB1(TNOM)$$

## Temperature Effects

$$R(TEMP) = RB0(TEMP) + RB1(TEMP) + 2RC$$

$$RB0(TEMP) = RB0(TNOM) \cdot [1 + TC1 \cdot (TEMP - TNOM) + TC2 \cdot (TEMP - TNOM)^2]$$

$$RB1(TEMP) = RB1(TNOM) \cdot [1 + CTC1 \cdot (TEMP - TNOM) + CTC2 \cdot (TEMP - TNOM)^2]$$

- Noise Model Sources

$$i_{nRs} = \left[ \frac{4kT}{R} \right]^{\frac{1}{2}}$$

## Diode Model Equations

$$T = TEMP + 273.15$$

$$T_0 = TNOM + 273.15$$

## Internal DC Model Parameters

$$\text{RATIO} = \frac{T - T_0}{T_0}$$

$$q = 1.6 \times 10^{-19} \quad k = 1.38 \times 10^{-23} \text{ J}^\circ\text{K}^{-1}$$

$$V_t = \frac{kT}{q} \quad V_{t0} = \frac{kT_0}{q}$$

$$I_s = IS \cdot \text{AREA} \cdot \left[ \frac{T}{T_0} \right]^{\text{PT}} \left[ e^{\left[ \frac{(EG)(\text{RATIO})}{V_t} \right]} \right]$$

$$I_{bv} = IBV \cdot \text{AREA}$$

$$V_{CRIT} = V_{t0} \ln \left[ \frac{V_{t0}}{1.414 \cdot IS} \right]$$

$V_{CRIT}$  is used as a starting point for DCOP.

## Internal Charge Related Model Parameters

$$Eg = 1.16 - 0.000702 \left[ \frac{T^2}{T + 1108} \right]$$

$$Eg_{ref} = 1.1150877$$

$$V_{ref} = Vt(T) \cdot \left( \frac{EG(TNOM)}{Vt(TNOM)} - \frac{EG(T)}{Vt(T)} + 3 \ln \frac{T}{TNOM} \right)$$

$$P_b = PB \left[ \frac{T}{T_0} \right] - V_{ref}$$

$$C_j = CJO \cdot AREA \cdot \left[ 1 + M \left\{ 0.0004(T - T_0) + \left[ 1 - \left( \frac{P_b}{PB} \right) \right] \right\} \right]$$

## DC Current Source Definitions

$V_d$  = (Internal Anode Voltage) – (Internal Cathode Voltage)

When  $BV$  is not specified by the user:

- If  $V_d \geq -5N \cdot V_t$

$$I_d = I_s \left[ e^{\left[ \frac{V_d}{(N)(V_t)} \right]} - 1 \right] + (GMIN)V_d$$

- If  $V_d < -5N \cdot V_t$

$$I_d = I_s (e^{-5} - 1) + \left[ \frac{I_s}{(N)(V_t)} \right] e^{-5} [V_d + 5N \cdot V_t] + (GMIN)V_d$$

When  $BV$  is specified by the user:

- If  $V_d \geq -5N \cdot V_t$

$$I_d = I_s \left[ e^{\left[ \frac{V_d}{(N)(V_t)} \right]} - 1 \right] + (GMIN)V_d$$

$X_{bv}$  is an internally generated voltage which facilitates the continuous representation of diode current.

- If  $X_{bv} \leq V_d < -5N \cdot V_t$

$$I_d = I_s(e^{-5} - 1) + \left[ \frac{I_s}{N \cdot V_t} \right] e^{-5} [V_d + 5N \cdot V_t] + (GMIN)V_d$$

- If  $V_d < X_{bv}$

$$I_d = I_s(e^{-5} - 1) + \left[ \frac{I_s}{N \cdot V_t} \right] e^{-5} [V_d + 5N \cdot V_t]$$

$$- e^{-5} (I_s) \left[ e^{\left[ \frac{-(V_d - X_{bv})}{N \cdot V_t} \right]} - 1 \right] + (GMIN)V_d$$

## Diode Diffusion Capacitance

$$C_{d2} = TT \left[ \frac{d(I_d)}{d(V_d)} \right]$$

## Diode Depletion Capacitance

- If  $V_d < FC \cdot P_b$

$$CDS = CS0 \cdot \left( 1 - \frac{V_d}{P_b} \right)^{-M}$$

$$\text{where } CS0 = \frac{1}{2} \cdot \frac{R - 2RC}{RSH} \cdot W^2 \cdot CJ0$$

- If  $V_d \geq FC \cdot P_b$

$$C_{d1} = C_j \left[ (1 - FC)^{-M} \left[ \frac{M(1 - FC)^{-M}}{P_b(1 - FC)} \right] (V_d - FC \cdot P_b) \right]$$

## Small Signal Model Equations

$$C_d = C_{d1} + C_{d2}$$

$$g_d = \frac{d(I_d)}{d(V_d)}$$

$$r_d = \frac{1}{g_d}$$

## Diode Model Noise Source Equations

- Resistance Generated Noise

$$i_{nrs} = \left[ \frac{4kT}{R_s} \right]^{\frac{1}{2}}$$

- Current Source Generated Shot and Flicker Noise

$$i_{nF} = \left[ \frac{KF \cdot I_d^{AF}}{R_s} \right]^{\frac{1}{2}}$$

$$i_{ns} = (2qI_d)^{\frac{1}{2}}$$

## Transmission Line

```
TXX NAP NAN NBP NBN [Z0=VAL1] TD=VAL2
TXX NAP NAN NBP NBN [Z0=VAL1] F=VAL3 [NL=VAL4]
```



For Lossy dispersive transmission lines, please refer to the LDTL description on [page 4-43](#).

### Parameters

- **XX**  
Transmission line name.
- **NAP**  
Positive A terminal of the transmission line.
- **NAN**  
Negative A terminal of the transmission line.
- **NBP**  
Positive B terminal of the transmission line.
- **NBN**  
Negative B terminal of the transmission line.
- **TD=VAL2**  
Transmission delay in seconds.
- **F=VAL3**  
Frequency in Hertz.
- **Z<sub>0</sub>=VAL1**  
Characteristic impedance in  $\Omega$ . Default value is  $50\Omega$ . Optional.
- **NL=VAL4**  
Number of wavelengths at frequency **F** required for a wave to propagate down the line. Default value is 0.25. Optional.

The above parameters can also be assigned via the **.PARAM** command.

The relationship between the **TD**, **F** and **NL** values is given by:

$$TD = \frac{1}{F} \times NL$$

### Examples

```
t1 1 2 3 4 zo=220 td=111ns
```

Specifies a transmission line **t1** between nodes 1 (+ve A) and 2 (-ve A) and nodes 3 (+ve B) and 4 (-ve B), with a characteristic impedance of  $220\Omega$ . The transmission delay of the line is 111ns.

```
t2 1 2 3 4 zo=220 f=2.25meg
```

Specifies the same transmission line as above, but in terms of a frequency for a quarter wavelength (as **NL** defaults to 0.25).

```
t3 1 2 3 4 zo=220 f=4.5meg nl=0.5
```

Specifies the same transmission line as above, but in terms of a frequency for half a wavelength.

## Lossy Transmission Line

LDTL (Lossy Dispersive Transmission Line) is a model implemented in Eldo to simulate transmission lines. It is dedicated for the simulation of lossy coupled uniform lines, also including dispersive effects. This model can be used in all analysis modes (DC, AC, Transient, SST, SSTNOISE, or MODSST). To specify the line parameters to the LDTL model, four different inputs are available:

1. the first level corresponds to R, L, C and G matrices,
2. the second level uses a file at XFX output format as input to specify the line parameters,
3. the third level corresponds to the electrical parameters for a single line,
4. the fourth level corresponds to the geometrical and physical parameters for a single stripline or up to two coupled microstrip lines.

Ways of instantiation are shown on the following pages. The user can also instantiate an LDTL model by using a model of MODFAS type (see [page 10-160](#)) directly in the parameter list. This model must be specified at the end of the instantiation line and can contain any parameter. For each level, an example is provided.

### Level 1

```
YXX LDTL [PIN:] P1...PN [REFin] PN+1...P2N REFout
+ [PARAM:] [LEVEL=1] [LENGTH=val] [SAVEFIT=val] [M=VAL]
+ [R(i)=val] [L(i,j)=val] [C(i,j)=val] [G(i,j)=val] [FR1=val]
```

The first level is dedicated to the simulation of an infinite number of coupled transmission lines. The Maxwell matrices (R, L, C and G) are used to describe the line system.

### Parameters

- **XX**  
Transmission line name.
- **P1...PN**  
The N nodes at one end of the line system for a system consisting of N lines.
- **REFin**  
Optional reference node for input signal, used to simulate differential lines.
- **PN+1...P2N**  
The N nodes at the other end of the line system. The line number i in the line system connects the nodes Pi and PN+i.
- **REFout**  
Reference node for output signal. If `REFin` is not specified, then both sides of the line system have the same reference plane.

- **LENGTH=val**  
Geometric length of the line system. Default value is 1 m. If **LENGTH=0**, Eldo uses the default value.
- **LEVEL=1**  
Keyword to specify the input format. Value 1 specifies the “R,L,G,C matrices” format. Default value is 1.
- **SAVEFIT=val**  
**SAVEFIT=1** ⇒ Saves the initialization of the transmission line model (in the file *circuit\_name.fit*), in order to speed up the following simulations of the same netlist. Default value is 0.
- **M=VAL**  
Device multiplier. Simulates the effect of multiple devices in parallel. In effect the current value is multiplied by **M**. Default is 1. **.OPTION YMFACT** must be selected in order for this option to work.
- **R(i)=val**  
Value of the (i,i) element per unit length of the resistance matrix: R. Default values are 50  $\Omega\text{m}^{-1}$ .
- **L(i,j)=val**  
Value of the (i,j) element per unit length of the inductance matrix: L. Default values  $1 \times 10^{-6}\text{Hm}^{-1}$  for the self inductance and 0 for the mutual inductances.
- **C(i,j)=val**  
Value of the (i,j) element per unit length of the capacitance matrix: C. Default values  $1 \times 10^{-9}\text{Fm}^{-1}$  for the self capacitance and 0 for the mutual capacitances.
- **G(i,j)=val**  
Value of the (i,j) element per unit length of the conductance matrix: G. Default values are 0  $\text{Sm}^{-1}$ .
- **FR1=val**  
Frequency at which dispersion starts (only affects resistance). Default: no dispersion will be considered.

If for a line only **R(1)** is specified, this value is used for all lines. Specification of the Transmission line matrix parameter can be done in one of the following ways:

- Complete matrix of coefficients consisting of  $N \times N$  values.
- Only the upper (or lower) triangular matrix because of the matrix symmetry.
- Only the first row (or column) of the matrix. This is normally sufficient if all lines in the system have the same width and spacing.

## Example

Three coupled lines defined by R, L, C and G matrices:

```

Y1 LDTL 2 3 4 0 5 6 7 0
+ param: LEVEL=1 length=0.677
+ R(1)=15 L(1,1)=418n C(1,1)=94p G(1,1)=0.02p
+ R(2)=15 L(2,2)=418n C(2,2)=94p G(2,2)=0.02p
+ C(1,2)=-22p C(2,3)=-22p
+ L(1,2)=125n L(2,3)=125n
+ R(3)=15 L(3,3)=418n C(3,3)=94p G(3,3)=0.02p

```

Same example using a **.MODEL** in the instantiation:

```

Y1 LDTL 2 3 4 0 5 6 7 0
+ param: LEVEL=1 length=0.677
+ model:level1_mod

.model level1_mod MODFAS R(1)=15
+ L(1,1)=418n C(1,1)=94p G(1,1)=0.02p
+ R(2)=15 L(2,2)=418n C(2,2)=94p G(2,2)=0.02p
+ C(1,2)=-22p C(2,3)=-22p
+ L(1,2)=125n L(2,3)=125n
+ R(3)=15 L(3,3)=418n C(3,3)=94p G(3,3)=0.02p

```

## Level 2

```

YXX LDTL [PIN:] P1...PN [REFin] PN+1...P2N REFout
+ [PARAM:] LEVEL=2 [LENGTH=val] [SAVEFIT=val]
+ [XFX_IDF=val] [FP=val] [MULTIDEBYE=val]

```

The second level uses a file at XFX output format as input to specify the line parameters. This level is dedicated to the simulation of an infinite number of coupled transmission lines.

## Parameters

The description of the global parameters (PINS, LENGTH and SAVEFIT) is as specified for the **LEVEL=1** format. Level 2 specific parameters are shown below.

- **LEVEL=2**  
Keyword to specify the input format. Value 2 specifies the XFX format. Default value is 1.
- **XFX\_IDF=val**  
Index relative to the XFX file name. If XFX\_IDF=val, Eldo will search for the file *XFX\_val.tlp*.
- **FP=val**  
Polarization frequency to control dispersive effect on the conductance (see “[Technical precision](#)” on page 4-51). Default:  $1.6 \times 10^9$ .

- **MULTIDEBYE=val**

`val=1` specifies the use of multi-pole debye model to model the dispersive effect on the conductance (recommended for modeling PCB-type dielectrics). `val=0`, this model is not used. (see “[Technical precision](#)” on page 4-51). Default: 0.

## Example

Two coupled lines defined by an XFX output file.

```
Y1 LDTL 1 2 3 4 0
+ param: LEVEL=2 length=0.1 xfx_idx=12
```

Same example using a **.MODEL** in the instantiation:

```
Y1 LDTL 1 2 3 4 0
+ param: model:level2_mod
.model level2_mod MODFAS LEVEL=2 length=0.1 xfx_idx=12
```

The parameter `xfx_idx=12` is a reference to the file *XFX\_12.tlp* containing the line information generated by XFX. Here is an example of such a file:

```
XFX V6.0.0.0 Report           5 Nov 15:57 1997      Setup File=ex3.xfx
Configuration Name: ANALOG1 Conductors: 2
Conductor index: 0 name: $$GND$$
Conductor index: 1 name: A
Conductor index: 2 name: C

    i      j      Lij      Cij      Ze      Zo      Se      So      Fwdx      Rvsx
from to    (nh/in) (pf/in) (ohms) (ohms) (ns/ft) (ns/ft) (s/s) (v/v)
-----
1     1     8.631     3.742     48.02      -     2.16      -      -      -
1     2     6.87e-10   2.98e-10   48.02    48.02     2.16     2.16    0.000    0.000
2     2     8.631     3.742     48.02      -     2.16      -      -      -

:      LOSS MATRICES
    i      j      Rsij      Gij      Rdci,j      Gdcij
from to    (ohm-nsec^.5) (mS-ns) (ohms) (mS) PER INCH
-----
1     1       0.7       0.0     0.47929    0.46827
1     2       0.00      0.0     0.00000    0.02912
2     2       0.7       0.0     0.34473    0.45333
;
```

## Level 3

```
YXX LDTL [PIN:] P1 [REFin] P2 REFout
+ [PARAM:] LEVEL=3 [LENGTH=val] [SAVEFIT=val]
+ [Zc=val] [VREL=val] [TD=val] [L=val] [C=val] [R=val] [FR1=val] [M=val]
```

The third level corresponds to the electrical parameters for a single line only.

## Parameters

The description of the global parameters (**PINS**, **LENGTH** and **SAVEFIT**) is as specified for the **LEVEL=1** format. Level 3 specific parameters are shown below.

- **LEVEL=3**  
Keyword to specify the input format. Value 3 specifies the electrical format. Default value is 1.
- **Zc=val**  
Characteristic impedance ( $\Omega$ ). If this value is not specified, it is calculated with the values of L and C.
- **VREL=val**  
Relative velocity. If this value is not specified, it is calculated with the values of L and C.
- **TD=val**  
Delay for **LENGTH** (implies total delay calculated is **LENGTH×TD**). If not specified, calculated.
- **L=val**  
Inductance per unit length. Default value:  $1 \times 10^{-6} \text{Hm}^{-1}$ .
- **C=val**  
Capacitance per unit length. Default value is:  $1 \times 10^{-9} \text{Fm}^{-1}$ .
- **R=val**  
Linear resistance. Default value is  $50 \Omega$ .
- **FR1=val**  
Frequency (Hz) at which dispersion starts (only affects resistance). Default: no dispersion will be considered.
- **M=VAL**  
Device multiplier. Simulates the effect of multiple devices in parallel. In effect the current value is multiplied by **M**. Default is 1. **.OPTION YMFAC**T must be selected in order for this option to work.

You can either specify directly the line parameters **R**, **L** and **C**, or use any combination of electrical parameters (L and C can be computed with electrical parameters, see “[Technical precision](#)” on page 4-51).

## Example

One dispersive line defined by electrical parameters.

```
Y1 ldtl 1 2 0
+ param: LEVEL=3 length=100 R=1
+ ZC=50 VREL=0.66 FR1=100Meg
```

Same example using a **.MODEL** in the instantiation:

```
Y1 ldtl 1 2 0
+ param: LEVEL=3 length=100 R=1
+ ZC=50 model: level3_mod

.model leve3_mod MODFAS VREL=0.66 FR1=100Meg
```

## Level 4

```
YXX LDTL [PIN:] P1 [REFin] P2 REFout
+ [PARAM:] LEVEL=4 [LENGTH=val] [SAVEFIT=val]
+ [DLEV=val] [PLEV=val] [ER=val] [H=val] [W=val] [T=val]
+ [RHO=val] [TAND=val] [H1=val] [FP=val] [H2=val] [S=val]
+ [THICKNESS=val] [DISPERSIVE=val] [USE_ER=val] [M=val] [MULTIDEBYE=val]
```

The fourth level corresponds to the geometrical parameters specification for microstrip line and stripline. Up to two coupled microstrip lines are allowed and only one single stripline can be taken into account. The structure of these transmission lines is illustrated in [Figure 4-4](#) (coupled pair of microstrip lines) and [Figure 4-5](#) (stripline).

## Parameters

The description of the global parameters (**PINs**, **LENGTH** and **SAVEFIT**) is as specified for the **LEVEL=1** format. Level 4 specific parameters are shown below.

- **LEVEL=4**  
Keyword to specify the input format. Value 4 specifies the geometrical and physical format. Default value is 1.
- **DLEV=val**  
Type of line: 1 for microstripline; 2 for stripline. Default is 1.
- **PLEV=val**  
Type of equations: 0 uses the equations from the references (1) and (2), 1 for simplified equations. Default is 0.
- **ER=val**  
Dielectric relative permittivity. Default value is 9.8 (alumina).
- **H=val**  
Dielectric thickness (m). Default value is  $400 \times 10^{-6}$  m.
- **W=val**  
Conductor width. Default value is  $50 \times 10^{-6}$  m.
- **T=val**  
Conductor thickness. Default value is  $5 \times 10^{-6}$  m.

- **RHO=val**  
Conductor resistivity. Default value is  $17 \times 10^{-9}$  Ωm (copper).
- **TAND=val**  
Dielectric loss tangent. Default value is 0.
- **H1=val**  
Conductor height, only for stripline configuration. Default value is  $197.5 \times 10^{-6}$  m.
- **FP=val**  
Polarization frequency to control dispersive effect on the conductance (see “[Technical precision](#)” on page 4-51). Default:  $1.6 \times 10^9$ .
- **H2=val**  
Height between substrate and a possible cover plate. Only for coupled microstrip configuration. Default: 0.0, means that no cover plate is taken into account.
- **S=val**  
Spacing between the two conductors. Default = conductor width value (W). Only for coupled microstrip configuration.
- **THICKNESS=val**  
Take into account effect of finite strip thickness if `val=1`. Default: 0. Only for coupled microstrip configuration.
- **DISPERSIVE=val**  
Take into account dispersive effect if `val=1`. Default: 0. Only for coupled microstrip configuration.
- **USE\_ER=val**  
Use directly the dielectric relative permittivity (ER) to compute the characteristic impedance ( $Z_c$ ) if `val=1`. Otherwise (if `val=0`), an effective relative permittivity will be calculated and used in the  $Z_c$  computation (see Technical Precision, “[LEVEL 4](#)” on page 4-53 for details). Default: 1.
- **M=VAL**  
Device multiplier. Simulates the effect of multiple devices in parallel. In effect the current value is multiplied by **M**. Default is 1. **.OPTION YMFACT** must be selected in order for this option to work. Only used for stripline (**DLEV=2**).

---

#### Note



When **PLEV=0** and **DLEV=2**, it is only possible to describe an off-centered microstrip line with the equations based on reference [1].

---

- **MULTIDEBYE=val**

`val=1` specifies the use of multi-pole debye model to model the dispersive effect on the conductance (recommended for modeling PCB-type dielectrics). `val=0`, this model is not used. (see “[Technical precision](#)” on page 4-51). Default: 0.

## Examples

A microstrip line based on equations from reference [1].

```
Y1 LDTL 1 2 0
+ param: LEVEL=4 length=10 PLEV=0 DLEV=1
+ h=400u w=50u t=5u rho=17E-09 er=9.8
```

Symmetric pair of coupled microstrip lines, including finite strip thickness and dispersive effects.

```
Y1 LDTL 1 2 0 3 4 0
+ param: LEVEL=4 length=10e-3 PLEV=1 DLEV=1
+ h=635u w=88u t=2u s=90u h2=935u
+ rho=1.72E-08 tand=0.01 thickness=1
+ dispersive=1
```

Same example using a **.MODEL** in the instantiation:

```
Y1 LDTL 1 2 0 3 4 0
+ param: LEVEL=4 length=10 PLEV=1 DLEV=1
+ h=635u w=88u model:level4_mod

.model level4_mod MODFAS t=2u s=90u h2=935u
+ rho=1.72E-08 tand=0.01 thickness=1
+ dispersive=1
```

## Error message treatment

A general problem which causes many errors is incorrect time delay values. These values are calculated by multiplying the L and C matrices. Negative or null time delay values can cause errors in the model. To avoid the simulation being stopped, here follows some advice on how to avoid errors. The error message is shown, together with advice on avoiding this type of error.

```
ERROR model Yxx : no time-delay in the transmission line(s), check your
input parameters.
```

This means that some value(s) of the L or C matrix (or both) are bad: the diagonal of  $L \times C$  matrix presents null value(s) (see [Technical precision](#), “[General Equation for Delay](#)” on page 4-51). It can appear when some off-diagonal term(s) of the C or L matrix are too large. Normally the coupling effect on L and C decreases when the distance between the two concerned lines increase. That means:  $|C(1,2)|$  should be larger than  $|C(1,3)|$ .

```
ERROR model Yxx: Non physical line model (negative time delay). Check C
and/or L matrices off-diagonal terms.
```

This message appears only for coupled transmission line models. It means that the time-delay of at least one line of the model is negative. So this modelization is not a physical one.

This means the diagonal of  $L \times C$  matrix presents negative value(s) (see [Technical precision](#), “[General Equation for Delay](#)” on page 4-51). It can appear when some off-diagonal term(s) of the  $C$  or  $L$  matrix are too large. Normally the coupling effect on  $L$  and  $C$  decreases when the distance between the two concerned lines increase. That means:  $|C(1,2)|$  should be larger than  $|C(1,3)|$ .

```
ERROR model Yxx : the diagonal of C is non-strictly-dominant : you should
have Sum{|(C(i,j)|} < |C(i,i)| (i != j)
```

This means some off-diagonal terms of the  $C$  matrix are too large. Therefore, the sum of all the off-diagonal terms of one line of the matrix is not lower than the diagonal term of the line: the strictly-dominant property is not verified. Such a property is required for the model.

```
WARNING model Y1 : negative diagonal value(s) : R[1][1]
```

This warning means that the value  $R[1][1]$  is negative. Therefore, the user has to check the parameters of the model instantiation according to the LEVEL used (see [Technical precision](#)).

## Technical precision

Here follows some technical information about the use of the `Yxx LDTL` model.

### General Equation for Delay

The time-delay ( $Td$ ) of a single transmission line is computed as follows:

$$Td = \text{Length} \times \sqrt{LC}$$

When we have a n-coupled transmission line model, the time-delay matrix is computed as follows ( $L$ ,  $C$  and  $Tdm$  are matrices):

$$Tdm = \text{Length} \times \text{diag}(L \times C)$$

$Tdm$  is a diagonal matrix. The  $n^{ieme}$  element of the diagonal is the time-delay of the  $n^{ieme}$  line of the model. Therefore, these diagonal values must be positive.

### LEVEL 1

To introduce skin effects in the line model (loss that is proportional to the square root of frequency), you just have to specify the *FR1* parameter. Then the resistivity value will be frequency dependent:

$$R = R(i) \times \left(1 + (1+i)\sqrt{\frac{f}{FR1}}\right) \text{ for the } i^{\text{th}} \text{ transmission line.}$$

## LEVEL 2

Here the skin effect is introduced by the parameter  $R_s$ :

$$R = R_{DC} + R_s \times (1+i)\sqrt{4\pi f}$$

You can also introduce frequency dependent conductance by using the parameters  $G_s$  and  $f_p$  (polarization frequency: **FP** parameter). The conductance dispersive effect can be modeled in two ways according to the **MULTIDEBYE** parameter:

- One-pole debye model (**MULTIDEBYE=0**)

The dispersive effect is obtain according to the following equation:

$$G = G_{DC} + G_s \times \frac{i2\pi f \times 2\pi f_p}{i2\pi f + 2\pi f_p}$$

- Multi-pole debye model (**MULTIDEBYE=1**)

By using this option, we build a complex frequency-dependent capacitance matrix:

$$C(\omega) = C_{inf} + f(j\omega)C_d$$

Therefore, line conductance per unit length becomes:

$$Y(j\omega) = G_{DC} + (j\omega)C_{inf} + (j\omega)f(j\omega)C_d$$

where:  $C_{inf} = C - \alpha G_s$  and  $C_d = \beta G_s$ ;  $C$  and  $G_s$  are the user defined matrices.

$$\text{with } \alpha = \frac{\ln\left(\frac{\omega_2^2 + \omega_0^2}{\omega_1^2 + \omega_0^2}\right)}{4\pi\left(\tan\left(\frac{\omega_0}{\omega_1}\right) - \tan\left(\frac{\omega_0}{\omega_2}\right)\right)} \text{ and } \beta = \frac{\ln(10^8)}{2\pi\left(\tan\left(\frac{\omega_0}{\omega_1}\right) - \tan\left(\frac{\omega_0}{\omega_2}\right)\right)}.$$

$$\text{Finally, the function } f(j\omega) = \frac{\ln\left(\frac{\omega_2 + j\omega}{\omega_1 + j\omega}\right)}{\ln(10^8)} \text{ which is fitted with 15 real poles.}$$

Note:  $\omega_0 = 2\pi f_p$ ,  $\omega_1 = 10^4$ ,  $\omega_2 = 10^{12}$ ,  $\omega = 2\pi f$  and  $f_p$  the polarization frequency (**FP** parameter).

## LEVEL 3

As already described, you can either specify directly the line parameters **R**, **L** and **C**, or use any combination of electrical parameters. In order to discard redundant parameter sets we use the following equations:

**Table 4-9. LDTL Level 3 Parameter Combinations**

Input Parameters	Equations
Zc, VREL	$C = \frac{1.0}{Zc \times VREL \times Clight}, L = \frac{Zc}{VREL \times Clight}$
Zc, TD	$C = \frac{TD}{Zc}, L = Zc \times TD$
TD, C	$L = \frac{TD^2}{C}$
VREL, C	$L = \frac{1.0}{VREL^2 \times Clight^2 \times C}$
TD, L	$C = \frac{TD^2}{L}$
VREL, L	$C = \frac{1.0}{VREL^2 \times Clight^2 \times L}$
any other	default values for <i>L</i> and <i>C</i>

*Clight* =  $3 \times 10^8$  ms<sup>-1</sup>—the speed of light.

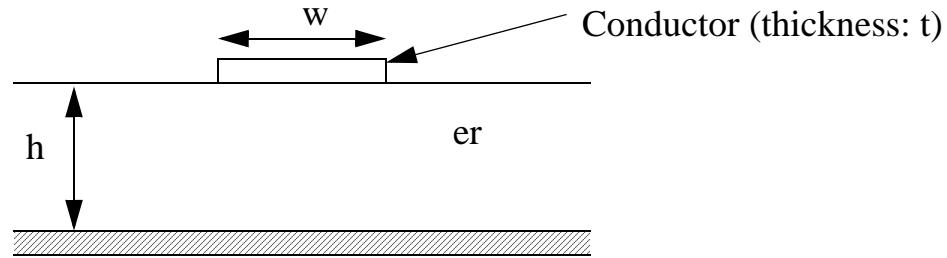
The skin effect is introduced by the parameter **FR1**:

$$R = R \times \left( 1 + (1 + i) \sqrt{\frac{f}{FR1}} \right)$$

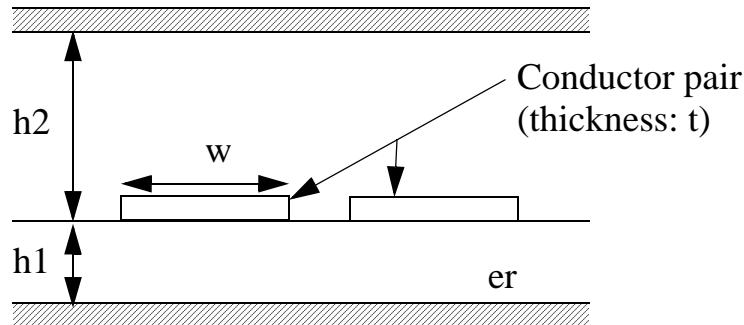
## LEVEL 4

Only two transmission line configurations can be described with this input format: microstrip line and stripline. The following figures provides the structure of these transmission lines. [Figure 4-3](#) and [Figure 4-4](#) provide the structure for a single and covered pair of microstrip models, [Figure 4-5](#) provides the structure for a stripline.

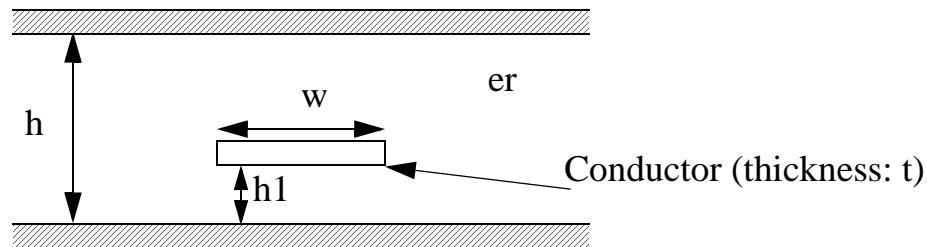
**Figure 4-3. Microstrip Line Structure**



**Figure 4-4. Covered Pair Microstrip Line Structure**



**Figure 4-5. stripline Structure**



The **PLEV** parameter allows the use of two sorts of equation: reference equations (**PLEV=0**) and simplified equations (**PLEV=1**). Provided here is the reference formulation.

### For a single microstrip line (**DLEV=1**)

- From reference (1) (**PLEV=0**), we have the following equations:

$$\text{Capacitance } C = \frac{\sqrt{\mu_0 \epsilon_0 \epsilon_r}}{Z_c}$$

$$\text{Inductance } L = Z_c \sqrt{\mu_0 \epsilon_0 \epsilon_r}$$

$$\text{Resistance } R = \frac{\rho}{w \times t}$$

with the characteristic impedance:

$$Z_C = \frac{\eta_0}{2\sqrt{2}\pi\sqrt{\epsilon_r+1}} \ln \left\{ 1 + \frac{4h}{w'} \left[ \frac{14+8/(\epsilon_r)}{11} \times \frac{4h}{w'} + \sqrt{\left( \frac{14+8/\epsilon_r}{11} \right)^2 \left( \frac{4h}{w'} \right)^2 + \frac{1+1/\epsilon_r}{2} \pi^2} \right] \right\}$$

where:

$$w' = w + \Delta w'$$

$$\Delta w' = \Delta w \left( \frac{1+1/(\epsilon_r)}{2} \right)$$

$$\Delta w = \frac{1}{\pi} \ln \left[ \frac{4e}{\sqrt{(t/h)^2 + \left( \frac{1/\pi}{w/t + 1.1} \right)^2}} \right]$$

With  $\eta_0$ , the wave impedance;  $\epsilon_0$ , the permittivity; and  $\mu_0$  the permeability of free space.

In reference (1), it is recommended to replace Er by Eeff in the characteristic impedance computation. It is done by specified the parameter **USE\_ER=0**.

for  $\frac{w}{h} \leq 1$ :

$$E_{\text{eff}} = \frac{Er+1}{2} + \frac{Er-1}{2} \left[ \left( 1 + \frac{12h}{w} \right)^{-0.5} + 0.04 \left( 1 - \frac{12h}{w} \right)^2 \right]$$

and for  $\frac{w}{h} \geq 1$ :

$$E_{\text{eff}} = \frac{Er+1}{2} + \frac{Er-1}{2} \left( 1 + \frac{12h}{w} \right)^{-0.5}$$

### For a symmetric pair of coupled microstrip lines (**DLEV=1**)

- From reference (2) (**PLEV=0**), we have the following equations:

Capacitance matrix

$$\begin{bmatrix} \hat{C} + \hat{C}_M & -\hat{C}_M \\ -\hat{C}_M & \hat{C} + \hat{C}_M \end{bmatrix}$$

where:  $\hat{C} = \frac{1.0}{v_{p,e} \times Z_{L,e}}$  and  $\hat{C}_M = \frac{1}{2} \left( \frac{1.0}{v_{p,o} \cdot Z_{L,o}} - \frac{1.0}{v_{p,e} \cdot Z_{L,e}} \right)$

Inductance matrix 
$$\begin{bmatrix} \hat{L} & \hat{L}_M \\ \hat{L}_M & \hat{L} \end{bmatrix}$$

where:  $\hat{L} = \frac{1}{2} \left( \frac{Z_{L,e}}{v_{p,e}} + \frac{Z_{L,o}}{v_{p,o}} \right)$  and  $\hat{L}_M = \frac{1}{2} \left( \frac{Z_{L,e}}{v_{p,e}} - \frac{Z_{L,o}}{v_{p,o}} \right)$

Resistance matrix 
$$\begin{bmatrix} \frac{Re + Ro}{2} & \frac{Re - Ro}{2} \\ \frac{Re - Ro}{2} & \frac{Re + Ro}{2} \end{bmatrix}$$

where:  $Re = \frac{\ln 10}{10} \cdot \alpha_{c,e} \cdot Z_{L,e}$  and  $Ro = \frac{\ln 10}{10} \cdot \alpha_{c,o} \cdot Z_{L,o}$

Conductance matrix 
$$\begin{bmatrix} \frac{Ge + Go}{2} & \frac{Ge - Go}{2} \\ \frac{Ge - Go}{2} & \frac{Ge + Go}{2} \end{bmatrix}$$

where:  $Ge = \frac{\ln 10}{10} \cdot \frac{\alpha_{c,e}}{Z_{L,e}}$  and  $Go = \frac{\ln 10}{10} \cdot \frac{\alpha_{c,o}}{Z_{L,o}}$

The indices *e* and *o* indicate even and odd mode parameters respectively. The expression of effective relative permittivity, characteristic impedance and attenuation coefficients are given in the single microstrip line description (for more details, see reference (2)). Dispersion is taken into account (when required) in the effective permittivity and characteristic impedance computations (see reference (2)). Therefore, effective parameter matrices (R, L, C and G) values change with frequency.

## For a single stripline (DLEV=2)

- From reference (1) (**PLEV=0**), we have the following equations:

$$\text{Capacitance } C = \frac{\sqrt{\mu_0 \epsilon_0 \epsilon_r}}{Z_0}$$

$$\text{Inductance } L = Z_0 \sqrt{\mu_0 \epsilon_0 \epsilon_r}$$

$$\text{Resistance } R_{DC} = \frac{\rho}{w \times t}$$

$$\text{with: } Z_0 = \frac{\eta_0}{\sqrt{\epsilon_r} \times \frac{C_1}{\epsilon}}$$

and:

$$\begin{aligned} \frac{C_1}{\epsilon} = & \frac{2w_1/h}{1-s/h-t/h} + \frac{2w_1/h}{1+s/h-t/h} + \\ & \frac{2}{\pi} \left( \frac{2}{1-t/(h-s)} \ln \left[ 1 + \frac{1}{1-t/(h-s)} \right] + \left( 1 - \frac{1}{1-t/(h-s)} \right) \ln \left[ \frac{1}{(1-t/(h-s))^2} - 1 \right] \right) + \\ & \frac{2}{\pi} \left( \frac{2}{1-t/(h+s)} \ln \left[ 1 + \frac{1}{1-t/(h+s)} \right] + \left( 1 - \frac{1}{1-t/(h+s)} \right) \ln \left[ \frac{1}{(1-t/(h+s))^2} - 1 \right] \right) \end{aligned}$$

where:

$$s = h - (2 \times h_1 + t), \text{ for a centered stripline } s = 0$$

$$\text{if } \left( \frac{w}{h-t} \right) < 0.35 \text{ then } w_1 = \left( \frac{0.07 \times (h-t) + w}{1.2} \right)$$

$$\text{else } w_1 = w$$

Dispersive effects are introduced according to the value of the geometrical parameters. So the resistivity can be frequency dependent:

$$R = R_{DC} \times \left( 1 + (1+i) \times \sqrt{\frac{f}{\rho \mu_0 t^2}} \right)$$

You can also introduce frequency dependent conductance by using the parameters *Gs* and *fp* (polarization frequency: **FP** parameter). The conductance dispersive effect can be modeled in two ways according to the **MULTIDEBYE** parameter:

- One-pole debye model (**MULTIDEBYE=0**)

$$G = \tan d \times C \times \frac{i2\pi f \times 2\pi fp}{i2\pi f \times 2\pi fp}$$

where *fp* is the polarization frequency (**FP** parameter).

- Multi-pole debye model (**MULTIDEBYE=1**)

By using this option, we build a complex frequency-dependent capacitance matrix:

$$C(\omega) = C_{\text{inf}} + f(j\omega)C_d$$

Therefore, line conductance per unit length becomes:

$$Y(j\omega) = (j\omega)C_{\text{inf}} + (j\omega)f(j\omega)C_d$$

where:  $C_{\text{inf}} = C - \alpha \tan dC$  and  $C_d = \beta \tan dC$ ;  $C$  is the user defined matrix and  $\tan d$  the dielectric loss tangent parameter.

$$\text{with } \alpha = \frac{\ln\left(\frac{\omega_2^2 + \omega_0^2}{\omega_1^2 + \omega_0^2}\right)}{4\pi\left(\tan\left(\frac{\omega_0}{\omega_1}\right) - \tan\left(\frac{\omega_0}{\omega_2}\right)\right)} \text{ and } \beta = \frac{\ln(10^8)}{2\pi\left(\tan\left(\frac{\omega_0}{\omega_1}\right) - \tan\left(\frac{\omega_0}{\omega_2}\right)\right)}.$$

$$\text{Finally, the function } f(j\omega) = \frac{\ln\left(\frac{\omega_2 + j\omega}{\omega_1 + j\omega}\right)}{\ln(10^8)} \text{ which is fitted with 15 real poles.}$$

Note:  $\omega_0 = 2\pi f_p$ ,  $\omega_1 = 10^4$ ,  $\omega_2 = 10^{12}$ ,  $\omega = 2\pi f$  and  $f_p$  the polarization frequency (**FP** parameter).

## Reference



- (1) *Transmission Line Design Handbook*, Brian C. Wadell, Artech House 1991.
  - (2) *Implementation of Single and Coupled Microstrip line in APLAC*, Luis Costa and Martti Valtonen, CT-33 December 1997.
-

## Lossy Transmission Line: W Model

The W model is implemented in Eldo to simulate lossy coupled uniform lines including dispersive effects. This model can be used in all analysis modes (DC, AC, Transient, SST, SSTNOISE, or MODSST). The general instantiation of a W model is:

### RLGCfile form

```
Wxx N=nb_line
+ P1...PN PGNDin PN+1...P2N PGNDout
+ RLGCfile=file_name L=length [FP=val]
+ [MULTIDEBYE=val] [SAVEFIT=val] [COMPAT=val] [FGD=val]
```

### Umodel form

```
Wxx N=nb_line
+ P1...PN PGNDin PN+1...P2N PGNDout
+ Umodel=model_name L=length [SAVEFIT=val]
```

### RLGCmodel form

```
Wxx N=nb_line
+ P1...PN PGNDin PN+1...P2N PGNDout
+ RLGCmodel=model_name L=length [FP=val]
+ [MULTIDEBYE=val] [SAVEFIT=val] [COMPAT=val] [FGD=val]
```

### Tabular RLGCmodel form

```
Wxx P1...PN PGNDin PN+1...P2N PGNDout
+ N=nb_line L=length
+ TABLEMODEL=table_model_name [SAVEFIT=val] [FITTABLEMODEL=val]
```

### Parameters

- **xx**  
W model transmission line name.
- **N=nb\_line**  
Number of lines.
- **P1...PN**  
The N nodes at one end of the line system for a system consisting of N lines.
- **PGNDin**  
Reference node for the P1...PN nodes of the line system.
- **PN+1...P2N**  
The N nodes at the other end of the line system. The line number i in the line system connects the nodes Pi and PN+i.

- **PGNDout**  
Reference node for the PN+1...P2N nodes of the line system.
- **RLGCfile=file\_name**  
Name of the file containing R, L, C, G,  $R_s$  and  $G_d$  matrices.
- **Umodel=model\_name**  
Name of the transmission line model. This entry allows the use of the U model ([page 4-70](#)) entries in the W model.
- **TABLEMODEL=table\_model\_name**  
Name of the model containing R, L, C and G tabular matrices description.
- **L=length**  
Geometric length of the system (meter). Default value is 1.0. If  $L=0$ , Eldo uses the default value.
- **FP=val**  
Polarization frequency to control dispersive effect on the conductance. Default:  $1.6 \times 10^9$ .
- **MULTIDEBYE=val**  
 $val=1$  specifies the use of multi-pole debye model to model the dispersive effect on the conductance (recommended for modeling PCB-type dielectrics).  $val=0$ , this model is not used. Default: 0.
- **SAVEFIT=val**  
If the value is 1, this option saves the initialization of the transmission line model (in the file *circuit\_name.fit*), in order to speed up the following simulations of the same netlist. Default value is 0.
- **COMPAT=val**  
If the value is 1, it specifies the model used for the dispersive effect is based on conductance (see the formula details of each W model instantiation on [page 4-61](#) and [page 4-64](#)). Default value is taken from the global option **COMPAT** (if **.OPTION COMPAT** is specified then **COMPAT=1**). Note that the **MULTIDEBYE** parameter priority is higher than **COMPAT**. If **MULTIDEBYE** is specified, then the value of **COMPAT** is zero.
- **FGD=val**  
Cut-off frequency value. Default is zero. Can only be specified in compat mode (**COMPAT=1**).
- **FITTABLEMODEL=val**  
When set to 1, it will enable a causal model (admittance and propagation) to be built from non-causal tabulated data. If set to 0 (default), the tabulated data will not be modified to build the model and the built-in models are considered to be causals.

## Examples

### RLGCfile entry

```
W1 N=2
+ 1 2 0 3 4 0
+ RLGCfile=2line.rlgc L=0.97e-3
```



See the `2line.rlgc` file example on [page 4-63](#).

---

### Umodel entry

```
.MODEL unamel U LEVEL=3 ELEV=1 PLEV=1 DLEV=2 NL=1
+ HT=1.0e-4 WD=2.0e-4 TH=5.0e-5 RHO=1.785e-8
W1 N=1 1 0 2 0 Umodel=uname L=1.0e-3
```

### RLGCmodel entry

```
W1 N=2 1 2 0 4 5 0 RLGCmodel=model_rlgc L=0.97e-3
```

### Tabular RLGCmodel entry

```
W1 i1 i2 0 o1 o2 0 N=2 L=0.1 TABLEMODEL=ex1
```

## RLGC file syntax

The RLGC file is a text file, which contains the values of R, L, C, G,  $R_s$  and  $G_d$  matrices per unit length. This file is order-dependent, and the order is the following:

- N Number of lines.
- $L_o$  DC inductance matrix (per unit length).
- $C_o$  DC capacitance matrix (per unit length).
- $R_o$  DC resistance matrix (per unit length).
- $G_o$  DC conductance matrix (per unit length).
- $R_s$  Skin effect resistance matrix (per unit length):

$$R = R_o + (1 + i)\sqrt{f}R_s$$

- $G_d$  Dielectric-loss conductance matrix (per unit length). The frequency dependent conductance uses the parameters  $G_s$  and  $f_p$  (polarization frequency: **FP** parameter). It can be modeled in two ways according to the **MULTIDEBYE** parameter:

- One-pole debye model (**MULTIDEBYE**=0)

$$G = G_o + \frac{G_d}{2\pi} \times \frac{i2\pi f \times 2\pi fp}{i2\pi f + 2\pi fp}$$

where  $fp$  is the polarization frequency (**FP** parameter).

- Multi-pole debye model (**MULTIODEBYE**=1)

By using this option, we build a complex frequency-dependent capacitance matrix:

$$C(\omega) = C_{inf} + f(j\omega)C_d$$

Therefore, line conductance per unit length becomes:

$$Y(j\omega) = G_o + (j\omega)C_{inf} + (j\omega)f(j\omega)C_d$$

where:  $C_{inf} = C - \alpha G_d$  and  $C_d = \beta G_d$ ;  $C$  and  $G$ s are the user defined matrices. with:

$$\alpha = \frac{\ln\left(\frac{\omega_2^2 + \omega_0^2}{\omega_1^2 + \omega_0^2}\right)}{4\pi\left(\tan\left(\frac{\omega_0}{\omega_1}\right) - \tan\left(\frac{\omega_0}{\omega_2}\right)\right)} \text{ and } \beta = \frac{\ln(10^8)}{2\pi\left(\tan\left(\frac{\omega_0}{\omega_1}\right) - \tan\left(\frac{\omega_0}{\omega_2}\right)\right)}$$

Finally, the function  $f(j\omega) = \frac{\ln\left(\frac{\omega_2 + j\omega}{\omega_1 + j\omega}\right)}{\ln(10^8)}$  which is fitted with 15 real poles.

Note:  $\omega_0 = 2\pi f_p$ ,  $\omega_1 = 10^4$ ,  $\omega_2 = 10^{12}$ ,  $\omega = 2\pi f$  and  $fp$  the polarization frequency (**FP** parameter).

- Compat dispersive model (**COMPAT**=1)

$$G = G_o + G_d \times \frac{f}{\sqrt{1 + \left(\frac{f}{f_{gd}}\right)^2}}$$

where  $f_{gd}$  is a cut-off frequency; if  $f_{gd}$  value is zero then  $G$  keeps linear dependency on the frequency. Default is zero.

The  $R_o$ ,  $G_o$ ,  $R_s$  and  $G_d$  matrices are optional (default value is zero).  $L_o$  and  $C_o$  matrices must be described in the RLGC file. Since these matrices are symmetrical, only the lower-triangular parts are specified in the RLGC file.

The diagonal terms of  $L_o$  and  $C_o$  matrices must be positive non-zero; the diagonal terms of  $R_o$ ,  $R_s$ ,  $G_o$  and  $G_d$  matrices must be non-negative. Off-diagonal terms of  $C_o$ ,  $G_o$  and  $G_d$  are non-positive.

## Comments

A comment line can be specified by an asterisk '\*' at the beginning of the line. This comments out the entire line.

## Separator

The number can be separated by any combination of the characters shown in the table below:

**Table 4-10. RLGC Separator Characters**

Character
Space
Tab
New line
,
;
(
)
[
{
}

## Example RLGC file: 2line.rlgc

```
*RLGC matrices for 2 frequency-dependent lines
*N (number of lines)
*****
2
* Lo
*****
0.3481e-6
0.5458e-7  0.3481e-6
* Co
*****
0.1593e-9
-0.2578e-10 0.1651e-9
* Ro
*****
75
0      50
* Go
*****
0.2421e-3
-0.4860e-4  0.2070e-3
```

```
* Rs
*****
0.0025
0    0.0014
* Gd
*****
1.2e-13
-4.1e-14  1.1e-13
```

## RLGC model syntax

The RLGCmodel is a model, which contains the values of R, L, C, G,  $R_s$  and  $G_d$  matrices per unit length. There is no limitation on the number of coupled lines. Since the matrices are symmetric, only the lower-triangular parts of the matrices have to be described in the RLGCmodel. Inductance and capacitance matrices ( $C_o$  and  $L_o$ ) have to be specified, the other matrices can be optional.

### General instantiation of the model

```
.MODEL model_name W MODELTYPE=RLGC N=nb_line
+ Lo=Lo_matrix_entries Co=Co_matrix_entries
+ [Ro=Ro_matrix_entries] [Go=Go_matrix_entries]
+ [Rs=Rs_matrix_entries] [Gd=Gd_matrix_entries]
```

### Parameters

- **N=nb\_line**  
Number of lines.
- **Lo=Lo\_matrix\_entries**  
Elements of the DC inductance matrix (per unit length).
- **Co=Co\_matrix\_entries**  
Elements of the DC capacitance matrix (per unit length).
- **Ro=Ro\_matrix\_entries**  
Elements of the DC resistance matrix (per unit length).
- **Go=Go\_matrix\_entries**  
Elements of the DC conductance matrix (per unit length).
- **Rs=Rs\_matrix\_entries**  
Elements of the skin-effect inductance matrix (per unit length):

$$R = R_o + (1 + i)\sqrt{f}R_s$$

- `Gd=Gd_matrix_entries`

Elements of the dielectric-loss conductance matrix (per unit length). The frequency dependent conductance uses the parameters  $G_s$  and  $fp$  (polarization frequency: **FP** parameter). It can be modeled in two ways according to the **MULTIDEBYE** parameter:

- One-pole debye model (**MULTIDEBYE**=0)

$$G = G_o + \frac{G_d}{2\pi} \times \frac{i2\pi f \times 2\pi fp}{i2\pi f + 2\pi fp}$$

where  $fp$  is the polarization frequency (**FP** parameter).

- Multi-pole debye model (**MULTIDEBYE**=1)

By using this option, we build a complex frequency-dependent capacitance matrix:

$$C(\omega) = C_{inf} + f(j\omega)C_d$$

Therefore, line conductance per unit length becomes:

$$Y(j\omega) = G_o + (j\omega)C_{inf} + (j\omega)f(j\omega)C_d$$

where:  $C_{inf} = C - \alpha G_d$  and  $C_d = \beta G_d$ ;  $C$  and  $G_s$  are the user defined matrices. with:

$$\alpha = \frac{\ln\left(\frac{\omega_2^2 + \omega_0^2}{\omega_1^2 + \omega_0^2}\right)}{4\pi\left(\tan\left(\frac{\omega_0}{\omega_1}\right) - \tan\left(\frac{\omega_0}{\omega_2}\right)\right)} \text{ and } \beta = \frac{\ln(10^8)}{2\pi\left(\tan\left(\frac{\omega_0}{\omega_1}\right) - \tan\left(\frac{\omega_0}{\omega_2}\right)\right)}$$

$$\text{Finally, the function } f(j\omega) = \frac{\ln\left(\frac{\omega_2 + j\omega}{\omega_1 + j\omega}\right)}{\ln(10^8)}$$

Note:  $\omega_0 = 2\pi f_p$ ,  $\omega_1 = 10^4$ ,  $\omega_2 = 10^{12}$ ,  $\omega = 2\pi f$  and  $fp$  the polarization frequency (**FP** parameter).

- Compat dispersive model (**COMPAT**=1)

$$G = G_o + G_d \times \frac{f}{\sqrt{1 + \left(\frac{f}{f_{gd}}\right)^2}}$$

where  $f_{gd}$  is a cut-off frequency; if  $f_{gd}$  value is zero then  $G$  keeps linear dependency on the frequency. Default is zero.

## Example RLGC Model

```
.MODEL model_rlgc W MODELTYPE=RLGC N=2
+ Lo = 0.3481e-6
+ 0.5458e-7 0.3481e-6
+ Co = 0.1593e-9
+ -0.2578e-10 0.1651e-9
+ Ro = 75
+ 0 50
+ Go = 0.2421e-3
+ -0.4860e-4 0.2070e-3
+ Rs = 0.0025
+ 0.0014
+ Gd = 1.2e-13
+ -4.1e-14 1.1e-13
```

## Tabular RLGCmodel syntax

The Tabular model is an extension of the RLGCmodel, which allows to model transmission line arbitrary frequency-dependent behavior. There is no limitation on the number of coupled lines. Inductance and capacitance tabular matrices ( $C_o$  and  $L_o$ ) have to be specified, the other tabular matrices are optional. Each tabular matrix is described in a **.MODEL** statement.

### General instantiation of the model

```
.MODEL model_name sp W MODELTYPE=TABLE N=nb_line
+ LMODEL=L_freq_model CMODEL=C_freq_model
+ [RMODEL=R_freq_model] [GMODEL=G_freq_model] [FITTABLEMODEL=val]
```

### Parameters

- **N=nb\_line**  
Number of lines.
- **LMODEL=L\_freq\_model**  
Name of the model containing the sampled values of the inductance matrix.
- **CMODEL=C\_freq\_model**  
Name of the model containing the sampled values of the capacitance matrix.
- **RMODEL=R\_freq\_model**  
Name of the model containing the sampled values of the resistance matrix. Default is zero.
- **GMODEL=G\_freq\_model**  
Name of the model containing the sampled values of the conductance matrix. Default is zero.

- FITTABLEMODEL=val

When set to 1, it will enable a causal model (admittance and propagation) to be built from non-causal tabulated data. If set to 0 (default), the tabulated data will not be modified to build the model and the built-in models are considered to be causals.

## Example Tablemodel

```
.model ex1 W MODELTYPE=TABLE N=2 LMODEL=lmod1
+ CMODEL=cmod1 Rmodel=rmod1 Gmodel=gmod1
```

## Sampled matrix model

This tabular matrix model gives a frequency-varying behavior of R, L, C and G matrices.

### General instantiation of the model

```
.MODEL model_name sp N=nb_line
+ SPACING=spacing_type VALTYPE=value_type
+ [ INFINITY=matrix_values]
+ DATA=tabular_matrix_values
```

### Parameters

- model\_name  
Name of the model.
- N=nb\_line  
Number of lines.
- SPACING=spacing\_type  
Data spacing format: only NONUNIFORM type is handled.
- VALTYPE=value\_type  
Type of matrix elements: only REAL type is handled.
- INFINITY=matrix\_values  
Data points at infinity.
- DATA=tabulated\_matrix\_values  
Specified frequency value and corresponding matrix data points. As the matrices are symmetric, only the lower-half portion is described. Syntax:  
DATA=(sampled\_number, f1 data1 f2 data2 ...).

## Example Tablemodel

As the model is a “two coupled transmission line”, the dimension of the matrices is 2. Therefore, on each line of the DATA specification, after the sample number, the first value is

the frequency, the second is the (1,1) diagonal value, the third is the (2,1) off-diagonal value, and the last is the (2,2) diagonal value.

```
.model cmod1 sp N=2 SPACING=NONUNIFORM VALTYPE=REAL
+ DATA=(1,( 6.602360e-11 -7.04724e-12 6.602360e-11))

.MODEL lmod1 sp N=2 SPACING=NONUNIFORM VALTYPE=REAL
+ INFINITY=(4.0076e-7 4.6030e-8 4.0076e-7)
+ DATA=( 20,
+ (0.000000e+00 3.934460e-07 4.6030e-08 3.933460e-07)
+ (3.746488e+06 4.151139e-07 4.6030e-08 4.151959e-07)
+ (7.726980e+06 4.084730e-07 4.6030e-08 4.085604e-07)
+ (1.196411e+07 4.054831e-07 4.6030e-08 4.055730e-07)
+ (1.648352e+07 4.037715e-07 4.6030e-08 4.037628e-07)
+ (3.204884e+07 4.008228e-07 4.6030e-08 4.008166e-07)
+ (5.911330e+07 3.988513e-07 4.6030e-08 3.988467e-07)
+ (7.650809e+07 3.981851e-07 4.6030e-08 3.981811e-07)
+ (8.650875e+07 3.978968e-07 4.6030e-08 3.978931e-07)
+ (9.756098e+07 3.976313e-07 4.6030e-08 3.976278e-07)
+ (1.098398e+08 3.973847e-07 4.6030e-08 3.973813e-07)
+ (1.235615e+08 3.971538e-07 4.6030e-08 3.971507e-07)
+ (2.962963e+08 3.958050e-07 4.6030e-08 3.958030e-07)
+ (3.428571e+08 3.956319e-07 4.6030e-08 3.956300e-07)
+ (4.010283e+08 3.954596e-07 4.6030e-08 3.954579e-07)
+ (5.753425e+08 3.951106e-07 4.6030e-08 3.951092e-07)
+ (7.145791e+08 3.949294e-07 4.6030e-08 3.949281e-07)
+ (9.230769e+08 3.947392e-07 4.6030e-08 3.947380e-07)
+ (1.269625e+09 3.945339e-07 4.6030e-08 3.945329e-07)
+ (4.000000e+09 3.940153e-07 4.6030e-08 3.940147e-07)
+ )

.MODEL rmod1 sp N=2 SPACING=NONUNIFORM VALTYPE=REAL
+ DATA=( 18,
+ (0.000000e+00 8.765530e-01 6.299210e-03 8.765530e-01)
+ (3.746488e+06 6.028640e+00 6.299210e-03 6.028640e+00)
+ (7.726980e+06 8.270684e+00 6.299210e-03 8.270684e+00)
+ (1.196411e+07 1.007694e+01 6.299210e-03 1.007694e+01)
+ (2.131439e+07 1.313620e+01 6.299210e-03 1.313620e+01)
+ (3.803487e+07 1.727973e+01 6.299210e-03 1.727973e+01)
+ (6.741573e+07 2.271433e+01 6.299210e-03 2.271433e+01)
+ (7.650809e+07 2.414031e+01 6.299210e-03 2.414031e+01)
+ (9.756098e+07 2.714662e+01 6.299210e-03 2.714662e+01)
+ (1.098398e+08 2.845071e+01 6.299210e-03 2.845071e+01)
+ (1.764706e+08 3.620734e+01 6.299210e-03 3.620734e+01)
+ (1.995249e+08 3.844427e+01 6.299210e-03 3.844427e+01)
+ (2.264151e+08 4.085570e+01 6.299210e-03 4.085570e+01)
+ (3.428571e+08 5.012232e+01 6.299210e-03 5.012232e+01)
+ (4.010283e+08 5.413628e+01 6.299210e-03 5.413628e+01)
+ (7.145791e+08 7.197077e+01 6.299210e-03 7.197077e+01)
+ (1.269625e+09 9.584070e+01 6.299210e-03 9.584070e+01)
+ (4.000000e+09 1.690795e+02 6.299210e-03 1.690795e+02)
+ )

.MODEL gmod1 sp N=2 SPACING=NONUNIFORM VALTYPE=REAL
+ DATA=( 22,
+ (0.000000e+00 5.977166e-11 0.000000e+00 5.977166e-11)
+ (3.746488e+06 1.451137e-05 -1.821096e-06 1.451043e-05)
```

```

+ (7.726980e+06 2.992905e-05 -3.755938e-06 2.992712e-05)
+ (1.196411e+07 4.634076e-05 -5.815525e-06 4.633777e-05)
+ (2.131439e+07 8.245729e-05 -1.036052e-05 8.245196e-05)
+ (3.803487e+07 1.473209e-04 -1.848803e-05 1.473114e-04)
+ (5.911330e+07 2.289642e-04 -2.873385e-05 2.289494e-04)
+ (6.741573e+07 2.611221e-04 -3.276951e-05 2.611062e-04)
+ (7.650809e+07 2.963396e-04 -3.718913e-05 2.963205e-04)
+ (9.756098e+07 3.778840e-04 -4.742254e-05 3.778596e-04)
+ (1.098398e+08 4.253437e-04 -5.339105e-05 4.254163e-04)
+ (1.389961e+08 5.383752e-04 -6.756338e-05 5.383404e-04)
+ (1.564859e+08 6.061286e-04 -7.606484e-05 6.060795e-04)
+ (1.995249e+08 7.728220e-04 -9.698528e-05 7.727721e-04)
+ (2.264151e+08 8.769759e-04 -1.100561e-04 8.769193e-04)
+ (2.962963e+08 1.146647e-03 -1.440240e-04 1.147553e-03)
+ (3.428571e+08 1.327992e-03 -1.666563e-04 1.327906e-03)
+ (4.757709e+08 1.842808e-03 -2.312632e-04 1.842689e-03)
+ (5.753425e+08 2.228580e-03 -2.796630e-04 2.228336e-03)
+ (9.230769e+08 3.575363e-03 -4.486902e-04 3.575132e-03)
+ (1.959184e+09 7.588526e-03 -9.523220e-04 7.588036e-03)
+ (4.000000e+09 1.549424e-02 -1.944324e-03 1.549234e-02)
+
)

```

## Error message treatment

Most of the errors you can meet with this model are the same as for the LDTL model, see [page 4-50](#). Additionally:

```
ERROR IN RLGC FILE 2lin.rlc : check matrix C
```

This means there is a lack of value(s) in the C matrix description.

## Lossy Transmission Line: U Model

```
Uxx P1...PN PGNDin PN+1...P2N PGNDout UNAME L=length [SAVEFIT=val]
```

The U model is implemented in Eldo to simulate lossy-coupled uniform lines. This model can be used in all analysis modes (DC, AC, Transient, SST, SSTNOISE, or MODSST).

### Parameters

- **xx**  
Transmission line name.
- **P1...PN**  
The N nodes at one end of the line system for a system consisting of N lines.
- **PGNDin**  
Reference node for the P1...PN nodes of the line system.
- **PN+1...P2N**  
The N nodes at the other end of the line system. The line number i in the line system connects the nodes Pi and PN+i.
- **PGNDout**  
Reference node for the PN+1...P2N nodes of the line system.
- **UNAME**  
Name of the lossy transmission line model.
- **L=length**  
Geometric length of the system (meter). Default value is 1.0. If L=0, Eldo uses the default value.
- **SAVEFIT=val**  
If the value is 1, this option saves the initialization of the transmission line model (in the file *circuit\_name.fit*), in order to speed up the following simulations of the same netlist. Default value is 0.

### Example

```
U1 1 0 2 0 Umodel L=1.0e-3
```

Specifies a lossy transmission line U1 between nodes 1 and 2, the reference plane is the ground (node 0). The length of this transmission line is 1.0e-3 and all the parameters are specified in the model called Umodel (**.MODEL U model**).

## Model Syntax

```
.MODEL UNAME U LEVEL=3 ELEV=elev_val PLEV=plev_val
+ [DEV=dlev_val] [LLEV=llev_val] [Param=p_val]
```

### Parameters

- **UNAME**  
Name of the model.
- **LEVEL=3**  
Selects the model of lossy transmission line.
- **ELEV=elev\_val**  
Selects the specification format:
  - ELEV=1** → geometrical description.
  - ELEV=2** → precomputed model parameters (R, L, C, and G matrices).
  - ELEV=3** → measured parameters.
- **PLEV=plev\_val**  
Selects the type of transmission line: planar structure (**PLEV=1**), coax (**PLEV=2**) or twinhead (**PLEV=3**). Only planar structure is supported.
- **DLEV=dlev\_val**  
Specifies the dielectric and ground reference configuration. Two configurations are proposed: microstrip layered dielectric (**DLEV=1**) and stripline (**DLEV=2**). Default value is 1.
- **LLEV=llev\_val**  
Reference plane inductance consideration (default is 0):
  - LLEV=0** → omit this inductance.
  - LLEV=1** → include this inductance (not supported).
- **Param=p\_val**  
Specifies parameters of the lines (depends on the specification format).

### Geometric description: ELEV=1

#### Restriction

Only single line can be described.

## Specific parameters

- DLEV  
Type of Line;  
DLEV=1 → Microstrip layered dielectric  
DLEV=2 → Stripline
- NL  
Number of line, default value is 1. (only single line can be described).
- HT  
Conductor height.
- WD  
Conductor width.
- TH  
Conductor thickness.
- KD  
Dielectric relative permittivity.
- RHO  
Conductor resistivity. Default value is 17e-9 Ωm (copper).

## Example

```
.MODEL Umodel U LEVEL=3 ELEV=1 PLEV=1 DLEV=2 NL=1
+ HT=1.0e-4 + WD=2.0e-4 TH=5.0e-5 RHO=1.785e-8
```

This model describes a lossy stripline.

## Precomputed model parameters: ELEV=2

The precomputed parameters correspond to the R, L, C and G matrices. Since these matrices are symmetric, only the upper-triangular parts are specified.

## Restriction

This description allows the specification of up to five signal conductors.

## Specific parameters

- crj  
Self capacitance per unit length ( $Fm^{-1}$ ). Default value is 1.0e-9.

- $c_{ij}$   
Mutual capacitance per unit length ( $Fm^{-1}$ ). Default value is 0.
- $l_{jj}$   
Self inductance per unit length ( $Hm^{-1}$ ). Default value is  $1.0e^{-6}$ .
- $l_{ij}$   
Mutual inductance per unit length ( $Hm^{-1}$ ). Default value is 0.
- $r_{jj}$   
Resistance per unit length ( $\Omega m^{-1}$ ). Default value is 0.
- $g_{rj}$   
Self conductance per unit length ( $Sm^{-1}$ ). Default value is 0.
- $g_{ij}$   
Mutual conductance per unit length ( $Sm^{-1}$ ). Default value is 0.

## Example

```
.MODEL Umodel U LEVEL=3 ELEV=2 PLEV=1 r11=34.48
+ r22=34.48 + r33=34.48 l11=49.76n l22=49.76n l33=49.76n
+ l12=7.65n + l23=7.65n cr1=10.82p cr2=11.24p cr3=10.82p
+ c12=-1.97p + c23=-1.97p gr1=0.15u gr2=0.15u gr3=0.15u
```

This model describes three coupled lossy transmission lines.

## Measured parameters: ELEV=3

This description corresponds to the electrical parameters.

## Restriction

Only single line can be described.

## Specific parameters

- **ZK**  
Characteristic impedance ( $\Omega$ ).
- **VREL**  
Relative velocity.
- **DELAY**  
Delay(s) for length **DELEN**.
- **CAPL**  
Linear capacitance in length **CLEN**. Default value is 1.

- AT1  
Attenuation factor in length **ATLEN**. Default value is 1.
- DELEN  
Unit of length (m) for **DELAY**. Default value is 1.
- CLEN  
Unit of length (m) for **CAPL**. Default value is 1.
- ATLEN  
Unit of length for **AT1**. Default value is 1.
- FR1  
Frequency at which dispersion starts (only affects resistance). If no value is specified, the dispersion will not be taken into account.

In order to discard redundant parameter sets, the following equations are used:

**Table 4-11. Lossy Transmission Line: U Model Parameter Combinations**

Input Parameters	Computation
ZK, DELAY, DELEN, CAPL and CLEN	Redundant, discard CAPL and CLEN
ZK, VREL, CAPL and CLEN	Redundant, discard CAPL and CLEN
ZK, DELAY and DLEN	$VREL = \frac{DLEN}{DELAY \times CLIGHT}$
ZK and VREL	$C = \frac{1.0}{ZK \times VREL \times CLIGHT}$ $L = \frac{ZK}{VREL \times CLIGHT}$
ZK, CAPL and CLEN	$C = \frac{CAPL}{CLEN}$ $L = C \times ZK^2$
CAPL, CLEN, DELAY and DELEN	$VREL = \frac{DELEN}{DELAY \times CLIGHT}$
CAPL, CLEN and VREL	$C = \frac{CAPL}{CLEN}$ $L = \frac{1.0}{C \times VREL^2 \times CLIGHT^2}$

## Example

```
.MODEL Umodel U LEVEL=3 ELEV=3 PLEV=1 ZK=50 DELAY=10n
+ AT1=1
```

This model describes a single lossy transmission line with a characteristic impedance of 50 Ω.

## Error message treatment

Most of the errors you can meet with this model are the same as for the LDTL model, see [page 4-50](#).

## Microstrip Models

Following are a set of microstrip and stripline layout discontinuity structures targeting RF simulations where a piece of microstrip or stripline discontinuity has to be included. They may also be used for integrated design of microstrip or stripline structures. The available models are as follows:

Microstrip Discontinuities:

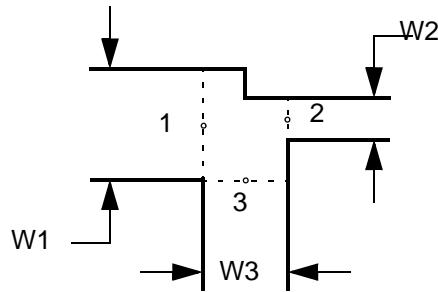
- [MTEE: Microstrip T Junction](#)
- [MBEND: Microstrip Bend \(Arbitrary Angle, Optimally Mitered\)](#)
- [MBEND2: 90-degree Microstrip Bend \(Mitered\)](#)
- [MBEND3: 90-degree Microstrip Bend \(Optimally Mitered\)](#)
- [MCORN: 90-degree Microstrip Bend \(Unmiterred\)](#)
- [MSTEP: Microstrip Step in Width](#)
- [VIA2: Cylindrical Via Hole in Microstrip](#)

Stripline Discontinuities:

- [SBEND: Unmiterred Stripline Bend](#)
- [STEE: Stripline T Junction](#)
- [SSTEP: Stripline Step in Width](#)

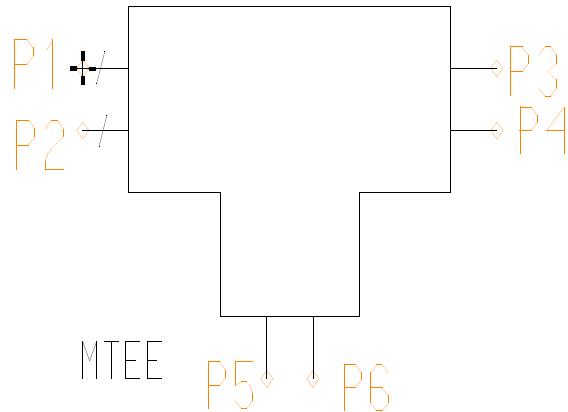
## MTEE: Microstrip T Junction

**Figure 4-6. Microstrip T Junction**



### Symbol

**Figure 4-7. Microstrip T Junction Symbol**



### Syntax

```
Yxx MTEE P1 P2 P3 P4 P5 P6 PARAM: [W1=val] [W2=val] [W3=val]
+ [T=val] [Er=val] [H=val]
```

### Parameters

**Table 4-12. Microstrip T Junction Parameters**

Parameter	Definition	Default	Units
W1	Conductor width of the first arm	2.0e-3	meter
W2	Conductor width of the second arm	2.0e-3	meter
W3	Conductor width of the third arm	3.0e-3	meter
T	Conductor thickness	5.0e-6	meter
ER	Dielectric relative permittivity	4	-

**Table 4-12. Microstrip T Junction Parameters**

Parameter	Definition	Default	Units
H	Dielectric thickness	1.6e-3	meter

### Model validity range

$$0.5 \leq W1/H \leq 2.0$$

$$0.5 \leq W2/H \leq 2.0$$

$$0.5 \leq W3/H \leq 2.0$$

### Simulation domains

DC, AC, TRANSIENT, and SST

### References

Brian C. Wadell, "Transmission Line Design Handbook", 1991 Artech House.

### Notes

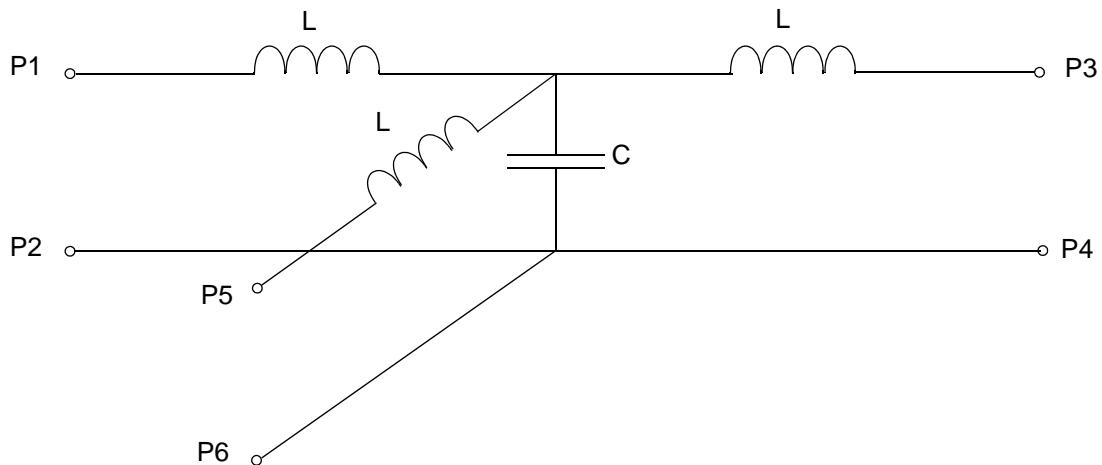
The model is based on the microstrip line symmetric T junction equations given in the mentioned reference.

The model handles symmetrical T-junction only. If the specified W1 and W2 parameters are not identical, the geometrical mean of W1 and W2 parameters is computed and used.

$$W1 = W2 = \sqrt{W1 \cdot W2} \text{ for non-symmetrical T-junction}$$

Figure 4-8 illustrates the model equivalent circuit and pins connections:

**Figure 4-8. Equivalent circuit Microstrip T Junction**



## Example

A simple MTEE s-parameter extraction example over a range of frequencies:

```

.param w1      = 2.0e-3
.param w2      = 2.0e-3
.param w3      = 3.0e-3
.param t       = 5.0e-6
.param Er      = 4
.param h       = 1.6e-3
.param frequency = 5e9

Ymtee MTEE t1a 0 t1b 0 t2 0 PARAM: W1=w1 W2=w2 W3=w3 T=t Er=Er + H=h

*** S-Parameters Extraction

V1a  t1a  0 IPORT=1 RPORT=50 FOUR fund1 PdBm (1) -100 -90
V1b  t1b  0 IPORT=2 RPORT=50 FOUR fund1 PdBm (1) -100 -90
V2   t2   0 IPORT=3 RPORT=50 FOUR fund1 PdBm (1) -100 -90

.step param frequency 1e9 7e9 100e6

.sst fund1=frequency nharm1=1

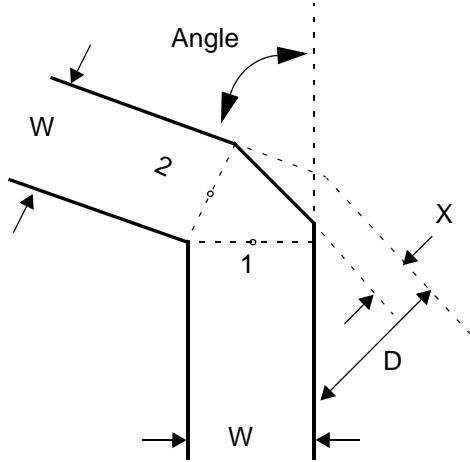
.extract fsst label=S11_Mag yval(SM(1,1),frequency)
.extract fsst label=S12_Mag yval(SM(1,2),frequency)
.extract fsst label=S13_Mag yval(SM(1,3),frequency)
.extract fsst label=S21_Mag yval(SM(2,1),frequency)
.extract fsst label=S22_Mag yval(SM(2,2),frequency)
.extract fsst label=S23_Mag yval(SM(2,3),frequency)
.extract fsst label=S31_Mag yval(SM(3,1),frequency)
.extract fsst label=S32_Mag yval(SM(3,2),frequency)
.extract fsst label=S33_Mag yval(SM(3,3),frequency)

.end

```

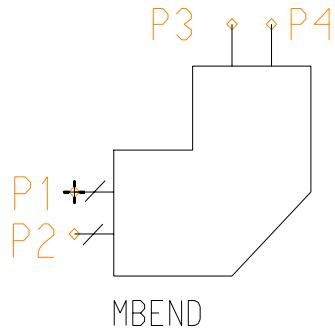
## MBEND: Microstrip Bend (Arbitrary Angle, Optimally Mitered)

**Figure 4-9. Microstrip Bend (Arbitrary Angle, Optimally Mitered)**



### Symbol

**Figure 4-10. Microstrip Bend (Arbitrary Angle, Optimally Mitered) Symbol**



### Syntax

```
Yxx MBEND P1 P2 P3 P4 PARAM: [W=val] [H=val] [Er=val] [T=val]
+ [RHO=val] [TAND=val] [M=val] [ANGLE=val]
```

### Parameters

**Table 4-13. Microstrip Bend (Arbitrary Angle, Optimally Mitered) Parameters**

Parameter	Definition	Default	Units
W	Conductor width	2.0e-3	meter
H	Dielectric thickness	1.6e-3	meter

**Table 4-13. Microstrip Bend (Arbitrary Angle, Optimally Mitered) Parameters**

Parameter	Definition	Default	Units
ER	Dielectric relative permittivity	4	-
T	Conductor thickness	5.0e-6	meter
RHO	Conductor resistivity	1.7e-8	Ohm.meter
TAND	Dielectric loss tangent	0.0	-
M	Optimal mitre percentage	60	%
ANGLE	Bend angle	60	degree

### Model validity range

$$1 \leq ER \leq 128$$

$$0 < ANGLE < 90$$

$$0.01 \leq W/H \leq 100$$

### Simulation domains

DC, AC, TRANSIENT, and SST

### References

Brian C. Wadell, "Transmission Line Design Handbook", 1991 Artech House.

### Equations

$$M = \frac{100 X}{d} \quad (\%)$$

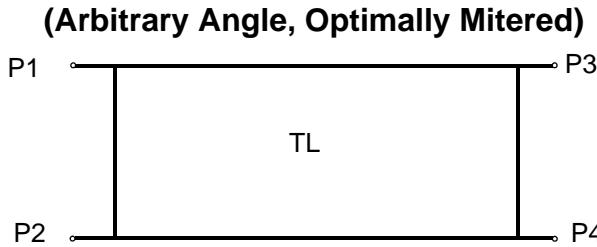
$$d - X = \sqrt{2} W \left( 1 - \frac{M}{100} \right)$$

The model is equivalent to a transmission line of length:

$$l = \frac{2 M}{100 \sin(ANGLE)}$$

[Figure 4-11](#) illustrates the model equivalent circuit and pins connections:

**Figure 4-11. Equivalent circuit Microstrip Bend**



## Example

A simple MBEND s-parameter extraction example over a range of frequencies:

```
.param W      = 2.0e-3
.param H      = 1.6e-3
.param Er     = 4
.param T      = 5.0e-6
.param RHO    = 1.7e-8
.param TAND   = 0
.param M      = 60
.param angle  = 60
.param fx     = 5e9

Ymbend MBEND in 0 out 0 PARAM: W=W H=H Er=Er T=T RHO=RHO
+ TAND=TAND M=M ANGLE=ANGLE

*** S-Parameters Extraction

Vin in 0 IPORT=1 RPORT=50 FOUR fund1 PdBm (1) -100 -90
Vout out 0 IPORT=2 RPORT=50 FOUR fund1 PdBm (1) -100 -90

.step param fx 1e9 7e9 100e6

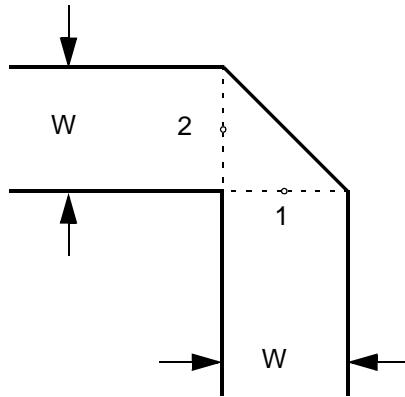
.sst fund1=fx nharm1=1

.extract fsst label=S11_Mag yval(SM(1,1),fx)
.extract fsst label=S12_Mag yval(SM(1,2),fx)

.end
```

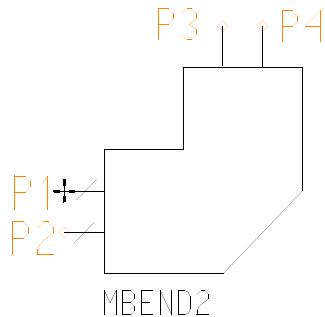
## MBEND2: 90-degree Microstrip Bend (Mitered)

**Figure 4-12. 90-degree Microstrip Bend (Mitered)**



### Symbol

**Figure 4-13. 90-degree Microstrip Bend (Mitered) Symbol**



### Syntax

```
YXX MBEND2 P1 P2 P3 P4 PARAM: [H=val] [W=val] [Er=val]
```

### Parameters

**Table 4-14. 90-degree Microstrip Bend (Mitered) Parameters**

Parameter	Definition	Default	Units
H	Substrate thickness	1.6e-3	meter
W	Conductor width	2.0e-3	meter
ER	Dielectric constant	4	-

### Model validity range

$0.2 < W/H < 6$

$2.36 < ER < 10.4$

Simulation frequency  $< 12/H$  (Frequency in GHz, H in mm)

## Simulation domains

DC, AC, TRANSIENT, and SST

## References

M. Kirschning, R. H. Jansen, and N. H. L. Koster. "Measurement and Computer-Aided Modeling of Microstrip Discontinuities by an Improved Resonator Method," 1983 IEEE MTT-S International Microwave Symposium Digest, May 1983, pp.495-497.

## Equations

The equivalent circuit of the MBEND2 consists of 2 inductors and a capacitor, shown in [Figure 4-14](#).

Equations used to calculate the equivalent circuit component values:

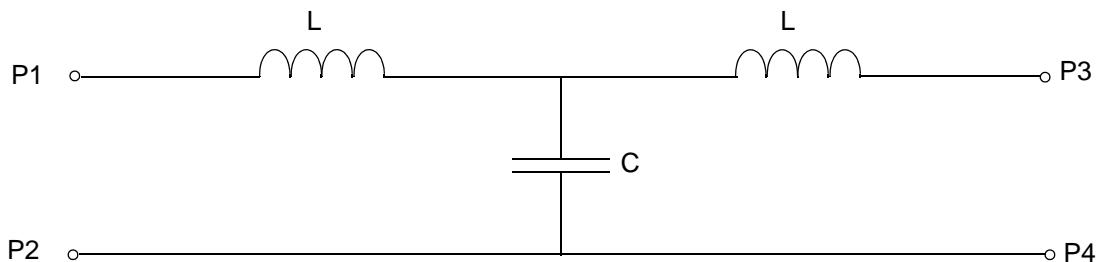
$$\frac{C}{H} = \frac{W}{H} \left[ 7.6 Er + 3.8 + \frac{W}{H} (3.93 Er + 0.62) \right] \quad \frac{\text{pF}}{\text{m}}$$

$$\frac{L}{H} = 441.2712 \left\{ 1 - 1.062 \exp \left[ -0.177 \left( \frac{W}{H} \right)^{0.947} \right] \right\} \quad \frac{\text{pF}}{\text{m}}$$

## Notes

The model parameters validity ranges were tested at the corners and some typical design values.

**Figure 4-14. Equivalent circuit MBEND2**



## Example

A simple MBEND2 s-parameter extraction example over a range of frequencies:

```
.param H = 1.6e-3
```

```

.param W  = 2.0e-3
.param Er = 4
.param fx = 1e9

Ymbend2 MBEND2 in 0 out 0 PARAM: H=H W=W Er=Er

*** S-Parameters Extraction

Vin in 0 IPORT=1 RPORT=50 FOUR fund1 PdBm (1) -100 -90
Vout out 0 IPORT=2 RPORT=50 FOUR fund1 PdBm (1) -100 -90

.step param fx 1e9 7e9 100e6

.sst fund1=fx nharm1=1

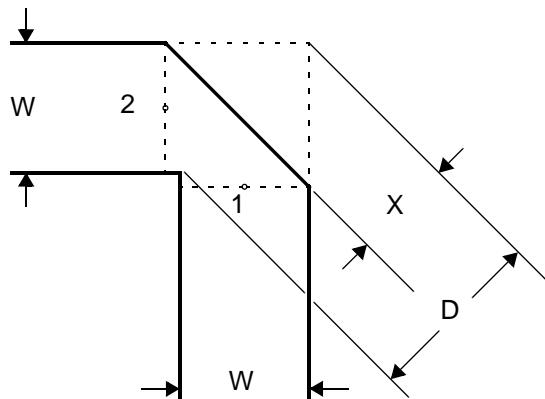
.extract fsst label=S11_Mag yval(SM(1,1),fx)
.extract fsst label=S12_Mag yval(SM(1,2),fx)

.end

```

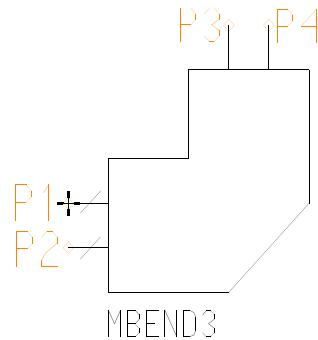
## MBEND3: 90-degree Microstrip Bend (Optimally Mitered)

**Figure 4-15. 90-degree Microstrip Bend (Optimally Mitered)**



### Symbol

**Figure 4-16. 90-degree Microstrip Bend (Optimally Mitered) Symbol**



## Syntax

```
Yxx MBEND3 P1 P2 P3 P4 PARAM: [W=val] [H=val] [ER=val] [T=val]
+ [RHO=val] [TAND=val]
```

## Parameters

**Table 4-15. 90-degree Microstrip Bend (Optimally Mitered) Parameters**

Parameter	Definition	Default	Units
W	Conductor width	2.0e-3	meter
H	Dielectric thickness	1.6e-3	meter
ER	Dielectric relative permittivity	4	-
T	Conductor thickness	5.0e-6	meter
RHO	Conductor resistivity	1.7e-8	Ohm.meter
TAND	Dielectric loss tangent	0.0	-

## Model validity range

$$0.25 \leq W/H \leq 2.75$$

$$2.5 \leq ER \leq 25$$

Simulation frequency < 15/h (Frequency in GHz, H in mm)

## Simulation domains

DC, AC, TRANSIENT, and SST

## References

Brian C. Wadell, "Transmission Line Design Handbook", 1991 Artech House.

## Equations

The optimal miter is given by:

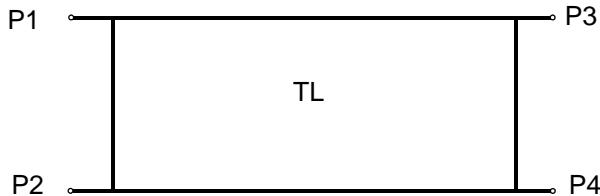
$$M = 52 + 65e^{-1.35\left(\frac{W}{H}\right)} = \frac{100 X}{d} \quad (\%)$$

and is modeled as a transmission line of length:

$$L = W \left[ 1.04 + 1.3e^{-1.35\left(\frac{W}{H}\right)} \right] \text{ m}$$

The following figure illustrates the model equivalent circuit and pins connections:

**Figure 4-17. Equivalent Circuit MBEND3**



## Example

A simple MBEND3 s-parameter extraction example over a range of frequencies:

```

.param W      = 2.0e-3
.param H      = 1.6e-3
.param Er     = 4
.param T      = 5.0e-6
.param RHO    = 1.7e-8
.param TAND   = 0
.param fx     = 5e9

Ymbend3 MBEND3 in 0 out 0 PARAM: W=W H=H Er=Er T=T RHO=RHO
+ TAND=TAND

*** S-Parameters Extraction

Vin in 0 IPORT=1 RPORT=50 FOUR fund1 PdBm (1) -100 -90
Vout out 0 IPORT=2 RPORT=50 FOUR fund1 PdBm (1) -100 -90

.step param fx 1e9 7e9 100e6

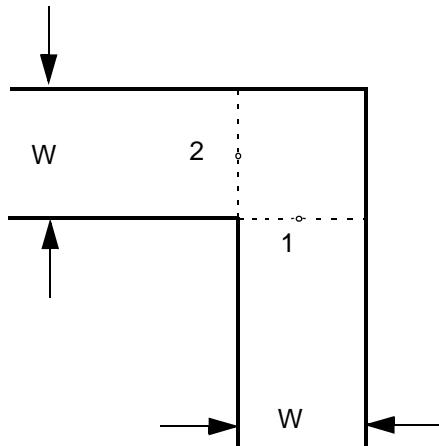
.sst fund1=fx nharm1=1

.extract fsst label=S11_Mag yval(SM(1,1),fx)
.extract fsst label=S12_Mag yval(SM(1,2),fx)

.end
    
```

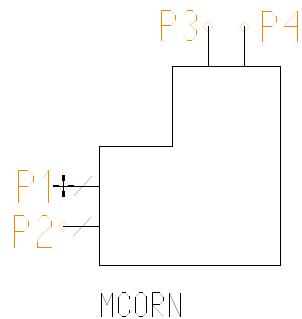
## MCORN: 90-degree Microstrip Bend (Unmitered)

Figure 4-18. 90-degree Microstrip Bend (Unmitered)



### Symbol

Figure 4-19. 90-degree Microstrip Bend (Unmitered) Symbol



### Syntax

```
Yxx MCORN P1 P2 P3 P4 PARAM: [W=val] [H=val] [Er=val]
```

### Parameters

Table 4-16. 90-degree Microstrip Bend (Unmitered) Parameters

Parameter	Definition	Default	Units
H	Substrate thickness	1.6e-3	meter
W	Conductor width	2.0e-3	meter
ER	Dielectric constant	4	-

## Model validity range

$$0.1 \leq W/H \leq 6$$

$$2 \leq ER \leq 15$$

## Simulation domains

DC, AC, TRANSIENT, and SST

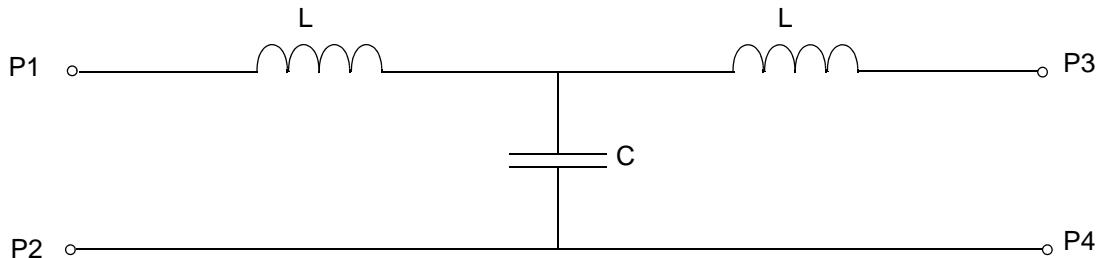
## References

M. Kirschning, R. H. Jansen, and N. H. L. Koster. "Measurement and Computer-Aided Modeling of Microstrip Discontinuities by an Improved Resonator Method," 1983 IEEE MTT-S International Microwave Symposium Digest, May 1983, pp. 495-497.

## Equations

The equivalent circuit of a microstrip corner is a lumped network of two inductors and a capacitor, as shown in [Figure 4-20](#).

**Figure 4-20. Equivalent circuit Microstrip corner**



The following equations are used to calculate the values of the model lumped components:

$$L = \left( 1 - 1.35 \times e^{\left\{ -0.18 \left( \frac{W}{H} \right)^{1.39} \right\}} \right) \times 0.2 \quad \text{nH}$$

$$C = \left\{ (10.35 \text{ } Er + 0.25) \left( \frac{W}{H} \right)^2 + (2.6 \text{ } Er + 5.44) \left( \frac{W}{H} \right) \right\} 0.001 \times H \quad \text{pF}$$

where H is in mm.

## Example

A simple MCORN s-parameter extraction example over a range of frequencies:

```

.param H  = 1.6e-3
.param W  = 2.0e-3
.param Er = 4
.param fx = 5e9

Ymcorn MCORN in 0 out 0 PARAM: H=H W=W Er=Er

*** S-Parameters Extraction

Vin in 0 IPORT=1 RPORT=50 FOUR fund1 PdBm (1) -100 -90
Vout out 0 IPORT=2 RPORT=50 FOUR fund1 PdBm (1) -100 -90

.step param fx 1e9 7e9 100e6

.sst fund1=fx nharm1=1

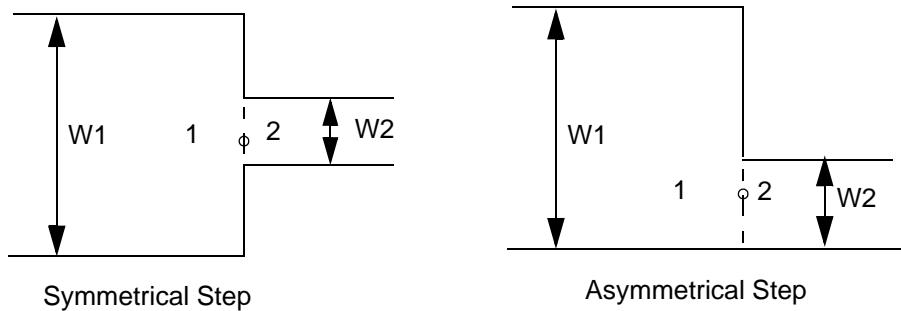
.extract fsst label=S11_Mag_Eldo yval(SM(1,1),fx)
.extract fsst label=S12_Mag_Eldo yval(SM(1,2),fx)

.end

```

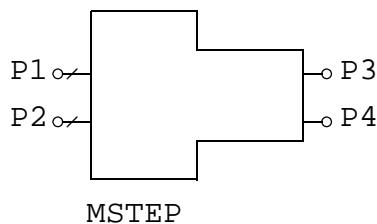
## MSTEP: Microstrip Step in Width

**Figure 4-21. Microstrip Step in Width**



## Symbol

**Figure 4-22. Microstrip Step in Width Symbol**



## Syntax

```

Yxx MSTEP P1 P2 P3 P4 PARAM: [W1=val] [W2=val] [ER=val]
+ [H=val] [F=val] [ASYMMETRICAL=val] [T=val]

```

## Parameters

**Table 4-17. Microstrip Step in Width Parameters**

Parameter	Definition	Default	Units
W1	Conductor width at port 1	2.0e-3	meter
W2	Conductor width at port 2	0.5e-3	meter
H	Substrate thickness	1.6e-3	meter
ER	Relative Dielectric constant	4	-
T	Conductor thickness	5.0e-6	meter
ASYMMETRICAL <sup>a</sup>	Selects between symmetrical and asymmetrical step structures	0	-
F	Operating frequency	1e9	Hz

a. ASYMMETRICAL only takes two values, 1 for asymmetrical, and 0 for symmetrical step.

## Simulation domains

DC, AC, TRANSIENT, and SST

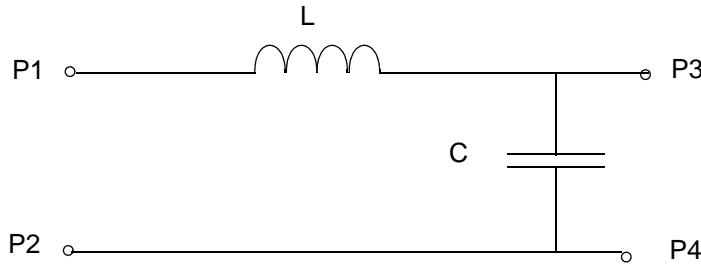
## References

Brian C. Wadell, "Transmission Line Design Handbook", 1991 Artech House.

## Equations

The equivalent circuit of a microstrip step is a lumped network of an inductor and a capacitor, as shown in [Figure 4-23](#).

**Figure 4-23. Equivalent circuit of a microstrip step in width**



The model equations to calculate the lumped component values are given in the mentioned reference.

## Example

A simple MSTEP s-parameter extraction example over a range of frequencies:

```
.param W1      = 2.0e-3
.param W2      = 0.5e-3
.param H       = 1.6e-3
.param ER      = 4
.param Frequency = 1e9
.param T       = 5.0e-6

Ymstep MSTEP t1a 0 t1b 0 PARAM: W1=W1 W2=W2 ER=ER H=H F=Frequency
ASYMMETRICAL=0 T=T

*** S-Parameters Extraction

V1a t1a 0 IPORT=1 RPORT=50 FOUR fund1 PdBm (1) -100 -90
V1b t1b 0 IPORT=2 RPORT=50 FOUR fund1 PdBm (1) -100 -90

.step param Frequency 1e9 5e9 100e6

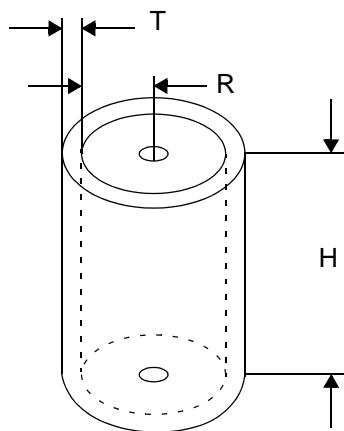
.sst fund1=Frequency nharm1=1

.extract fsst label=S11_Mag yval(SM(1,1),Frequency)
.extract fsst label=S12_Mag yval(SM(1,2),Frequency)

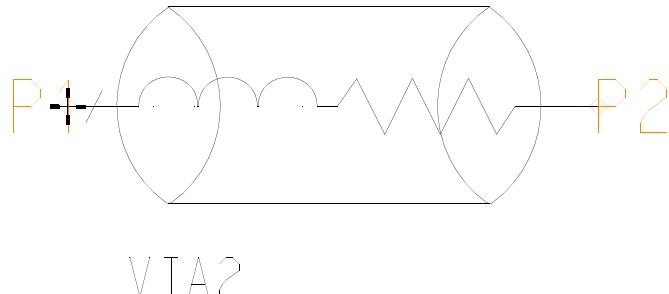
.end
```

## VIA2: Cylindrical Via Hole in Microstrip

Figure 4-24. Cylindrical Via Hole in Microstrip



## Symbol

**Figure 4-25. Cylindrical Via Hole in Microstrip Symbol**

## Syntax

```
Yxx VIA2 P1 P2 PARAM: [H=val] [R=val] [COND=val] [T=val] [F=val]
```

## Parameters

**Table 4-18. Cylindrical Via Hole in Microstrip Parameters**

Parameter	Definition	Default	Units
H	Substrate thickness	200.0e-6	meter
R	Via radius	100.0e-6	meter
COND	Conductor conductivity	58.842e6	1/(Ohm.meter)
T	Conductor thickness	5.0e-6	meter
F	Operating center frequency	1.0e9	Hz

## Model validity range

$100\mu\text{m} < H < 635\mu\text{m}$

$0.1 < R/H < 1.5$

$0 < T < R$

## Simulation domains

DC, AC, TRANSIENT, and SST

## References

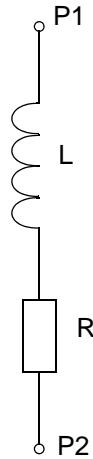
M. Goldfarb and R. Pucel. "Modeling Via Hole Grounds in Microstrip," IEEE Microwave and Guided Wave Letters, Vol. 1, No. 6, June, pp.135-137.

## Notes

The VIA2 is modeled as a series resistor and inductor network. The resistor and inductor values are based on equations given in the mentioned reference.

Figure 4-26 illustrates the model equivalent circuit and pins connections:

**Figure 4-26. Equivalent circuit Cylindrical Via Hole in Microstrip**



## Example

A simple VIA2 s-parameter extraction example over a range of frequencies:

```
.param H      = 200e-6
.param R      = 100e-6
.param COND   = 58.824e6
.param T      = 5.0e-6
.param fx     = 5e9

Yvia2 VIA2 in out PARAM: H=H R=R COND=COND T=T F=fx

*** S-Parameters Extraction

Vin in 0 IPORT=1 RPORT=50 FOUR fund1 PdBm (1) -100 -90
Vout out 0 IPORT=2 RPORT=50 FOUR fund1 PdBm (1) -100 -90

.step param fx 1e9 7e9 100e6

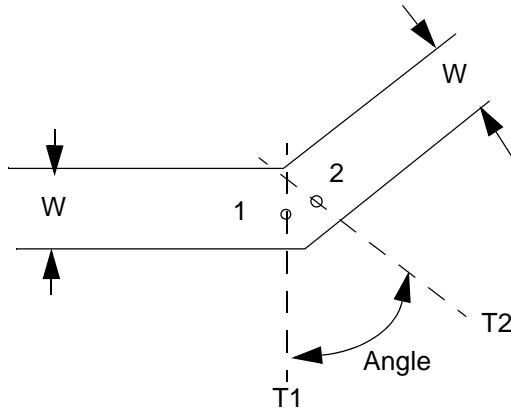
.sst fund1=fx nharm1=1

.extract fsst label=S11_Mag yval(SM(1,1),fx)
.extract fsst label=S12_Mag yval(SM(1,2),fx)

.end
```

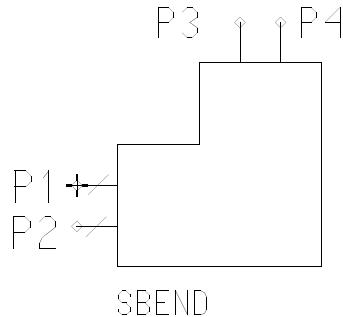
## SBEND: Unmitered Stripline Bend

**Figure 4-27. Unmitered Stripline Bend**



### Symbol

**Figure 4-28. Unmitered Stripline Bend Symbol**



### Syntax

```
Yxx SBEND P1 P2 P3 P4 PARAM: [W=val] [B=val] [ER=val] [T=val]
+ [ANGLE=val] [F=val]
```

### Parameters

**Table 4-19. Unmitered Stripline Bend Parameters**

Parameter	Definition	Default	Units
W	Conductor width	0.1e-3	meter
B	Ground plane spacing	280e-6	meter
T	Conductor thickness	17e-6	meter
ER	Relative dielectric constant	4.2	-

**Table 4-19. Unmitered Stripline Bend Parameters**

Parameter	Definition	Default	Units
ANGLE	Bend angle	60	degree
F	Simulation frequency	2e9	Hz

## Simulation domains

DC, AC, TRANSIENT, and SST

## References

Altschuler, H.M., and A.A. Oliner, "Discontinuities in the Center Conductor of Symmetric Strip Transmission Line," IRE Transactions on Microwave Theory and Techniques, Vol. MTT-8, May 1960, pp. 328-339 and "Addendum to 'Discontinuities in the Center Conductor of Symmetric Strip Transmission Line,'" Vol. MTT-10, No. 2, March 1962, p. 143.

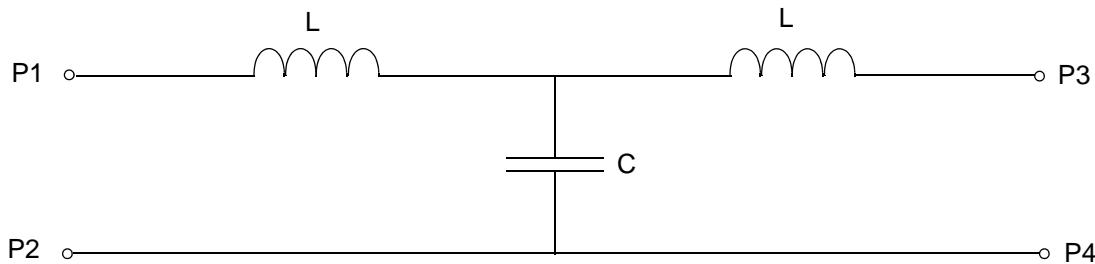
K.C. Gupta, "Computer-Aided Design of Microwave Circuits", 1981, ARTECH HOUSE, INC.

Arthur A. Oliner, "Equivalent Circuits for Discontinuities in balanced Strip Transmission Line", Microwave Theory and Techniques, IEEE Transactions on, Volume: 3 Issue: 2, Mar 1955.

## Notes

The equivalent circuit of an unmitered stripline bend is a lumped network of two inductors and a capacitor whose values are based on equations given in the mentioned references. The equivalent circuit is shown in [Figure 4-29](#).

**Figure 4-29. Equivalent circuit of an unmitered stripline bend**



## Example

A simple SBEND s-parameter extraction example over a range of frequencies:

```
.param W      = 0.1e-3
.param B      = 280e-6
```

```

.param ER      = 4.2
.param T       = 17e-6
.param ANGLE   = 60
.param F       = 2e9

Ysbend SBEND in 0 out 0 PARAM: W=W B=B ER=ER T=T ANGLE=ANGLE F=F

*** S-Parameters Extraction

Vin in 0 IPORT=1 RPORT=50 FOUR fund1 PdBm (1) -100 -90
Vout out 0 IPORT=2 RPORT=50 FOUR fund1 PdBm (1) -100 -90

.step param f 1e9 5e9 100e6

.sst fund1=f nharm1=1

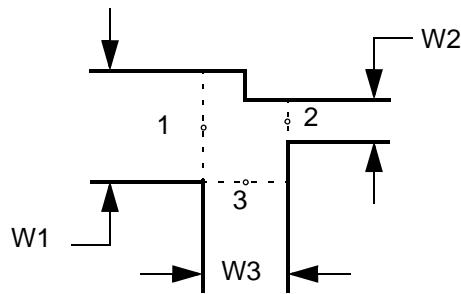
.extract fsst label=S11_Mag yval(SM(1,1),f)
.extract fsst label=S12_Mag yval(SM(1,2),f)

.end

```

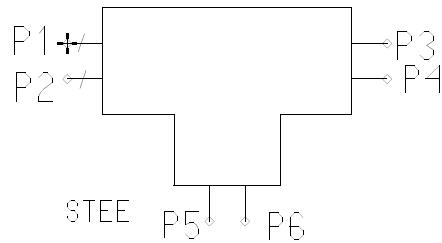
## STEE: Stripline T Junction

**Figure 4-30. Stripline T Junction**



## Symbol

**Figure 4-31. Stripline T Junction Symbol**



## Syntax

```

Yxx STEE P1 P2 P3 P4 P5 P6 PARAM: [W1=val] [W2=val] [W3=val]
+ [B=val] [ER=val] [T=val] [F=val]

```

## Parameters

**Table 4-20. Stripline T Junction Parameters**

Parameter	Definition	Default	Units
W1	First arm conductor width	0.1e-3	meter
W2	Second arm conductor width	0.1e-3	meter
W3	Third arm conductor width	0.2e-3	meter
B	Ground plane spacing	280e-6	meter
T	Conductor thickness	17e-6	meter
ER	Relative dielectric constant	4.2	-
F	Simulation frequency	2e9	Hz

## Simulation domains

DC, AC, TRANSIENT, and SST

## References

Altschuler, H.M., and A.A. Oliner, "Discontinuities in the Center Conductor of Symmetric Strip Transmission Line," IRE Transactions on Microwave Theory and Techniques, Vol. MTT-8, May 1960, pp. 328-339 and "Addendum to 'Discontinuities in the Center Conductor of Symmetric Strip Transmission Line,'" Vol. MTT-10, No. 2, March 1962, p. 143.

K.C. Gupta, "Computer-Aided Design of Microwave Circuits", 1981, ARTECH HOUSE, INC.

Arthur A. Oliner, "Equivalent Circuits for Discontinuities in balanced Strip Transmission Line", Microwave Theory and Techniques, IEEE Transactions on, Volume: 3 Issue: 2, Mar 1955.

## Notes

The model is based on the microstrip line symmetric T junction equations given in the mentioned references.

The model handles symmetrical T-junction only. If the specified W1 and W2 parameters are not identical, the geometrical mean of W1 and W2 parameters is computed and used.

$$W_1 = W_2 = \sqrt{W_1 \cdot W_2} \text{ for non-symmetrical T-junction}$$

## Example

A simple STEE s-parameter extraction example over a range of frequencies:

```

.param W1      = 0.1e-3
.param W2      = 0.1e-3
.param W3      = 0.2e-3
.param B       = 280e-6
.param ER      = 4.2
.param T       = 17e-6
.param frequency = 2e9

Ystee STEE tla 0 t1b 0 t2 0 PARAM: W1=W1 W2=W2 W3=W3 B=B ER=ER T=T
F=frequency

*** S-Parameters Extraction

V1a  t1a  0 IPORT=1 RPORT=50 FOUR fund1 PdBm (1) -100 -90
V1b  t1b  0 IPORT=2 RPORT=50 FOUR fund1 PdBm (1) -100 -90
V2   t2   0 IPORT=3 RPORT=50 FOUR fund1 PdBm (1) -100 -90

.step param frequency 1e9 5e9 100e6

.sst fund1=frequency nharm1=1

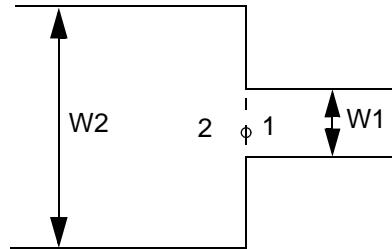
.extract fsst label=S11_Mag yval(SM(1,1),frequency)
.extract fsst label=S12_Mag yval(SM(1,2),frequency)
.extract fsst label=S13_Mag yval(SM(1,3),frequency)
.extract fsst label=S33_Mag yval(SM(3,3),frequency)

.end

```

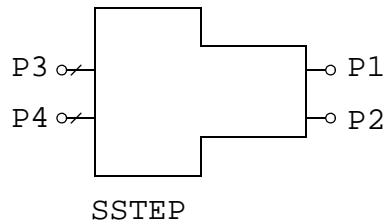
## SSTEP: Stripline Step in Width

**Figure 4-32. Stripline Step in Width**



### Symbol

**Figure 4-33. Stripline Step in Width Symbol**



## Syntax

```
Yxx SSTEP P1 P2 P3 P4 PARAM: [W1=val] [W2=val] [B=val] [T=val]  
+ [ER=val] [F=val]
```

## Parameters

**Table 4-21. Stripline Step in Width Parameters**

Parameter	Definition	Default	Units
W1	Conductor width at port 1	0.10e-3	meter
W2	Conductor width at port 2	0.15e-3	meter
B	Ground plane spacing	280e-6	meter
T	Conductor thickness	17e-6	meter
ER	Relative dielectric constant	4.2	-
F	Simulation frequency	1e9	Hz

## Simulation domains

DC, AC, TRANSIENT, and SST

## References

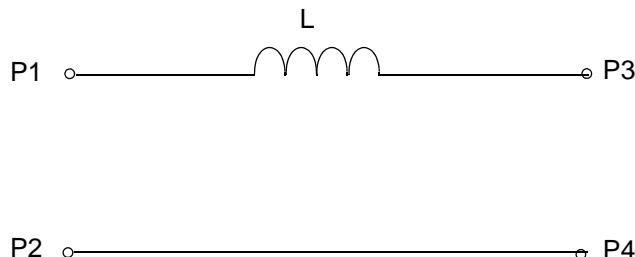
Brian C. Wadell, "Transmission Line Design Handbook", 1991 Artech House.

## Notes

The SSTEP is modeled as a series inductor. The inductor value is based on equations given in the mentioned reference.

[Figure 4-34](#) illustrates the model equivalent circuit and pins connections:

**Figure 4-34. Equivalent circuit for a Stripline Step in Width**



## Example

A simple SSTEP s-parameter extraction example over a range of frequencies:

```
.param W1      = 0.10e-3
.param W2      = 0.15e-3
.param B       = 280e-6
.param T       = 17e-6
.param ER      = 4.2
.param frequency = 1e9

Ysstep SSTEP t1a 0 t1b 0 PARAM: W1=W1 W2=W2 B=B T=T ER=ER F=frequency

*** S-Parameters Extraction

V1a t1a 0 IPORT=1 RPORT=50 FOUR fund1 PdBm (1) -100 -90
V1b t1b 0 IPORT=2 RPORT=50 FOUR fund1 PdBm (1) -100 -90

.step param frequency 1e9 5e9 100e6

.sst fund1=frequency nharm1=1

.extract fsst label=S11_Mag yval(SM(1,1),frequency)
.extract fsst label=S12_Mag yval(SM(1,2),frequency)

.end
```

## Junction Diode

```
DXX NP NN [NM] MNAME [[AREA=]AREA_VAL] [PERI=PERIVAL] [PGATE=PGATE_VAL]  
+ [T[EMP]=VAL] [DTEMP=VAL] [M=VAL] [OFF=0|1] [NOISE=0|1] [NONOISE]  
DXX NP NN [NM] MNAME [FMIN=VAL] [FMAX=VAL] [NBF=VAL]
```

### Parameters

- **XX**  
Junction diode name.
- **NP**  
Name of the positive node.
- **NN**  
Name of the negative node.
- **NM**  
Name of the **M** node. Any number of pins can be specified. However, only some proprietary models will make use of additional pins, if any.
- **MNAME**  
Name of the model used, as described in a **.MODEL** command.
- **AREA=AREA\_VAL**  
Model area factor. Default value is 1.
- **PERI=PERIVAL**  
Perimeter of the diode. Default value is 0.
- **PGATE=PGATE\_VAL**  
Length of the diode gate-edge. Default value is 0 (used for JUNCAP model only).
- **T=TVAL**  
Sets temperature for the individual diode. Default nominal temperature is 27 °C.
- **M=VAL**  
Device multiplier, simulating the effect of multiple devices in parallel. All currents, capacitances and resistances are affected by **M**. Default value is 1.  
The device is first evaluated without the **M** factor, and at the very end of the device computation, all scaling quantities are multiplied / divided by **M**. Input values **W** and **L** are not affected. Models are chosen depending on input **W** and **L**, if required. Options **MINL**, **MAXL**, **MINW**, **MAXW**, etc. do not apply either, since they check the input values of **W** and **L**.

---

 **Note**

Using an M factor value less than 1 could lead to simulating devices that cannot be physically realized.

---

- **T [EMP ]=VAL**

Sets temperature for the individual device, in degrees Celsius. Default nominal temperature=27°C.

- **DTEMP=VAL**

Temperature difference between the device and the rest of the circuit, in degrees Celsius. Default value is 0.0.

---

 **Note**

**TEMP** and **DTEMP** are mutually exclusive. If both are specified, the last one is utilized.

---

- **OFF=0 | 1**

When set to 1, causes no initial operating point to be calculated for the device during DC analysis, i.e. the device is “off”. When set to 0, the option is ignored.

- **NONOISE**

Specifies that no noise model will be used for this device when performing noise analysis. Therefore, the device presents no noise contribution to the noise analysis.

- **NOISE=0|1**

When set to 0, is equivalent to specifying the **NONOISE** flag. When set to 1, the option is ignored.

- **FMIN=VAL**

Lower limit of the noise frequency band.

- **FMAX=VAL**

Upper limit of the noise frequency band.

- **NBF=VAL**

Specifies the number of sinusoidal sources with appropriated amplitude and frequency and with randomly distributed phase from which the noise source is composed. Default value is 50. This parameter has no effect when **FMIN** is set to 0.

**Note**

 **FMIN** and **FMAX** define the frequency band of the noise sources. This frequency range may sometimes not correspond to the noise frequency band at the output of the circuit. For instance, the band (**FMIN**, **FMAX**) does not correspond to the output noise frequency band in the case of filters or oscillators and mixers that exhibit frequency conversion. **FMIN** is also used to specify the algorithm used to generate the noise source generated by the diode. When **FMIN**>0 the diode noise source is generated with sinusoids; when **FMIN**=0 it is generated with a continuous spectrum between **FMIN** and **FMAX**.

---

## Handling floating gates

When **.OPTION FLOATGATECHECK** is set, will enable Eldo to issue a warning when a floating gate is detected and resume the simulation. Eldo will not consider reverse-biased diodes as active elements when checking for floating gates. **.OPTION FLOATGATERR** can be set to enable Eldo to issue an error when a floating gate is detected and stop the simulation. In addition Eldo can force detected floating gates to 0 using **.OPTION FLOATGATE0**, which can be useful to change the topology of a circuit to achieve better convergence.

For more information please refer to [page 11-26](#).

## Combining identical Diodes

When **.OPTION REDUCE** is set, and when multiple identical diodes follow each other, those diodes are reduced into a single instance using the **M** parameter, for example:

```
d1 1 2 diode1
d2 1 2 diode1
d3 1 3 diode1
.OPTION REDUCE
```

Here, diode instances **d1** and **d2** will be replaced by:

```
d1 1 2 diode1 M = 2
```

but **d3** will remain as it is because it is connected to different nodes.

**Note**

 It will only work when no parameters are given on the instance.

---



This also applies to BJTs, MOSFETs and sub circuits. For more information see [page 4-111](#), [page 4-127](#) and [page 4-172](#) respectively.

---

## Noise in Diodes



Noise models are available for diodes, see “[Noise Equations for All Levels](#)” on page 1-39 of the *Eldo Device Equations Manual*.

### Example

```
d1 1 2 diode1 T=50
```

Specifies a diode **d1** placed between nodes 1 and 2 of model name **diode1**, and at temperature of 50°C.

## Diode Model Syntax

```
.MODEL MNAME D [PAR=VAL]
```

Eldo provides several predefined diode models. The **LEVEL** parameter is placed first in the parameter list and specifies the model to be used. The options are listed in the table below. Parameters specific to the selected model are then assigned values following the **LEVEL** parameter. Default diode level is 1.

**Table 4-22. Diode Models**

LEVEL Value	Model Name
1	Berkeley Level 1 (Eldo Level 1)
2	Modified Berkeley Level 1 (Eldo Level 2)
3	Fowler-Nordheim Model (Eldo Level 3)
4	STMicroelectronics LEVEL 1
5	STMicroelectronics LEVEL 2
6	STMicroelectronics LEVEL 3
8 <sup>a</sup>	JUNCAP (Eldo Level 8) ( <b>DIOLEV</b> =9)
8 <sup>b</sup>	JUNCAP2 (Eldo Level 8, <b>DIOLEV</b> =11)
9	Philips Diode Level 500 (Eldo Level 9)
21	Diode Level 21

a. Diode LEVEL 8 has a selector (**DIOLEV**) which when set to **DIOLEV**=9, the MOS Model 9 model is used.

b. Diode LEVEL 8 has a selector (**DIOLEV**) which when set to **DIOLEV**=11, the JUNCAP2 model is used.

**i** Please refer to the [Diode Model Equations](#) of the *Eldo Device Equations Manual*.

---

## Using -compat with Diodes

Using the **-compat** runtime flag or selecting **.option compat** has the following effect on Diode Level values.

**Table 4-23. Diode Models with -compat**

LEVEL	Model Name
2	Fowler-Nordheim (Eldo LEVEL 3)
3	Berkeley Level 1 (Eldo LEVEL 1), by default SCALEV is set to 3
4	JUNCAP Diode Model (Eldo LEVEL 8), by default DIOLEV is set to 9

## Berkeley Level 1 (Eldo Level 1)

The DC characteristics of the diode are determined by the parameters **IS** and **N**. An ohmic resistance, **RS**, is included. Charge storage effects are modeled by a transit time, **TT**, and a non-linear depletion layer capacitance which is determined by the parameters **CJO**, **VJ** and **M**. The temperature dependence of the saturation current is defined by the parameters **EG**, the energy and **XTI**, and the saturation current temperature exponent. Reverse breakdown is modeled by an exponential increase in the reverse diode current and is determined by the parameters **BV** and **IBV** (both of which are positive numbers).

**i** For model parameters, please refer to the [Berkeley Level 1 Model \(Eldo Level 1\) Parameters](#) of the *Eldo Device Equations Manual*.

---

## Modified Berkeley Level 1 (Eldo Level 2)

The Eldo Level 2 model is based on the Berkeley diode model. It includes modified calculations for the reverse breakdown effects, the recombination effect and for the temperature behavior.

**i** For model parameters, please refer to the [Modified Berkeley Level 1 Model \(Eldo Level 2\) Parameters](#) of the *Eldo Device Equations Manual*.

---

## Fowler-Nordheim Model (Eldo Level 3)

Fowler-Nordheim diode models are formed as a metal-insulator-semiconductor device or as a semiconductor-insulator-semiconductor layer device. The insulator is sufficiently thin to permit

the tunneling of carriers. This element models electrically-alterable memory cells, air-gap switches, and other insulation breakdown devices.



For model parameters, please refer to the [Fowler-Nordheim Model \(Eldo Level 3\) Parameters](#) of the *Eldo Device Equations Manual*.

## Example

```
*DIODE model definition
.model dio d level=3
...
*main circuit
d1 2 10 dio
```

Specifies the diode **d1** placed between the nodes 2 and 10 of model name **dio**. Default parameter values are used for this Fowler-Nordheim model.

```
*DIODE model definition
.model diodel d rs=4.68 bv=6.10 cjo=346p
+ tt=50n m=0.33 vj=0.75 is=1e-11 n=1.27
+ ibv=20ma
...
*main circuit
dbridge 2 10 diodel
```

Specifies the diode **dbridge** placed between nodes 2 and 10. The electrical parameters are defined in the **diodel** model.

### Note

**LEVEL** is not defined and so the Berkeley Level 1 diode model is used (**LEVEL=1**).

## JUNCAP (Eldo Level 8)

The JUNCAP model is the replica of the bulk-source/drain parasitic diode model included in the common MOS structure. This model may be used to define the bulk diode as an external component or to define new elements such as the substrate diode with similar behaviors.

LEVEL 8 has a selector (**DIOLEV**) which when set to **DIOLEV=9** means the MOS Model 9 model is used.



For model parameters, please refer to the [JUNCAP Model \(Eldo Level 8\) Parameters](#) of the *Eldo Device Equations Manual*.

## JUNCAP2 (Eldo Level 8, DIOLEV=11)

The JUNCAP2 model (LEVEL=8, DIOLEV=11) is the evolution of the standard JUNCAP1 junction diode model (LEVEL=8, DIOLEV=9).



For model parameters, please refer to the [JUNCAP2 Model \(Eldo Level 8, DIOLEV=11\) Parameters](#) of the *Eldo Device Equations Manual*.

---

## Philips Diode Level 500 (Eldo Level 9)

The Philips Diode Level 500 model provides a detailed description of the diode currents in forward and reverse biased Si-diodes. It is meant to be used for DC, transient and AC analysis.



For model parameters, please refer to the [Philips Diode Level 500 Model \(Eldo Level 9\) Parameters](#) of the *Eldo Device Equations Manual*.

---

## Diode Level 21

Level 21 is a combination of the Level 9 current equations and Level 1 capacitance model with a mix in the temperature dependencies of both models.



For model parameters, please refer to the [Diode Model Level 21 Parameters](#) of the *Eldo Device Equations Manual*.

---

## BJT—Bipolar Junction Transistor

```
Qxx NC NB NE [NS] [TH] MNAME [[AREA=]AREA_VAL] [AREAB=AREA_VAL]
+ [AREAC=AREA_VAL] [T[EMP]=VAL] [DTEMP=VAL] [M=VAL] [OFF=0|1]
+ [NOISE=0|1] [NONOISE]
Qxx NC NB NE [NS] MNAME [FMIN=VAL] [FMAX=VAL] [NBF=VAL]
```

### Parameters

- **xx**  
Name of the bipolar junction transistor.
- **NC**  
Name of the collector node.
- **NB**  
Name of the base node.
- **NE**  
Name of the emitter node.
- **MNAME**  
Name of the model used, as described in a **.MODEL** command.
- **NS**  
Substrate node. Default value is ground. This node can be specified in the **.MODEL** card as **BULK=<node\_name>**.
- **TH**  
Thermal node, available only for the HICUM model (level 9). Voltage in this node represents the temperature increase of the device due to self-heating.
- **AREA=AREA\_VAL**  
Relative device area. Default value is 1.
- **AREAB=AREA\_VAL**  
Base relative device area. Default is **AREA**.
- **AREAC=AREA\_VAL**  
Collector relative device area. Default is **AREA**.
- **T=VAL**  
Sets temperature for the individual BJT. Default value is the current simulation temperature.
- **M=VAL**  
Device multiplier, simulating the effect of multiple devices in parallel. All currents, capacitances and resistances are affected by **M**. Default value is 1.

The device is first evaluated without the **M** factor, and at the very end of the device computation, all scaling quantities are multiplied / divided by **M**. Input values **W** and **L** are not affected. Models are chosen depending on input **W** and **L**, if required. Options **MINL**, **MAXL**, **MINW**, **MAXW**, etc. do not apply either, since they check the input values of **W** and **L**.

**Note**



Using an M factor value less than 1 could lead to simulating devices that cannot be physically realized.

---

- **T [EMP ]=VAL**

Sets temperature for the individual device, in degrees Celsius. Default nominal temperature=27°C.

- **DTEMP=VAL**

Temperature difference between the device and the rest of the circuit, in degrees Celsius. Default value is 0.0.

**Note**



**TEMP** and **DTEMP** are mutually exclusive. If both are specified, the last one is utilized.

---

- **OFF=0 | 1**

When set to 1, causes no initial operating point to be calculated for the device during DC analysis, i.e. the device is “off”. When set to 0, the option is ignored.

- **NONOISE**

Specifies that no noise model will be used for this device when performing noise analysis. Therefore, the device presents no noise contribution to the noise analysis.

- **NOISE=0|1**

When set to 0, is equivalent to specifying the **NONOISE** flag. When set to 1, the option is ignored.

- **FMIN=VAL**

Lower limit of the noise frequency band.

- **FMAX=VAL**

Upper limit of the noise frequency band.

- **NBF=VAL**

Specifies the number of sinusoidal sources with appropriated amplitude and frequency and with randomly distributed phase from which the noise source is composed. Default value is 50. This parameter has no effect when **FMIN** is set to 0.

---

**Note**

 **FMIN** and **FMAX** define the frequency band of the noise sources. This frequency range may sometimes not correspond to the noise frequency band at the output of the circuit. For instance, the band (**FMIN**, **FMAX**) does not correspond to the output noise frequency band in the case of filters or oscillators and mixers that exhibit frequency conversion. **FMIN** is also used to specify the algorithm used to generate the noise source generated by the BJT. When **FMIN**>0 the BJT noise source is generated with sinusoids; when **FMIN**=0 it is generated with a continuous spectrum between **FMIN** and **FMAX**.

---

## Example

```
q1 1 2 3 bipoll t=50
```

Specifies a bipolar junction transistor **q1** connected to nodes 1, 2 and 3, of model type **bipoll**, at a temperature of 50°C.

## Combining identical BJTs

When **.OPTION REDUCE** is set, multiple identical BJTs that follow each other are reduced into a single instance using the **M** parameter, for example:

```
q1 1 2 3 bipoll
q2 1 2 3 bipoll
q3 1 2 4 bipoll
.OPTION REDUCE
```

Here, BJT instances **q1** and **q2** will be replaced by:

```
q1 1 2 3 bipoll M=2
```

but **q3** will remain as it is because the nodes are different.

---

**Note**

 This will also work even if the **AREA** parameter is specified. If two BJTs are merged and in addition have the same connectivity, the models will also have the same **AREA**.

---



This also applies to MOSFETs, diodes and subcircuits. For more information see [page 4-127](#), [page 4-104](#) and [page 4-172](#) respectively.

---

## BJT Model Syntax

```
.MODEL MNAME NPN SUBS=VAL [ PAR=VAL ]
.MODEL MNAME PNP SUBS=VAL [ PAR=VAL ]
.MODEL MNAME LPNP SUBS=VAL [ PAR=VAL ]
```

Both NPN and PNP model types are to be used for vertical structures. This means that most of the current flows in a vertical direction. With regard to a silicon integrated transistor, this flow can also be parallel to the surface of the silicon. This type of BJT is also called a lateral transistor. It is possible in Eldo for a **PNP** BJT model to be declared as lateral by using the **LPNP** keyword in the **.MODEL** command. The current flow can also be affected by the **SUBS** parameter as follows:

- If **SUBS=-1**, BJT is lateral.
- If **SUBS=1**, BJT is vertical.

PRECISE mode is selected by using the **.OPTION PRECISE** command. In PRECISE mode, Level 1 models describe a vertical structure, whereas Level 2 models describe a lateral structure.

The **LEVEL** parameter is used to determine which model is to be used as well as the mode of operation depending on whether Eldo or PRECISE mode is required, as shown by the table below:

**Table 4-24. BJT Models**

LEVEL	Model Name (Eldo Mode)	Structure (PRECISE Mode)
1	<a href="#">Modified Gummel-Poon Model (Eldo Level 1)</a> (Berkeley Standard Model)	Vertical
2	STMicroelectronics LEVEL 1	Lateral
3	BNR-HICUM Model	
4	<a href="#">Philips Mextram 503.2 Model (Eldo Level 4)</a>	
5	<a href="#">Improved Berkeley Model (Eldo Level 5)</a>	
7	ROCK-HICUM Model	
8	<a href="#">VBIC v1.2 Model (Eldo Level 8)</a> <b>(VERSION=1.15) VBIC v1.1.5 Model (Eldo Level 8)</b>	Vertical Vertical
9	<a href="#">HICUM Model (Eldo Level 9)</a>	
22	<a href="#">Philips Mextram 504 Model (Eldo Level 22)</a>	
23	<a href="#">Philips Modella Model (Eldo Level 23)</a>	Lateral
24	<a href="#">HICUM Level0 Model (Eldo Level 24)</a>	

## Using -compat with BJT Models

Using the `-compat` runtime flag or selecting `.OPTION COMPAT` has the following effect on BJT Level values.

**Table 4-25. BJT Models with -compat**

LEVEL	Model Name
2	Improved Berkeley Model (Eldo LEVEL 5)
2	STMicroelectronics LEVEL 1 (Eldo LEVEL 2)
4	VBIC v1.2 (Eldo LEVEL 8)
	( <code>VERSION=1.15</code> ) VBIC v1.1.5 (Eldo LEVEL 8)
6	Philips Mextram 503.2 Model (Eldo LEVEL 4)
8	HICUM Model (Eldo LEVEL 9)

If `-compat` is set, NPN are vertical, PNP are lateral.

## Noise in BJTs

Noise models are available for BJTs Level 1 and Level 5 only, see the relevant sections of the *Eldo Device Equations Manual* in “Noise” on page 2-15 and “Noise” on page 4-15 respectively. Other BJT models have their own separate noise models.

## Automatic Selection of BJT Model Via AREA Specifications

BJT model versions can be selected via `.MODEL` command `AREAMIN` and `AREAMAX` parameters. Whenever Eldo finds a BJT device for which the model name has no `.MODEL` command, it searches through all defined models for a model of the same root name and whose `AREAMIN/AREAMAX` range matches the specified device size.

The BJT model is selected if the instance parameter `AREA` is consistent with `AREAMIN/AREAMAX` of the `.MODEL` command.

### Example

```
Q1 C B S QND AREA = 10
.MODEL QND.1 NPN AREAMIN=0 AREAMAX=5
.MODEL QND.2 NPN AREAMIN=0 AREAMAX=20
```

In this example, the model selected for `Q1` will be `QND.2`.

## Modified Gummel-Poon Model (Eldo Level 1)

The bipolar junction transistor model in Eldo is an adaptation of the integral charge control model of Gummel and Poon. This modified Gummel-Poon model extends the original model to include several effects at high bias levels. The model automatically simplifies to the less complex Ebers-Moll model when using default values. Parameter names used in the modified Gummel-Poon model have been chosen to be more easily understood by the program user, and to better reflect both physical and circuit design thinking.

- 
-  Please refer to the [BJT Level 1 Equations](#) of the *Eldo Device Equations Manual*.  
For model parameters, please refer to the [Modified Gummel-Poon Model \(Eldo Level 1\) Parameters](#) of the *Eldo Device Equations Manual*.
- 

## Philips Mextram 503.2 Model (Eldo Level 4)

This is the implementation of the Philips Mextram Bipolar Model 503.2 in Eldo. The basis of this work is the unclassified report 006/94 published by Philips Nat.lab., June 1995. In this model, the current through the epilayer is an explicit and continuous function of the internal and external base-collector junction voltages. The model covers all possible modes of operation such as ohmic current flow, saturated current flow and base push out both in the forward and reverse mode of operation.

- 
-  Please refer to the [Mextram Equations](#) of the *Eldo Device Equations Manual*.  
For model parameters, please refer to the [Philips Mextram 503.2 Model \(Eldo Level 4\) Parameters](#) of the *Eldo Device Equations Manual*.
- 

## Improved Berkeley Model (Eldo Level 5)

The improved Berkeley model is based on the Level 1 model (Gummel Poon). It includes several additional effects such as:

- Variable RC resistance due to velocity saturation.
- Quasi-saturation model based on a publication from G.M. Kull et al.
- Additional temperature effects.
- Variable exponent for high current roll off.

- 
-  Please refer to the [BJT Level 5 Equations](#) of the *Eldo Device Equations Manual*.  
For model parameters, please refer to the [Improved Berkeley Model \(Eldo Level 5\) Parameters](#) of the *Eldo Device Equations Manual*.
-

## Example

```
*BJT model definition
.model qmod npn bf=160 rb=100 cjs=2p
+ tf=0.3n tr=6n cje=3p cjc=2p vaf=100
...
*main circuit
q23 10 24 13 qmod
```

Specifies the bipolar transistor `q23`, with the collector connected to node 10, base to node 24 and emitter to node 13. Electrical parameters are specified in the model `qmod`.

## VBIC v1.2 Model (Eldo Level 8)

The VBIC model is a Bipolar Junction Transistor (BJT) model. VBIC stands for **V**ertical **B**ipolar **I**ntercompany **M**odel. The VBIC model was developed as an industry-standard, public domain replacement for the SPICE Gummel-Poon (SGP) model. VBIC is designed to be as similar as possible to the SGP model, yet overcomes its major deficiencies. VBIC improvements on SGP:

- Improved Early effect modeling
- Quasi-saturation modeling
- Parasitic substrate transistor modeling
- Parasitic fixed (oxide) capacitance modeling
- Includes an avalanche multiplication model
- Improved temperature modeling
- Base current is decoupled from collector current
- Electro-thermal modeling.

Eldo recognizes two versions of the VBIC model. To select each model the **VERSION** parameter must be used as shown below:

**Table 4-26. VBIC Version Selection**

Parameter Value	VBIC Version
<b>VERSION=1.2</b>	VBIC v1.2 (default)
<b>VERSION=1.15</b>	VBIC v1.1.5

The different model parameters are described in [Parameter List for v1.2](#) and [Parameter List for v1.1.5](#).

**i** Please refer to the [VBIC Equations](#) of the *Eldo Device Equations Manual*.  
For model parameters, please refer to the [VBIC v1.2 Model \(Eldo Level 8\) Parameters](#) of the *Eldo Device Equations Manual*.

---

## VBIC v1.1.5 Model (Eldo Level 8)

**Note**

v1.1.5 of the VBIC model was formerly Level 21 in Eldo.

---

Eldo recognizes two versions of the VBIC model. To select each model the **VERSION** parameter must be used as shown above.

For information on Version 1.2 see “[VBIC v1.2 Model \(Eldo Level 8\)](#)” on page 4-115

**i** Please refer to the [VBIC Equations](#) of the *Eldo Device Equations Manual*.  
For model parameters, please refer to the [VBIC v1.1.5 Model \(Eldo Level 8\) Parameters](#) of the *Eldo Device Equations Manual*.

---

## HICUM Model (Eldo Level 9)

HICUM (HIGh Current bipolar compact transistor Model) is an advanced transistor model for bipolar transistors with main emphasis on circuit design for high-speed applications.

HICUM development resulted from the experience that the SPICE Gummel-Poon model (SGPM) is not accurate enough for high-speed large-signal transient applications and the required high collector current densities. Other major disadvantages of the SGPM are lack of sufficient physical background, poor description of base resistance and (quasi-)saturation effects.

When the model parameter **VERSION** is set to a value of 2.2, some new equations are used.  
When set to 2.1, the old equations are used.

**Table 4-27. HICUM Version Selection**

Parameter Value	HICUM Version
<b>VERSION=2.2</b>	HICUM 2.2 (Default)
<b>VERSION=2.1</b>	HICUM 2.1

**i** For HICUM v2.1, please refer to the [HICUM v2.1 Equations](#) of the *Eldo Device Equations Manual*. For HICUM v2.1 model parameters, please refer to the [HICUM v2.1 Model \(Eldo Level 9\) Parameters](#) of the *Eldo Device Equations Manual*.

---



For HICUM v2.2, please refer to the [HICUM v2.2 Equations](#) of the *Eldo Device Equations Manual*. For HICUM v2.2 model parameters, please refer to the [HICUM v2.2 Model \(Eldo Level 9\) Parameters](#) of the *Eldo Device Equations Manual*.

---

## Philips Mextram 504 Model (Eldo Level 22)

This is the implementation of the Philips Mextram Bipolar Model 504 in Eldo. The basis of this work is the unclassified report NL-UR 2000/811 published by Philips Nat.Lab., April 2001. The Mextram 504 model is implemented in Eldo as LEVEL=22.

Mextram 504 has two main improvements with respect to Mextram 503. The description of the currents and charges is much smoother as a function of bias, and the parameter extraction has been improved.

Mextram 504.1 is the new version of Mextram 504 released by Philips in September 2001 and updated in March 2002.



Please refer to the [Mextram 504 Equations](#) of the *Eldo Device Equations Manual*. For model parameters, please refer to the [Philips Mextram 504 Model \(Eldo Level 22\) Parameters](#) of the *Eldo Device Equations Manual*.

---

## Printing/Plotting States

If a state is to be monitored by the user, the user has to type in the netlist for a given transistor Q1 to monitor, for example In:

```
.PLOT DC S(Q1->In)    for DC or  
.PLOT AC S(Q1->In)    for AC or  
.PLOT TRAN S(Q1->In)   for TRAN
```

## Philips Modella Model (Eldo Level 23)

The Modella (*Model lateral*) model (Eldo Level = 23) developed by Philips provides an accurate model dedicated to lateral PNP devices. This new model is based on a totally new approach, accounting for the complex bi-dimensional structure of lateral transistors. The Modella model allows the simulation of lateral devices using real physically based parameters, instead of using less accurate empirically-modified vertical models, such as Gummel-Poon.

In the design of bipolar analog integrated circuits, greater flexibility is often achieved when both NPN and PNP transistors are incorporated in the circuit design. Many present day bipolar production processes use the conventional lateral PNP as the standard PNP transistor structure. For accurate modeling of such a lateral PNP transistor it is important to take the complex two-dimensional nature of the transistor into account. The physics based Modella model (*Model*

*lateral*) does exactly this. Using a modeling approach whereby the main currents and charges are independently related to bias-dependent minority carrier concentrations. Current crowding effects, high injection effects, and a bias dependent output impedance are all taken into account.



Please refer to the [Modella Equations](#) of the *Eldo Device Equations Manual*.  
For model parameters, please refer to the [Philips Modella Model \(Eldo Level 23\)](#)  
[Parameters](#) of the *Eldo Device Equations Manual*.

---

## Example of a Modella Model Card

```
.MODEL modella_model LPNP LEVEL=23
+ IS= 1.8E-16 BF= 131.0 IBF= 2.6E-14 VLF= 0.54 IK= 1.1E-04
+ XIFV=0.43 EAFL= 20.5 EAFV= 75.0 BR= 25.0 IBR= 1.2E-13
+ VLR= 0.48 XIRV= 0.43 EARL= 13.1 EARV= 104.0 XES= 2.7E-3
+ XHES= 0.70 XCS= 3.0 XHCS= 1.0 ISS= 4.0E-13 RCEX= 5.0
+ RCIN= 47.0 RBCC= 10.0 RBCV= 10.0 RBEC= 10.0 RBEV= 50.0
+ REEX= 27.0 REIN= 66.0 RSB= 1.E15 TLAT= 2.4E-9 TFVR= 3.0E-8
+ TFM= 2.0E-10 CJE= 6.1E-14 VDE= 0.52 PE= 0.3 TRVR= 1.0E-9
+ TRN= 3.0E-9 CJC= 3.9E-13 VDC= 0.57 PC= 0.36 CJS= 1.3E-12
+ VDS= 0.52 PS= 0.35 TREF= 25.00 DTA= 0.0 VGEB = 1.206
+ VGCB= 1.206 VGSB= 1.206 VGB= 1.206 VGE= 1.206 VGJE= 1.123
+ AE= 4.48 SPB= 2.853 SNB= 2.6 SNBN= 0.3 SPE= 0.73 SPC= 0.73
+ SX= 1.0 KF= 1.0 AF= 1.0 EXPHI= 1
```

## DC Operating Point Output

The DC operating point output facility gives information on the state of a device at its operating point. The following table shows the DC operating points that are printed in the *.chi* file in an OP and AC analysis.

Please refer to the [DC Operating Point Output](#) of the *Eldo Device Equations Manual*.

To print or plot a state for Q1, a BJT transistor, the following syntax is required:

- For DC

```
.PLOT DC S(Q1->GBE)
```

- For AC

```
.PLOT AC S(Q1->GBE)
```

- For Transient

```
.PLOT TRAN S(Q1->GBE)
```

## HICUM Level0 Model (Eldo Level 24)

The HICUM (HIgh CUrrent bipolar compact transistor Model) Level0 model for bipolar transistors is a simplified version of HICUM v2.x with less computational effort while keeping more accurate simulation results than with the well-known Spice-Gummel-Poon-Model. The latter is caused by the more accurate transit time and transfer current formulation. However HICUM v2.x is recommended for more accurate results in physical based device modeling of bipolar transistors.

The HICUM/Level0 model is implemented in Eldo as LEVEL=24.



Please refer to the [HICUM Level0 Equations](#) of the *Eldo Device Equations Manual*.  
For model parameters, please refer to the [HICUM Level0 Model \(Eldo Level 24\) Parameters](#) of the *Eldo Device Equations Manual*.

---

## JFET—Junction Field Effect Transistor

```
JXX ND NG NS MNAME [[AREA=]AREA_VAL] [L=VAL] [W=VAL]
+ [T[EMP]=VAL] [DTEMP=VAL] [OFF] [NONOISE]
JXX ND NG NS MNAME [FMIN=VAL] [FMAX=VAL] [NBF=VAL]
```

### Parameters

- **XX**  
JFET transistor name.
- **ND**  
Name of the drain node.
- **NG**  
Name of the gate node.
- **NS**  
Name of the source node.
- **MNAME**  
Name of the model used, as described in a **.MODEL** command.

### Optional Parameters

- **AREA**  
Relative device area (optional). Default value is 1.
- **L=VAL**  
Channel length in meters.
- **W=VAL**  
Channel width in meters.
- **T [ EMP ]=VAL**  
Sets temperature for the individual device, in degrees Celsius. Default nominal temperature=27°C.
- **DTEMP=VAL**  
Temperature difference between the device and the rest of the circuit, in degrees Celsius. Default value is 0.0.

---

**Note**

 **TEMP** and **DTEMP** are mutually exclusive. If both are specified, the last one is utilized.

---

- **OFF**  
Causes no initial operating point to be calculated for the device during DC analysis, i.e. the device is “off.”
- **NONOISE**  
Specifies that no noise model will be used for this device when performing noise analysis. Therefore, the device presents no noise contribution to the noise analysis.
- **FMIN=VAL**  
Lower limit of the noise frequency band.
- **FMAX=VAL**  
Upper limit of the noise frequency band.
- **NBF=VAL**  
Specifies the number of sinusoidal sources with appropriated amplitude and frequency and with randomly distributed phase from which the noise source is composed. Default value is 50. This parameter has no effect when **FMIN** is set to 0.

**Note**



**FMIN** and **FMAX** define the frequency band of the noise sources. This frequency range may sometimes not correspond to the noise frequency band at the output of the circuit. For instance, the band (**FMIN**, **FMAX**) does not correspond to the output noise frequency band in the case of filters or oscillators and mixers that exhibit frequency conversion.

**Note**



**FMIN** is also used to specify the algorithm used to generate the noise source generated by the JFET. When **FMIN>0** the JFET noise source is generated with sinusoids; when **FMIN=0** it is generated with a continuous spectrum between **FMIN** and **FMAX**.

## Noise in JFETs



Noise models are available for JFETs, see “[Noise](#)” on page 11-20 of the *Eldo Device Equations Manual*.

## Example

```
j1 1 2 3 fet2 t=50
```

Specifies a junction field effect transistor **j1** placed between nodes 1, 2 and 3 of model type **fet2**, and at a temperature of 50°C.

## JFET Model Syntax

```
.MODEL MNAME NJF [ PAR=VAL ]
.MODEL MNAME PJF [ PAR=VAL ]
```

The JFET model is derived from the FET model of Schichman & Hodges. The DC characteristics are defined by the **VTO** and **BETA** parameters, which determine the variation of drain current with gate voltage, **LAMBDA**, which determines the output conductance, and **IS**, the saturation current of the two gate junctions. The ohmic resistances, **RD** and **RS** are included. Charge Storage is modeled by non-linear depletion layer capacitances for both gate junctions. These capacitances vary with the  $-1/2$  power of junction voltage and are defined by **CGS**, **CGD** and **PB**. The following predefined JFET device models can be selected using the **LEVEL** parameter:

**Table 4-28. JFET Models**

LEVEL	Model Name
1	Schichman & Hodges Model
2	Enhanced Schichman & Hodges Model
3	Extended Schichman & Hodges Model



Please refer to the [JFET & MESFET Equations](#) of the *Eldo Device Equations Manual*. For model parameters, please refer to the [JFET Model \(Eldo Level 1, 2 and 3\) Parameters](#) of the *Eldo Device Equations Manual*.

## Example

```
*JFET model definition
.model je20 njf vto=-3.2 beta=0.98m
+ lambda=2.5m cgs=5p cgd=1.3p is=7p
...
*main circuit
j1 3 2 0 je20
```

Specifies the transistor **j1** with drain connected to node 3, gate to node 2 and source to ground. The electrical parameters are specified in the model **je20**.

## MESFET—Metal Semiconductor Field Effect Transistor

```
JXX ND NG NS MNAME [AREA] [L=VAL] [W=VAL] [T[EMP]=VAL] [DTEMP=VAL]
+ [OFF] [NONOISE]
JXX ND NG NS MNAME [FMIN=VAL] [FMAX=VAL] [NBF=VAL]
```



See “[JFET—Junction Field Effect Transistor](#)” on page 4-120 for more details and the **.MODEL** syntax.

The following predefined MESFET device models can be selected using the **LEVEL** parameter:

**Table 4-29. MESFET Models**

LEVEL	Model Name
6	Curtice Model
7	Schichman & Hodges Model
8	Statz Model
9	TriQuint Model



Please refer to the [JFET & MESFET Equations](#) of the *Eldo Device Equations Manual*. For model parameters, please refer to the [MESFET Model \(Eldo Level 6, 7 and 8\)](#) [Parameters](#) of the *Eldo Device Equations Manual*. For model parameters, please refer to the [MESFET Model \(Eldo Level 9, Update 1, 2 and 3\)](#) [Parameters](#) of the *Eldo Device Equations Manual*.

## MOSFET

```
MXX ND NG NS [NB] [{NN}] [MOD[EL]=]MNAME [[L=]VAL] [[W=]VAL]
+ [AD=VAL] [AS=VAL] [PD=VAL] [PS=VAL] [GEO=VAL] [NRD=VAL]
+ [NRS=VAL] [M=VAL] [RDC=VAL] [RSC=VAL] [T[EMP]=VAL] [DTEMP=VAL] [NONOISE]
MXX ND NG NS [NB] [{NN}] [MOD[EL]=]MNAME [W=VAL] [L=VAL]
+ [FMIN=VAL] [FMAX=VAL] [NBF=VAL]
```

### Parameters

- **XX**  
MOS transistor name.
- **ND**  
Name of the drain node.
- **NG**  
Name of the gate node.
- **NS**  
Name of the source node.
- **NB**  
Name of the bulk node. If not specified, the user can specify the parameter **BULK=node\_name** with the **.MODEL** command, Eldo will connect the bulk node to **node\_name**. By default **node\_name** is 0.
- **NN**  
Name of the **N** node. Any number of pins can be specified. However, only the BSIM3SOI model and some other proprietary models will make use of additional pins, if any.
- **[ MOD [ EL ] = ] MNAME**  
Name of the model used, as described in a **.MODEL** command. Using the optional **MOD[EL]** keyword, allows the user to avoid any instantiation errors that may occur due to the MOS instantiation allowing the specification of more than four pins.

Eldo will assume that the model name is the string that precedes the first string that is followed by an “=” sign. If there is no such string, the model name is assumed to be the 5th string, e.g. In the example below, **E** is assumed to be the model name.

```
M1 A B C D E F ...
```

In the example below, **F** is assumed to be the model name.

```
M1 A B C D E F <PNAME>=<value>
```

In the example below, **G** is the model name because the **MOD** keyword is present.

```
M1 A B C D E MOD=G ...
```

- **L=VAL**  
Channel length in meters. Default value is 100 $\mu$ m.
- **W=VAL**  
Channel width in meters. Default value is 100 $\mu$ m.
- **AD=VAL**  
Drain area in m<sup>2</sup>.
- **AS=VAL**  
Source area in m<sup>2</sup>.
- **PD=VAL**  
Drain perimeter in meters.
- **PS=VAL**  
Source perimeter in meters.
- **GEO=VAL**  
Switch which determines the layout of the device:
  - GEO=0** (default value) indicates the drain and source of the device are not shared by the other devices
  - GEO=1** indicates the drain is shared with another device
  - GEO=2** indicates the source is shared with another device
  - GEO=3** indicates the source and drain are shared with another device
- **NRD=VAL**  
Equivalent number of squares of the drain diffusion. **NRD** is multiplied with sheet resistance RSH to obtain parasitic series drain resistance of each transistor. RSH is specified in the **.MODEL** command.
- **NRS=VAL**  
Equivalent number of squares of the source diffusion. **NRS** is multiplied with sheet resistance RSH to obtain the parasitic series source resistance of each transistor.
- **M=VAL**  
This is the device “multiplier,” simulating the effect of multiple devices in parallel. Effective width, overlap, junction capacitances, and junction currents are multiplied by **M**. Parasitic resistance values are divided by **M**. Default value is 1.  
  
The device is first evaluated without the **M** factor, and at the very end of the device computation, all scaling quantities are multiplied / divided by **M**. Input values **W** and **L** are not affected. Models are chosen depending on input **W** and **L**, if required. Options **MINL**, **MAXL**, **MINW**, **MAXW**, etc. do not apply either, since they check the input values of **W** and **L**.

**Note**



Using an M factor value less than 1 could lead to simulating devices that cannot be physically realized.

---

- **RDC=VAL**  
Additional drain resistance due to contact resistance.
- **RSC=VAL**  
Additional source resistance due to contact resistance.
- **T [ TEMP ]=VAL**  
Sets temperature for the individual device, in degrees Celsius. Default nominal temperature=27°C.
- **DTEMP=VAL**  
Temperature difference between the device and the rest of the circuit, in degrees Celsius. Default value is 0.0.

**Note**



**TEMP** and **DTEMP** are mutually exclusive. If both are specified, the last one is utilized.

---

- **NONOISE**  
Specifies that no noise model will be used for this device when performing noise analysis. Therefore, the device presents no noise contribution to the noise analysis.
- **FMIN=VAL**  
Lower limit of the noise frequency band.
- **FMAX=VAL**  
Upper limit of the noise frequency band.
- **NBF=VAL**  
Specifies the number of sinusoidal sources with appropriated amplitude and frequency and with randomly distributed phase from which the noise source is composed. Default value is 50. This parameter has no effect when **FMIN** is set to 0.

**Note**



**FMIN** and **FMAX** define the frequency band of the noise sources. This frequency range may sometimes not correspond to the noise frequency band at the output of the circuit. For instance, the band (**FMIN**, **FMAX**) does not correspond to the output noise frequency band in the case of filters or oscillators and mixers that exhibit frequency conversion.

---

**Note**

 **FMIN** is also used to specify the algorithm used to generate the noise source generated by the MOSFET. When **FMIN>0** the MOSFET noise source is generated with sinusoids; when **FMIN=0** it is generated with a continuous spectrum between **FMIN** and **FMAX**.

---

**Note**

 Default value for all optional parameters is zero unless otherwise stated. If **L** or **W** are not specified, via the device instantiation statement, then they will take the values of the **DEFL** and **DEFW** parameters in the **.OPTION** command on [page 11-31](#). The MOS geometry parameters can be assigned directly or via “**.PARAM**” on page 10-205.

---

## Combining identical MOSFETs

When **.OPTION REDUCE** is set, multiple identical MOS instances that follow each other are reduced into a single instance using the **M** parameter, for example:

```
M1 1 2 3 4 MOS1 w=10u l=3u
M2 1 2 3 4 MOS1 w=10u l=3u
M3 1 2 3 4 MOS2 w=10u l=3u
```

Here, MOS instances **M1** and **M2** will be replaced by:

```
M1 1 2 3 4 MOS1 w=10u l=3u M=2
```

**M3** will remain because it has a different model name.



This also applies to BJTs, diodes and sub circuits. For more information see [page 4-111](#), [page 4-104](#) and [page 4-172](#) respectively.

---

## MOSFET Models

```
.MODEL MNAME NMOS [ PAR=VAL ]
.MODEL MNAME PMOS [ PAR=VAL ]
```

- **PAR=VAL**

Names and values of the model parameters.

The following predefined MOSFET device models can be selected using the **LEVEL** parameter.

**Table 4-30. MOSFET Models**

LEVEL	Model Name	Type	
		Capacitive	Charge
1 (BERK1)	Berkeley LEVEL 1 <a href="#">Berkeley SPICE Models</a>	Yes	

**Table 4-30. MOSFET Models**

LEVEL	Model Name	Type	
		Capacitive	Charge
2 (BERK2)	Berkeley LEVEL 2 <a href="#">Berkeley SPICE Models</a>	$x_{QC} \geq 0.5$	$x_{QC} < 0.5$
3 (BERK3)	Berkeley LEVEL 3 <a href="#">Berkeley SPICE Models</a>	$x_{QC} \geq 0.5$	$x_{QC} < 0.5$
4 (MERCK1)	Merckel Charge Control <a href="#">MERCKEL MOSFET Models</a>		Yes
5 (MERCK2)	Merckel Simplified <a href="#">MERCKEL MOSFET Models</a>	Yes	
6 (MERCK3)	Merckel Submicron <a href="#">MERCKEL MOSFET Models</a>	Yes	
7 (MERCK4)	Merckel 3 (MHS) <a href="#">MERCKEL MOSFET Models</a>	Yes	
8 (BSIM1)	<a href="#">Berkeley SPICE BSIM1 Model (Eldo Level 8)</a>		Yes
9	reserved		
10	reserved		
11 (BSIM2)	<a href="#">Berkeley SPICE BSIM2 Model (Eldo Level 11)</a>		Yes
12 (ELDO2)	<a href="#">Modified Berkeley Level 2 (Eldo Level 12)</a>	$x_{QC} \geq 0.5$	$x_{QC} < 0.5$
13 (ELDO3)	<a href="#">Modified Berkeley Level 3 (Eldo Level 13)</a>	$x_{QC} \geq 0.5$	$x_{QC} < 0.5$
14	reserved		
16 (ELDO6)	<a href="#">Modified Lattin-Jenkins-Grove Model (Eldo Level 16)</a>	$x_{QC} \geq 0.5$	$x_{QC} < 0.5$
17 (ELDO7)	<a href="#">Enhanced Berkeley Level 2 Model (Eldo Level 17)</a>	$x_{QC} \geq 0.5$	$x_{QC} < 0.5$
18 (STMOS1)	STMicroelectronics LEVEL 1		Yes
19 (STMOS3)	STMicroelectronics LEVEL 3		Yes
20 to 40	User Defined Models		
41 (STPROM1)	STMicroelectronics PROM LEVEL1		Yes
42 (STPROM3)	STMicroelectronics PROM LEVEL 3		Yes
43 (STMOS4)	STMicroelectronics MOS LEVEL 4		Yes
44 (EKV)	<a href="#">EKV MOS Model (Eldo Level 44 or EKV)</a>	$x_{QC} \geq 0.5$	$x_{QC} < 0.5$
47 (BSIM3)	<a href="#">Berkeley SPICE BSIM3v2 Model (Eldo Level 47)</a>		Yes
53 (BSIM3V3)	<a href="#">Berkeley SPICE BSIM3v3 Model (Eldo Level 53)</a> BSIM3v3.0, BSIM3v3.1 & BSIM3v3.2		Yes
54 (SSIM)	<a href="#">Motorola SSIM Model (Eldo Level 54 or SSIM)</a>		Yes
55 (BSIM3SOI)	<a href="#">Berkeley SPICE BSIM3SOI v1.3 Model (Eldo Level 55)</a>		Yes

**Table 4-30. MOSFET Models**

<b>LEVEL</b>	<b>Model Name</b>	<b>Type</b>	
		<b>Capacitive</b>	<b>Charge</b>
56 (BSIM3SOI)	Berkeley SPICE BSIM3SOI v2.x and v3.x Model (Eldo Level 56)		Yes
59 (MOSP9)	Philips MOS 9 Model (Eldo Level 59 or MOSP9)		Yes
60 (BSIM4)	Berkeley SPICE BSIM4 Model (Eldo Level 60) BSIM4.0, BSIM4.1, BSIM4.2, BSIM4.3 & BSIM4.4		Yes
62 (TFT)	TFT Polysilicon Model (Eldo Level 62) v1.0 & v2.0		Yes
63 (MOSP11)	Philips MOS Model 11 Level 1101 (Eldo Level 63)		Yes
64 (TFT)	TFT Amorphous-Si Model (Eldo Level 64)		Yes
65 (MOSP11)	Philips MOS Model 11 Level 1100 (Eldo Level 65)		Yes
66 (HISIM)	HiSIM Model (Eldo Level 66)		Yes
67 (SP)	SP Model (Eldo Level 67)		Yes
68 (BSIM5)	Berkeley SPICE BSIM5 Model (Eldo Level 68)		Yes
69	Philips MOS Model 11 Level 1102 (Eldo Level 69)		Yes
70 (PSP)	Philips PSP Model (Eldo Level 70)		Yes
101 (HVMOS)	BTA HVMOS Model (Eldo Level 101)		Yes

## Using -compat with MOSFET

Using the `-compat` runtime flag or selecting `.OPTION COMPAT` has the following effect on MOSFET Level values.

**Table 4-31. MOS Levels with -compat**

<b>LEVEL</b>	<b>Model Name (Eldo Level)</b>
2	Eldo2 (Eldo LEVEL 12)
3	Eldo3 (Eldo LEVEL 13)
6	Modified Lattin-Jenkins Grove Model (Eldo LEVEL 16)
8	Enhanced Berkeley LEVEL 2 (Eldo LEVEL 17)
13	Berkeley BSIM1 (Eldo LEVEL 8)
39	Berkeley BSIM2 (Eldo LEVEL 11)
49	Berkeley BSIM3 v3.0 & BSIM3 v3.1 (Eldo LEVEL 53)

**Table 4-31. MOS Levels with -compat**

LEVEL	Model Name (Eldo Level)
50	Philips MOS Model 9 (Eldo LEVEL 59)
54	Berkeley BSIM4.0.0 (Eldo LEVEL 60)
57	Berkeley BSIM3SOIv2 PD (Eldo LEVEL 56, SOIMOD=1)
59	Berkeley BSIM3SOIv2 FD (Eldo LEVEL 56, SOIMOD=3)

If using MOSFET without specifying parasitic areas and perimeters, please refer to [.OPTION XA](#) on page 11-24 for more information about computation of effective parasitic areas and perimeters.

By specifying `UDMP=1` on UDM models, Eldo will add the extrinsic contribution to the intrinsic part defined by the user. The parasitic effects are computed according to the Berkeley Level 1 common approach.

```
.model M NMOS level=21 UDMP=1
```



For more information on User Defined Models see the “[Introduction](#)” on page 1-1 of the *Eldo UDM User’s Manual*.

---

## Noise in MOSFETs



Noise models are available for MOSFETs, see “[Noise](#)” on page 11-20 of the *Eldo Device Equations Manual*.

---

In Eldo, MOS noise models are selected using either the device model parameters `THMLEV` and `FLKLEV`, or the global Eldo options `THERMAL_NOISE=VAL` and `FLICKER_NOISE=VAL`.

**Table 4-32. MOS Noise Models**

THERMAL_NOISE/THMLEV Value	Noise Model
0	SPICE noise model—default <sup>a</sup>
1	Strong inversion model
2	Weak and strong inversion model
3	Previous model with short channel effects
4	Strong inversion model with short channel effects

- a. Please note that the conductance used in the SPICE thermal noise equation is ***gm***. However, Eldo uses a more sophisticated computation of the total transistor conductance. This is given by the following equation:

$$g_{tot} = gm + gds + gmb$$

**Table 4-33. MOS Noise Models**

FLICKER_NOISE/FLKLEV Value	Noise Model
0	SPICE noise model—default
1	Improved SPICE noise model
2	Improved SPICE noise model
3	Improved SPICE noise model

## Selection of MOSFET Models via W/L Specifications (Binning)

- **.OPTION MODWL**

By default, Eldo behaves as if this option is set. It enables the use of MOS model versions which can be selected via **.MODEL** command **W** and **L** parameters (binning parameters). Whenever Eldo finds a MOS device for which the model name has no **.MODEL** command, it searches through all defined models for a model of the same root name and whose **W/L** range matches the specified device size. For example, in the case below, Eldo will assign the model **MODROOT.2** to the MOSFET **M1**:

```
M1 1 2 3 4 MODROOT w=2u l=3u
.MODEL MODROOT.1 NMOS VT0=1 WMIN=3u WMAX=5u LMIN=1u LMAX=5u
.MODEL MODROOT.2 NMOS VT0=2 WMIN=1u WMAX=5u LMIN=1u LMAX=5u
```

**Note**



Eldo selects the model to be assigned to MOS devices according to the geometric size of each device, even if these geometric sizes are modified at run-time via **.STEP** commands. In previous versions, the selection of the model was done just once at the very beginning of the simulation, and was not changed at run time.

The following formulae are used in conjunction with **.OPTION MODWL**:

$$. MIN + XLREF \leq L + XL < LMAX + XLREF$$

$$WMIN + XWREF \leq W + XW < WMAX + XWREF$$

**XL** and **XW** default to 0.0. If **XLREF** is not specified, **XLREF** is set to **XL**. Similarly for **XWREF**, which is set to **XW**.

The **.OPTION MODWL** algorithm works in the following way:

A model with dimensions W, L is chosen if:

$$WMIN \leq W < WMAX \text{ and } LMIN \leq L < LMAX$$

If when applying this rule, no match is found, Eldo will repeat the search with two inclusive inequalities:

$$WMIN \leq W \leq WMAX \text{ and } LMIN \leq L \leq LMAX$$

If a match is still not found, Eldo will issue an error.



Please also refer to **.OPTION MODWL** on [page 11-35](#).

---

## BSIM4 Model Selection

The BSIM4 model has unique conditions for W/L specification model selection. The conditions for this model are shown below:

$$WMIN \leq W/NF < WMAX \text{ and } LMIN \leq L/NF < LMAX$$

Where NF is a parameter specified in the model instantiation card which defines the number of device fingers. An example is shown below:

```
.model nmos.1 nmos level=60 lmin=0.8u lmax=1.2u wmin=8u wmax=12u
.model nmos.2 nmos level=60 lmin=0.8u lmax=1.2u wmin=80u wmax=120u

m1 1 2 0 0 nmos w=10u nf=1 l=1u
m2 1 2 0 0 nmos w=100u nf=10 l=1u
```

In this case m1 and m2 will both use the nmos.1 model.

## MOS Parasitics Common Approach

The need arises to unify the handling of the parasitic elements for all Eldo models, which are under our management. That is to say, that proprietary models are not concerned by those changes. To summarize, Eldo Levels 1, 2, 3, 8 (BSIM1), 11 (BSIM2), 12, 13, 16, 17, 44 (EKV), 47 (BSIM3v2), 53 (BSIM3v3) and 59 (MOSP9) are relevant to this feature. This parasitics common approach deals with the following effects:

- Parasitics access resistance calculation.
- Drain, source area and perimeter calculations.
- Bulk diode current calculations.
- Bulk diode capacitance calculations.

**Note**

 Noise calculations and geometric range definitions are also common.



Please refer to the [Common Equations](#) of the *Eldo Device Equations Manual*.

Four independent selectors **ALEV**, **RLEV**, **CAPLEV** and **DCAPLEV** ensure the switching between the different sets of equations. To ensure compatibility with previous versions of the software, an additional parameter **ARLEV** is necessary. **ARLEV** has priority over **ALEV** and **RLEV**, so if **ARLEV** is specified: **ALEV=RLEV=ARLEV**.

**Table 4-34. MOSFET Common Parameters**

Nr.	Name	Description	Default	Units
1	ACM	Flag to control which parasitic models are to be used	0	
2	OPTACM <sup>a</sup>	Flag to enable ACM control over parasitic models	0	
3	CALCACM <sup>b</sup>	Flag to control the area and perimeter calculations when ACM=12	0	

**Parasitic Resistance Related Parameters**

4	RLEV	Switch selector	*	
5	RD	Drain ohmic resistance	0	Ω
6	RS	Source ohmic resistance	0	Ω
7	RSH	Drain and Source diffusion sheet resistance	0	Ω/sq
8	RDC	Additional drain resistance due to contact resistance	0	Ω
9	RSC	Additional source resistance due to contact resistance	0	Ω
10	LD	Lateral diffusion into channel from source and drain diffusion	0*	m
11	LDIF	Length of lightly doped diffusion adjacent to gate	0	m
12	HDIF	Length of heavily doped diffusion from contact to lightly doped region	0	m
13	LGCD	Gate to contact length of drain side	0	m
14	LGCS	Gate to contact length of source side	0	m
15	SC	Spacing between contacts	-1e20	m
16	RDD	Scalable drain resistance	0	Ω m
17	RSS	Scalable source resistance	0	Ω m

**Table 4-34. MOSFET Common Parameters**

Nr.	Name	Description	Default	Units
<b>Area and Perimeter Related Parameters</b>				
18	ALEV	Switch selector	*	
19	DL	Length account for masking and etching effects	0	
20	DW <sup>c</sup>	Width account for masking and etching effects	0	m
21	LMLT	Length diffusion layer shrink reduction factor	1	
22	WMLT	Width diffusion layer shrink reduction factor	1	
23	WD	Lateral diffusion into channel from bulk along width	0	
<b>Bulk Diode Current Related Parameters</b>				
24	DIOLEV	Switch selector	*	
25	JS	Bottom bulk junction saturation current density	0	Am <sup>-2</sup>
26	JSW	Sidewall bulk junction saturation current density	0	Am <sup>-1</sup>
27	IS	Bulk junction saturation current	1×10 <sup>-14</sup>	A
28	N	Emission coefficient	1	-
29	NDS	Reverse bias slope coefficient	1	-
30	VNDS	Reverse bias transition voltage	-1	V
31	SBTH	Flag for reverse diode behavior (for PRECISE compatibility) 0 sets DIOLEV=1 and 1 sets DIOLEV=4	0	-
<b>JUNCAP Diode Current Related Parameters (DIOLEV=9)</b>				
32	VR	Voltage at which the parameters have been determined	0	V
33	JSGBR	Bottom saturation-current density due to electron-hole generation at $V = V_R$	1×10 <sup>-3</sup>	Am <sup>-2</sup>
34	JSDBR	Bottom saturation-current density due to diffusion from back contact	1×10 <sup>-3</sup>	Am <sup>-2</sup>
35	JSGSR	Sidewall saturation-current density due to electron-hole generation at $V = V_R$	1×10 <sup>-3</sup>	Am <sup>-1</sup>
36	JSDSR	Sidewall saturation-current density due to diffusion from back contact	1×10 <sup>-3</sup>	Am <sup>-1</sup>
37	JSGGR	Gate-edge saturation-current density due to electron-hole generation at $V = V_R$	1×10 <sup>-3</sup>	Am <sup>-1</sup>
38	JSDGR	Gate-edge saturation-current density due to diffusion from back contact	1×10 <sup>-3</sup>	Am <sup>-1</sup>

**Table 4-34. MOSFET Common Parameters**

Nr.	Name	Description	Default	Units
39	NBJ	Emission coefficient of the bottom forward current	<i>I</i>	
40	NSJ	Emission coefficient of the sidewall forward current	<i>I</i>	
41	NGJ	Emission coefficient of the gate-edge forward current	<i>I</i>	
42	CJBR	Bottom junction capacitance at $V = V_R$	$I \times 10^{-12}$	Fm $^{-2}$
43	CJSR	Sidewall junction capacitance at $V = V_R$	$I \times 10^{-12}$	Fm $^{-1}$
44	CJGR	Gate-edge junction capacitance at $V = V_R$	$I \times 10^{-12}$	Fm $^{-1}$
45	VDBR	Diffusion voltage of the bottom junction at $T = T_{nom}$	<i>I</i>	V
46	VDSR	Diffusion voltage of the sidewall junction at $T = T_{nom}$	<i>I</i>	V
47	VDGR	Diffusion voltage of the gate-edge junction at $T = T_{nom}$	<i>I</i>	V
48	PB	Bottom-junction grading coefficient	0.4	
49	PS	Sidewall junction grading coefficient	0.4	
50	PG	Gate-edge-junction grading coefficient	0.4	
51	TRDIO9	Nominal temperature for Juncap junction diode model	<b>TNOM</b> ( <b>TR</b> )	°C

**Bulk Diode Capacitance Related Parameters (DIOLEV≠9)**

52	DCAPLEV	Switch selector	*	
53	CBD	Zero bias Bulk-Drain capacitance	0	F
54	CBS	Zero bias Bulk-Source capacitance	0	F
55	CJ	Zero bias bottom Bulk junction capacitance	0*	Fm $^{-2}$
56	CJGATE	Zero bias gate-edge sidewall Bulk junction capacitance (only used when <b>ALEV=3</b> and <b>DCAPLEV=0</b> )	0	Fm $^{-1}$
57	CJSW	Zero bias sidewall Bulk junction capacitance	0	Fm $^{-1}$
58	FC	Bulk junction forward bias capacitance coefficient	0.5*	
59	MJ	Bulk junction bottom grading coefficient	0.5	
60	MJSW	Bulk junction sidewall grading coefficient	0.33	
61	PB	Bulk bottom junction potential	0.8*	V
62	PBSW	Bulk sidewall junction potential	<b>PB</b>	V

**Table 4-34. MOSFET Common Parameters**

Nr.	Name	Description	Default	Units
63	TT	Transit time	0	s
<b>Temperature Effect Related Model Parameters</b>				
64	TNOM (TR)	Nominal temperature (TR for MOSP9 only)	27	°C
65	TMOD	Model temperature	TNOM	°C
66	TLEV	Temperature equation level selector	0	
67	TLEV <sub>C</sub>	Temperature equation level selector for capacitances and potentials	0	
68	CTA (TCJ)	Junction capacitance <b>C<sub>J</sub></b> temperature coefficient	0	°K <sup>-1</sup>
69	CTP (TCJSW)	Junction sidewall capacitance <b>C<sub>JSW</sub></b> temperature coefficient	0	°K <sup>-1</sup>
70	PTA (TPB)	Junction potential <b>P<sub>B</sub></b> temperature coefficient	0	V°K <sup>-1</sup>
71	PTP (TPBSW)	Junction potential <b>P<sub>H<sub>B</sub></sub></b> temperature coefficient	0	V°K <sup>-1</sup>
72	EG	Energy gap for PN junction diode	1.11 <sup>d</sup>	eV
73	GAP1	First bandgap correction factor	7.02×10 <sup>-4</sup>	eV°K <sup>-1</sup>
74	GAP2	Second bandgap correction factor	1108	°K
75	TLEVI (LIS)	Saturation current temperature selector	1	
76	ISTMP	Number of degrees that doubles <b>I<sub>S</sub></b> value	10	°C
77	XTI	Saturation current temperature exponent	0	
78	TLEVR	Access resistances temperature selector	1	
79	TRD1 (TC1,TRD)	Temperature coefficient (linear) for <b>R<sub>D</sub></b>	0	°K <sup>-1</sup>
80	TRD2 (TC2)	Temperature coefficient (quadratic) for <b>R<sub>D</sub></b>	0	°K <sup>-2</sup>
81	TRS1 (TRS)	Temperature coefficient (linear) for <b>R<sub>S</sub></b>	0	°K <sup>-1</sup>
82	TRS2	Temperature coefficient (quadratic) for <b>R<sub>S</sub></b>	0	°K <sup>-2</sup>
83	TRSH1	Temperature coefficient (linear) for <b>R<sub>SH</sub></b>	0	°K <sup>-1</sup>
84	TRSH2	Temperature coefficient (quadratic) for <b>R<sub>SH</sub></b>	0	°K <sup>-2</sup>
<b>Noise Effect Related Model Parameters</b>				
85	AF	Flicker noise exponent	1	
86	KF	Flicker noise coefficient	0	
87	FLKLEV	Flicker noise level selector	0	
88	THMLEV	Thermal noise level selector	0	

**Table 4-34. MOSFET Common Parameters**

Nr.	Name	Description	Default	Units
<b>Geometric Range Related Model Parameters</b>				
89	WMIN	Model geometric range parameters. These model parameters give the range of the physical length and width dimensions to which the MOSFET model applies; used in conjunction with the .option modwl command	1	μm
90	WMAX		2000	μm
91	LMIN		1	μm
92	LMAX		100	μm

- a. The model parameter OPTACM, when set to 1, has the same effect as .option ACM. The only difference is that the latter affects all of the model cards in the netlist, while the former affects the model card that it is specified in.
- b. The model parameter CALCACM will accept either 0 (default) or 1. It is used only with .option ACM or OPTACM=1, and ACM=12. By default (CALCACM=0), sets the value of ALEV to 0 but if it is set to 1, it ALEV will be set to 2.
- c. for **ALEV**=7, the definition is “total width connection.”
- d. if **TLEV**=0 or 1, **EG**=1.11, else **EG**=1.16
- \*. For parameters marked with a \* in their default parameter column, the default value depends upon the model selected, see tables below.

**Table 4-35. RLEV Default Values**

Model (Eldo Level)	RLEV default value
Level (12,13,16,17), BSIM1 (8)	0
EKV (44)	2
BSIM3v2 (47), BSIM3v3 (53)	4
Level (1,2,3), MOS9 (53), BSIM2 (11), MM11 (63,65)	6

**Table 4-36. ALEV Default Values**

Model (Eldo Level)	ALEV default value
Level (12,13,16,17), BSIM1 (8), BSIM3v3 (53)	0
EKV (44), BSIM3v2 (47)	2
Level (1,2,3), MOS9 (53), BSIM2 (11), MM11 (63,65)	6

**Table 4-37. DIOLEV Default Values**

Model (Eldo Level)	DIOLEV default value
Level (1,2,3), MOS9 (53), BSIM3v2 (47)	1
Level (12,13,16,17),	2
BSIM1 (8), BSIM2 (11)	3

**Table 4-37. DIOLEV Default Values**

Model (Eldo Level)	DIOLEV default value
EKV (44)	6
BSIM3v3 (53)	7
MM11 (63,65)	9

**Table 4-38. DCAPLEV Default Values**

Model (Eldo Level)	DCAPLEV default value
Level (12,13,16,17), BSIM1 (8)	0
BSIM3v2 (47)	1
Level (1,2,3), MOS9 (53), BSIM2 (11), EKV (44)	2
BSIM3v3 (53)	4
MM11 (63,65)	-

## Scaling Rules

$$LDIF_{scal} = LDIF \cdot scalm \quad HDIF_{scal} = HDIF \cdot scalm \cdot WMLT$$

$$DL_{scal} = DL \cdot scalm \quad DW_{scal} = DW \cdot scalm$$

$$LD_{scal} = LD \cdot scalm \quad WD_{scal} = WD \cdot scalm$$

$$AD_{scal} = \frac{AD}{scalm \cdot scalm} \quad AS_{scal} = \frac{AS}{scalm \cdot scalm} v_{tvt}$$

$$PD_{scal} = \frac{PD}{scalm} \quad PS_{scal} = \frac{PS}{scalm}$$

$$JS_{scal} = \frac{JS}{scalm \cdot scalm} \quad JSW_{scal} = \frac{JSW}{scalm}$$

$$CJ_{scal} = \frac{CJ}{scalm \cdot scalm} \quad CJSW_{scal} = \frac{CJSW}{scalm}$$

$$CJGATE_{scal} = \frac{CJGATE}{scalm}$$

### Note



The model parameter scaling factor **SCALM** can be defined for all model cards in the netlist using **.OPTION SCALM=val**, or can be individually defined for each model card using the model parameter **SCALM**. This overrides the global **SCALM** value defined using the **.OPTION** command.

## Berkeley SPICE Models

The DC characteristics of the MOSFET are defined by the device parameters **VTO**, **KP**, **LAMBDA**, **PHI** and **GAMMA**. These are computed by Eldo, if the process parameters (**NSUB**, **TOX**,...) are given. User specified values always override calculated values. **VTO** is positive (negative) for enhancement mode and negative (positive) for depletion mode N-channel (P-channel) devices.

Charge storage is modeled by three constant capacitors, **CGSO**, **CGDO** and **CGBO**. Capacitances taken into account are the non-linear thin-oxide capacitance distributed among the gate, source, drain and bulk regions, and the non-linear depletion-layer capacitances for both substrate junctions divided into bottom and periphery, which vary as the **MJ** and **MJSW** power of junction voltage respectively, and are determined by the parameters **CBD**, **CBS**, **CJ**, **CJSW**, **MJ**, **MJSW** and **PB**. The model is the piecewise linear voltage dependent capacitance model proposed by Meyer.

Some overlap among the parameters describing the junctions exists, e.g. Reverse current can be input either as **IS** (in Amperes) or as **JS** (in  $\text{Am}^{-2}$ ). The first is an absolute value, while the second is multiplied by **AD** and **AS** to give the reverse current of the drain and source junctions respectively. This methodology is used as there is no sense in always relating junction characteristics with **AD** and **AS** of the device card; these areas can be defaulted. The same idea also applies to the zero-bias junction capacitances **CBD** and **CBS** (in Farads) on one hand, and **CJ** (in  $\text{Fm}^{-2}$ ) on the other. Parasitic drain and source series resistance can be expressed as either **RD** and **RS** (in  $\Omega$ ) or **RSH** (in  $\Omega/\text{Square}$ ), the latter being multiplied by the number of squares **NRD** and **NRS** input on the device card.

The temperature of a semiconductor model can be defined in Eldo using the **TMOD** parameter. If this parameter is not present the global circuit temperature **TNOM** is assumed. The individual model temperature of a device can be set by using the optional instance parameter **T**.



Please refer to the [MOS Level 1, 2, 3 Equations](#) of the *Eldo Device Equations Manual*.  
For model parameters, please refer to the [Berkeley SPICE Model \(Eldo Level 1, 2, 3\)](#)  
[Parameters of the Eldo Device Equations Manual](#).

Specific Levels 1, 2, and 3 initialization for parasitics common approach MOS parameters:

- In Eldo mode:  
**ALEV=6; RLEV=6; DIOLEV=1; DCAPLEV=2**  
if **XJ** is specified then **LD=0.75\*XJ** otherwise **LD=0**;
- In PRECISE mode:  
**ALEV=4; RLEV=4; DIOLEV=1; DCAPLEV=2**.



For details of the common parameters, see “[MOS Parasitics Common Approach](#)” on page 4-132.

- Equivalent name definitions:  
**P<sub>HP</sub>** parameter is equivalent to **P<sub>BSW</sub>**.  
**X<sub>L</sub>** and **X<sub>W</sub>** are equivalent to **D<sub>L</sub>** and **D<sub>W</sub>** respectively.

For model parameters, please refer to the [Berkeley SPICE Model \(Eldo Level 1, 2, 3\)](#) [Parameters with Precise Option](#) of the *Eldo Device Equations Manual*.

## MERCKEL MOSFET Models

Three Merckel model levels are presently available as Levels 4, 5 and 6. These models are defined by their own set of parameters, which can be easily extracted from measurements.

---

### Note

Contact Mentor Graphics for more details about the above models.

---

## Key Parameters

- **V<sub>T0</sub>**  
Threshold voltage at **V<sub>GS</sub>=0**.
- **M<sub>U0</sub>**  
Low field surface mobility.
- **D**  
Correction term issued from development of the ‘3/2’ model. **D** is extracted from the model parameter.
- **T<sub>G</sub>**  
Gate field effect coefficient on mobility.
- **T<sub>D</sub>**  
Drain field effect coefficient on mobility, which equals  $\frac{M_{U0}}{(L \times V_S)}$  where **V<sub>S</sub>** is the saturation velocity.
- **V<sub>EA</sub>**  
Early voltage coefficient.

The Level 4 model is a charge control model and the DC characteristics are the same as in Level 6.

The Level 5 model is very similar to that of SPICE Level 1, but it also takes into account the effect of mobility modulation by **TG**. The early effect is modeled by **KE** instead of **LAMBDA** as in SPICE models.

The Level 6 model is more accurate, and it is especially dedicated to submicron technologies; the threshold dependencies on geometries are modeled by **VE** instead of **LAMBDA** as in SPICE models.

The capacitance models are close to those of SPICE. The bulk junction capacitances vary with the power of  $-1/2$  of the junction voltage. Overlap capacitances are computed from the **REC** parameter. **LDIF** is used for computation of the drain and source perimeters and areas, if the parameters for **AD**, **AS**, **PD** and **PS** are missing in the declaration of the MOS transistor.



Please refer to the [MOS Level 4, 5, 6 Equations](#) of the *Eldo Device Equations Manual*. For model parameters, please refer to the [MERCKEL MOSFET Model \(Eldo Level 4, 5, 6\) Parameters](#) of the *Eldo Device Equations Manual*.



For **FLKLEV** and **THMLEV**, see the [“Noise in MOSFETs”](#) on page 4-130.

## Correspondence of Merckel to SPICE Model Parameters

For historical reasons, some SPICE parameter names have equivalent Merckel parameter names, which can be interchanged within a model definition. The following is a list of these parameters:

**Table 4-39. Merckel-Spice**

Merckel	SPICE
NIV	LEVEL
EOX	TOX
DPHIF	PHI
VTO	VT0
MUO	UO
KB	GAMMA
TG	THETA
NB	NSUB
CDIFS0	CJ
CDIFP0	CJSW
DL	2PLD

**Table 4-39. Merckel-Spice**

Merckel	SPICE
KO	KP/2

## Berkeley SPICE BSIM1 Model (Eldo Level 8)

The Berkeley Short Channel IGFET Model (BSIM1) is a semi-empirical model proposed by the University of Berkeley (California). It was adopted to cope with rapid advances of technologies and fits well for submicron technologies.

The major effects modeled in BSIM1 include:

- mobility reduction due to the vertical field
- channel length modulation
- carrier velocity saturation
- non-uniform channel doping
- subthreshold conduction
- drain-induced barrier lowering
- source/drain charge sharing

A representation with 17 parameters per device size was found to be adequate for modeling the DC characteristics. Geometric dependencies are included to introduce parameter sensitivity to length and width. The charge model was derived from its drain-current counterpart to ensure model consistency of device physics. Charge conservation is also guaranteed. Also, three possible drain-source sharings are possible.



Please refer to the [BSIM1 Equations](#) of the *Eldo Device Equations Manual*.

For model parameters, please refer to the [Berkeley SPICE BSIM1 Model \(Eldo Level 8 Parameters\)](#) of the *Eldo Device Equations Manual*.

---

Specific BSIM1 initialization for parasitics common approach MOS parameters:

```
ALEV=0; RLEV=0; DIOLEV=3; DCAPLEV=0
if XJ is specified then LD=0.75*XJ otherwise LD=0;
in PRECISE mode, CJ=0 otherwise CJ=4.5*10**-5;
FC is not a BSIM1 parameter, therefore it is set to 0.
```



For details of the common parameters, see “[MOS Parasitics Common Approach](#)” on page 4-132.

---

## Berkeley SPICE BSIM2 Model (Eldo Level 11)

Based on BSIM1 model, the BSIM2 was developed by the University of Berkeley (California) to correct BSIM1 problems. BSIM2 has been successfully used to model the drain current and output resistance of MOSFETs with gate oxide as thin as 3.6 nm and channel length as small as 0.2  $\mu\text{m}$ .

Based on recent physical understanding of deep-submicron MOSFETs, it has been found that the important effects that should be included in MOSFET modeling are:

- mobility reduction due to the vertical field
- carrier velocity saturation
- non-uniform channel doping
- subthreshold conduction
- drain-induced barrier lowering
- source/drain charge sharing
- channel length modulation
- source/drain parasitic resistance
- hot electron induced output resistance reduction
- inversion layer capacitance.

Except for the three last effects, most of these were already present in the BSIM1 model. In order to develop a consistent model, these effects were re-examined using new deep-submicron MOSFET considerations and their implementation. The charge model was derived from its drain-current counterpart to ensure model consistency of device physics. Charge conservation is also guaranteed. Also 3 possible drain-source sharings are possible.



Please refer to the [BSIM2 Equations](#) of the *Eldo Device Equations Manual*.

For model parameters, please refer to the [Berkeley SPICE BSIM2 Model \(Eldo Level 11 Parameters\)](#) of the *Eldo Device Equations Manual*.

Specific BSIM2 initialization for parasitics common approach MOS parameters:

```
ALEV=6; RLEV=0; DIOLEV=3; DCAPLEV=2
if XJ is specified then LD=0.75*XJ; otherwise LD=0;
CJ=0; FC is not a BSIM2 parameter, therefore it is set to 0.
```



For details of the common parameters, see the “[MOS Parasitics Common Approach](#)” on page 4-132.

## Modified Berkeley Level 2 (Eldo Level 12)

A modified version of Level 2, Level 12 includes specification of Meyer capacitance parameters **CF1**, **CF2**, **CF3**, **CF5** and **CGBEX**. These parameters are used to solve the dynamic part of the model. The diode currents, impact ionization currents and access resistances are also calculated differently when using this model.

- 
- i** Please refer to the [Modified Berkeley Level 2 \(Eldo Level 12\)](#) of the *Eldo Device Equations Manual*.  
For model parameters, please refer to the [Modified Berkeley SPICE Level 2 Parameters](#) of the *Eldo Device Equations Manual*.
- 

## Modified Berkeley Level 3 (Eldo Level 13)

A modified version of Level 3, Level 13 includes specification of Meyer capacitance parameters **CF1**, **CF2**, **CF3**, **CF5** and **CGBEX**. These parameters are used to solve the dynamic part of the model. The diode currents, impact ionization currents and access resistances are also calculated differently when using this model.

- 
- i** Please refer to the [Modified Berkeley Level 3 \(Eldo Level 13\)](#) of the *Eldo Device Equations Manual*.  
For model parameters, please refer to the [Modified Berkeley SPICE Level 3 Parameters](#) of the *Eldo Device Equations Manual*.
- 

### Example

```
*MOSFET model definition
.model me21 nmos level=4 vt0=1v eox=25n uo=600
+ nsub=2.0e16 phi=0.6 vmax=2.0e5 kw=2.24u
+ kl=2.24u gw=3.91u gl=0.7u dinf=0.1 kb=0.1
+ ve=1.0e4 ldif=10u cj=0.0001 cjsw=0 dw=0
+ dl=0.8u rec=0.15u tg=0.06
...
*main circuit
m1 vdd n3 n2 vbb me21 w=40u l=3u
```

Specifies the transistor **m1** with a 40 micron channel width, a 3 micron channel length, drain connected to node **vdd**, gate to node **n3**, Source to node **n2** and bulk to voltage **vbb**. The electrical parameters are specified in the model **me21**.

## Modified Lattin-Jenkins-Grove Model (Eldo Level 16)

Level 16 model, also called Lattin-Jenkins-Grove model, is a model based on Level 2 equations and represents the ASPEC, ISPICE compatible model.



Please refer to the [Modified Lattin-Jenkins-Grove Model \(Eldo Level 16\)](#) of the *Eldo Device Equations Manual*.

For model parameters, please refer to the [Modified Lattin-Jenkins-Grove Model Parameters](#) of the *Eldo Device Equations Manual*.

---

## Enhanced Berkeley Level 2 Model (Eldo Level 17)

An enhanced version of Berkeley Level 2, Level 17 includes modified equations for the description of the threshold voltage, subthreshold current, effective mobility and effective substrate doping.

The additional developments of the standard Berkeley SPICE Level 2 Model in this Eldo Level 17 are derived from model equations developed by the General Electric/Intersil Research Institutes.



Please refer to the [Enhanced Berkeley Level 2 Model \(Eldo Level 17\)](#) of the *Eldo Device Equations Manual*.

For model parameters, please refer to the [Enhanced Berkeley SPICE Level 2 Model Parameters](#) of the *Eldo Device Equations Manual*.

---

## EKV MOS Model (Eldo Level 44 or EKV)

The EKV model was originally developed by the EPFL (Ecole Polytechnique Fédérale de Lausanne), namely MM Enz, Krummenacher and Vittoz, hence the name. It is the result of many years of investigation to find a model that deals correctly with analog problems.

Consequently, it may solve the problems of analog designers mainly in low voltage and low current applications and in terms of realistic behavior for currents, conductances and capacitances.

A compatible version of the EKV model is available in Eldo. This signifies that former versions of the model are also available. The compatibility covers versions v2.3 up to, and including, the new v2.6 (revision 2). The different versions are accessible through the model parameter **UPDATE** which can be used as shown in the table below. By default, the EKV model version v2.3 is selected.

Table 4-40. EKV MOS Models

Parameter Value	EKV Version
UPDATE=2.3 or UPDATE=23	EKV v2.3 (Default)
UPDATE=2.5 or UPDATE=25	EKV v2.5
UPDATE=2.61 or UPDATE=26.1	EKV v2.6 (Revision 1)

**Table 4-40. EKV MOS Models**

Parameter Value	EKV Version
UPDATE=2.6 or UPDATE=26 UPDATE=2.62 or UPDATE=26.2	EKV v2.6 (Revision 2)
UPDATE=2.63 or UPDATE=26.3	EKV v2.6 (Revision 3)

---

**Note**

 EKV versions v1.3 and v2.2 are no longer supported.

---



Please refer to the [EKV MOSFET Equations](#) of the *Eldo Device Equations Manual*.

For model parameters, please refer to the [EKV MOS Model \(Eldo Level 44\) Setup Parameters](#) of the *Eldo Device Equations Manual*.

For model parameters, please refer to the [EKV MOS Model \(Eldo Level 44\) Parameters](#) of the *Eldo Device Equations Manual*.

Please refer to the [Parameter units](#) of the *Eldo Device Equations Manual*.

Please refer to the [Parameter preprocessing \(intrinsic parameters initialization, version 2.6 revision 2\)](#) of the *Eldo Device Equations Manual*.

---

## Berkeley SPICE BSIM3v2 Model (Eldo Level 47)

BSIM3 version 2 is a physical model developed by the University of Berkeley (California). It is based on a coherent quasi two-dimensional analysis of the MOSFET device structure, taking into account the effects of device geometry and process parameters. It allows users to accurately model MOSFET behavior for up-to-date submicron technologies.



Please refer to the [BSIM3v2 Equations](#) of the *Eldo Device Equations Manual*.

For model parameters, please refer to the [Berkeley SPICE BSIM3v2 Model \(Eldo Level 47\) Parameters](#) of the *Eldo Device Equations Manual*.

---

### Notes about Parameters

- **NPEAK**, **NGATE**, **ND** and **U0** may be entered in meters or centimeters:  
**NPEAK** is converted to  $\text{cm}^{-3}$  as follows: if **NPEAK** is greater than  $1 \times 10^{20}$ , it is multiplied by  $1 \times 10^{-6}$ .  
**NGATE** is converted to  $\text{cm}^{-3}$  as follows: if **NGATE** is greater than  $1 \times 10^{23}$ , it is multiplied by  $1 \times 10^{-6}$ .  
**ND** is converted to  $\text{cm}^{-3}$  as follows: if **ND** is greater than  $1 \times 10^{24}$ , it is multiplied by  $1 \times 10^{-6}$ .  
**U0** is converted to  $\text{m}^2(\text{Vs})^{-1}$  as follows: if **U0** is greater than 1, it is multiplied by  $1 \times 10^{-4}$ .  
**NSUB** must be entered in  $\text{cm}^{-3}$ .

- Specific BSIM3v2 initialization for parasitics common approach MOS parameters:  
**ARLEV=ALEV=2; RLEV=4; DCAPLEV=1; DIOLEV=1.**  
**CJ=5×10<sup>-4</sup>; CJSW=5×10<sup>-10</sup>; PB=1; PBSW=1;**  
**FC** is not a BSIM3v2 parameter, therefore it is set to 0.  
As a consequence, you may use **LD**, **WD** instead of **DL**, **DW**. Respectively, if **DL**, **DW** are given, they will be used for common equation initializations instead of **LD** and **WD**.



For details of the common parameters, see the “[MOS Parasitics Common Approach](#)” on page 4-132.

- For derivative computation in BSIM3v2, a parameter **DERIV** has been added. By default, **DERIV=1** for analytical derivatives. **DERIV** may be set to 0 (or with the command **.option mnumer**) for the finite difference method.

## Berkeley SPICE BSIM3v3 Model (Eldo Level 53)

BSIM3v3 has been extensively modified from its previous release BSIM3v2. The physical effects modeled are the same as BSIM3v2. In addition to those, the new advancements are:

- A single drain current expression to describe current and output conductance characteristics from subthreshold to strong inversion as well as from the linear to the saturation operating region,
- New width dependencies for bulk charge and source/drain resistance (**Rds**),
- New capacitance models for both intrinsic and extrinsic capacitances,
- A new noise model,
- A new relaxation time model for characterizing the non-quasi-static effect for improved transient modeling.



Please refer to the [BSIM3v3 Equations](#) of the *Elodo Device Equations Manual*.

BSIM3v3 is available in three versions: BSIM3v3, BSIM3v3.1, and the most recent, BSIM3v3.2. According to Berkeley’s recommendation, version BSIM3v3.2 is split into four versions.

The different versions, BSIM3v3, BSIM3v3.1 and BSIM3v3.2.x, are accessible through the model parameter **VER**. By default, BSIM3v3.2.3 (**VER=3.24**) is selected.

**Table 4-41. BSIM3v3 Models**

Parameter Value	BSIM3v3 Version
VER(VERSION)=3.24	BSIM3v3.2.4 (Default)

**Table 4-41. BSIM3v3 Models**

Parameter Value	BSIM3v3 Version
VER(VERSION)=3.23	BSIM3v3.2.3
VER(VERSION)=3.22	BSIM3v3.2.2 (Also if <b>VER</b> = 3.2)
VER(VERSION)=3.21	BSIM3v3.2.1
VER(VERSION)=3.1	BSIM3v3.1
VER(VERSION)=3.0	BSIM3v3

### BSIM4 gate current inside BSIM3v3

Due to the continual decrease in device dimensions, the oxide thickness in new technologies is very small, so much so that the gate current is noticeable. Many companies would model the gate current with BSIM3v3 model in the form of a SUBCKT, however, this would slow the simulation a great deal. The gate current model of BSIM4 has, therefore, been incorporated inside BSIM3v3.

Having the gate current inside the compact model should be much faster than in the form of a SUBCKT. The model parameters used in BSIM4 should be used for the gate current. The two main selectors are **IGCMOD** & **IGBMOD**. If either of these flags equal 1, the gate current will be calculated and the values taken will not be equal to 0. If both of these selectors are equal to 0 or not specified, the gate current is not calculated and the simulation will proceed as before.



These parameters can be found in the [BSIM4 Gate Current Model](#) parameter table.

---

### BSIM4 Gate-Induced Drain and Source Leakage Current inside BSIM3v3

The gate induced drain leakage (GIDL) and gate induced source leakage (GISL) currents are gaining more importance in new technologies from day to day. Many companies would model the GIDL current with BSIM3v3 model, using a SUBCKT, however, this would slow the simulation. The GIDL (and GISL, see Note below) current model of BSIM4 has therefore been incorporated inside BSIM3v3 to increase the simulation speed. The model parameters used in BSIM4 should be used for the GIDL current. These four main parameters are **AGIDL**, **BGIDL**, **CGIDL** & **EGIDL**. If any of these parameters are specified using the model command, the GIDL current will be calculated. If none of the parameters are specified however, the GIDL current is not calculated and the simulation will proceed as before.



These parameters can be found in the [BSIM4 Gate-Induced Drain Leakage Model Parameters](#) table.

---

**Note**

 BSIM4.21 has the addition of Gate Induced Source Leakage (GISL) current. This is enabled by default in BSIM3v3 when enabling the GIDL current. If you want to use the GIDL model of BSIM4 versions prior to version 4.21, please set the **IGIDLVER** model parameter to a value less than 4.21 (default value). See the [BSIM4.2.1 enhancements](#) for further information on the GISL in BSIM4.

## BSIM3SOI DTOXCV Parameter inside BSIM3v3

There have been a number of problems regarding **CAPMOD=3** of BSIM3v3. The most serious symptom is the capacitance reduction ( $C_{gg}$ ) in the strong inversion region if **CAPMOD** was changed from 0 to 3. BSIMPD is an SOI model formulated on top of the BSIM3v3 framework and also has the same trouble. The Berkeley team proposed an additional parameter, **DTOXCV**, in **CAPMOD=3** of BSIMPDv2.2.3 to solve this issue. This has also been added to BSIM3v3.



This parameter can be found in the [Process Parameters](#) section of the parameter table.

## Compatibility option for negative **Rds** values

By default, negative **Rds** values are clipped to zero. An Eldo option is available to allow negative **Rds** values: **.OPTION RDSWTPOS=0 | 1**. Default value is 1, which means Eldo clips negative **Rds** values (after temperature update) to zero. To allow negative **Rds** values, set the option to 0.

**Caution**

When this option is used to allow negative **Rds** values, the results may be unpredictable, or even cause the simulation to fail.



For model parameters, please refer to the [Berkeley SPICE BSIM3v3 Model \(Eldo Level 53\) Parameters](#) of the *Eldo Device Equations Manual*.

## Notes about Parameters

- **NCH**, **NGATE** and **U0** may be entered in meters or centimeters:  
**NCH** is converted to  $\text{cm}^{-3}$  as follows: if **NCH** is greater than  $1 \times 10^{20}$ , it is multiplied by  $1 \times 10^{-6}$ .  
**NGATE** is converted to  $\text{cm}^{-3}$  as follows: if **NGATE** is greater than  $1 \times 10^{23}$ , it is multiplied by  $1 \times 10^{-6}$ .  
**U0** is converted to  $\text{m}^2/\text{Vsec}$  as follows: if **U0** is greater than 1, it is multiplied by  $1 \times 10^{-4}$ .  
**NSUB** must be entered in  $\text{cm}^{-3}$ .

- Specific BSIM3v3 initialization for parasitics common approach MOS parameters:  
**ARLEV=ALEV=0; RLEV=4; DCAPLEV=1; DIOLEV=1.**  
**CJ=** $5 \times 10^{-4}$ **; CJSW=** $5 \times 10^{-10}$ **; PB=1; PBSW=1;**  
**FC** is not a BSIM3v3 parameter, therefore it is set to 0.  
As a consequence, you may use **LD**, **WD** instead of **LINT**, **WINT** or **DL**, **DW** instead of **DLC**, **DWC**. Respectively, **LINT**, **WINT** and **DLC**, **DWC** will be used for common equation initialization.



For details of the common parameters, see the “[MOS Parasitics Common Approach](#)” on page 4-132.

---

- As **LINT** and **LD** are both the same the user should only specify one parameter (**LINT** or **LD**). If both **LINT** and **LD** are specified then Eldo will return the following warning:

Double definition for parameter(s) LD.

Only the value of **LD** will be printed in the .chi file.

- For derivative computation in BSIM3v3, a parameter **DERIV** has been added. By default, **DERIV=1** for analytical derivatives. **DERIV** may be set to 0 (or with the command **.option mnumer**) for the finite difference method.
- The different versions, BSIM3v3.2, BSIM3v3.1 and BSIM3v3, can also be selected by using the command **.option bsim3ver = 3.2, 3.1 or 3.0**.
- Parameter checking is added in BSIM3v3.2 to avoid bad values of certain parameters as follows:  
If **PSCBE2**  $\leq 0.0$ , the user will be warned for the poor value used.  
If **(MOIN < 5.0)** or **(MOIN > 25.0)**, a warning message will be given.  
If **(ACDE < 0.4)** or **(ACDE > 1.6)**, a warning message will be given.  
If **(NOFF < 0.1)** or **(NOFF > 4.0)**, a warning message will be given.  
If **(VOFFCV < -0.5)** or **(VOFFCV > 0.5)**, a warning message will be given.  
If **(IJTH < 0.0)**, fatal error occurs.  
If **(TOXM <= 0.0)**, fatal error occurs

## Printing and plotting BSIM3v3 Output States

The three capacitance states can be printed/plotted for any BSIM3v3 instance using the syntax **S (Mxx->state)**. For example, **CAPGDO** can be printed by specifying:

```
.PLOT DC S(Mxx->CAPGDO)      for DC or
.PLOT AC S(Mxx->CAPGDO)      for AC or
.PLOT TRAN S(Mxx->CAPGDO)     for TRAN
```

**Table 4-42. Berkeley BSIM3v3 States**

State	Description
CAPGDO	Gate-Drain overlap capacitance
CAPGSO	Gate-Source overlap capacitance
CAPGBO	Gate-Bulk overlap capacitance

## Motorola SSIM Model (Eldo Level 54 or SSIM)

The Motorola SSIM model can be invoked in two ways:

1. By specifying explicitly the level SSIM inside the **.MODEL** card:

```
.MODEL mod NMOS LEVEL=SSIM <end of the model ...>
or:
```

```
.MODEL mod NMOS LEVEL=54 <end of the model ...>
```

2. By running Eldo in Motorola mode:

There are two ways to use the Motorola mode of Eldo:

- a. by invoking Eldo with the **-ssim** switch at runtime
- b. by adding **.OPTION MOTOROLA** at the beginning of the netlist (before the first **.MODEL** card).

When Eldo is in Motorola mode, the level of the SSIM model is 6:

```
.MODEL mod NMOS LEVEL=6
```

---

**Note**

This is useful when the user wants to utilize a **.MODEL** card from Motorola. The same system exists for the STMicroelectronics models.

---

## Berkeley SPICE BSIM3SOI v1.3 Model (Eldo Level 55)

BSIM3SOI v1.3 is an officially released SOI (Silicon On Insulator) MOSFET model from the Device Group at the University of California at Berkeley. The model can be used for both Partially Depleted (PD) and Fully Depleted (FD) devices. Many advanced concepts are introduced so as to allow transition between PD and FD operation dynamically and continuously, namely the *Dynamic Depletion Approach*. The basic I-V model is modified from the BSIM3v3.1 equation set.



Please refer to the [BSIM3SOI Equations](#) of the *Eldo Device Equations Manual*.

---

## Command Line Information

```
Mname <D node> <G node> <S node> <E node> [P node] <model>
+ [L=<VAL>] [W=<VAL>]
+ [AD=<VAL>] [AS=<VAL>] [PD=<VAL>] [PS=<VAL>]
+ [NRS=<VAL>] [NRD=<VAL>] [NRB=<VAL>]
+ [OFF] [BJTOFF=<val>] [RTH0=<VAL>] [CTH0=<VAL>]
+ [M=<VAL>] [DEBUG=<VAL>]
```

## Parameters

- D node  
Drain node
- G node  
Gate node
- S node  
Source node
- E node  
Substrate node
- [ P node ]  
Optional external body contact  
if not specified, it is a 4-terminal device  
if specified, it is a 5-terminal device. The P node and B node will be connected by a resistance.
- model  
Level 55 BSIM3SOI model name
- **L**=VAL  
Channel length
- **W**=VAL  
Channel width
- **AD**=VAL  
Drain diffusion area
- **AS**=VAL  
Source diffusion area
- **PD**=VAL  
Drain diffusion perimeter length
- **PS**=VAL  
Source diffusion perimeter length

- **NRS=VAL**  
Number of squares in source series resistance
- **NRD=VAL**  
Number of squares in drain series resistance
- **NRB=VAL**  
Number of squares in body series resistance
- **OFF=VAL**  
Device simulation off
- **BJTOFF=VAL**  
Turn off BJT current if equal to 1
- **RTH0=VAL**  
Thermal resistance per unit width  
if not specified, **RTH0** is extracted from the model card.  
if specified, it will override the one in the model card.
- **CTH0=VAL**  
Thermal capacitance per unit width  
if not specified, **CTH0** is extracted from model card.  
if specified, it will over-ride the one in model card.
- **M=VAL**  
This is the device “multiplier,” simulating the effect of multiple devices in parallel.  
Effective width, overlap, junction capacitances, and junction currents are multiplied by **M**.  
Parasitic resistance values are divided by **M**. Default value is 1.
- **DEBUG=VAL**  
Please see the notes below.

## Printing/Plotting States

The instance parameter **DEBUG** allows users to turn on debugging information selectively.  
Internal parameters (e.g. **par**) for an instance (e.g. **m1**) can be plotted by this command:

```
.plot <Analysis_Type> S(m1 -> par)
```

## Example

```
.plot DC S(m1 -> body)
```



For model parameters, please refer to the [Berkeley SPICE BSIM3SOI v1.3 Model \(Eldo Level 55\) Parameters](#) of the *Eldo Device Equations Manual*.

---

## Berkeley SPICE BSIM3SOI v2.x and v3.x Model (Eldo Level 56)

BSIM3SOI is the officially released SOI (Silicon On Insulator) MOSFET model from the Device Group at the University of California at Berkeley. Both BSIMSOI v2.x and v3.x models can be used for Partially Depleted (**PD**) and Fully Depleted (**FD**) devices. For the BSIMSOIv2.x model many advanced concepts were introduced so as to allow transition between **PD** and **FD** operation dynamically and continuously, namely the *Dynamic Depletion Approach (DD)*. The user is able to select one of these three modes using a parameter selector called **SOIMOD**. The basic I-V model is modified from the **BSIM3v3.1** equation set.

---

 Please refer to the [BSIM3SOI v2.x and v3.x Equations](#) of the *Eldo Device Equations Manual*.

---

The different versions are accessible through the model parameter **VERSION** as shown in the table below. By default, BSIM3SOIv3.2 (**VERSION**=3.2) is selected.

**Table 4-43. BSIM3SOI Version Selection**

Parameter Value	BSIM3SOI Version
VERSION=3.2	BSIM3SOIv3.2 (Default)
VERSION=3.11	BSIM3SOIv3.1.1
VERSION=3.1	BSIM3SOIv3.1
VERSION=3.0	BSIM3SOIv3.0
VERSION=2.23	BSIM3SOIv2.2.3
VERSION=2.22	BSIM3SOIv2.2.2
VERSION=2.21	BSIM3SOIv2.2.1
VERSION=2.1	BSIM3SOIv2.1

Parameter selector **SOIMOD** was an Eldo specific model parameter in the v2.x versions. Beginning version v3.0, it is a Berkeley standard model parameter to select between various SOI models: PD, FD and DD. For versions v3.0 and v3.1, the **SOIMOD** values have changed in Eldo to be compatible with the Berkeley standard values. Please see the following table:

**Table 4-44. SOIMOD Selection**

Model	SOIMOD for v2.x (Eldo specific)	SOIMOD for v3.0 (Spice compatible)	SOIMOD for v3.1 (Spice compatible)
<b>PD</b>	1	0 (default)	0 (default)
<b>DD</b>	2 (default)	-	-
<b>FD</b>	3	-	2

**Table 4-44. SOIMOD Selection**

Model	SOIMOD for v2.x (Eldo specific)	SOIMOD for v3.0 (Spice compatible)	SOIMOD for v3.1 (Spice compatible)
FD module over PD <sup>a</sup>	-	1	1

a. The FD module is an addition of some equations over the PD module to make the PD module also fit FD devices.

The different versions are handled separately inside this chapter. See “[BSIMPDv2.x](#)” on page 4-156, “[BSIMFDv2.1](#)” on page 4-156, “[BSIMDDv2.1](#)” on page 4-156, and “[BSIM3SOIv3.x](#)” on page 4-156.

## TNODEOUT keyword

**TNODEOUT** is a keyword that can be specified in the instantiation statement of an SOI device as follows:

```
Mxx nd ng ns ne <np> <nb> <nT> mname <L=val> <W=val> TNODEOUT
```

- If **TNODEOUT** is not specified, the user can specify four nodes for a device with floating body. Specifying five nodes implies that the fifth node is the external body contact node, with a body resistance between the internal and external terminals. This configuration applies to a distributed body resistance simulation. Specifying six nodes implies a body contacted case with an accessible internal body node (sixth node). Specifying seven nodes implies that the seventh node is the temperature node. This may be used to model thermal coupling.
- If **TNODEOUT** is specified, simulation interprets the last node as the temperature node. You can specify five nodes for a device with floating body. Specifying six nodes implies body contact. Seven nodes is a body contacted case with an accessible internal body node.

The major features are summarized as follows:

- Dynamic depletion approach is applied on both I-V and C-V. Charge and Drain current are scalable with **TBOX** and **TSI** continuously.
- Supports external body bias and backgate bias; a total of 6 nodes.
- Real floating body simulation in both I-V and C-V. Body potential is properly bounded by diode and C-V formulation.
- Self heating implementation improved over the alpha version.
- An improved impact ionization current model.
- Various diode leakage components and parasitic bipolar current included.
- New depletion charge model (**EBCI**) introduced for better accuracy in capacitive coupling prediction. An improved BSIM3v3 based model is also included.

- Dynamic depletion can suit different requirements for SOI technologies.
- Single I-V expression as in BSIM3v3.1 to guarantee the continuity of  $I_{ds}$ ,  $G_{ds}$  and  $G_m$  and their derivatives for all bias conditions.

## BSIM3SOIv2.x

### BSIMPDv2.x

BSIMPD is a Partially Depleted (**PD**) Silicon-on-Insulator (SOI) MOSFET model for SPICE simulation. This model is formulated on top of the BSIM3v3 framework. It shares the same basic equations with the bulk model so that the physical nature and smoothness of BSIM3v3 are retained. Most parameters related to general MOSFET operation (non-SOI specific) are directly imported from BSIM3v3 to ensure parameter compatibility.

Many enhanced features are included in BSIMPD through the joint effort of the BSIM Team at UC Berkeley and IBM Semiconductor Research and Development Center (SRDC) at East Fishkill. In particular, the model has been tested extensively within IBM on its state-of-the-art high speed SOI technology.

The latest version, BSIMPDv3.1.1, is implemented in Eldo. A version control parameter **VERSION** allows the use of older versions if required.

For model parameters, please refer to the [Berkeley SPICE BSIM3SOI PD Model \(Eldo Level 56\) Parameters](#) of the *Eldo Device Equations Manual*.

### BSIMFDv2.1

For model parameters, please refer to the [Berkeley SPICE BSIM3SOI FD v2.1 Model \(Eldo Level 56\) Parameters](#) of the *Eldo Device Equations Manual*.

### BSIMDDv2.1

For model parameters, please refer to the [Berkeley SPICE BSIM3SOI DD Model \(Eldo Level 56\) Parameters](#) of the *Eldo Device Equations Manual*.

## BSIM3SOIv3.x

Using BSIMPD as a foundation, a unified model is implemented for both PD and FD SOI circuit designs based on the concept of body-source built-in potential lowering.

In this version, BSIMSOI is constructed based on the concept of body-source built-in potential lowering,  $\Delta V_{bi}$ . There are three modes ( $soiMod = 0, 1, 2$ ) in BSIMSOI: BSIMPD ( $soiMod = 0$ ) can be used to model the PD SOI device, where the body potential is independent of  $\Delta V_{bi}$  ( $V_{BS} > \Delta V_{bi}$ ). Therefore the calculation of  $\Delta V_{bi}$  is skipped in this mode. On the other hand, the ideal FD model ( $soiMod = 2$ ) is for the FD device with body potential equal to  $\Delta V_{bi}$ . Hence the

calculation of body current/charge, which is essential to the PD model, is skipped. For the unified SOI model (*soiMod* = 1), however, both  $\Delta V_{bi}$  and body current/charge are calculated to capture the floating-body behavior exhibited in FD devices.

---

**Note**

 BSIMDD is not available for BSIM3SOIv3.x.

---

## BSIMSOIv3.x Parameter List

For model parameters, please refer to the [BSIMSOIv3.x Parameter List](#) of the *Eldo Device Equations Manual*.

## Philips MOS 9 Model (Eldo Level 59 or MOSP9)

Philips MOS 9 model is a compact model for MOS transistor, intended for the simulation of circuit behavior with emphasis on analog applications. This model has been developed originally by Philips Electronics, N.V and is now in the public domain.



Please refer to the [MOS Model 9 Equations](#) of the *Eldo Device Equations Manual*.

---

The implementation in Eldo is based on the unclassified report NL-UR 003/94 “MOS MODEL 9, level 902” issued in June 1995. The current implementation is MOS Model 9, level 903. In addition to the Philips syntax for the parameters, some equivalences to common approach parameters are made:

```
philips = eldo
LVAR = DL
LAP = LD
WVAR = DW
WOT = WD
NFR = KF
TR = TNOM
```

In these cases, the same default values are used.

Additionally, all the parameters available for the common approach are available for this model together with the corresponding equations set for parasitics. Furthermore, **AF** slope for noise has been added in Eldo. The overlap capacitances may be defined through the Philips parameter **COL** or through **CGDO**, **CGSO**. **CGBO** is also introduced.

The instantiation parameter **MULT** that indicates the number of devices in parallel is called **M** in Eldo.

The present restrictions in Eldo w.r.t. Philips implementation is that noise equations only include **Sfl** and **Sth** terms.

The model parameter **VERSION** can be set to values of 903.1 (default) or 903.2. When set to 903.2, it allows the model parameter **THE3R** to take negative values, otherwise **THE3R** is clipped to zero if negative.

**Table 4-45. Philips MOS9 Version Selection**

Parameter Value	Effect on THE3R
VERSION=903.1	Clip <b>THE3R</b> to zero if negative
VERSION=903.2	Allows <b>THE3R</b> to take negative values



For model parameters, please refer to the [Philips MOS 9 Model \(Eldo Level 59 Reference and Scaling Parameters](#) of the *Eldo Device Equations Manual*.

---

## Berkeley SPICE BSIM4 Model (Eldo Level 60)

There are six versions of this model, BSIM4.0.0, BSIM4.1.0, BSIM4.2.0, BSIM4.2.1, BSIM4.3.0, and BSIM4.4.0 (default).

---

**Note**



BSIM4 accepts the M factor.

---

The BSIM4 model has been developed to explicitly address many issues in modeling sub-0.13 micron CMOS technology and RF high-speed CMOS circuit simulation.



Please refer to the [BSIM4 Equations](#) of the *Eldo Device Equations Manual*.

---

### Use of Juncap diode (**DIOLEV=9.0**) with BSIM4

The Juncap diode (**DIOLEV=9**) can be used as a parasitic diode for the BSIM4 MOSFET model instead of that provided by Berkeley.

The Juncap diode can be chosen by specifying the model card parameter **DIOLEV=9.0**. Values for **DIOLEV** other than 9.0 will give a warning and the diode quantities will be calculated using the Berkeley parasitic diodes.

The parameters and equations of the Juncap diode (**DIOLEV= 9.0**) can all be found in the section “[Level 8 Equations](#)” on page 1-17 of the *Eldo Device Equations Manual*.

---

**Note**



The initialization of device parameters (*PS*, *AS*, *PD*, *AD*) and calculation of geometrical quantities (*PSeff*, *ASEff*, *PDeff*, *ADeff*) are all Berkeley standard for BSIM4; not those of the common equations.

---

## BSIM4 Model Selection via W/L Specifications

The BSIM4 model has unique conditions for W/L specification model selection. See “[Selection of MOSFET Models via W/L Specifications \(Binning\)](#)” on page 4-131.

The different versions—BSIM4.0.0, BSIM4.1.0, BSIM4.2.0, BSIM4.2.1, BSIM4.3.0, and BSIM4.4.0—are accessible through the model parameter **VERSION**. By default, BSIM4.4.0 (**VERSION**=4.4) is selected.

**Table 4-46. Berkeley BSIM4 Version Selection**

Parameter Value	BSIM4 Version
<b>VERSION</b> =4.4	BSIM4.4.0 (Default)
<b>VERSION</b> =4.3	BSIM4.3.0
<b>VERSION</b> =4.21	BSIM4.2.1
<b>VERSION</b> =4.2	BSIM4.2.0
<b>VERSION</b> =4.1	BSIM4.1.0
<b>VERSION</b> =4.0	BSIM4.0.0

## BSIM4.x Model Parameters

For model parameters, please refer to the [Berkeley BSIM4 Model \(Eldo Level 60\) Parameters](#) of the *Eldo Device Equations Manual*.

## BSIM4.4.0 Model Parameters

For model parameters, please refer to the [Berkeley BSIM4.4.0 Specific Model Parameters](#) of the *Eldo Device Equations Manual*.

## TFT Polysilicon Model (Eldo Level 62)

This is the modified polysilicon TFT model based on the original work at Rensselaer Polytechnic Institute (RPI).

This is a complete model developed for CAD and is based on the new universal charge control concept which guarantees stability and conversion. The unified DC model covers all regimes of operation and the AC model accurately reproduces frequency dispersion of capacitances (because of low mobility, carrier transit time is quiet; the signal period for even low frequency signals). This model provides automatic scaling of model parameters to accurately model a wide range of device geometries and physical based parameters can be easily extracted from experimental data.

**i** Please refer to the [TFT Polysilicon Model](#) of the *Eldo Device Equations Manual*.  
For model parameters, please refer to the [TFT Polysilicon Model \(Eldo Level 62\)](#)  
[Parameters](#) of the *Eldo Device Equations Manual*.

---

## Philips MOS Model 11 Level 1101 (Eldo Level 63)

A new compact model for MOS transistors has been developed. Philips MOS Model 11 (MM11), the successor of Philips MOS Model 9, not only gives an accurate description of charges and currents and their first-order derivatives (transconductance, conductance, capacitances), but also of their higher-order derivatives. In other words it gives an accurate description of MOSFET distortion behavior, and as such MM11 is suitable for digital, analog as well as RF circuit design.

**i** Please refer to the [MOS Model 11 Level 1100 & 1101 Equations](#) of the *Eldo Device Equations Manual*.

---

MOS Model 11 is a symmetrical, surface-potential-based model. It includes an accurate description of all physical effects important for modern and future CMOS technologies, such as, for example, gate tunnelling current, influence of pocket implants, poly-depletion, quantum-mechanical effects and bias-dependent overlap capacitances.

Level 1101 is an updated version of Level 1100. It uses the same basic equations as Level 1100, but uses different geometry scaling rules. It includes two types of geometrical scaling rules: physical rules and binning rules. Moreover, the temperature scaling has been implemented on the “miniset” level instead of the “maxiset” level as was the case for Level 1100.

The MM11 model (Level 1101) is implemented in Eldo as LEVEL=63.

**i** For information on the MM11 model (Level 1100), see [Philips MOS Model 11 Level 1100 \(Eldo Level 65\)](#).

---

Binning is used with this MM11 Level 1101 to decide which geometrical scaling rule is used. For Physical rule **BINNING**=0.0 and for Binning rule **BINNING**=1.0. By Default **BINNING**=0.0 (Physical rule) is used.

**Table 4-47. Philips MOS Model 11 Version Selection**

Parameter Value	Geometrical scaling rule
BINNING=0.0	Physical rule (Default)
BINNING=1.0	Binning rule

## Model Parameters

MM11 Level 1101(0) is specified with **BINNING**=0.0 (Physical rule).

For model parameters, please refer to the [MM11 Level 1101\(0\) Model Parameters Physical Rule](#) of the *Eldo Device Equations Manual*.

MM11 Level 1101(1) is specified with **BINNING**=1.0 (Binning rule).

For model parameters, please refer to the [MM11 Level 1101\(1\) Model Parameters Binning Rule](#) of the *Eldo Device Equations Manual*.

## TFT Amorphous-Si Model (Eldo Level 64)

This is the modified amorphous-silicon TFT model based on the original work at Rensselaer Polytechnic Institute (RPI).

The model provides the following features and benefits:

- Uses the new, universal charge control concept, which guarantees stability and convergence
- Unified DC models cover all regimes of operation
- AC models accurately reproduces **Cgc** frequency dispersion
- Automatic scaling of model parameters to accurately model a wide range of device geometries
- Temperature dependence included
- A minimum number of physically based parameters that can easily be extracted from experimental data and related back to the fabrication steps



Please refer to the [TFT Amorphous-Si Model](#) of the *Eldo Device Equations Manual*.

For model parameters, please refer to the [TFT Amorphous-Si Model \(Eldo Level 64\) Parameters](#) of the *Eldo Device Equations Manual*.

For model parameters, please refer to the [TFT Amorphous-Si Model Parasitic Parameters \(Access Resistance and Overlap Capacitance Parameters\)](#) of the *Eldo Device Equations Manual*.

## Printing or plotting a state from the TFT A-Si States structure

If a state from the list below is to be monitored by the user, the user has to type in the netlist for a given transistor M1 to monitor, for example Gm:

```
.PLOT DC S(M1->Gm) for DC or  
.PLOT AC S(M1->Gm) for AC or  
.PLOT TRAN S(M1->Gm) for TRAN
```

## Philips MOS Model 11 Level 1100 (Eldo Level 65)

A new compact model for MOS transistors has been developed. Philips MOS Model 11 (MM11), the successor of Philips MOS Model 9, not only gives an accurate description of charges and currents and their first-order derivatives (transconductance, conductance, capacitances), but also of their higher-order derivatives. In other words it gives an accurate description of MOSFET distortion behavior, and as such MM11 is suitable for digital, analog as well as RF circuit design.

MOS Model 11 is a symmetrical, surface-potential-based model. It includes an accurate description of all physical effects important for modern and future CMOS technologies, such as, for example, gate tunnelling current, influence of pocket implants, poly-depletion, quantum-mechanical effects and bias-dependent overlap capacitances.

There are two versions of this model. Level 1100 and Level 1101. Level 1101 is an updated version of Level 1100.

The MM11 model (Level 1100) is implemented in Eldo as LEVEL=65.

For information on the MM11 model (Level 1101), see [Philips MOS Model 11 Level 1101 \(Eldo Level 63\)](#).



Please refer to the [MOS Model 11 Level 1100 & 1101 Equations of the Eldo Device Equations Manual](#).

For model parameters, please refer to the [MM11 Level 1100 Model \(Eldo Level 65\) Parameters of the Eldo Device Equations Manual](#).

---

## HiSIM Model (Eldo Level 66)

HiSIM (Hiroshima University STARC IGFET Model) MOSFET model is a complete MOSFET surface potential model for circuit simulation based on the drift-diffusion approximation. The most important advantage of the drift-diffusion approximation is the unified description of device characteristics for all bias conditions. Under the gradual-channel approximation all device characteristics are described analytically by channel-surface potentials at the source side and at the drain side. These surface potentials are functions of applied voltages on four terminals; the gate voltage  $V_{gs}$ , the drain voltage  $V_{ds}$ , the bulk voltage  $V_{bs}$  and the earthed source. All phenomena such as short-channel and reverse-short-channel effects are therefore treated as results of the surface potential modification.

Since the surface potentials are implicit functions of applied voltages, iteration procedures are required in addition to global time-step iteration in circuit simulation. Therefore specific attention is paid on calculating the surface potentials with enough accuracy even with small CPU time. Up to now validity of HiSIM has been tested for the channel length down to  $0.1\mu m$  with the pocket-implanted technology. Though all descriptions are given for the n-channel MOSFET, they are also valid for the p-channel case.

The different versions, HiSIM1.0.2, HiSIM1.1.2, and HiSIM1.2.0 are accessible through the model parameter **VERSION**. By default, HiSIM1.2.0 (**VERSION**=120) is selected. Below is a table showing the different HiSIM versions selection.

**Table 4-48. HiSIM Version Selection**

Parameter Value	HiSIM Version
VERSION=120	HiSIM1.2.0 (Default)
VERSION=112	HiSIM1.1.2
VERSION=102	HiSIM1.0.2

It is recommended to use version 1.2.0 as earlier versions (1.1.2 and 1.0.2) are less stable and prone to numerical problems and errors.



- Please refer to the [HiSIM Model](#) of the *Eldo Device Equations Manual*.  
For model parameters, please refer to the [HiSIM Model \(Eldo Level 66\) Parameters](#) of the *Eldo Device Equations Manual*.  
For model parameters, please refer to the [HiSim Model Selector Parameters](#) of the *Eldo Device Equations Manual*.

## SP Model (Eldo Level 67)

SP is a generic compact MOSFET model developed at The Pennsylvania State University. It is surface-potential-based, free from unphysical behavior often associated with more traditional models and contains a relatively small number of parameters. The development of SP is based on solution of several long standing problems of compact MOSFET modeling.

Consequently SP is a surface potential based model that does not contain iterative loops or channel segmentation in both the intrinsic and the extrinsic submodels.



- Please refer to the [Surface-Potential-Based Compact MOSFET Model](#) of the *Eldo Device Equations Manual*.  
For model parameters, please refer to the [Ranges of SP Parameters](#), [Temperature dependence \(-55 to 150\)](#), and [Extrinsic Model Parameters](#) of the *Eldo Device Equations Manual*.

## Berkeley SPICE BSIM5 Model (Eldo Level 68)

---

**Note**



The BSIM5 model delivered with Eldo v6.6\_1 is a pre-version release. It is provided to interested users for testing purposes only. Use of this model for production activity should be avoided.

---

The BSIM5 model is based on the surface-potential-plus (SPP) core. The physical nature of the model results in smooth and continuous I-V and C-V equations. The flexible architecture of the model facilitates the easy incorporation of device physics effects and adaptation to new process technologies. For example, the model can be extended to non-classical devices such as the Double-Gate, Ultra-Thin-Body, and Multi-Gate MOSFETs. These extensions are under research in the BSIM group currently. The flexible architecture also enables the carry-over of BSIM4's accurate modeling of numerous device behaviors attributable to device physics or technologies.



Please refer to the [BSIM5 Equations](#) of the *Eldo Device Equations Manual*.

For model parameters, please refer to the [Berkeley BSIM5 Model \(Eldo Level 68\)](#) Parameters of the *Eldo Device Equations Manual*.

---

## Philips MOS Model 11 Level 1102 (Eldo Level 69)

MOS Model 11 (MM11, level 1102) is a new compact MOSFET model, intended for digital, analog and RF circuit simulation in modern and future CMOS technologies. MM1102 gives not only an accurate description of currents and charges and their first-order derivatives (i.e. transconductance, conductance, capacitances), but also of the higher order derivatives, resulting in an accurate description of electrical distortion behavior. The latter is especially important for analog and RF circuit design. The model furthermore gives an accurate description of the noise behavior of MOSFETs. Additionally, in order for the model to be valid for modern and future MOS devices, several important physical effects have been included in the model.

MM1102, is an updated version of MM1101. It uses slightly different equations. The surface potential generally is implicitly related to the terminal voltages and has to be calculated iteratively. Since the iterative procedure was assumed to be time consuming, the surface potential has been approximated by an explicit expression in MM1101. In the MM1102, the surface potential is calculated iteratively using a second-order Newton-Raphson procedure, resulting in a much more accurate description of surface potential which is obtained within three iterations. Owing to the increased accuracy, some of the basic equations used in MM1101 can be simplified, and as a result MM1102 is computationally as fast as MM1101. In addition, a more physical and simpler velocity saturation expression is used, and as a consequence the saturation voltage expression has changed slightly as well. This all results in a more accurate description of transconductance in saturation.

The MM11 model (Level 1102) is implemented in Eldo as LEVEL=69.



Please refer to the [MOS Model 11 Level 1102 Equations](#) of the *Eldo Device Equations Manual*.

For model parameters, please refer to the [MM11 Level 1102 Model Parameters \(Electrical\)](#), [MM11 Level 1102 Model Parameters \(Physical\)](#), and [MM11 Level 1102 Model Parameters \(Binning\)](#) of the *Eldo Device Equations Manual*.

---

## Philips PSP Model (Eldo Level 70)

The PSP model is a new compact MOSFET model, which has been jointly developed by Philips Research and Penn State University. It is a surface-potential based MOS Model, containing all relevant physical effects (mobility reduction, velocity saturation, DIBL, gate current, lateral doping gradient effects, etc.) to model present-day and upcoming deep-submicron CMOS technologies. Unlike previous Philips MOS models, the source/drain junction model, c.q. the JUNCAP2 model, is an integrated part of the PSP model.

The PSP model is a symmetrical, surface-potential-based model, giving an accurate physical description of the transition from weak to strong inversion. The PSP model includes an accurate description of all physical effects important for modern and future CMOS technologies, such as:

- mobility reduction
- bias-dependent series resistance
- velocity saturation
- conductance effects (CLM, DIBL, etc.)
- lateral doping gradient effect
- mechanical stress related to STI
- gate leakage current
- gate-induced drain leakage
- gate depletion
- quantum-mechanical effects bias-dependent overlap capacitances

In addition, it gives an accurate description of charges and currents and their first-order derivatives (transconductance, conductance, capacitances), but also of their higher-order derivatives. In other words, it gives an accurate description of MOSFET distortion behavior, and as such the PSP model is suitable for digital, analog as well as RF circuit design.



Please refer to the [PSP Model Equations](#) of the *Eldo Device Equations Manual*.

For model parameters, please refer to the [Parameter Scaling and Parameter Sets](#) of the *Eldo Device Equations Manual*.

---

## BTA HVMOS Model (Eldo Level 101)

HV (High-Voltage) MOS transistor model is based on the BSIM3v3 model. Major enhancements include current-crowding effect at high gate bias, asymmetric source-drain structure, self-heating, and more flexible gate-dependent output characteristics. Like BSIM3v3, the HVMOS transistor model also allows the binning option to achieve even higher accuracy. The binning equation is given by:

$$P = P_0 + \frac{P_l}{L_{eff}} + \frac{P_w}{W_{eff}} + \frac{P_p}{L_{eff} \cdot W_{eff}}$$

---

**Note**



Eldo defaults **L<sub>MIN</sub>** and **W<sub>MIN</sub>** to  $1 \times 10^{-6}$  m if the user did not specify these parameters.

To avoid mistakes in finding the correct models, please set the **L<sub>MIN</sub>** and **W<sub>MIN</sub>** values in your model cards.

---

### HVMOS License

The HVMOS model is a proprietary model of BTA Technology. To use the BTA HVMOS model, a license for Eldo from Mentor Graphics is required, *as well as* a license for the HVMOS model from BTA.

To setup the BTA license daemon for the HVMOS library, please refer to BTA's "License Installation and Management User Guide."



Please refer to the [BTA HVMOS Model](#) of the *Eldo Device Equations Manual*.  
For model parameters, please refer to the [BTA HVMOS Model \(Eldo Level 101\)](#)  
[Parameters](#) of the *Eldo Device Equations Manual*.

---

## S-Domain Filter

**FNS**xx IN OUT [**RIN**=val] [**ROUT**=val] NN {NN}, DN {DN}

The Eldo S-Domain filter is defined by the Transfer Function:

$$H(s) = \frac{N_0 + N_1 s + N_2 s^2 + N_3 s^3 + \dots + N_n s^n}{D_0 + D_1 s + D_2 s^2 + D_3 s^3 + \dots + D_m s^m}$$

### Parameters

- **xx**  
S-Domain filter name.
- **IN**  
Name of the input node.
- **OUT**  
Name of the output node.
- **RIN**  
Input resistance. Default is infinity. This resistance of value **RIN** is located between node **IN** and ground. Note that setting **RIN** to 0 will have the default effect, i.e. that **RIN** will be infinity.
- **ROUT**  
Output resistance. Default is 0.0. FNS output is equivalent to a voltage source in series with an output resistance **ROUT**.
- **NN**  
Coefficients of the Transfer Function numerator, starting from  $N_0$ .
- **DN**  
Coefficients of the Transfer Function denominator, starting from  $D_0$ .

### Example

```
fns1 n1 n2 1 2, 1 3 2
```

Specifies an S-Domain filter **fns1** with input and output nodes **n1** and **n2**. The Transfer Function is:

$$H(s) = \frac{1 + 2s}{1 + 3s + 2s^2}$$

It is possible to specify the order of the coefficient in brackets after giving the coefficient value. All non-specified coefficients will be set to 0, e.g.

```
fns2 1 2 rout=1 1.0e0(0), 1.0e0(0) 3.0e-8(1) 1.0e-16(3)
```

is equivalent to:

```
fns2 1 2 rout=1 1.0e0, 1.0e0 3.0e-8 0 1.0e-16
```

## Z-Domain Filter

```
FNZXX IN OUT FREQ=VAL [RIN=val] [ROUT=val] NN {NN}, DN {DN}
```

The Eldo Z-Domain filter is defined by the Transfer Function:

$$H(z) = \frac{N_0 + N_1 z^{-1} + N_2 z^{-2} + N_3 z^{-3} + \dots + N_n z^{-n}}{D_0 + D_1 z^{-1} + D_2 z^{-2} + D_3 z^{-3} + \dots + D_m z^{-m}}$$

### Parameters

- **xx**  
Z-Domain filter name.
- **IN**  
Name of the input node.
- **OUT**  
Name of the output node.
- **FREQ=VAL**  
Sampling frequency in Hertz.
- **RIN**  
Input resistance. Default is infinity. This resistance of value **RIN** is located between node **IN** and ground. Note that setting **RIN** to 0 will have the default effect, i.e. that **RIN** will be infinity.
- **ROUT**  
Output resistance. Default is 0.0. FNZ output is equivalent to a voltage source in series with an output resistance **ROUT**.
- **NN**  
Coefficients of the Transfer Function numerator, starting from  $N_0$ .
- **DN**  
Coefficients of the Transfer Function denominator, starting from  $D_0$ .

The AC response of Z transforms was modified, beginning Eldo v6.3, to be consistent with that of ADMS. The term  $\sin x/x$  is taken into account by Eldo. Use the option **NOZSINXX** to remove the effect of that term for pre-v6.3 functionality.

### Example

```
fnz1 n1 n2 freq=1k 1 3, 1 2 4
```

A Z-Domain filter **f nz1** with input and output nodes **n1** and **n2**, clocked at 1kHz. The Transfer Function is:

$$H(z) = \frac{1 + 3z^{-1}}{1 + 2z^{-1} + 4z^{-2}}$$

It is possible to specify the order of the coefficient in brackets after giving the coefficient value. All non-specified coefficients will be set to 0, e.g.

```
f nz2 1 2 rout=1 1.0e0(0), 1.0e0(0) 3.0e-8(1) 1.0e-16(3)
```

is equivalent to:

```
f nz2 1 2 rout=1 1.0e0, 1.0e0 3.0e-8 0 1.0e-16
```

## Subcircuit Instance

```
xxx NN {NN} NAME [PAR=VAL] [PAR={EXPR}] [M=VAL] [TEMP=VAL]
+ [(SWITCH|ANALOG|OSR|DIGITAL)] [NONOISE|NOISE=0]
```

This statement is used to instantiate a subcircuit that has been previously defined using a **.SUBCKT** command. Subcircuit definitions and instances can be nested. Parameters contained in a subcircuit can be assigned explicitly or via the **.PARAM** command, direct assignment always taking precedence over **.PARAM** commands.

In order to improve execution speed the subcircuit can be optionally solved using the differentiated accuracy system.



Before using this system, refer to “[Speed and Accuracy](#)” on page 16-1.

### Parameters

- **xx**  
Subcircuit name.
- **NN**  
Names of the nodes to be connected externally. Nodes are referenced in the order they appear in a **.SUBCKT** command. **SWITCH** is an invalid node name since this name is interpreted as an optional parameter in Eldo-XL.
- **NAME**  
Name of the subcircuit being instantiated, as specified by the **.SUBCKT** command.
- **PAR=VAL**  
Specifies that the parameter **PAR** is assigned the value **VAL** inside the subcircuit. This parameter assignment takes precedence over any parameter assignments occurring in the **.SUBCKT** command.
- **M=VAL**  
Multiplication factor for the subcircuit instantiation. This effectively places **M** instances of the subcircuit in parallel, all connected to the specified nodes **NN**. If **M=0** is specified, the corresponding subcircuit instance will be ignored (as if the subcircuit instance is commented out). Every element defined in the subcircuit will be duplicated by this multiplication factor.
- **TEMP=VAL**  
Sets temperature for the individual subcircuit. This overrides the temperature of devices which are instantiated in the X instance. Default nominal temperature=27 °C.
- **(ANALOG)**  
Keyword used with the differentiated accuracy system indicating that the subcircuit should be solved using Newton block iteration techniques. These techniques are used in

conjunction with the **EPS** parameter in the **.OPTION** command. (**ANALOG**) basically means “high accuracy.”

- **(SWITCH)**

Keyword used with the differentiated accuracy system. See also **.OPTION NOSWITCH** on [page 11-18](#).

---

**i** Before using the differentiated accuracy system see the relevant section in the [Speed and Accuracy](#) chapter.

---

- **(OSR | DIGITAL)**

Used to stop propagation of the **ANALOG** flag across the hierarchy. In addition the flag will request Eldo to use OSR in the selected blocks, if possible (i.e. MOS subcircuit with **OSR** flag could then be solved by OSR, but BJT subcircuit will still be solved by Newton even if flag **OSR** is set).

- **NONOISE**

Specifies that no noise model will be used for this subcircuit when performing noise analysis. Therefore, the subcircuit presents no noise contribution to the noise analysis. For more details see [“.SUBCKT” on page 10-306](#).

- **NOISE=0**

Synonymous with **NONOISE** keyword. Specifies that no noise model will be used for this subcircuit when performing noise analysis. Therefore, the subcircuit presents no noise contribution to the noise analysis. Specifying any value other than zero will cause Eldo to issue a warning, and the parameter will be ignored.

## Combining identical subcircuits

When **.OPTION REDUCE** is set, multiple identical **subckt** instances that follow each other are reduced into a single instance using the **M** parameter, for example:

```
x1 1 2 FOO A=1 B=1
x2 1 2 FOO A=1 B=1
x3 1 2 FOO A=1.0 B=1
.OPTION REDUCE
```

Here, X instances **x1** and **x2** will be replaced by:

```
x1 1 2 FOO A=1 B=1 M=2
```

but **x3** will remain as it is because the character string for **A** does not match.

---

**i** This also applies to BJTs, MOSFETs and diodes. For more information see [page 4-111](#), [page 4-127](#) and [page 4-104](#) respectively.

---

## Example

```
*SUBCKT definition
.subckt inv 1 2
r1 1 3 2k
r2 3 4 4k
r3 4 2 3k
.ends inv
...
*subcircuit instance
x1 1 48 inv
.print v(x1.1)
.plot v(x1.1)
```

Specifies the instantiation of subcircuit `inv` with instance name `x1` placed between nodes 1 and 48. Subcircuit node 1 is shown both printed and plotted.

```
*SUBCKT definition
.subckt inv 1 2
r1 1 3 rval
r2 3 4 rval1
r3 4 2 rval2
.ends inv
...
*subcircuit instance
x1 1 2 inv rval=6 rval2={rval1+1k}
.param rval1 = 10
x2 2 3 inv rval=4
.ic v(x1.1)=0
```

Specifies two instantiations of subcircuit `inv` with instance names `x1` and `x2`. Parameters are assigned directly, using expressions and via the `.PARAM` command. Subcircuit node 2 is given the initial condition of 0V via the `.IC` command.



# Chapter 5 Sources

---

## Introduction

Eldo provides a number of sources (stimuli generators) which can be divided into four groups as shown below:

## Independent Sources

Two types of independent sources are provided:

Independent Voltage Source	<b>V</b>
Independent Current Source	<b>I</b>

Independent sources can be assigned a time-dependent value for transient analysis. The time zero values of time dependent sources are used for DC analysis. Eight types of time dependent source are provided:

Amplitude Modulation Function	<b>AM</b>
Exponential Function	<b>EXP</b>
Noise Function	<b>NOISE</b>
Pattern Function	<b>PATTERN</b>
Pulse Function	<b>PULSE</b>
Piece Wise Linear Function	<b>PWL</b>
Single Frequency FM Function	<b>SFFM</b>
Sine Function	<b>SIN</b>
Trapezoidal Pulse with Bit Pattern Function	<b>PBIT</b>
Exponential Pulse with Bit Pattern Function	<b>EBIT</b>

## Linear Dependent Sources

Four types of linear dependent sources are provided:

Linear Voltage Controlled Voltage Source ( $v = e \cdot v$ )	<b>E</b>
Linear Current Controlled Current Source ( $i = f \cdot i$ )	<b>F</b>

Linear Voltage Controlled Current Source ( $i = g \cdot v$ ) G

Linear Current Controlled Voltage Source ( $v = h \cdot i$ ) H

where E, F, G and H are constants representing voltage gain, current gain, transconductance and transresistance respectively.

## Non-linear Dependent Sources

Four types of non-linear dependent sources are provided, defined by:

Non-linear Voltage Controlled Voltage Source ( $v = f(v)$ ) E

Non-linear Current Controlled Current Source ( $i = f(i)$ ) F

Non-linear Voltage Controlled Current Source ( $i = f(v)$ ) G

Non-linear Current Controlled Voltage Source ( $v = f(i)$ ) H

where the function is a polynomial and the arguments multi-dimensional. The polynomial functions are specified by the coefficients  $p_0 \dots p_n$ . The significance of the coefficients depends upon the order of the polynomial, as shown below:

### 1st order polynomial

$f_a$  is the function argument. The function value  $f_v$  is computed in the following manner:

$$f_v = P_0 + (P_1 \cdot f_a) + (P_2 \cdot f_a^2) + (P_3 \cdot f_a^3) + (P_4 \cdot f_a^4) + \dots$$

### 2nd order polynomial

$f_a$  and  $f_b$  are the function arguments. The function value  $f_v$  is computed in the following manner:

$$f_v = P_0 + (P_1 \cdot f_a) + (P_2 \cdot f_b) + (P_3 \cdot f_a^2) + (P_4 \cdot f_a \cdot f_b) + (P_5 \cdot f_b^2) + (P_6 \cdot f_a^3) + (P_7 \cdot f_a^2 \cdot f_b) + (P_8 \cdot f_a \cdot f_b^2) + \dots$$

### 3rd order polynomial

$f_a$ ,  $f_b$  and  $f_c$  are the function arguments. The function value  $f_v$  is computed in the following manner:

$$f_v = P_0 + (P_1 \cdot f_a) + (P_2 \cdot f_b) + (P_3 \cdot f_c) + (P_4 \cdot f_a^2) + (P_5 \cdot f_a \cdot f_b) + (P_6 \cdot f_a \cdot f_c) + (P_7 \cdot f_b^2) + (P_8 \cdot f_b \cdot f_c) + (P_9 \cdot f_c^2) + (P_{10} \cdot f_a^3) + \dots$$

The following pages contain syntax and parameter explanations, together with a number of worked examples, of each of the source options provided by Eldo.

## S, Y, Z Parameter Extraction

A set of commands in Eldo allow the user to extract the S parameters (Scattering parameters), the Y parameters (Admittance) or the Z parameters (Impedance) in the frequency domain for a specified circuit. The circuit can have any number of ports. Special sources must be added at each port of the circuit to be analyzed.



See “S, Y, Z Parameter Extraction” on page 5-61.

---

# Independent Voltage Source

## Independent Source Element

```
Vxx NP NN [[DC] DCVAL] [AC [ACMAG [ACPHASE]]]
+ [TIME_DEPENDENT_FUNCTION1] [TC1=val] [TC2=val]
+ [RPORT=val [NONOISE]] [RPORT_TC1=val] [RPORT_TC2=val]
+ [IPORT=val] [CPORT=val] [LPORT=val] [MODE=keyword] [NOISETEMP=val]
Vxx NP NN [[DC] DCVAL] [AC [ACMAG [ACPHASE]]]
+ [TIME_DEPENDENT_FUNCTION1] [TC1=val] [TC2=val]
+ ZPORT_FILE=string [IPORT=val] [CPORT=val] [LPORT=val] [MODE=keyword]
+ [NOISETEMP=val]
```

## Noise Source

```
Vxx NP NN [[DC] DCVAL] [AC [ACMAG [ACPHASE]]]
+ [TC1=val] [TC2=val] [RPORT=val [NONOISE]]
+ [RPORT_TC1=val] [RPORT_TC2=val] [IPORT=val] [CPORT=val]
+ [LPORT=val] [MODE=keyword] [NOISETEMP=val] NOISE [THN=VAL] [FLN=VAL]
+ [ALPHA=VAL] [FC=VAL] [N=VAL] [FMIN=VAL] [FMAX=VAL]
+ [NBF=VAL]
Vxx NP NN [[DC] DCVAL] [AC [ACMAG [ACPHASE]]]
+ [TC1=val] [TC2=val] ZPORT_FILE=string [IPORT=val] [CPORT=val]
+ [LPORT=val] [MODE=keyword] [NOISETEMP=val] NOISE [THN=VAL] [FLN=VAL]
+ [ALPHA=VAL] [FC=VAL] [N=VAL] [FMIN=VAL] [FMAX=VAL]
+ [NBF=VAL]
```

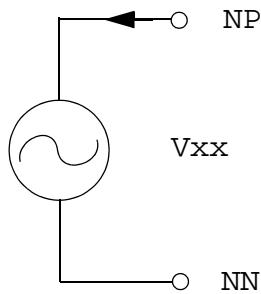
## Tabular Noise Source

```
Vxx NP NN [[DC] DCVAL] [AC [ACMAG [ACPHASE]]]
+ [TC1=val] [TC2=val] [RPORT=val [NONOISE]]
+ [RPORT_TC1=val] [RPORT_TC2=val] [IPORT=val] [CPORT=val]
+ [LPORT=val] [MODE=keyword] [NOISETEMP=val] NOISE TABLE
+ [[INTERP=]DEC|OCT|LIN|LOG|HARM_DEC|HARM_OCT] [DB|MA]
+ (f1 val1) (f2 val2) ...
Vxx NP NN [[DC] DCVAL] [AC [ACMAG [ACPHASE]]]
+ [TC1=val] [TC2=val] ZPORT_FILE=string [IPORT=val] [CPORT=val]
+ [LPORT=val] [MODE=keyword] [NOISETEMP=val] NOISE TABLE
+ [[INTERP=]DEC|OCT|LIN|LOG|HARM_DEC|HARM_OCT] [DB|MA]
+ (f1 val1) (f2 val2) ...
```

## Multi-Tone Source

```
Vxx NP NN [[DC] DCVAL] [AC [ACMAG [ACPHASE]]]
+ [TC1=val] [TC2=val] [RPORT=val [NONOISE]]
+ [RPORT_TC1=val] [RPORT_TC2=val] [IPORT=val] [CPORT=val]
+ [LPORT=val] [MODE=keyword] [NOISETEMP=val] FOUR
+ fund1 [fund2 [fund3]] MA|RI|DB|PMA|PDB|PDBM
+ (int_val1 [,int_val2 [,int_val3]]) real_val1 real_val2
+ {(int_val1 [,int_val2 [,int_val3]]) real_val1 real_val2}
Vxx NP NN [[DC] DCVAL] [AC [ACMAG [ACPHASE]]]
+ [TC1=val] [TC2=val] ZPORT_FILE=string
+ [IPORT=val] [CPORT=val] [LPORT=val] [MODE=keyword] [NOISETEMP=val] FOUR
+ fund1 [fund2 [fund3]] MA|RI|DB|PMA|PDB|PDBM
+ (int_val1 [,int_val2 [,int_val3]]) real_val1 real_val2
+ {(int_val1 [,int_val2 [,int_val3]]) real_val1 real_val2}
```

1. Refer to the EXP, PATTERN, PULSE, PWL, SFFM and SIN source functions.



**Note**



The current flows from the positive node NP, through the voltage source, to the negative node NN.

---

## Parameters

- **xx**  
Independent voltage source name.
- **NP**  
Name of the positive node.
- **NN**  
Name of the negative node.
- **DCVAL**  
Value of the DC voltage.
- **ACMAG**  
AC magnitude in volts. Default value is 1.
- **ACPHASE**  
AC phase in degrees. Default value is zero.
- **TIME\_DEPENDENT\_FUNCTION**  
Refers to the time dependence of the voltage source (**PWL**, **PULSE**, **EXP**, **PATTERN**, **SSFM** and **SIN**). A multi-tone sine wave time dependence can also be defined with the **FOUR** keyword as detailed in the separate syntax. (See below for more details).
- **TC1, TC2**  
First and Second order temperature coefficients. Default values are zero. **TC2** can be specified even if **TC1** is not.

$$'VAL(T) = VAL(T_{nom})(1 + TC1(T - T_{nom}) + TC2(T - T_{nom})^2)$$

The above equation defines the value of the voltage (*VVAL*) as a function of temperature, where **T** is the operating temperature specified either by the **.TEMP** command, or the **T** parameter. **T<sub>nom</sub>** is the nominal temperature for which the voltage source has voltage **VAL**. Default value of **T<sub>nom</sub>** is 27 °C and is adjustable using **.OPTION TNOM**.

- **NOISE**

Generates a noise source.

---

**Note**



For the Noise Source, at least one parameter has to be specified after the **NOISE** keyword, otherwise Eldo issues the following message: “Incorrect number of parameters for noise source VNOISE”

---

- **THN**

Defines the white noise level in A<sup>2</sup>/Hz or V<sup>2</sup>/Hz respectively.

- **FLN**

Defines the 1/f or Flicker Noise level at 1 Hz in A<sup>2</sup>.Hz<sup>(1-alpha)</sup> or V<sup>2</sup>.Hz<sup>(1-alpha)</sup> respectively. Default is 0.

- **ALPHA**

Frequency exponent for 1/f noise.

- **FC**

Cut-off frequency of the low pass noise filter.

- **N**

Filter order.

- **FMIN**

Lower limit of the noise frequency band.

- **FMAX**

Upper limit of the noise frequency band.

---

**Note**



**FMIN** and **FMAX** define the frequency band of the noise sources. This frequency range may sometimes not correspond to the noise frequency band at the output of the circuit. For instance, the band (**FMIN**, **FMAX**) does not correspond to the output noise frequency band in the case of filters or oscillators and mixers that exhibit frequency conversion.

---

- **NBF**

Specifies the number of sinusoidal sources with randomly distributed amplitude and phase from which the noise source is composed. Default is 50.

**Note**



Parameters **FMIN**, **FMAX** and **NBF** of Noise Sources are taken into account for Transient analysis only.



Please refer to “[.NOISETRAN](#)” on page 10-182 for further information.

- **TABLE**

Keyword indicating that the source has a tabular description.

- **INTERP**

Specifies how to interpolate between different frequency values: **LIN** means linear interpolation; **LOG**, **OCT** or **DEC** are all used in the same sense which is logarithmic, octal, or decimal interpolation; **HARM\_DEC** or **HARM\_OCT** specify a logarithmic or octal interpolation around each harmonic of the **.SSTNOISE** analysis.

- **f1, f2**

Frequency values in Hertz.

- **vall, val2**

Values in V<sup>2</sup>/Hz.

- **FOUR**

Keyword specifying that the source is multi-tone.

- **fund1 [fund2 [fund3]]**

Parameters to define fundamental frequencies. A source can be defined with up to 3 tones.

- **MA | RI | DB | PMA | PDB | PDBM**

Keyword defining the format (**MA**—Magnitude Angle, **RI**—Real Imaginary, **DB**—Magnitude in dB Angle, **PMA**—Power in Watt Angle, **PDB**—Power in dB Angle, **PDBM**—Power in dBm Angle). Power formats (**PMA**, **PDB** and **PDBM**) are only allowed on port sources (voltage or current sources where **IPORT** is specified). For multi-tone, the format is used in conjunction with the **real\_vall**, **real\_val2** specification below. For tabular (only **DB | MA**), values are couples **f1 vall**, see above.

- **int\_vall**

Defines the index of the harmonic according to fund1.

- **int\_val2**

Defines the index of the harmonic according to fund2.

- **int\_val3**

Defines the index of the harmonic according to fund3.

The group of 1 to 3 index values define a frequency. For example, for a source with 3 fundamental frequencies, (**int\_vall**, **int\_val2**, **int\_val3**) specifies the frequency:

$F = \text{int\_val1} * \text{fund1} + \text{int\_val2} * \text{fund2} + \text{int\_val3} * \text{fund3}$

- **real\_val1, real\_val2**

Defines the complex value of the source for the corresponding frequency in the specified format (MA, RI or DB).

- **RPORT**

Resistance of the port value which defaults to  $50\ \Omega$  whenever **RPORT\_TC1**, **RPORT\_TC2** or **IPORT** is specified. The possibility of having **NONOISE** on this resistance is allowed.

- **ZPORT\_FILE**

Specifies the Touchstone file name that contains the complex port impedance.

- **NONOISE**

Used in conjunction with **RPORT**. Specifies that no noise model will be used for this device when performing noise analysis. Therefore, the port resistance presents no noise contribution to the noise analysis.

- **RPORT\_TC1 & RPORT\_TC2**

Temperature coefficients of **RPORT**: they both default to 0. If **RPORT** happens to be 0 (in parametric simulations for instance), there will be no S extraction performed.

- **IPORT**

This is a strictly positive number that is unique and is used as the port number: this number is used for naming the outputs (for instance, `.PLOT AC S(1, 2)`). An error message will be issued if two port instances have the same value for **IPORT**, or if an **IPORT** is missing (e.g. maximum **IPORT** number found in the netlist is 4, and there is no instance with **IPORT 3**).

- **CPORT**

Capacitor placed in series with **RPORT**. Defaults to 0, in which case it behaves like a zero voltage source (i.e. **CPORT** has no effect).

- **LPORT**

Inductor placed in series with **RPORT**. Defaults to 0.

- **MODE=SINGLE | COMMON | DIFFERENTIAL**

Mixed-mode S parameter selection.

**SINGLE** specifies the port as single ended, it is dedicated to S parameter extraction.  
Default.

**COMMON** and **DIFFERENTIAL** specify that the port is not single ended. Such ports are split into two linked sources that are either common (same amplitude and same phase) or differential (same amplitude but opposite phases). If the S parameters are extracted a “non single ended” port is equally common and differential depending on which display is required. During simulation (DC, AC or TRAN) this port is either common or differential depending on the specified mode keyword.

- **NOISETEMP**

Corresponds to the temperature value used for the calculation of the NOISE generated by **RPORT**. The default is the temperature of the circuit, but should be set to 16.85 to comply with IEEE specifications. When this parameter is specified, it overrides the **INPUT\_TEMP** parameter in the **.SNF** command.

## Examples

```
vplus n12 n13 24
```

Specifies a fixed voltage of 24V between nodes n12 and n13.

```
v7 n4 n9 dc 1.2 ac 1.0e-3
```

Specifies a DC voltage of 1.2V placed between nodes n4 and n9 and an AC voltage of 1mV.

```
v7 n4 n9 ac 1.2 pwl (0 3 5n 0 10n 0)
```

Specifies an AC voltage source of 1.2V together with a Piece Wise Linear (**PWL**) voltage source definition placed between nodes n4 and n9.

```
V1 n1 n2 FOUR fund1 MA
+ (0) 5 0
+ (1) 2.5 -90
.param fund1=100meg
```

The above example defines a source having the following time dependence:

```
V1(t) = 5 + 2.5*sin(2*pi*100meg*t)
V2 n3 n4 FOUR fund1 fund2 DB
+ (0, 0) 0 0
+ (1, 0) 6 -90
+ (0, 1) -6 0
+ (-2, 2) 0 45
.param fund1=900meg fund2=1.2giga
```

The value of V2 will be computed as follows:

$$V2 = A + B\sin(2\pi \cdot 900\text{e}6 \cdot t) + C\cos(2\pi \cdot 1.2\text{e}9 \cdot t) + A\cos((-2\pi \cdot 900\text{e}6 + 2\pi \cdot 1.2\text{e}9) \cdot 2\pi \cdot t + \pi/4)$$

with:

A is computed as 0dB from 1V, i.e. A is 1V

B is 6dB with reference to 1V, then it is roughly 2.0V

C is -3dB with reference to 1V, then it is roughly 0.5V

The  $\pi/4$  term at the end of the expression corresponds to 45 degrees, as specified in the SPICE card.

**Note**

 Independent voltage sources defined to have a voltage of 0V may be removed by Eldo in order to simplify calculations. If you wish to use a voltage source as a current probe, use **.OPTION AMMETER** to prevent Eldo from removing such voltage sources defined to have a voltage of 0V. Moreover, currents through components may be measured directly, therefore, the use of voltage sources as current probes is not usually necessary.

The next example specifies a current source of 1A between nodes 1 and 0. It has a first order temperature coefficient (TC1) of 1. The second order temperature coefficient (TC2) will default to 0.

```
v1 1 0 1 tc1=1
```

**Table noise input source**

It is possible to provide Eldo some input noise source values depending on the frequency via the **TABLE** keyword. Eldo accepts the following:

```
Vxx P1 P2 NOISE TABLE [DEC|LOG|LIN] (f1,val1) (f2,val2)....
```

Eldo will assume zero current source or zero voltage source in any mode other than **NOISE** (i.e. for **DC**, **AC**, **TRAN**, etc.).

# Independent Current Source

## Independent Source Element

```
Ixx NP NN [[DC] DCVAL] [AC [ACMAG [ACPHASE]]]
+ [TIME_DEPENDENT_FUNCTION1] [TC1=val] [TC2=val]
+ [RPORT=val [NONOISE]] [RPORT_TC1=val] [RPORT_TC2=val]
+ [IPORT=val] [CPORT=val] [LPORT=val] [NOISETEMP=val]
Ixx NP NN [[DC] DCVAL] [AC [ACMAG [ACPHASE]]]
+ [TIME_DEPENDENT_FUNCTION1] [TC1=val] [TC2=val]
+ ZPORT_FILE=string [IPORT=val] [CPORT=val] [LPORT=val] [MODE=keyword]
+ [NOISETEMP=val]
```

## Noise Sources

```
Ixx NP NN [[DC] DCVAL] [AC [ACMAG [ACPHASE]]]
+ [TC1=val] [TC2=val] [RPORT=val [NONOISE]]
+ [RPORT_TC1=val] [RPORT_TC2=val] [IPORT=val] [CPORT=val]
+ [LPORT=val] [NOISETEMP=val] NOISE [THN=VAL] [FLN=VAL]
+ [ALPHA=VAL] [FC=VAL] [N=VAL] [FMIN=VAL] [FMAX=VAL]
+ [NBF=VAL]
Ixx NP NN [[DC] DCVAL] [AC [ACMAG [ACPHASE]]]
+ [TC1=val] [TC2=val] [IPORT=val] [CPORT=val]
+ ZPORT_FILE=string [LPORT=val] [MODE=keyword] [NOISETEMP=val]
+ NOISE [THN=VAL] [FLN=VAL] [ALPHA=VAL] [FC=VAL] [N=VAL] [FMIN=VAL]
+ [FMAX=VAL] [NBF=VAL]
```

## Tabular Noise Source

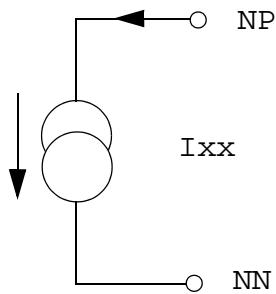
```
Ixx NP NN [[DC] DCVAL] [AC [ACMAG [ACPHASE]]]
+ [TC1=val] [TC2=val] [RPORT=val [NONOISE]]
+ [RPORT_TC1=val] [RPORT_TC2=val] [IPORT=val] [CPORT=val]
+ [LPORT=val] [NOISETEMP=val] NOISE TABLE
+ [[INTERP=]DEC|OCT|LIN|LOG|HARM_DEC|HARM_OCT] (f1 val1) (f2 val2) ...
Ixx NP NN [[DC] DCVAL] [AC [ACMAG [ACPHASE]]]
+ [TC1=val] [TC2=val] ZPORT_FILE=string [IPORT=val] [CPORT=val]
+ [LPORT=val] [MODE=keyword] [NOISETEMP=val] NOISE TABLE
+ [[INTERP=]DEC|OCT|LIN|LOG|HARM_DEC|HARM_OCT] (f1 val1) (f2 val2) ...
```

## Multi-Tone Source

```
Ixx NP NN [[DC] DCVAL] [AC [ACMAG [ACPHASE]]]
+ [TC1=val] [TC2=val] [RPORT=val [NONOISE]]
+ [RPORT_TC1=val] [RPORT_TC2=val] [IPORT=val] [CPORT=val]
+ [LPORT=val] [NOISETEMP=val] FOUR fund1 [fund2 [fund3]]
+ MA|RI|DB|PMA|PDB|PDBM
+ (int_val1 [,int_val2 [,int_val3]]) real_val1 real_val2
+ {(int_val1 [,int_val2 [,int_val3]]) real_val1 real_val2}
Ixx NP NN [[DC] DCVAL] [AC [ACMAG [ACPHASE]]]
+ [TC1=val] [TC2=val] ZPORT_FILE=string [IPORT=val] [CPORT=val]
+ [LPORT=val] [MODE=keyword] [NOISETEMP=val] FOUR fund1 [fund2 [fund3]]
+ MA|RI|DB|PMA|PDB|PDBM
+ (int_val1 [,int_val2 [,int_val3]]) real_val1 real_val2
+ {(int_val1 [,int_val2 [,int_val3]]) real_val1 real_val2}
```

---

1. Refer to the EXP, PULSE, PWL, SFFM and SIN source functions.



**Note**



The current flows from the positive node **NP**, through the current source, to the negative node **NN**.

---

**Parameters**

- **xx**  
Independent current source name.
- **NP**  
Name of the positive node.
- **NN**  
Name of the negative node.
- **DCVAL**  
Value of the DC current source in amperes.
- **ACMAG**  
AC magnitude in amperes. Default value is 1.
- **ACPHASE**  
AC phase in degrees. Default value is 0.
- **TIME\_DEPENDENT\_FUNCTION**  
Refers to the time dependence of the voltage source (**PWL**, **PULSE**, **EXP**, **PATTERN**, **SSFM** and **SIN**). A multi-tone sine wave time dependence can also be defined with the **FOUR** keyword as detailed in the separate syntax. (See below for more details).
- **TC1, TC2**  
First and Second order temperature coefficients. Default values are zero. **TC2** can be specified even if **TC1** is not.

$$IVAL(T) = VAL(T_{nom})(1 + TC1(T - T_{nom}) + TC2(T - T_{nom})^2)$$

The above equation defines the value of the current (*IVAL*) as a function of temperature, where **T** is the operating temperature specified either by the **.TEMP** command, or the **T** parameter. **T<sub>nom</sub>** is the nominal temperature for which the current source has current **VAL**. Default value of **T<sub>nom</sub>** is 27 °C and is adjustable using **.OPTION TNOM**.

- **NOISE**

Generates a noise source.

**Note**



For the Noise Source, at least one parameter has to be specified after the **NOISE** keyword, otherwise Eldo issues the following message: “Incorrect number of parameters for noise source INOISE”

---

- **THN**

Defines the white noise level in A<sup>2</sup>/Hz or V<sup>2</sup>/Hz respectively.

- **FLN**

Defines the 1/f or Flicker Noise level at 1 Hz in A<sup>2</sup>.Hz<sup>(1-alpha)</sup> or V<sup>2</sup>.Hz<sup>(1-alpha)</sup> respectively. Default is 0.

- **ALPHA**

Frequency exponent for 1/f noise.

- **FC**

Cut-off frequency of the low pass noise filter.

- **N**

Filter order.

- **FMIN**

Lower limit of the noise frequency band.

- **FMAX**

Upper limit of the noise frequency band.

**Note**



**FMIN** and **FMAX** define the frequency band of the noise sources. This frequency range may sometimes not correspond to the noise frequency band at the output of the circuit. For instance, the band (**FMIN**, **FMAX**) does not correspond to the output noise frequency band in the case of filters or oscillators and mixers that exhibit frequency conversion.

---

- **NBF**

Specifies the number of sinusoidal sources with randomly distributed amplitude and phase from which the noise source is composed. Default is 50.

**Note**

Parameters **FMIN**, **FMAX** and **NBF** of Noise Sources are taken into account for Transient analysis only.



Please refer to “[.NOISETRAN](#)” on page 10-182 for further information.

- **TABLE**

Keyword indicating that the source has a tabular description.

- **INTERP**

Specifies how to interpolate between different frequency values: **LIN** means linear interpolation; **LOG**, **OCT** or **DEC** are all used in the same sense which is logarithmic, octal, or decimal interpolation; **HARM\_DEC** or **HARM\_OCT** specify a logarithmic or octal interpolation around each harmonic of the **.SSTNOISE** analysis..

- **f1, f2**

Frequency values in Hertz.

- **val1,val2**

Values in A<sup>2</sup>/Hz.

- **FOUR**

Keyword specifying that the source is multi-tone.

- **fund1 [fund2 [fund3]]**

Parameters to define fundamental frequencies. A source can be defined with up to 3 tones.

- **MA | RI | DB | PMA | PDB | PDBM**

Keyword defining the format (**MA**—Magnitude Angle, **RI**—Real Imaginary, **DB**—Magnitude in dB Angle, **PMA**—Power in Watt Angle, **PDB**—Power in dB Angle, **PDBM**—Power in dBm Angle). Power formats (**PMA**, **PDB** and **PDBM**) are only allowed on port sources (voltage or current sources where **IPORT** is specified). The format is used in conjunction with the `real_val1, real_val2` specification below.

- **int\_val1**

Defines the index of the harmonic according to fund1.

- **int\_val2**

Defines the index of the harmonic according to fund2.

- **int\_val3**

Defines the index of the harmonic according to fund3.

The group of 1 to 3 index values define a frequency. For example, for a source with 3 fundamental frequencies, (int\_val1, int\_val2, int\_val3) specifies the frequency:

---

```
F = int_val1*fund1+int_val2*fund2+int_val3*fund3
```

- **real\_val1, real\_val2**

Defines the complex value of the source for the corresponding frequency in the specified format (MA, RI or DB).

- **RPORT**

Resistance of the port value which defaults to  $50 \Omega$  whenever **RPORT\_TC1**, **RPORT\_TC2** or **IPORT** is specified. The possibility of having **NONOISE** on this resistance is allowed.

- **ZPORT\_FILE**

Specifies the Touchstone file name that contains the complex port impedance.

- **NONOISE**

Used in conjunction with **RPORT**. Specifies that no noise model will be used for this device when performing noise analysis. Therefore, the port resistance presents no noise contribution to the noise analysis.

- **RPORT\_TC1 & RPORT\_TC2**

Temperature coefficients of **RPORT**: they both default to 0. If **RPORT** happens to be 0 (in parametric simulations for instance), there will be no S extraction performed.

- **IPORT**

This is a strictly positive number that is unique and is used as the port number: this number is used for naming the outputs (for instance, `.PLOT AC S(1,2)`). An error message will be issued if two port instances have the same value for **IPORT**, or if an **IPORT** is missing (e.g. maximum **IPORT** number found in the netlist is 4, and there is no instance with **IPORT 3**).

- **CPORT**

Capacitor placed in parallel to **RPORT**. Defaults to 0, in which case it behaves like a zero voltage source (i.e. **CPORT** would have no effect).

- **LPORT**

Inductor placed in series with **RPORT**. Defaults to 0.

- **MODE=SINGLE | COMMON | DIFFERENTIAL**

Mixed-mode S parameter selection.

**SINGLE** specifies the port as single ended, it is dedicated to S parameter extraction. Default.

**COMMON** and **DIFFERENTIAL** specify that the port is not single ended. Such ports are split into two linked sources that are either common (same amplitude and same phase) or differential (same amplitude but opposite phases). If the S parameters are extracted a “non single ended” port is equally common and differential depending on which display is required. During simulation (DC, AC or TRAN) this port is either common or differential depending on the specified mode keyword.

- **NOISETEMP**

Corresponds to the temperature value which will be used for the calculation of the NOISE generated by **PORT**. The default is the temperature of the circuit, but should be set to 16.85 to comply with IEEE specifications. When this parameter is specified, it overrides the **INPUT\_TEMP** parameter in the **.SNF** command.

## Examples

```
i23 n2 n3 1.0e-4
```

Specifies a 0.1mA current flowing from node n2 towards node n3.

```
i41 n2 n4 dc 1.0e-3 ac 1.0e-6 45
```

Specifies a DC current of 1mA flowing from node n2 towards node n4 and an AC current of  $1.0 \times 10^{-6}$ A with a phase of 45 degrees.

```
i1 1 0 1 tc1 = 1
```

Specifies a current source of 1A between nodes 1 and 0. It has a first order temperature coefficient of 1. The second order temperature coefficient (TC2) will default to 0.

## Table noise input source

It is possible to provide Eldo some input noise source values depending on the frequency via the **TABLE** keyword. Eldo accepts the following:

```
Ixx P1 P2 NOISE TABLE [DEC|LOG|LIN] (f1,val1) (f2,val2)....
```

Eldo will assume zero current source or zero voltage source in any mode other than **NOISE** (i.e. for **DC**, **AC**, **TRAN**, etc.).

## Amplitude Modulation Function

**AM** (AMPLITUDE OFFSET FM FC TD)

Generates a time-dependent Amplitude Modulated signal. To be used in combination with independent voltage (**Vxx**) or current (**Ixx**) sources.

### Parameters

- **AMPLITUDE**  
Signal amplitude. Default is 0.
- **OFFSET**  
Offset of the signal. Default is 0.
- **FM**  
Modulation frequency. Default is 1/TSTOP.
- **FC**  
Carrier frequency. Default is 0.
- **TD**  
Delay before signal starts. Default is 0.

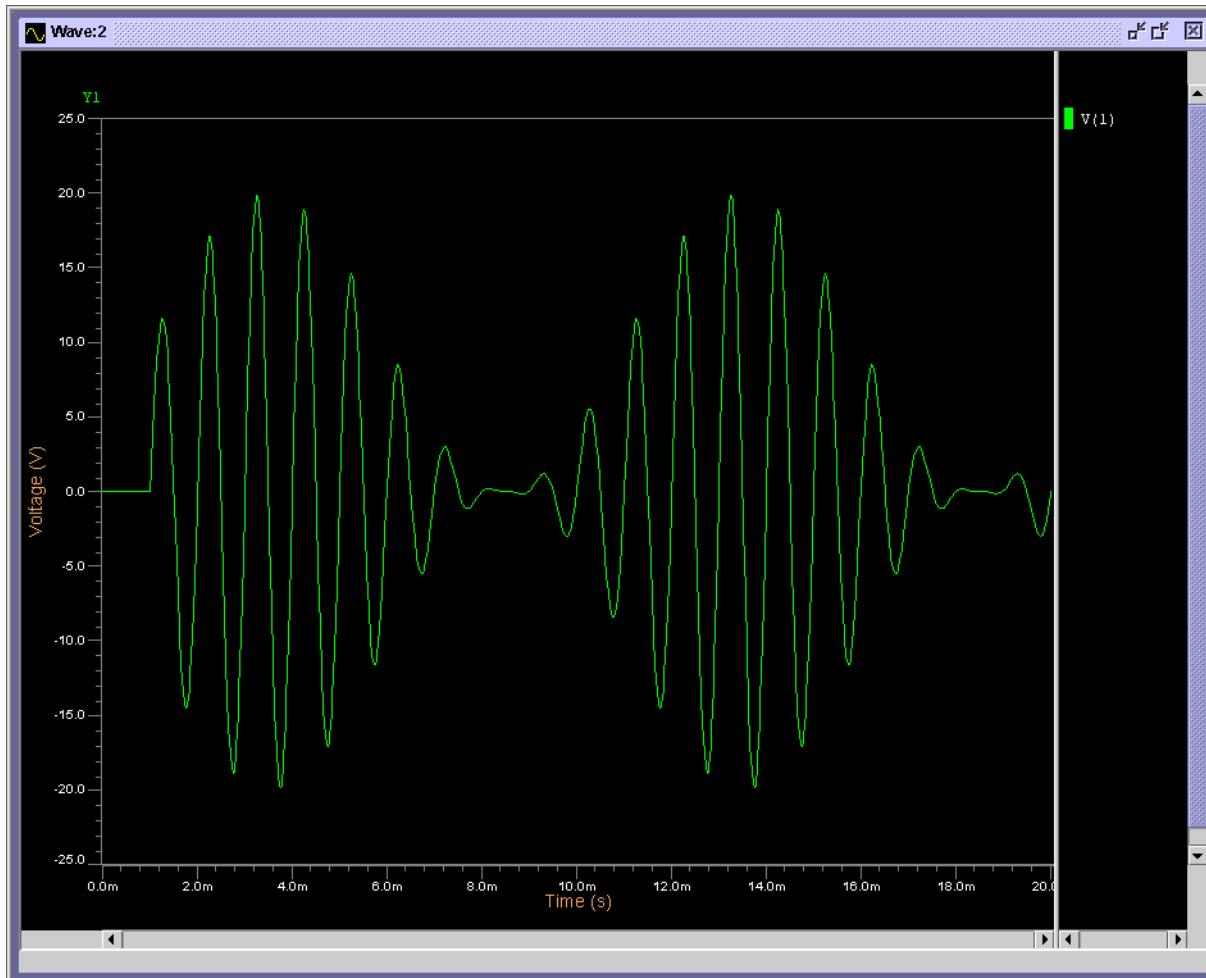
The waveform is described by:

$$V = \text{AMPLITUDE} \times (\text{OFFSET} + \sin(2\pi \times \text{FM} \times (\text{time} - \text{TD}))) \\ \times \sin(2\pi \times \text{FC} \times (\text{time} - \text{TD}))$$

### Example

```
vam 1 0 am (10 1 p100 1k 1m)
r1 1 0 1
.tran 1m 20m
.plot tran v(1)
.param p100 = 100
.end
```

The diagram below, has been generated by the above netlist. It shows amplitude modulation for a signal with amplitude 10 and an offset of 1. There is a signal delay of 1 millisecond and the modulation frequency and carrier frequency is set at 100 and 1000 respectively.

**Figure 5-1. AM Function Example**

## Exponential Function

**EXP** (V1 V2 [TD1 [TAU1 [TD2 [TAU2]]]])

Generates an exponentially damped pulse as defined below. To be used in combination with independent voltage (**Vxx**) or current (**Ixx**) sources.

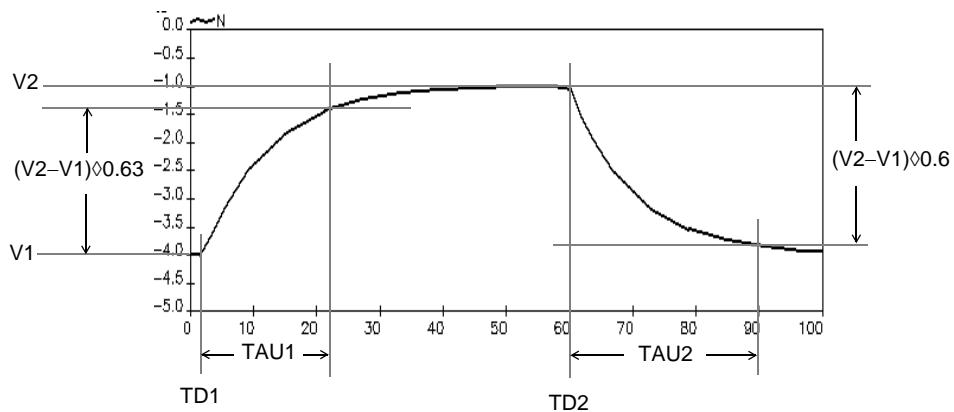
### Parameters

- V1  
Initial value in volts or amperes.
- V2  
Asymptotic, or target value of the pulse in volts or amperes.
- TD1  
Rise delay time in seconds. Default value is zero.
- TAU1  
Rise time constant in seconds. Default value is TPRINT.
- TD2  
Fall delay time in seconds. Default value is TD1+TPRINT.
- TAU2  
Fall time constant in seconds. Default value is TPRINT.

The waveform is described by the following relationships:

TIME	VOLTAGE
0 to <b>TD1</b>	V1
<b>TD1</b> to <b>TD2</b>	$V1 + (V2 - V1) \left( 1 - \exp \frac{-(time - TD1)}{TAU1} \right)$
<b>TD2</b> to <b>TSTOP</b>	$V1 + (V2 - V1) \left( 1 - \exp \frac{-(time - TD1)}{TAU1} \right) + (V1 - V2) \left( 1 - \exp \frac{-(time - TD2)}{TAU2} \right)$

**Figure 5-2. Exponential Function**



### Example

```
vin n3 0 exp (-4 -1 2n 10n 60n 10n)
```

Specifies a voltage source **vin** between node **n3** and ground. The time dependent voltage is described by:

TIME	VOLTAGE
0ns to 2ns	-4 V.
2ns to 60ns	Exponential rise from -4 V to -1 V with the rise time constant 10ns.
60ns – <b>TSTOP</b>	Exponential fall from -1 V to -4 V with the fall time constant 10ns. The <b>TSTOP</b> value is set in the <b>.TRAN</b> command.

## Noise Function

```
NOISE THN FLN ALPHA [FC N] [FMIN] [FMAX] [NBF]
```

Generates a noise source and is used in combination with independent voltage (**V<sub>xx</sub>**) or current (**I<sub>xx</sub>**) sources.

---

### Note



This source is effective only during **.NOISE** and **.NOISETRAN** analyses. A noise source has no effect during an AC or Transient simulation.

---

### Parameters

- **THN**  
Defines the white noise level in  $A^2/Hz$  or  $V^2/Hz$  respectively. Default is 0.
- **FLN**  
Defines the 1/f or Flicker Noise level at 1 Hz in  $A^2.Hz^{(1-\alpha)}$  or  $V^2.Hz^{(1-\alpha)}$  respectively. Default is 0.
- **ALPHA**  
Frequency exponent for 1/f noise. Default is 1.
- **FC**  
Cut-off frequency of the low pass noise filter.
- **N**  
Filter order.
- **FMIN**  
Lower limit of the noise frequency band.
- **FMAX**  
Upper limit of the noise frequency band.
- **NBF**  
Specifies the number of sinusoidal sources with randomly distributed amplitude and phase from which the noise source is composed. Default is 50.

---

### Note



**FMIN** and **FMAX** define the frequency band of the noise sources. This frequency range may sometimes not correspond to the noise frequency band at the output of the circuit. For instance, the band (**FMIN**, **FMAX**) does not correspond to the output noise frequency band in the case of filters or oscillators and mixers that exhibit frequency conversion.

---

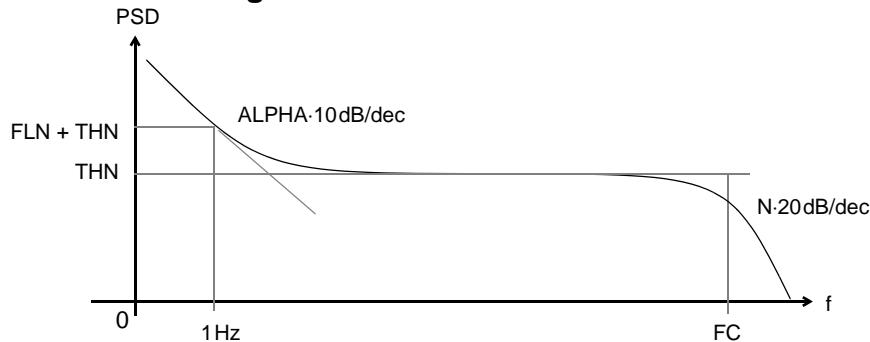
The Power Spectral Density (PSD) may be described as:

$$S_x = \left( THN + \frac{FLN}{f^{\text{ALPHA}}} \right) \left( \frac{1}{(1 + \Omega^2)^N} \right)$$

where:

$$\Omega = \frac{f}{FC}$$

**Figure 5-3. Noise Function**



### Examples

```
v1 n1 n2 noise 1e-17 0 1
```

Specifies a voltage noise source **v1** placed between nodes **n1** and **n2**. The White Noise level of the PSD has a value of  $1 \times 10^{-17} \text{V}^2/\text{Hz}$ , the Flicker Noise level is 0 and the frequency exponent for 1/f noise is 1.

```
i5 n5 0 noise 1e-20 1e-15 1.3 1meg 2
```

Specifies a current noise source **i5** placed between node **n5** and ground. The White Noise level of the PSD has a value of  $1 \times 10^{-20} \text{A}^2/\text{Hz}$ , the Flicker Noise level is  $1 \times 10^{-15}$ , the frequency exponent for 1/f noise is 1.3, the Cut-off frequency of the low pass filter is 1MHz and the order of the filter is 2.

## Pattern Function

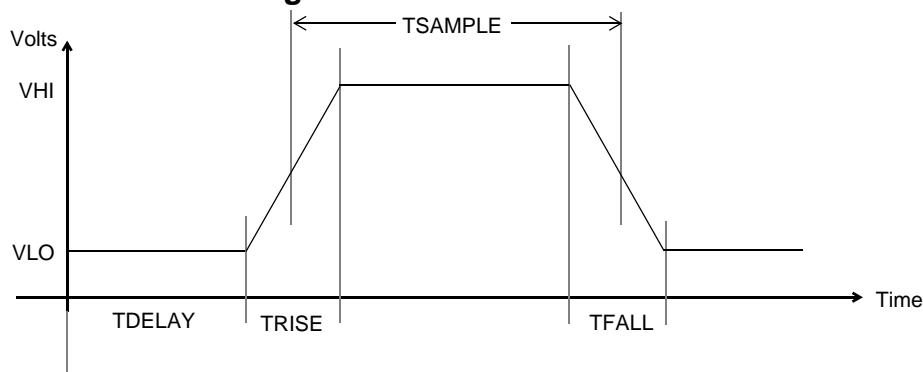
```
PATTERN VHI VLO TDELAY TRISE TFALL TSAMPLE BITS R
```

Generates a pulsating (digital like) source whose sequential values are defined as a distinct series of 1 and 0 values. To be used in combination with independent voltage (**Vxx**) or current (**Ixx**) sources. Periodic patterns can also be specified.

### Parameters

- **VHI**  
Voltage representing the 1 string value for **PATTERN** sources.
- **VLO**  
Voltage representing the 0 string value for **PATTERN** sources.
- **TDELAY**  
Delay before the **PATTERN** series is started. The value assigned during this time is the first string value of the series.
- **TRISE**  
Rise time between **PATTERN** values in seconds.
- **TFALL**  
Fall time between **PATTERN** values in seconds.
- **TSAMPLE**  
Time spent at 1 or 0 **PATTERN** value.
- **BITS**  
String of 1, 0 and Z values representing a **PATTERN** source. This can also be specified via a parameter using `$(PAT)` where `PAT` is the parameter as specified in the **.PARAM** statement. Z means high impedance, and this releases the applied signal, i.e. when the Z state is active, the signal will be disconnected from the node, and the node will be computed by Eldo as if it were a non-input signal.
- **R**  
Specifying `R` at the end of the pattern means that it is periodic.

**Figure 5-4. Pattern Function**



## Examples

```
v3 20 0 pattern 5 0 1u 2u 2u 8u 110010101 R
```

Specifying **R** at the end of the pattern means that it is periodic. Therefore, the equivalent pattern here is: 110010101110010101110010101110010101110.....

```
v1 1 2 pattern 5 0 10n 5n 10n 20n 001111000
```

Specifies a DC voltage source placed between nodes 1 and 2 defined by a string of 1 and 0 values. The voltage representing the 0 string value is 0V and the voltage level representing the 1 voltage level is 5V. Rise and fall times of the voltage source are 5 and 10ns respectively. The voltage source has the following characteristics:

1. Between time 0 and 10ns, the voltage source remains at 0V (**delay=10n**).
2. The voltage source remains at 0V for  $2 \times 20\text{ns}$  (**tbit=20n**).
3. The voltage source rises to a 5V level in 5ns (**trise=5n**).
4. The voltage source remains at 5V for  $4 \times 20\text{ns}$ .
5. The voltage level falls to 0V in 10ns (**tfall=10n**).
6. The voltage level remains at 0V for  $3 \times 20\text{ns}$ .

## Pulse Function

**PULSE (V0 V1 [TD [TR [TF [PW [PER]]]]])**

Generates a periodic pulse as described below. To be used in combination with independent voltage (**Vxx**) or current (**Ixx**) sources.

### Parameters

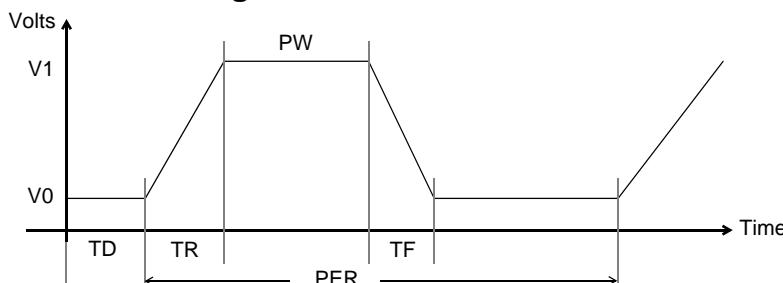
- **V0**  
Initial value of DC voltage or current.
- **V1**  
Pulse magnitude in volts or amperes.
- **TD**  
Delay time in seconds. Default value is zero.
- **TR**  
Rise time in seconds. Default value is **TPRINT**.
- **TF**  
Fall time in seconds. Default value is **TPRINT**.
- **PW**  
Pulse width in seconds. Default value is **TSTOP**.
- **PER**  
Pulse period in seconds. Default value is **TSTOP**.

#### Note



If the parameters **TR** and **TF** are both set to 0, they are assigned a value of **TPRINT**, the time interval used for the printing of results of the transient analysis, defined in the **.TRAN** command.

**Figure 5-5. Pulse Function**



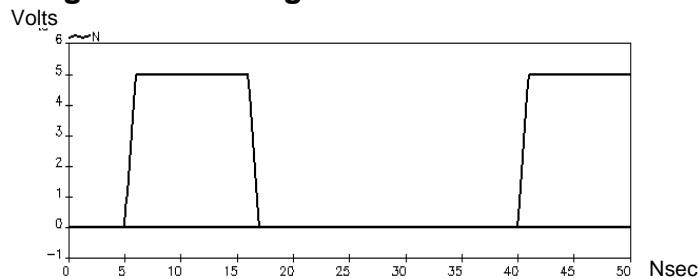
### Example

```
vp n12 n13 pulse(0 5 5n 1n 1n 10n 35n)
```

Specifies the voltage source **vp** placed between the node **n12** and **n13**. The voltage source has the following characteristics:

1. 0V from 0 to 5ns ( $\text{td}=5\text{n}$ ,  $\text{v0}=0$ ).
2. Rise from 0 to 5V in the time period 5 to 6ns ( $\text{tr}=1\text{n}$ ,  $\text{v1}=5$ ).
3. Remain at 5V from 6 to 16ns ( $\text{pw}=10\text{n}$ ).
4. Fall from 5 to 0V in the time period 16 to 17ns ( $\text{tf}=1\text{n}$ ).
5. 0V from 17 to 35ns ( $\text{per}=35\text{n}$ ).
6. Second cycle starting at 35ns.

**Figure 5-6. Voltage Source Characteristics**



## Piece Wise Linear Function

```
PWL (TN VN {TN VN} [TD=val] [R=val] [SHIFT=val] [R] [SCALE=val]
+ [STRETCH=val])
PWL (FILE=<pwl_file> [TD=val] [R=val] [SHIFT=val] [R] [SCALE=val]
+ [STRETCH=val])
PWL (FILE=<pwl_file> [COL=val] [ISTEP=val] [ISTART=val] [ISTOP=val]
+ [TD=val] [R=val] [SHIFT=val] [R] [SCALE=val] [STRETCH=val])
```

Generates a Piece Wise Linear function using straight lines between specified voltage points until **TN** is reached. To be used in combination with independent voltage (**Vxx**) or current (**Ixx**) sources.

The second two syntaxes above shows how Eldo can read PWL corner points from a file, with the last syntax supporting multi-column files.

**SHIFT=val** and **R=val** can be used together in the same PWL statement. The keyword **R** alone (i.e. without a value specified) can be used in conjunction with **SHIFT=val**, but must be placed at the end of the statement.

The **TN VN** pairs can be separated by spaces or commas.

### Parameters

- **VN**  
Value of the source at time **Ti** in volts or amperes. The source value at intermediate times is provided by linear interpolation. A high-impedance can be specified with the **Z** keyword.
- **TN**  
Time in seconds, at which **Vi** is supplied, where  $T_i < T_{i+1}$ .
- **TD=VAL**  
Delay time in seconds. Default value is zero.

---

#### Note

 TD and SHIFT are synonymous.

---



---

#### Note

 If the rise or fall times are 0, they are assigned a value of **TPRINT**, the time interval used for the printing of results of the transient analysis, defined in the **.TRAN** command.

---

- **R**  
Specifies a periodically repetitive signal of period (**TN-T1**: the difference between the last and first specified time parameters) in which case the values **V1** and **VN** must be the same.

- **R=VAL**

This means that the period will be equal to the last time value specified in the PWL statement minus the **R** value.

- **SHIFT=VAL**

This acts as if the **SHIFT** value was added to all time values specified in the PWL card.

- **SCALE=VAL**

Element scale factor. The value of the element is multiplied by **SCALE**, which defaults to 1.

- **STRETCH=VAL**

Time scale factor applied to the waveform. Default value is 1.

- **FILE=pwl\_file**

Allows Eldo to read PWL corner points from the specified file *pwl\_file*. This is a text file that supplies the time-current (tn in) or time-voltage (tn vn) pairs. Engineering units (for example 9ns) are allowed. The time-value pairs are separated by spaces, commas or newline characters. The file can have multiple lines and can have any number of point pairs per line. Continuation signs “+” are not needed.

- **COL=VAL**

For use with multi-column file specification. Selects the column number of the files. The time values column is the column 0. So a value less than 1 is not allowed for col.

- **ISTART=VAL**

For use with multi-column file specification. Selects the starting line of data. First data line is 1.

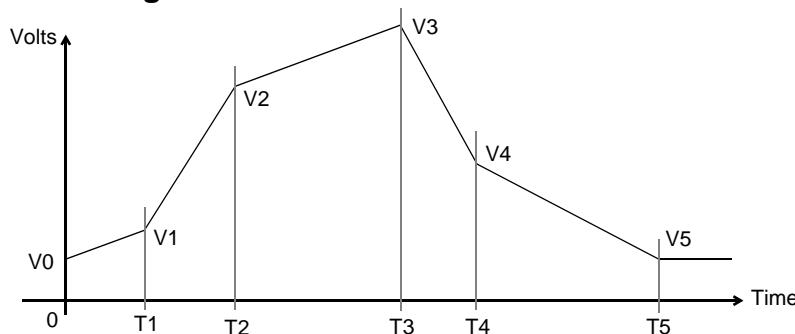
- **ISTOP=VAL**

For use with multi-column file specification. Selects the stopping line of data.

- **ISTEP=VAL**

For use with multi-column file specification. Selects the data interval. A value of nb means Eldo peaks data at each nb lines.

**Figure 5-7. Piece Wise Linear Function**

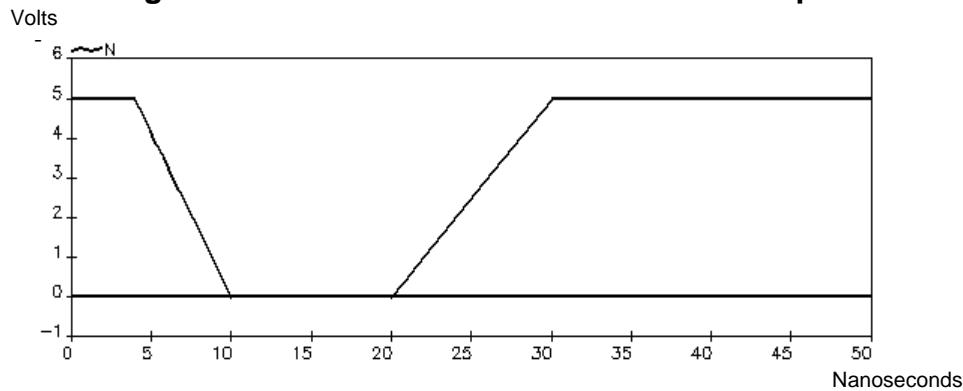


## Examples

```
v1 n3 n4 pwl (4n 5 10n 0 20n 0 30n 5)
```

The voltage source **v1** placed between node **n3** and **n4** specifies a signal which is defined by linear interpolation between the values enclosed in the parentheses.

**Figure 5-8. Piece Wise Linear Function Example 1**

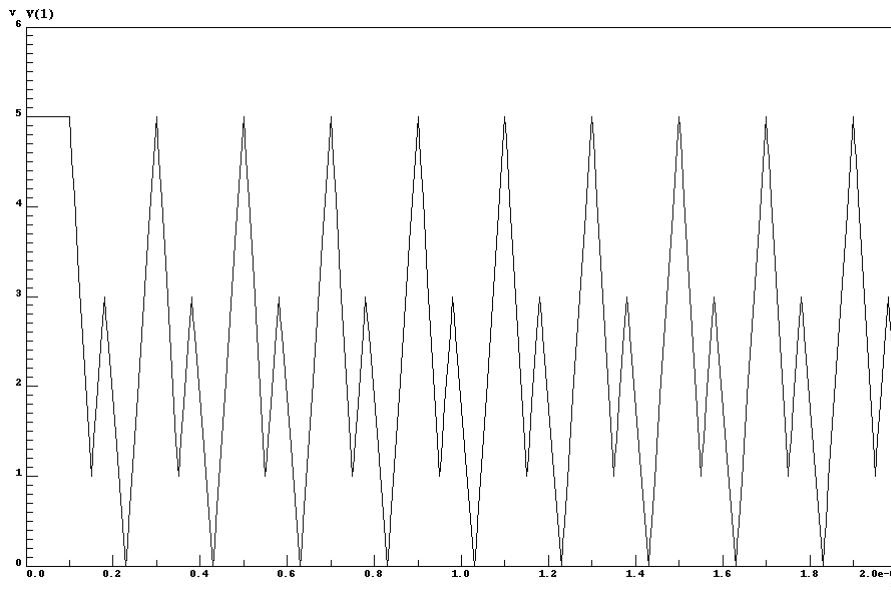


A periodically repetitive signal can be specified as shown in the following example:

```
v1 1 0 pwl (100n 5 150n 1 180n 3 230n 0 300n 5 R)
r1 1 0 1
.tran 1u 2u
.plot tran v(1)
.end
```

The voltage source **v1** between nodes **1** and **0** will be interpolated between the specified values of **VN** at time **TN** and plotted until the time **2μs** is reached. The repetitive part of the waveform starts at **T1=100ns**, and in this case, has a period of **TN-T1=200ns**. This is illustrated in [Figure 5-9](#).

**Figure 5-9. Piece Wise Linear Function Example 2**



A high-impedance can be specified in PWL statements as shown in the following example:

```
v1 1 0 PWL (0 0 10n 15 15n Z 25n Z 30n 0)
```

At 15ns, **v1** is disconnected which means that the rest of the circuit will impose a value on node 1, that is, node 1 becomes a node to be solved as any other node in the circuit. At 30ns, **v1** is connected back.

The example below shows the usage of parameters **SCALE** and **STRETCH**:

```
v1 1 0 pwl ( 0 0 10n 10 SCALE=2)
*v1 1 0 pwl ( 0 0 10n 10 STRETCH=2)
r1 1 0 1
.tran 1n 10n
.plot tran v(1)
.end
```

When the voltage source with the PWL function is specified with the **SCALE** parameter the element is multiplied by a scale factor of 2 along the y-axis. When **STRETCH** is specified the element is multiplied by a time scale factor along the x-axis.

The example below shows two ways of specifying the **FILE** parameter. The name can be specified directly or via a parameter value.

```
v1 1 0 pwl file="stim1.txt" R
.param STIMFILE='stim.txt'
v2 2 0 pwl file=$(STIMFILE) R
```

The example below shows a PWL source with multi-column file specification.

```
*test with multicolm files
*
v1 1 0 dc 0 pwl(file="stim4.txt" col=1 istep=1)
```

```
v2 2 0 dc 0 pwl(file="stim4.txt" col=2 istep=2)
r1 1 0 1
r2 2 0 1
.tran 1u 6u
.plot tran v(1) v(2)
.end
```

Contents of *stim4.txt* file:

```
#time v1 v2
0 1.0 9.0
500e-9 2.0 2.0
1e-6 3.0 13.0

2e-6 4.0 4.0
3e-6 5.0 15.0
4e-6 6.0 6.0
5e-6 7.0 7.0

v1 1 0 pwl file="stim1.txt" R

.param STIMFILE=' "stim.txt" '
v2 2 0 pwl file=$(STIMFILE) R
```

The file *stim4.txt* is a multi-column set of data. This file is structured in a such a way that each line consists of a time value, TN, and source values, VN1, VN2. This example is a three column file. The columns are internally labeled beginning 0 (0, 1, 2).

For the first PWL source in the main netlist:

```
v1 1 0 dc 0 pwl(file="stim4.txt" col=1 istep=1)
```

Eldo will parse the source value in column 1 with a step of 1. It is equivalent to writing:

```
v1 1 0 dc 0 pwl(0 1.0 500ns 2.0 1us 3.0 2us 4 3us 5 4us 6 5us 7)
```

For the second PWL source in the main netlist:

```
v2 2 0 dc 0 pwl(file="stim4.txt" col=2 istep=2)
```

Eldo will parse the source value in column 2 with a step of 2. It is equivalent to writing:

```
v2 2 0 dc 0 pwl(0 9.0 1us 13.0 3us 15 5us 7)
```

When the voltage source with the PWL function is specified with the **SCALE** parameter the element is multiplied by a scale factor of 2 along the y-axis. When **STRETCH** is specified the element is multiplied by a time scale factor along the x-axis.

## Single Frequency FM Function

**SFFM** (SO SA [FC [MDI [FS]])

Generates a single frequency FM modulated signal. To be used in combination with independent voltage (**Vxx**) or current (**Ixx**) sources.

### Parameters

- SO  
Offset voltage in volts or current in amperes.
- SA  
Magnitude of signal in volts or amperes.
- FC  
Carrier frequency in Hertz. Default value is 1/TSTOP.
- MDI  
Modulation index. Default value is zero.
- FS  
Signal frequency in Hertz. Default value is 1/TSTOP.

The shape of the waveform is described by:

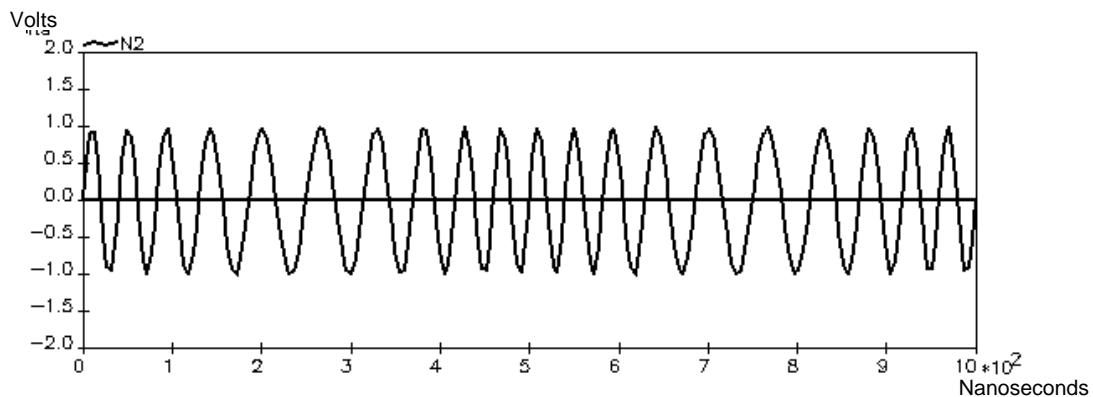
$$\text{value} = \text{SO} + \text{SA} \times \sin((2\pi \times \text{FC} \times \text{time}) + \text{MDI} \times \sin(2\pi \times \text{FS} \times \text{time}))$$

### Example

```
v1 n12 0 sffm (0 1 20meg 2.5 2meg)
```

Specifies a voltage source **v1** placed between node **n12** and ground. The voltage has 0 offset and an amplitude of 1V. The carrier frequency of 20MHz is modulated with the signal frequency of 2MHz, the modulation index being 2.5.

**Figure 5-10. Single Frequency FM Function**



## Sine Function

**SIN (VO VA [FR [TD [THETA [PHASE]]]])**

Generates a sinusoidal or a damped sine wave. To be used in combination with independent voltage (**Vxx**) or current (**Ixx**) sources.

### Parameters

- **VO**  
Offset voltage in volts or offset current in amperes.
- **VA**  
Sine wave nominal starting amplitude in volts or amperes.
- **FR**  
Frequency in Hertz. Default value is 1/TSTOP.
- **TD**  
Delay time in seconds. Default value is zero.
- **THETA**  
Damping factor in 1/sec. Default value is zero.
- **PHASE**  
Phase delay in degrees. Default value is zero.

The waveform is described by:

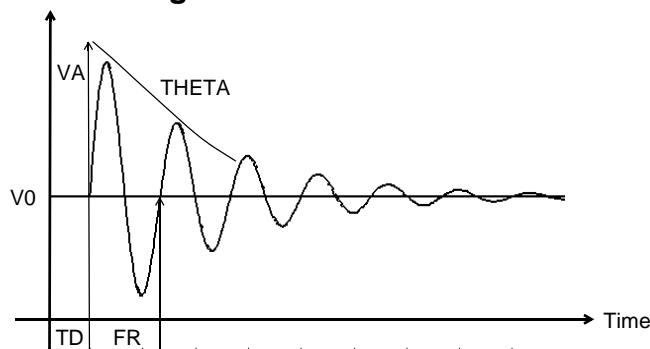
for  $t < TD$

$$V = VO + VA \times \sin\left(2\pi \times \frac{PHASE}{360}\right)$$

for  $t \geq TD$

$$V = VO + VA \times \exp(-(t - TD) \times THETA) \times \sin\left(2\pi \times \left(FR \cdot (t - TD) + \frac{PHASE}{360}\right)\right)$$

**Figure 5-11. Sine Function**

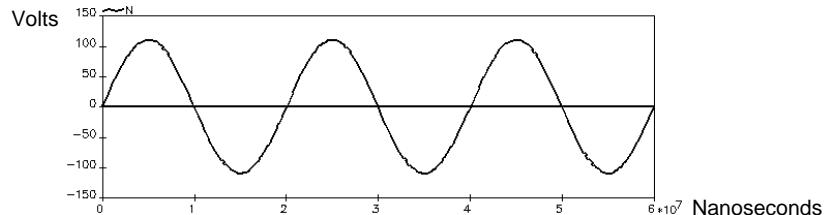


## Examples

```
vsin n2 n3 sin (0 110 50 0 0)
```

Specifies the voltage source `vsin` between nodes `n2` and `n3` of amplitude 110V and frequency 50Hz.

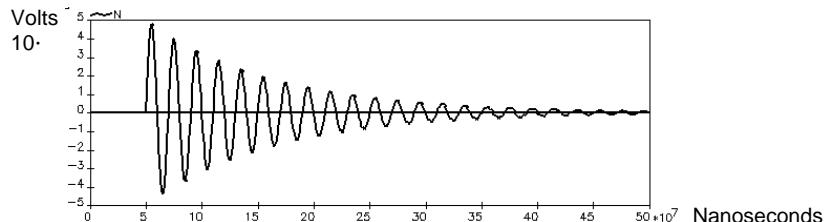
**Figure 5-12. 1st Sine Function Example**



```
vsin n4 n9 sin(0 50 50 .05 9)
```

Specifies the voltage source `vsin` placed between nodes `n4` and `n9` with an amplitude of 50V and frequency of 50Hz. It has a delay of 0.05s with a decay factor of 9.

**Figure 5-13. 2nd Sine Function Example**



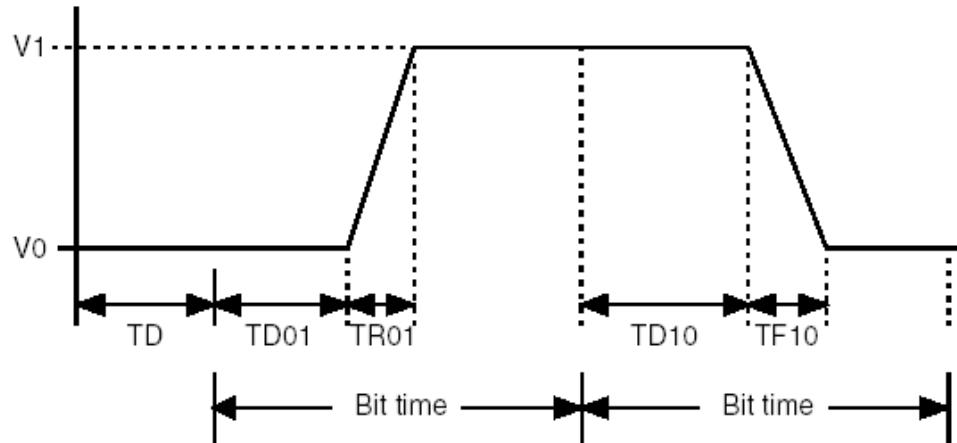
## Trapezoidal Pulse With Bit Pattern Function

**PBIT** V0 V1 TD TD01 TR01 TD10 TF10 BITTIME {PATTERN} [R]

Generates a trapezoidal pulse with bit pattern source. This describes a bit pattern as a series of trapezoidal pulses with linear rise and fall waveforms. To be used in combination with independent voltage (**Vxx**) or current (**Ixx**) sources.

### Parameters

- **V0**  
Voltage (or current) representing the zero bit value.
- **V1**  
Voltage (or current) representing the one bit value.
- **TD**  
Delay before the series is started. The value assigned during this time is the first string value of the series.
- **TD01**  
Time delay for 0-1 bit transition in seconds. Default is 0.
- **TR01**  
Rise time for 0-1 bit transition in seconds. Default is TSTEP.
- **TD10**  
Time delay for 1-0 bit transition in seconds. Default is 0.
- **TF10**  
Fall time for 1-0 bit transition in seconds. Default is TSTEP.
- **BITTIME**  
Bit time for complete transition in seconds.
- **PATTERN**  
Pattern depicting value at end of each bit time.
- **R**  
Specifying **R** at the end of the pattern means that it is periodic.

**Figure 5-14. Trapezoidal Pulse with Bit Pattern****Examples**

```
v1 1 0 pbit 1 5 0n 1n 0.5n 0n 1n 5n 101011101000011 R
```

Specifies the voltage source,  $V_1$  as a trapezoidal voltage pulse source between nodes 1 and 0. The source has the following characteristics:

The zero bit value is 0V. The one bit value is 5V.

There is no delay before the series is started.

Time delay for 0-1 bit transition is 1ns.

Rise time for 0-1 bit transition is 0.5ns.

Time delay for 1-0 bit transition is 0ns.

Fall time for 1-0 bit transition is 1ns.

Bit time for complete transition is 5ns.

The bit pattern has fifteen bits which is repeated until the simulation run ends.

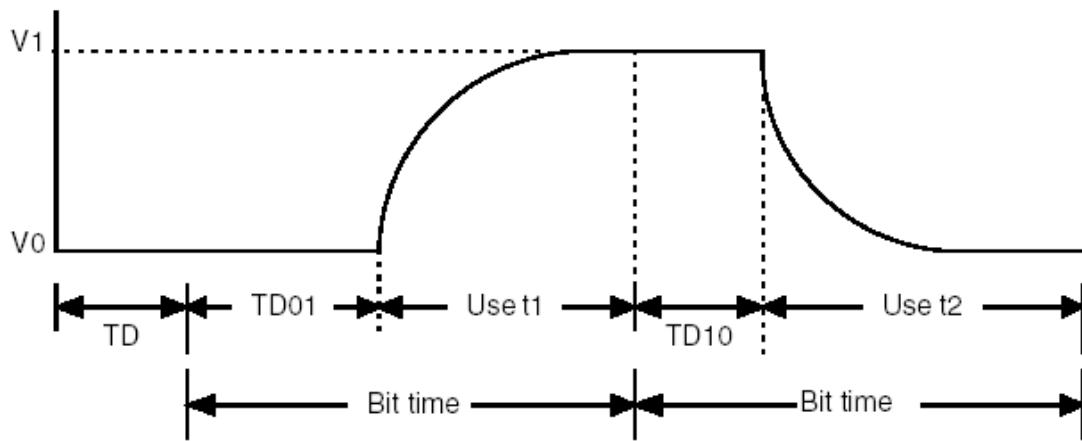
## Exponential Pulse With Bit Pattern Function

**E<sub>BIT</sub>** V0 V1 TD TD01 TAU01 TD10 TAU10 BITTIME {PATTERN} [R]

Generates an exponential pulse with bit pattern source. This describes a bit pattern as a series of exponential pulses with exponential rise and fall waveforms. To be used in combination with independent voltage (**Vxx**) or current (**Ixx**) sources.

### Parameters

- **V0**  
Voltage (or current) representing the zero bit value.
- **V1**  
Voltage (or current) representing the one bit value.
- **TD**  
Delay before the series is started. The value assigned during this time is the first string value of the series.
- **TD01**  
Time delay for 0-1 bit transition in seconds. Default is 0.
- **TAU01**  
Rise time constant for 0-1 bit transition in seconds. Default is TSTEP.
- **TD10**  
Time delay for 1-0 bit transition in seconds. Default is 0.
- **TAU10**  
Fall time constant for 1-0 bit transition in seconds. Default is TSTEP.
- **BITTIME**  
Bit time for complete transition in seconds.
- **PATTERN**  
Pattern depicting value at end of each bit time.
- **R**  
Specifying **R** at the end of the pattern means that it is periodic.

**Figure 5-15. Exponential Pulse with Bit Pattern****Examples**

```
v1 1 0 ebit 1 5 0n 1n 0.5n 0n 1n 5n 101011101000011 R
```

Specifies the voltage source, V1 as an exponential voltage pulse source between nodes 1 and 0. The source has the following characteristics:

- The zero bit value is 0V. The one bit value is 5V.
- There is no delay before the series is started.
- Time delay for 0-1 bit transition is 1ns.
- Rise time constant for 0-1 bit transition is 0.5ns.
- Time delay for 1-0 bit transition is 0ns.
- Fall time constant for 1-0 bit transition is 1ns.
- Bit time for complete transition is 5ns.

The bit pattern has fifteen bits which is repeated until the simulation run ends.

## Voltage Controlled Voltage Source

### Linear (Voltage Gain Block)

```
Exx NP NN [VCVS] NCP NCN VAL [MIN=VAL] [MAX=VAL]
+ [TC1=VAL] [TC2=VAL] [SCALE=VAL] [ABS=VAL]
Exx NP NN [VCVS] NCP NCN VAL0 {VALn} [MIN=VAL] [MAX=VAL]
+ [TC1=VAL] [TC2=VAL] [SCALE=VAL] [ABS=VAL]
```

### Polynomial

```
Exx NP NN [VCVS] POLY(ND) PCP PCN {PCP PCN} PN {PN}
+ [MIN=VAL] [MAX=VAL] [TC1=VAL] [TC2=VAL] [SCALE=VAL] [ABS=VAL]
```

### Piece Wise Linear

```
Exx NP NN PWL(1) NCP NCN PWL_LIST [DELTA=val]
+ [TC1=VAL] [TC2=VAL] [SCALE=VAL] [ABS=VAL]
```

### Multi-Input Gate

```
Exx NP NN NAND(ND) | AND(ND) | OR(ND) | NOR(ND) PCP PCN {PCP PCN}
+ PWL_LIST [TC1=VAL] [TC2=VAL] [SCALE=VAL] [ABS=VAL]
```

### Delay Element

```
Exx NP NN DELAY NCP NCN [TD=val] [ABS=VAL]
```

### Arithmetic Expression

```
Exx NP NN VALUE={EXPR} [MIN=VAL] [MAX=VAL]
+ [TC1=VAL] [TC2=VAL] [SCALE=VAL] [ABS=VAL]
```

### Tabular

```
Exx NP NN [MIN=VAL] [MAX=VAL] [TC1=VAL] [TC2=VAL] [SCALE=VAL]
+ TABLE EXPR=(XN YN) {(XN YN)} [ABS=VAL]
```

### Integral/Derivative

```
Exx NP NN INTEGRATION|DERIVATION NCP NCN VAL
+ [MIN=VAL] [MAX=VAL] [TC1=VAL] [TC2=VAL] [SCALE=VAL] [ABS=VAL]
```

### S-domain

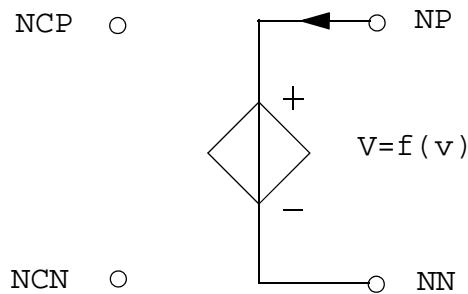
```
Exx NP NN FNS NCP NCN n0 n1 ... nm, p0 p1 ... pn
Exx NP NN PZ NCP NCN a zrl zil ... zrm zim, b prl pil ... prn pin
```

### Frequency Dependent

```
Exx NP NN FREQ NCP NCN f0 a0 ph0 f1 a1 ph1... fn an phn
+ [RESTORE_CAUSALITY=val]
```

### Ideal Transformer

```
Exx NP NN TRANS[FORMER] NCP NCN VAL [MIN=VAL] [MAX=VAL]
+ [TC1=VAL] [TC2=VAL] [SCALE=VAL] [ABS=VAL]
```

**Note**

The current flows from the positive node NP, through the current source, to the negative node NN.

**Parameters**

- **xx**  
Voltage controlled voltage source name.
- **NP**  
Name of the positive node.
- **NN**  
Name of the negative node.
- **NCP**  
Name of the positive controlling node.
- **NCN**  
Name of the negative controlling node.
- **VAL**  
Voltage gain.

The above parameters can be related by:

$$V(NP) - V(NN) = VAL \times (V(NCP) - V(NCN))$$

- **VALn**  
Coefficients of the polynomial function:  
 VAL0 is a voltage shift,  
 VAL1 is the 1st order gain,  
 VAL2 is the 2nd order gain, etc.

The above parameters can be related by:

$$V(NP, NN) = VAL0 + VAL1 \times V(NCP, NCN) + VAL2 \times V(NCP, NCN)^2 \\ + VAL3 \times V(NCP, NCN)^3 + \dots$$

- **MIN=VAL**

Minimum output voltage value. Unit Volts.

- **MAX=VAL**

Maximum output voltage value. Unit Volts.

- **TC1, TC2**

First and Second order temperature coefficients. Default values are zero. The output value is updated with temperature according to the formula:

$$VAL(T) = VAL(TNOM) \times (1 + TC1(T - TNOM) + TC2(T - TNOM)^2)$$

- **SCALE=VAL**

Element scale factor. The value of the element is multiplied by **SCALE**, which defaults to 1.

- **ABS=val**

Can be set to 1 or 0 (default is 0). If **ABS=1**, the output value is the absolute value of the signal.

- **POLY**

Keyword indicating the source has a non-linear polynomial description.

- **ND**

Order of the polynomial when **POLY** is specified. Number of Inputs when **NAND, NOR, AND, OR** is specified. **ND** must be greater than or equal to 1. If **ND** is not specified it will default to 1.

- **PCP**

Name of the positive controlling node producing the voltage difference for the function arguments of the polynomial. Number is equivalent to the order of the polynomial.

- **PCN**

Name of the negative controlling node producing the voltage difference for the function arguments of the polynomial. Number is equivalent to the order of the polynomial.

- **PN**

Coefficients of the polynomial.

- **NAND, NOR, AND, OR**

One of these can be specified in place of the existing keyword **POLY**. If **NAND** or **AND** are used, the lower valued command will be used to compute the output. In the case of **NOR** or **OR**, the higher valued command will be used. In such cases of **NAND/AND/OR/NOR** types, Eldo expects a list of couple values (x,y) specified as a **PWL\_LIST**, the output will be the interpolated value y. Commas used as delimiters are optional.

- **PWL\_LIST**

Consists of a list of couple values (x,y); interpolation will be made to extract the value, depending on  $(v(ncp)-v(ncn))$ .

- **PWL(1)**

When specified, a simple interpolation will be performed to evaluate the output. A list of couple values (x,y) specified as a **PWL\_LIST** is expected.

- **DELTA=VAL**

This parameter must be in the range of 0 to 0.5, and is used to smooth out the output. The smooth-out occurs in the portion of the interval determined by the **DELTA** value. If **DELTA** is 0, smooth-out does not occur and strict linear interpolation is performed to compute the output. If **DELTA** is set to 0.5 the smooth-out occurs over the whole interval.

- **DELAY**

The **E** element is controlled by the voltage, therefore, the item specified after the **DELAY** operator must be the values of the positive and negative control nodes. The output is shifted by a delay value of **TD**.

- **TD=VAL**

Delay value. The default value of **TD** is 0 if left unspecified.

- **VALUE**

Keyword indicating that the source has a functional description. A set of expressions is specified in **EXPR**.

- **TABLE**

Keyword indicating that the source has a tabular description. The table itself contains pairs of values. **EXPR** is evaluated, and its value is used to look up an entry in the table. Linear interpolation is made between entries.

- **EXPR**

A set of expressions, used to set the source value or an entry look up for a tabular description of the source.




---

Expression formats are described in the [Eldo Control Language](#) chapter.

---

Controlled source expressions may also contain voltages, currents or time. Voltages may be the voltage through a node, e.g. **v(6)**, or the voltage across two nodes **v(5, 6)**. Currents must be the current through a voltage source, such as **i(v1)**.

**EXPR** is also able to make use of the operators **DDT** and **IDT**. **DDT** stands for derivative, and **IDT** for integral. **DDT** and **IDT** operators utilize the integration scheme which is used by Eldo for computing the derivative/integral.




---

Please refer to “[Arithmetic Functions](#)” on page 3-7.

---

- **XN, YN**

Input and corresponding output source values for tabular source definitions.

- **INTEGRATION | DERIVATION**

Specifies that the voltage drop across **NP** and **NN** should be equal to the *integral* (or *derivative*) of  $v(NCP) - v(NCN)$  multiplied by **VAL**.

- **FNS**

Specifies a s-domain function for which the transfer function is:

$$H = \frac{n_0 + n_1 \cdot s + \dots + n_m \cdot s^m}{p_0 + p_1 \cdot s + \dots + p_n \cdot s^n}$$

If **P2** is 0 and **NCN** is 0, this is equivalent to the existing **FNS** device:

**FNS NCP NCN n0 n1 ... nm, p0 p1 ... pn**

- $n_0 \ n_1 \ \dots \ n_m, \ p_0 \ p_1 \ \dots \ p_n$

Polynomial coefficients for the transfer function of **FNS**.

- **PZ**

When the poles and zeroes are known, the transfer function is:

$$H = \frac{a \cdot \prod((s - (z_{ri} + j \cdot 2 \cdot \pi \cdot z_{ii})) \cdot (s - (z_{ri} - 2 \cdot \pi \cdot z_{ii})))}{b \cdot \prod((s - (p_{ri} + j \cdot 2 \cdot \pi \cdot p_{ii})) \cdot (s - (p_{ri} - 2 \cdot \pi \cdot p_{ii})))}$$

For the numerator, **i** is from 1 to m. For the denominator, **i** is from 1 to n.

**Note**



The conjugate appears only if the imaginary part is not 0.

- **a, b**

Coefficients for the transfer function of **PZ**.

- **pr, pi**

Poles of the transfer function. **pr** is the real part, **pi** the imaginary part.

- **zr, zi**

Zeroes of the transfer function. **zr** is the real part, **zi** the imaginary part.

- **FREQ**

Specifies a frequency domain description. Enables a frequency dependent voltage controlled source to be simulated for AC, Transient, and MODSST analyses.

- **f0 a0 ph0 f1 a1 ph1... fn an phn**

Coefficients for the frequency domain. **fi**, **ai** and **phi** are the frequency, the amplitude in dB, and the phase in degrees respectively. At each frequency point, Eldo will

evaluate the amplitude in dB by making a log interpolation, and will evaluate the phase by making a linear interpolation.

- **RESTORE\_CAU<sub>S</sub>LITY=val**

The device characteristics are defined by an equation, when set to 1 the causality of the equation will be restored (if required) by building the corresponding imaginary part, this will produce results with an even higher level of accuracy. When set to 0 (default) the causality of the equation will not be restored.

- **TRANS [ FORMER ]**

Keyword specifying that an ideal transformer is used. It differs from a linear VCVS source by the equations used:

$$V(NP) - V(NN) = \frac{V(NCP) - V(NCN)}{VAL}$$

## Examples

```
e23 n2 n3 14 0 2.0
```

Specifies that the voltage applied between nodes n2 and n3 is twice the potential difference between node 14 and ground.

```
e1 2 0 poly (2) 3 0 5 0 0 1 1 2 4.5
```

Specifies a 2nd order non-linear voltage controlled voltage source e1 between node 2 and ground. The two controlling voltages contributing to the voltage difference for the function arguments  $f_a$  and  $f_b$  occur between node 3 (NCP1) and ground (NCN1), and node 5 (NCP2) and ground (NCN2). Polynomial coefficients are P0=0, P1=1, P2=1, P3=2 and P4=4.5. The resultant non-linear voltage function has the following form:

$$f_v = f_a + f_b + 2f_a^2 + 4.5f_a f_b$$

where:  $f_a$  is the voltage difference between the controlling nodes NCP1 and NCN1  
 $f_b$  is the voltage difference between the controlling nodes NCP2 and NCN2.

```
e1 1 2 value = {v(3,4)*i(v5)}
```

Specifies that the voltage applied between nodes 1 and 2 is the instantaneous power calculated by multiplying the voltage across nodes 3 and 4 and the current through V5.

```
V1 A 0 1
r1 A 0 1
V2 B 0 5
r2 B 0 5
E3 3 0 NAND(2) A 0 B 0 (-10,-30) (10,30)
r3 3 0 1
```

This example shows the use of the **NAND** keyword. In this case, the command of lower value is used, i.e. it is **v(A,0)** which will be used, the voltage on node 3 will be 3V (interpolation for x in [-10,10] and y in [-30,30]).

```
V1 A 0 1
r1 A 0 1
E3 3 0 PWL(1) A 0 (-10,-30) (10,30)
r3 3 0 1
.DC V1 1 10 1
```

**V(3)** will vary from 3 to 30 when **V(1)** varies from 1 to 10.

The next example specifies a voltage controlled voltage source in the frequency domain. It is between nodes 2 and 0 and the voltage across those two pins is equal to the GAIN(f) multiplied by **V(1,0)**.

```
*SEARCH MAX must be 3.7276 and min must be 2.2758
E2 2 0 FREQ 1 0
+1k 20 0
+1e10 5 0
v1 1 0 2 ac 1
r1 1 2 1k
r2 2 0 1k
.ac dec 10 1e7 1e9
```

## Current Controlled Current Source

### Linear (Current Gain Block)

```
Fxx NP NN [CCCS] VN VAL [MIN=VAL] [MAX=VAL]
+ [TC1=VAL] [TC2=VAL] [SCALE=VAL] [ABS=VAL]
```

### Polynomial

```
Fxx NP NN [CCCS] POLY(ND) VN {VN} PN {PN}
+ [MIN=VAL] [MAX=VAL] [TC1=VAL] [TC2=VAL] [SCALE=VAL]
+ [ABS=VAL]
```

### Piece Wise Linear

```
Fxx NP NN PWL(1) VN PWL_LIST [DELTA=val]
+ [TC1=VAL] [TC2=VAL] [SCALE=VAL] [ABS=VAL]
```

### Multi-Input Gate

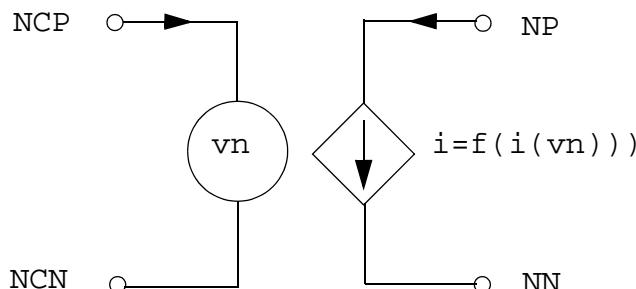
```
Fxx NP NN NAND(ND) | AND(ND) | OR(ND) | NOR(ND) VN {VN}
+ PWL_LIST [TC1=VAL] [TC2=VAL] [SCALE=VAL] [ABS=VAL]
```

### Delay Element

```
Fxx NP NN DELAY VN [TD=val] [ABS=VAL]
```

### Integral/Derivative

```
Fxx NP NN INTEGRATION|DERIVATION VN VAL [MIN=VAL] [MAX=VAL]
+ [TC1=VAL] [TC2=VAL] [SCALE=VAL] [ABS=VAL]
```



#### Note



The current flows from the positive node NP, through the current source, to the negative node NN. NCP and NCN are the positive and negative controlling nodes for source vn.

### Parameters

- xx

Current controlled current source name.

- **NP**  
Name of the positive node.
- **NN**  
Name of the negative node.
- **VN**  
Name of the voltage source through which the controlling current flows. The direction of positive controlling current flow is from the positive node, through the source, to the negative node of **VN**. When **POLY** is specified, one name must be added for each dimension of the polynomial. If **POLY** is not specified, it is assumed that there is only one dimension for which a name should be added.
- **VAL**  
Current gain.

The above parameters can be related by:

$$I(Fxx) = I(VNAM) \times VAL$$

- **MIN=VAL**  
Minimum output current value. Unit Amps.
- **MAX=VAL**  
Maximum output current value. Unit Amps.
- **TC1, TC2**  
First and Second order temperature coefficients. Default values are zero. The output value is updated with temperature according to the formula:

$$VAL(T) = VAL(TNOM) \times (1 + TC1(T - TNOM) + TC2(T - TNOM)^2)$$

- **SCALE=VAL**  
Element scale factor. The value of the element is multiplied by **SCALE**, which defaults to 1.
- **ABS=val**  
Can be set to 1 or 0 (default is 0). If **ABS=1**, the output value is the absolute value of the signal.
- **POLY**  
Keyword indicating the source has a non-linear polynomial description.
- **ND**  
Order of the polynomial when **POLY** is specified. Number of Inputs when **NAND, NOR, AND, OR** is specified. **ND** must be greater than or equal to 1. If **ND** is not specified it will default to 1.

- **PN**  
Coefficients of the polynomial.
- **NAND, NOR, AND, OR**  
One of these can be specified in place of the existing keyword **POLY**. If **NAND** or **AND** are used, the lower valued command will be used to compute the output. In the case of **NOR** or **OR**, the higher valued command will be used. In such cases of **NAND/AND/OR/NOR** types, Eldo expects a list of device names (x,y) specified as a **PWL\_LIST**, the output will be the interpolated value y. Commas used as delimiters are optional.
- **PWL\_LIST**  
Consists of a list of couple values (x,y); interpolation will be made to extract the value, depending on  $(v(ncp)-v(ncn))$ .
- **PWL(1)**  
When specified, a simple interpolation will be performed to evaluate the output. A list of couple values (x,y) specified as a **PWL\_LIST** is expected.
- **DELTA=VAL**  
This parameter must be in the range of 0 to 0.5, and is used to smooth out the output. The smooth-out occurs in the portion of the interval determined by the **DELTA** value. If **DELTA** is 0, smooth-out does not occur and strict linear interpolation is performed to compute the output. If **DELTA** is set to 0.5 the smooth-out occurs over the whole interval.
- **DELAY**  
The **F** element is controlled by the current, therefore, the item specified after the **DELAY** operator must be a device name, **VN**. The output is shifted by a **DELAY** value of **TD**.
- **TD=VAL**  
Delay value. The default value of **TD** is 0 if left unspecified.
- **INTEGRATION | DERIVATION**  
Specifies that the current flowing through **Fxx** should be equal to the *integral* (or *derivative*) of  $i(VN)$  multiplied by **VAL**.

## Examples

**f7 n4 n6 v9 7**

Specifies that the current through **f7** flowing from node **n4** to node **n6** is seven times the current of **v9**.

**f1 2 0 poly (2) v1 v2 0 1 1 3**

Specifies a 2nd order non-linear current controlled current source **f1** placed between node **2** and ground. The names of the zero voltage sources sensing the current for the function arguments of the polynomial are **v1** and **v2**. Polynomial coefficients are **PN0=0**, **PN1=1** and **PN2=3**. The resultant non-linear current function has the following form:

$$f_v = f_a + f_b + 3f_a^2$$

where  $f_a$  and  $f_b$  are equal to the current flowing through V1 and V2 respectively.

```
f2 3 0 poly (3) v1 v2 v3 0 1 1 3 2.5
```

Specifies a 3rd order non-linear current controlled current source **f2** placed between node 3 and ground. The names of the zero voltage sources sensing the current for the function arguments of the polynomial are v1, v2 and v3. Polynomial coefficients are PN0=0, PN1=1, PN2=1, PN3=3 and PN4=2.5.

The resultant non-linear current function has the following form:

$$f_v = f_a + f_b + 3f_c + 2.5f_a^2$$

where  $f_a$ ,  $f_b$  and  $f_c$  are equal to the current flowing through V1, V2 and V3 respectively.

```
V1 A 0 1
r1 A 0 1
V2 B 0 5
r2 B 0 5
F3 3 0 NAND(2) V1 V2 (-10,-30) (10,30)
r3 3 0 1
```

This example shows the use of the **NAND** keyword. In this case, the command of lower value is used, i.e. it is **v(A,0)** which will be used, the voltage on node 3 will be 3V (interpolation for x in [-10,10] and y in [-30,30]).

```
V1 A 0 1
r1 A 0 1
F3 3 0 PWL(1) V1 (-10,-30) (10,30)
r3 3 0 1
.DC V1 1 10 1
```

V(3) will vary from 3 to 30 when V(1) varies from 1 to 10.

# Voltage Controlled Current Source

## Linear (Transconductance Gain Block)

```
Gxx NP NN [VCR|VCCAP|VCCS] NCP NCN VAL [MIN=VAL] [MAX=VAL]
+ [TC1=VAL] [TC2=VAL] [SCALE=VAL] [ABS=VAL]
```

## Polynomial

```
Gxx NP NN [VCR|VCCAP|VCCS] POLY(ND) PCP PCN {PCP PCN} PN {PN}
+ [MIN=VAL] [MAX=VAL] [TC1=VAL] [TC2=VAL] [SCALE=VAL] [ABS=VAL]
```

## Piece Wise Linear

```
Gxx NP NN [VCR|VCCAP|VCCS] [PWL(1)|NPWL(1)|PPWL(1)] NCP NCN PWL_LIST
+ [DELTA=val] [MIN=VAL] [MAX=VAL] [TC1=VAL] [TC2=VAL] [SCALE=VAL]
+ [ABS=VAL]
```

## Multi-Input Gate

```
Gxx NP NN NAND(ND) | AND(ND) | OR(ND) | NOR(ND) PCP PCN {PCP PCN}
+ PWL_LIST [TC1=VAL] [TC2=VAL] [SCALE=VAL] [ABS=VAL]
```

## Delay Element

```
Gxx NP NN DELAY NCP NCN [TD=val] [ABS=VAL]
```

## Arithmetic Expression

```
Gxx NP NN VALUE={EXPR} [MIN=VAL] [MAX=VAL]
+ [TC1=VAL] [TC2=VAL] [SCALE=VAL] [ABS=VAL]
```

## Tabular

```
Gxx NP NN [VCR|VCCAP|VCCS] [MIN=VAL] [MAX=VAL] [TC1=VAL] [TC2=VAL]
+ [SCALE=VAL] [ABS=VAL] TABLE EXPR=(XN YN) {(XN YN)}
```

## Integral/Derivative

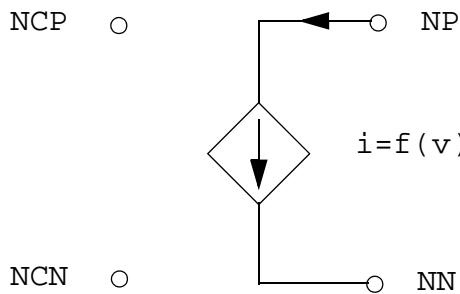
```
Gxx NP NN INTEGRATION|DERIVATION NCP NCN VAL
+ [MIN=VAL] [MAX=VAL] [TC1=VAL] [TC2=VAL] [SCALE=VAL] [ABS=VAL]
```

## Pole-Zeros

```
Gxx NP NN PZ NCP NCN a zrl zil ... zrm zim, b prl pil ... prn pin
```

## Frequency Dependent

```
Exx NP NN FREQ NCP NCN f0 a0 ph0 f1 a1 ph1... fn an phn
+ [RESTORE_CAUSALITY=val]
```



**Note**



The current flows from the positive node NP, through the current source, to the negative node NN.

---

### Parameters

- **xx**  
Voltage controlled current source name.
- **NP**  
Name of the positive node.
- **NN**  
Name of the negative node.
- **NCP**  
Name of the positive controlling node.
- **NCN**  
Name of negative controlling node.
- **VAL**  
Transadmittance in  $\Omega^{-1}$ .

The above parameters can be related by:

$$I(G_{xx}) = VAL(V(NCP) - V(NCN))$$

- **MIN=VAL**  
Minimum output current value. Unit Amps.
- **MAX=VAL**  
Maximum output current value. Unit Amps.

- **TC1, TC2**

First and Second order temperature coefficients. Default values are zero. The output value is updated with temperature according to the formula:

$$\text{VAL}(T) = \text{VAL}(\text{TNOM}) \times (1 + \text{TC1}(T - \text{TNOM}) + \text{TC2}(T - \text{TNOM})^2)$$

- **SCALE=VAL**

Element scale factor. The value of the element is multiplied by **SCALE**, which defaults to 1.

- **ABS=val**

Can be set to 1 or 0 (default is 0). If **ABS=1**, the output value is the absolute value of the signal.

- **POLY**

Keyword indicating the source has a non-linear polynomial description.

- **ND**

Order of the polynomial when **POLY** is specified. Number of Inputs when **NAND, NOR, AND, OR** is specified. **ND** must be greater than or equal to 1. If **ND** is not specified it will default to 1.

- **PCP**

Name of the positive controlling node giving the voltage difference for the function arguments of the polynomial. Number is equal to the order of the polynomial.

- **PCN**

Name of the negative controlling node giving the voltage difference for the function arguments of the polynomial. Number is equal to the order of the polynomial.

- **PN**

Coefficients of the polynomial.

- **VCCAP**

Used to instantiate a Voltage Controlled CAPacitor. The value of C is computed from the controlling nodes and polynomial coefficients in the same way that the **POLY** voltage control source values are computed.

- **VCR**

Used to instantiate a Voltage Controlled Resistor. The value of R is computed from the controlling nodes and polynomial coefficients in the same way that the **POLY** voltage control source values are computed. VCR output has an additional formulation, the **PWL**:

- **PWL\_LIST**

Consists of a list of couple values (x,y); interpolation will be made to extract the value, depending on `(v(ncp)-v(ncn))`.

- **PWL(1)**

When specified, a simple interpolation will be performed to evaluate the output. A list of couple values (x,y) specified as a **PWL\_LIST** is expected.

- **NPWL(1)**

When specified, the command value used to evaluate the output will be:  $V(ncp) - V(ncn)$  if  $V(NP) > V(NN)$ , otherwise the command value will be  $v(ncp) - v(NP)$ .

- **PPWL(1)**

When specified, the command value used to evaluate the output will be:  $V(ncp) - V(ncn)$  if  $V(NP) < V(NN)$ , otherwise the command value will be  $v(ncp) - v(NP)$ .

---

**Note**



For **NPWL(1)**, node **NCN** and **NN** must be the same node.

For **PPWL(1)**, node **NCN** and **NP** must be the same node.

Also note that if VCR values are limited to [-1.0e-15 1.0e-15] for instance, then zero value for VCR is excluded.

---

- **NAND, NOR, AND, OR**

One of these can be specified in place of the existing keyword **POLY**. If **NAND** or **AND** are used, the lower valued command will be used to compute the output. In the case of **NOR** or **OR**, the higher valued command will be used. In such cases of **NAND/AND/OR/NOR** types, Eldo expects a list of couple values (x,y) specified as a **PWL\_LIST**, the output will be the interpolated value y. Commas used as delimiters are optional.

- **DELTA=VAL**

This parameter must be in the range of 0 to 0.5, and is used to smooth out the output. The smooth-out occurs in the portion of the interval determined by the **DELTA** value. If **DELTA** is 0, smooth-out does not occur and strict linear interpolation is performed to compute the output. If **DELTA** is set to 0.5 the smooth-out occurs over the whole interval.

- **DELAY**

The **G** element is controlled by the voltage, therefore, the item specified after the **DELAY** operator must be the values of the positive and negative control nodes. The output is shifted by a delay value of **TD**.

- **TD=VAL**

Delay value. The default value of **TD** is 0 if left unspecified.

- **VALUE**

Keyword indicating that the source has a functional description. A set of expressions is specified in **EXPR**.

- **TABLE**

Keyword indicating that the source has a tabular description. The table itself contains pairs of values. **EXPR** is evaluated, and its value is used to look up an entry in the table. Linear interpolation is made between entries.

- **EXPR**

A set of expressions, used to set the source value or an entry look up for a tabular description of the source.



Expression formats are described in the [Eldo Control Language](#) chapter.

Controlled source expressions may also contain voltages, currents or time. Voltages may be the voltage through a node, e.g. **v(6)**, or the voltage across two nodes **v(5, 6)**. Currents must be the current through a voltage source, such as **i(v1)**.

**EXPR** is able to make use of the operators **DDT** and **IDT**. **DDT** stands for derivative, and **IDT** for integral. **DDT** and **IDT** operators utilize the integration scheme which is used by Eldo for computing the derivative/integral.



Please refer to “[Arithmetic Functions](#)” on page 3-7.

- **XN, YN**

Input and corresponding output source values for tabular source definitions.

- **INTEGRATION | DERIVATION**

Specifies that the current flowing through **Gxx** should be equal to the *integral* (or *derivative*) of **v(NCP)–v(NCN)** multiplied by **VAL**.

- **PZ**

When the poles and zeroes are known, the transfer function is:

$$H = \frac{a \cdot \prod((s - (zri + j \cdot 2 \cdot pi \cdot zii)) \cdot (s - (zri - 2 \cdot pi \cdot zii)))}{b \cdot \prod((s - (pri + j \cdot 2 \cdot pi \cdot pii)) \cdot (s - (pri - 2 \cdot pi \cdot pii)))}$$

For the numerator, **i** is from 1 to m. For the denominator, **i** is from 1 to n.

**Note**

The conjugate appears only if the imaginary part is not 0.

- **a, b**

Coefficients for the transfer function of **PZ**.

- **pr, pi**

Poles of the transfer function. **pr** is the real part, **pi** the imaginary part.

- **zr, zi**  
Zeroes of the transfer function. **zr** is the real part, **zi** the imaginary part.
- **FREQ**  
Specifies a frequency domain description. Enables a frequency dependent voltage controlled source to be simulated for AC, Transient, and MODSST analyses.
- **f0 a0 ph0 f1 a1 ph1... fn an phn**  
Coefficients for the frequency domain. **fi**, **ai** and **phi** are the frequency, the amplitude in dB, and the phase in degrees respectively. At each frequency point, Eldo will evaluate the amplitude in dB by making a log interpolation, and will evaluate the phase by making a linear interpolation.
- **RESTORE\_CAU<sub>S</sub>LITY=val**  
The device characteristics are defined by an equation, when set to 1 the causality of the equation will be restored (if required) by building the corresponding imaginary part, this will produce results with an even higher level of accuracy. When set to 0 (default) the causality of the equation will not be restored.

## Examples

```
g2 n2 n0 5 0 0.0005
```

Specifies that the current through **g2** from node **n2** to ground is equal to 0.0005 times the potential difference between node **5** and ground.

```
g1 3 0 poly (3) 3 0 9 0 5 0 0 1 1 3 2 1
```

Specifies a 3rd order non-linear voltage controlled current source **g1** placed between node **3** and ground. The three controlling voltages giving the voltage difference for the function arguments occur between node 3 (NCP1) and ground (NCN1), node 9 (NCP2) and ground (NCN2), and node 5 (NCP3) and ground (NCN3). Polynomial coefficients are P0=0, P1=1, P2=3, P3=2 and P5=1.

The resultant non-linear voltage function has the following form:

$$f_v = f_a + f_b + 3f_c + 2f_a^2 + f_a f_b$$

where: **f<sub>a</sub>** is the voltage difference between the controlling nodes NCP1 and NCN1.

**f<sub>b</sub>** is the voltage difference between the controlling nodes NCP2 and NCN2.

**f<sub>c</sub>** is the voltage difference between the controlling nodes NCP3 and NCN3.

```
g1 1 2 value={v(6)*v(7)}
```

**Voltage Controlled Current Source**

Specifies current through **g1** from node 1 to node 2 to be equal to the product of voltages at nodes 6 and 7.

```
g3 5 0 table v(5)=(0,0) (0.02,2.6e-3) (0.04,4.4e-3)
+ (0.06,4.2e-3)(0.08,3.7e-3) (0.10,3.5e-3)(0.12,3.9e-3)
```

Specifies a voltage controlled current source between nodes 5 and 0. Current out of node 5 is controlled via the table by the voltage at the node 5. Thus, values in the table describe the I–V device characteristics.

```
V1 A 0 1
r1 A 0 1
V2 B 0 5
r2 B 0 5
G3 3 0 NAND(2) A 0 B 0 (-10,-30) (10,30)
r3 3 0 1
```

This example shows the use of the **NAND** keyword. In this case, the command of lower value is used, i.e. it is **v(A,0)** which will be used, the voltage on node 3 will be 3V (interpolation for x in [-10,10] and y in [-30,30]).

```
V1 A 0 1
r1 A 0 1
G3 3 0 PWL(1) A 0 (-10,-30) (10,30)
r3 3 0 1
.DC V1 1 10 1
```

V(3) will vary from 3 to 30 when V(1) varies from 1 to 10.

## Current Controlled Voltage Source

### Linear (Transresistance Gain Block)

```
HXX NP NN [CCVS] VN VAL [MIN=VAL] [MAX=VAL]
+ [TC1=VAL] [TC2=VAL] [SCALE=VAL] [ABS=VAL]
```

### Polynomial

```
HXX NP NN [CCVS] POLY(ND) VN {VN} PN {PN}
+ [MIN=VAL] [MAX=VAL] [TC1=VAL] [TC2=VAL] [SCALE=VAL] [ABS=VAL]
```

### Piece Wise Linear

```
HXX NP NN PWL(1) VN PWL_LIST [DELTA=val]
+ [TC1=VAL] [TC2=VAL] [SCALE=VAL] [ABS=VAL]
```

### Multi-Input Gate

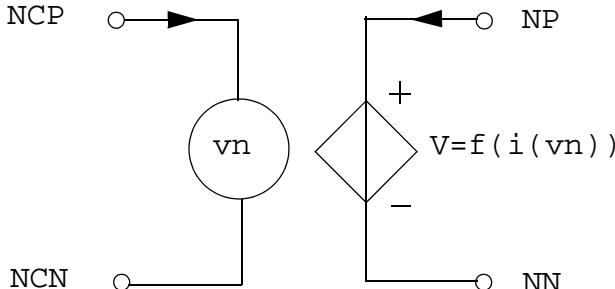
```
HXX NP NN NAND(ND) | AND(ND) | OR(ND) | NOR(ND) VN {VN}
+ PWL_LIST [TC1=VAL] [TC2=VAL] [SCALE=VAL] [ABS=VAL]
```

### Delay Element

```
HXX NP NN DELAY VN [TD=val] [ABS=VAL]
```

### Integral/Derivation

```
HXX NP NN INTEGRATION|DERIVATION VN VAL [MIN=VAL] [MAX=VAL]
+ [TC1=VAL] [TC2=VAL] [SCALE=VAL] [ABS=VAL]
```



#### Note

The current flows from the positive node NP, through the current source, to the negative node NN. NCP and NCN are the positive and negative controlling nodes for source vn.

### Parameters

- **xx**  
Current controlled voltage source name.
- **NP**  
Name of the positive node.

- **NN**  
Name of the negative node.
- **VN**  
Name of the voltage source through which the controlling current flows. The direction of positive controlling current flow is from the positive node, through the source, to the negative node of **VN**. When **POLY** is specified, one name must be added for each dimension of the polynomial. If **POLY** is not specified, it is assumed that there is only one dimension for which a name should be added.
- **VAL**  
Transresistance in  $\Omega$ .

The above parameters can be related by:

$$V(NP) - V(NN) = I(VNAM) \times VAL$$

- **MIN=VAL**  
Minimum output voltage value. Unit Volts.
- **MAX=VAL**  
Maximum output voltage value. Unit Volts.
- **TC1, TC2**  
First and Second order temperature coefficients. Default values are zero. The output value is updated with temperature according to the formula:

$$VAL(T) = VAL(TNOM) \times (1 + TC1(T - TNOM) + TC2(T - TNOM)^2)$$

- **SCALE=VAL**  
Element scale factor. The value of the element is multiplied by **SCALE**, which defaults to 1.
- **ABS=val**  
Can be set to 1 or 0 (default is 0). If **ABS=1**, the output value is the absolute value of the signal.
- **POLY**  
Keyword indicating the source has a non-linear polynomial description.
- **ND**  
Order of the polynomial when **POLY** is specified. Number of Inputs when **NAND, NOR, AND, OR** is specified. **ND** must be greater than or equal to 1. If **ND** is not specified it will default to 1.
- **PN**  
Coefficients of the polynomial.

- **NAND, NOR, AND, OR**

One of these can be specified in place of the existing keyword **POLY**. If **NAND** or **AND** are used, the lower valued command will be used to compute the output. In the case of **NOR** or **OR**, the higher valued command will be used. In such cases of **NAND/AND/OR/NOR** types, Eldo expects a list of device names (x,y) specified as a **PWL\_LIST**, the output will be the interpolated value y. Commas used as delimiters are optional.

- **PWL\_LIST**

Consists of a list of couple values (x,y); interpolation will be made to extract the value, depending on **(v(ncp)-v(ncn))**.

- **PWL(1)**

When specified, a simple interpolation will be performed to evaluate the output. A list of couple values (x,y) specified as a **PWL\_LIST** is expected.

- **DELTA=VAL**

This parameter must be in the range of 0 to 0.5, and is used to smooth out the output. The smooth-out occurs in the portion of the interval determined by the **DELTA** value. If **DELTA** is 0, smooth-out does not occur and strict linear interpolation is performed to compute the output. If **DELTA** is set to 0.5 the smooth-out occurs over the whole interval.

- **DELAY**

The **H** element is controlled by the current, therefore, the item specified after the **DELAY** operator must be a device name **VN**. The output is shifted by a **DELAY** value of **TD**.

- **TD=VAL**

Delay value. The default value of **TD** is 0 if left unspecified.

- **INTEGRATION | DERIVATION**

Specifies that the current flowing across **NP** and **NN** should be equal to the *integral* (or *derivative*) of **i(VN)** multiplied by **VAL**.

## Examples

```
h7 n4 n8 v12 7.5
```

Specifies the voltage applied between nodes **n4** and **N8** to be equal to the current through **v12** multiplied by  $7.5\Omega$ .

```
h1 2 0 poly(1) v1 1 2 4
```

Specifies a 1st order non-linear current controlled voltage source **h1** placed between node **2** and ground. The name of the zero voltage source sensing the current for the function arguments is **v1**. Polynomial coefficients are **P0=1**, **P1=2** and **P2=4**.

The resultant non-linear voltage function has the following form:

$$f_v = 1 + 2f_a + 4f_a^2$$

where  $f_a$  is equal to the current flowing through V1.

```
h2 3 0 poly(3) v1 v2 v3 0 1 1 1
```

Specifies a 3rd order non-linear current controlled voltage source H2 placed between node 3 and ground. The names of the zero voltage sources sensing the current for the function arguments are V1, V2 and V3. Polynomial coefficients are P0=0, P1=1, P2=1 and P3=1. The resultant non-linear current function has the following form:

$$f_v = f_a + f_b + f_c$$

where  $f_a, f_b$  and  $f_c$  are equal to the current flowing through V1, V2 and V3 respectively.

```
v1 A 0 1
r1 A 0 1
v2 B 0 5
r2 B 0 5
H3 3 0 NAND(2) v1 v2 (-10,-30) (10,30)
r3 3 0 1
```

This example shows the use of the **NAND** keyword. In this case, the command of lower value is used, i.e. it is  $v(A, 0)$  which will be used, the voltage on node 3 will be 3V (interpolation for x in [-10,10] and y in [-30,30]).

```
v1 A 0 1
r1 A 0 1
H3 3 0 PWL(1) v1 (-10,-30) (10,30)
r3 3 0 1
.DC V1 1 10 1
```

V(3) will vary from 3 to 30 when V(1) varies from 1 to 10.

## S, Y, Z Parameter Extraction

```
Vyy NP NN IPORT=VAL [RPORT=VAL] [CPORT=VAL] [LPORT=VAL] [MODE=KEYWORD]
Vyy NP NN IPORT=VAL ZPORT_FILE=string [CPORT=VAL] [LPORT=VAL]
+ [MODE=KEYWORD]
Iyy NP NN IPORT=VAL [RPORT=VAL] [CPORT=VAL] [LPORT=VAL] [MODE=KEYWORD]
Iyy NP NN IPORT=VAL ZPORT_FILE=string [CPORT=VAL] [LPORT=VAL]
+ [MODE=KEYWORD]
```

### Parameters

- **YY**  
Name of the port.
- **NP**  
Name of the positive Node.
- **NN**  
Name of the Negative Node.
- **IPORT**  
This is a strictly positive number that is unique and is used as the port number: this number is used for naming the outputs (for instance, .PLOT AC S(1,2)). An error message will be issued if two port instances have the same value for **IPORT**, or if an **IPORT** is missing (e.g. maximum **IPORT** number found in the netlist is 4, and there is no instance with **IPORT 3**).

### Optional Parameters

- **RPORT**  
Value of the Reference Impedance in Ohms. Default value is  $50\Omega$ .
- **CPORT**  
Capacitor placed in series with **RPORT**. Defaults to 0, in which case it behaves like a zero voltage source (i.e. **CPORT** would have no effect).
- **LPORT**  
Inductor placed in series with **RPORT**. Defaults to 0.
- **ZPORT\_FILE**  
Specifies the Touchstone file name that contains the port source with a complex impedance from which the S parameters will be extracted from.
- **MODE=SINGLE | COMMON | DIFFERENTIAL**  
Mixed-mode S parameter selection.  
**SINGLE** specifies the port as single ended, it is dedicated to S parameter extraction. Default.  
**COMMON** and **DIFFERENTIAL** specify that the port is not single ended. Such ports are split into two linked sources that are either common (same amplitude and same phase) or

differential (same amplitude but opposite phases). During S parameter extraction a “non single ended” port is equally common and differential depending on which display is required. During simulation (DC, AC or TRAN) this port is either common or differential depending on the specified mode keyword.

---

**Note**

---

 Port numbers in `VYY` instances should range from 1 to the total number of ports without discontinuity. The simulation parameters `FMIN`, `FMAX`, and Number of frequency points for the analysis are specified with a `.AC` command.

---



For further information, please see “[Working with S, Y, Z Parameters](#)” on page 15-1.

---

# Chapter 6

## Analog Macromodels

---

### Eldo Analog Macromodels

The following analog macromodels are provided in Eldo:

Comparator (Single Output)	<b>COMP</b>
Comparator (Differential Output)	<b>COMPD</b>
Op-amp (Linear, Single Output)	<b>OPAMP0</b>
Op-amp (Linear, Differential Output)	<b>OPAMP0D</b>
Op-amp (Linear 1-pole, Single Output)	<b>OPAMP1</b>
Op-amp (Linear 1-pole, Differential Output)	<b>OPAMP1D</b>
Op-amp (Linear 2-pole, Single Output)	<b>OPAMP2</b>
Op-amp (Linear 2-pole, Differential Output)	<b>OPAMP2D</b>
Delay	<b>DEL</b>
Saturating Resistor	<b>SATR</b>
Voltage Limiter	<b>SATV</b>
Voltage Controlled Switch	<b>VSWITCH</b>
Current Controlled Switch	<b>CSWITCH</b>
Triangular to Sine Wave Converter	<b>TRI2SIN</b>
Staircase Waveform Generator	<b>STAIRGEN</b>
Sawtooth Waveform Generator	<b>SAWGEN</b>
Triangle Waveform Generator	<b>TRIGEN</b>
Amplitude Modulator	<b>AMM</b>
Pulse Amplitude Modulator	<b>PAM</b>
Sample and Hold	<b>SA_HO</b>
Track and Hold	<b>TR_HO</b>
Pulse Width Modulator	<b>PWM</b>
Voltage Controlled Oscillator	<b>VCO</b>
Peak Detector	<b>PEAK_D</b>

Level Detector (Single & Differential Output)	<a href="#">LEV_D</a>
Logarithmic Amplifier	<a href="#">LOGAMP</a>
Anti-logarithmic Amplifier	<a href="#">EXPAMP</a>
Differentiator	<a href="#">DIFF</a>
Integrator	<a href="#">INTEG</a>
Adder, Subtractor, Multiplier, Divider	<a href="#">ADD</a> , <a href="#">SUB</a> , <a href="#">MULT</a> , <a href="#">DIV</a>

## General Notes on the Use of FAS Macromodels

For FAS macromodels, i.e. all those using the **Y** prefix, parameters can be declared in a number of ways. These are listed below in order of priority.

---

### Note

---



#### Note 1

In the instantiation line, using the **PARAM** keyword. The **PARAM:** command has to be followed by a white space.

#### Note 2

In a model command line, the syntax is as follows:

```
.model mname modfas par=val [par=val]
```

If this syntax is used, a model name must be declared using the **MODEL:** keyword in the instantiation.

#### Note 3

If parameters are not explicitly declared, the parameter default values are used.

---



For further information on FAS models, please refer to the *CFAS User's Manual*.

---

# Comparator

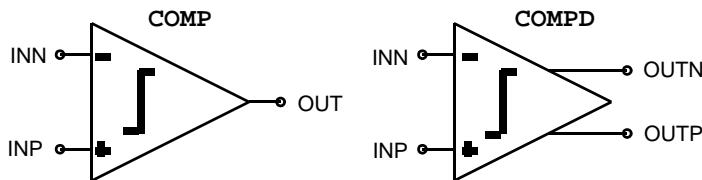
## Single Output

```
COMPXX INP INN OUT [MNAME] [VHI=VAL1] [VLO=VAL2]
+ [VOFF=VAL3] [VDEF=VAL4] [TCOM=VAL5] [TPD=VAL6]
```

## Differential Output

```
COMPDXX INP INN OUTP OUTN [MNAME] [VHI=VAL1] [VLO=VAL2]
+ [VOFF=VAL3] [VDEF=VAL4] [TCOM=VAL5] [TPD=VAL6]
```

**Figure 6-1. Comparator Macromodel**



Eldo offers two comparator macromodels, the single output comparator **COMP** and the differential output comparator **COMPD**.

## Model Pins

INP	Name of the positive input node.
INN	Name of the negative input node.
OUT	Output node for the single output comparator.
OUTP	Positive output node for the differential output comparator.
OUTN	Negative output node for the differential output comparator.
MNAME	Name of a model described with the <b>.MODLOGIC</b> or <b>MODEL ... LOGIC</b> command.
VHI=VAL1	Upper voltage level. Default value is 5V.
VLO=VAL2	Lower voltage level. Default value is 0V.
VOFF=VAL3	Offset input voltage. The voltage on the positive input node is compared with that on the negative added to <b>VOFF</b> . Default value is 0V.
VDEF=VAL4	Hysteresis voltage in volts. Default is 0V.
TCOM=VAL5	Commutation time. This is the time for the output to switch from <b>VHI</b> to <b>VLO</b> or vice versa. Default value is 1ns.
TPD=VAL6	Transit time through the comparator. Default value is zero.

The above parameters can be related in the following way:

**Table 6-1. Comparator Parameters**

VIN-	Voltage on INN
VIN+	Voltage on INP
VOUT-	Voltage on OUT (OUTN)
VOUT+	Voltage on OUTP

If  $VIN+(t) > VIN-(t) + VOFF + VDEF$

and  $VIN+(t - 1) < VIN-(t - 1) + VOFF + VDEF$

then the output rises.

If  $VIN+(t) < VIN-(t) + VOFF - VDEF$

and  $VIN+(t - 1) > VIN-(t - 1) + VOFF - VDEF$

then the output falls.

### For a differential comparator

$$VOUT- = (VHI + VLO) - VOUT+$$

### Examples

```
compd2 vp vn voutp voutn voff=0.9 vdef=0.1
+ vhi=2.5 vlo=-2.5
```

Specifies a differential output comparator **compd2** with input nodes **vp** (+ve), **vn** (-ve) and output nodes **voutp** (+ve), **voutn** (-ve). The upper and lower thresholds of the comparator are +2.5V and -2.5V respectively and input offset is 0.9V. The comparator exhibits a hysteresis of 0.1V.

```
* .MODLOGIC definition
.modlogic compar vhi=2.5 vlo=-2.5
...
comp1 vinp vinn vout compar voff=0.9 vdef=0.1
```

Specifies a single output comparator **comp1** of model type **compar** with input nodes **vinp** (+ve), **vinn** (-ve) and output node **vout**. The comparator input offset voltage and hysteresis are 0.9V and 0.1V respectively. The comparator thresholds are  $\pm 2.5V$ .



For more information, refer to “[.MODLOGIC](#)” on page 10-164.

---

## Op-amp (Linear)

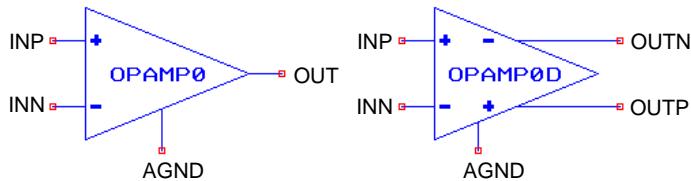
### Single Output

```
YXX OPAMP0 [PIN:] INP INN OUT AGND
+ [PARAM: PAR=VAL {PAR=VAL}] [MODEL: MNAME]
```

### Differential Output

```
YXX OPAMP0D [PIN:] INP INN OUTN OUTP AGND
+ [PARAM: PAR=VAL {PAR=VAL}] [MODEL: MNAME]
```

**Figure 6-2. Op-amp (Linear) Macromodel**



Two general linear operational amplifier macromodels are available, the single output linear gain op-amp **OPAMP0**, and the differential output linear gain op-amp **OPAMP0D**. Voltage clipping effects can be described using the voltage limiter macromodel (**SATV**).

### Model Pins

INP	Name of the positive input node.
INN	Name of the negative input node.
OUT	Output node for the single output op-amp.
OUTP	Positive output node for the differential output op-amp.
OUTN	Negative output node for the differential output op-amp.
AGND	Name of the ground node.

**Table 6-2. Op-amp (Linear) Model Parameters**

Nr.	Name	Default	Units	Definition
1	<b>GAIN</b> <sup>a</sup>	$1.0 \times 10^5$		Amplifier gain
2	<b>RIN</b>	$1.0 \times 10^7$	$\Omega$	Input resistance
3	<b>M</b> <sup>b</sup>	1		Device multiplier

a. Can also be specified in dB.

b. **.OPTION YMFACT** must be specified for **M** to work.



See “General Notes on the Use of FAS Macromodels” on page 6-2 for additional notes on using this model.

## Examples

```
yop1 opamp0 n2 n1 n3 0 param: gain=1000 rin=10meg
```

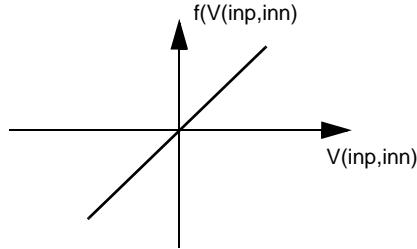
Specifies a linear gain operational amplifier **yop1** of type **opamp0** having input nodes **n2** (+ve) and **n1** (-ve) with output node **n3**. Gain and input resistance parameters are declared explicitly in the instantiation.

```
.model tdk modfas gain=2000 rin=20 meg
...
yop1 opamp0d n2 n1 n3 n4 0 model: tdk
```

Specifies a differential linear gain operational amplifier **yop1** of type **opamp0d** with input nodes **n2** (+ve) and **n1** (-ve) and output nodes **n3** (-ve) and **n4** (+ve). Gain and input resistance parameters are declared using the **.MODEL** command.

## Model Equations

**Figure 6-3. Op-amp (Linear) Model Characteristics**



## General

$$I(\text{inp}) = \frac{V(\text{inp},\text{inn})}{R_{\text{IN}}}$$

$$I(\text{inn}) = -\frac{V(\text{inp},\text{inn})}{R_{\text{IN}}}$$

## Single Output

$$V(\text{out,agnd}) = \text{GAIN} \times V(\text{inp},\text{inn})$$

## Differential Output

$$V(\text{outn,agnd}) = -\frac{1}{2}\text{GAIN} \times V(\text{inp},\text{inn})$$

$$V(\text{outp,agnd}) = \frac{1}{2}\text{GAIN} \times V(\text{inp},\text{inn})$$

## Op-amp (Linear 1-pole)

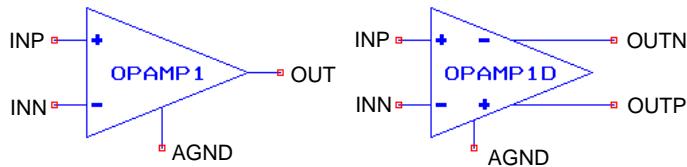
### Single Output

```
YXX OPAMP1 [PIN:] INP INN OUT AGND
+ [PARAM: PAR=VAL {PAR=VAL}] [MODEL: MNAME]
```

### Differential Output

```
YXX OPAMP1D [PIN:] INP INN OUTN OUTP AGND
+ [PARAM: PAR=VAL {PAR=VAL}] [MODEL: MNAME]
```

**Figure 6-4. Op-amp (Linear 1-pole) Macromodel**



Two operational amplifier macromodels are implemented, namely the single output linear gain one pole op-amp **OPAMP1** and the differential output one pole op-amp **OPAMP1D**.

### Model Pins

INP	Name of the positive input node.
INN	Name of the negative input node.
OUT	Output node for the single op-amp.
OUTP	Positive output node for the differential output op-amp.
OUTN	Negative output node for the differential output op-amp.
AGND	Name of the ground node.

**Table 6-3. Op-amp (Linear 1-pole) Model Parameters**

Nr.	Name	Default	Units	Definition
1	<b>GAIN</b> <sup>a</sup>	$1.0 \times 10^5$		Open-loop gain
2	<b>VOFF</b>	0	V	Offset voltage
3	<b>P1</b>	$1.0 \times 10^2$	Hz	Dominant pole frequency
4	<b>RIN</b>	$1.0 \times 10^7$	$\Omega$	Input resistance
5	<b>CMRR</b> <sup>ab</sup>	0		Common mode rejection ratio
6	<b>M</b> <sup>c</sup>	1		Device multiplier

a. Can also be specified in dB.

b. If  $CMRR=0$ , then  $1/CMRR$  is ignored in the model equations.

c. .OPTION YMFACT must be specified for **M** to work.



See “General Notes on the Use of FAS Macromodels” on page 6-2 for additional notes on using this model.

---

## Examples

```
yop1 opamp1 n2 n1 n3 0 param: voff=100e-6
```

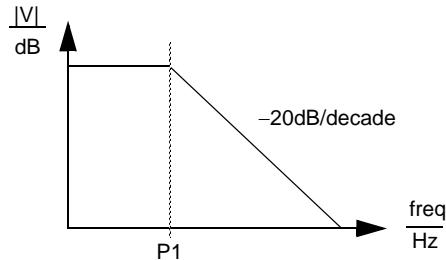
Specifies a linear gain operational amplifier **yop1** of model type **opamp1** having input nodes **n2** (+ve) and **n1** (-ve) with output node **n3**. The op-amp offset voltage is declared explicitly in the instantiation line.

```
.model tdk modfas voff=100e-6
...
yop1 opamp1d n1 n2 n3 n4 0 model: tdk
```

Specifies a differential linear gain operational amplifier **yop1** of type **opamp1d** with input nodes **n1** (+ve) and **n2** (-ve) and output nodes **n3** (-ve) and **n4** (+ve). The offset voltage parameter is declared using the **.MODEL** command.

## Model Equations

**Figure 6-5. Op-amp (Linear 1-pole) Model Characteristics**



## General

$$I(\text{inp}) = \frac{V(\text{inp},\text{inn}) + \text{VOFF}}{\text{RIN}}$$

$$I(\text{inn}) = - \frac{V(\text{inp},\text{inn}) + \text{VOFF}}{\text{RIN}}$$

$$\text{TAU} = \frac{1}{(2\pi \times P1)}$$

$$VA = \text{GAIN} \times \left( (V(\text{inp},\text{inn}) + \text{VOFF}) + \frac{1}{\text{CMRR}} \times \frac{(V(\text{inp}) + V(\text{inn}))}{2} \right)$$

**Note**

If  $CMRR = 0$ , then  $1/CMRR$  is ignored. VA is then calculated as if no value for  $CMRR$  had been specified.

## Single Output

$$\frac{V_{(out,agnd)}}{VA} = \frac{1}{1 + TAU \times p}$$

$p$  is the complex frequency.

## Differential Output

$$\frac{V_{(outn,agnd)}}{VA} = -\frac{1}{2} \left( \frac{1}{1 + TAU \times p} \right)$$

$$\frac{V_{(outp,agnd)}}{VA} = \frac{1}{2} \left( \frac{1}{1 + TAU \times p} \right)$$

## Application Area

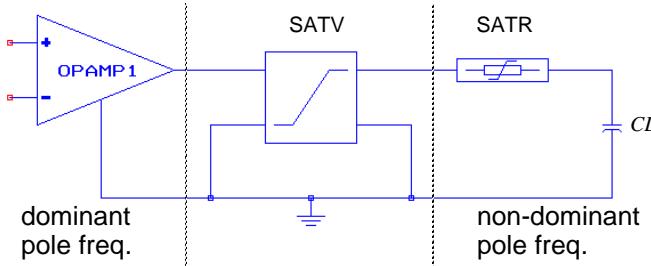
A typical application area for the linear one-pole op-amp is in the modeling of two-stage amplifiers including real saturation effects. This application is achieved in three parts as explained below:

1. The first stage of the amplifier is described using the **OPAMP1** macromodel with the equation:

$$\frac{V_{out}}{V_{in}} = \frac{GAIN}{(1 + T \times p)}$$

2. Clipping of the voltage is achieved using the voltage limiter macromodel **SATV** at the output of the **OPAMP1** macromodel.
3. The second stage of the amplifier is described using an external capacitor **CL**, together with a non-linear resistor **R**, implemented as a saturating resistor (**SATR** macromodel).

**Figure 6-6. Op-amp (Linear 1-pole) Application Area**



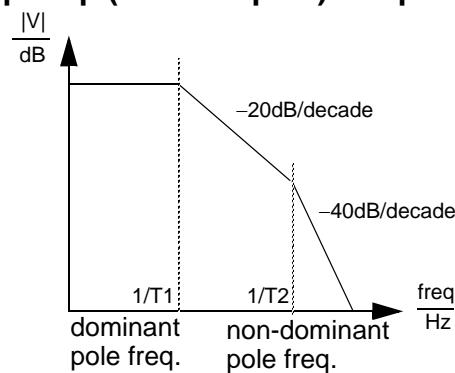
The dominant pole of the amplifier is determined by the following equation:

$$T1 = \frac{1}{(2\pi \times P1)} + R \times CL$$

The non-dominant pole of the amplifier is determined by the following equation:

$$T2 = \frac{1}{(2\pi \times P1)} \times (R \times CL)$$

**Figure 6-7. Op-amp (Linear 1-pole) Frequency Response**



An example of the above application can be found in the [Examples](#) appendix of this manual.

---

## Op-amp (Linear 2-pole)

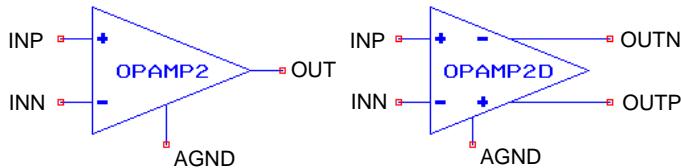
### Single Output

```
YXX OPAMP2 [PIN:] INP INN OUT AGND
+ [PARAM: PAR=VAL {PAR=VAL}] [MODEL: MNAME]
```

### Differential Output

```
YXX OPAMP2D [PIN:] INP INN OUTN OUTP AGND
+ [PARAM: PAR=VAL {PAR=VAL}] [MODEL: MNAME]
```

**Figure 6-8. Op-amp (Linear 2-pole) Macromodel**



Two linear 2-pole op-amp macromodels are provided, a single output linear gain two pole op-amp **OPAMP2**, and a differential output two pole op-amp **OPAMP2D**.

### Model Pins

INP	Name of the positive input node.
INN	Name of the negative input node.
OUT	Output node for the single op-amp.
OUTP	Positive output node for the differential output op-amp.
OUTN	Negative output node for the differential output op-amp.
AGND	Name of the ground node.

**Table 6-4. Op-amp (Linear 2-pole) Model Parameters**

Nr.	Name	Default	Units	Definition
1	<b>GAIN</b> <sup>a</sup>	$1.0 \times 10^5$		Open-loop gain
2	<b>VOFF</b>	0	V	Offset voltage
3	<b>P1</b>	$1.0 \times 10^2$	Hz	Dominant pole frequency
4	<b>P2</b>	$1.0 \times 10^6$	Hz	Non-dominant pole frequency
5	<b>RIN</b>	$1.0 \times 10^7$	$\Omega$	Input resistance
6	<b>CMRR</b> <sup>ab</sup>	0		Common mode rejection ratio
7	<b>M</b> <sup>c</sup>	1		Device multiplier

- a. Can also be specified in dB.
- b. If  $CMRR=0$ , then  $1/CMRR$  is ignored in the model equations.
- c. **.OPTION YMFACT** must be specified for **M** to work.



See “[General Notes on the Use of FAS Macromodels](#)” on page 6-2 for additional notes on using this model.

---

## Examples

```
yopa1 opamp2 pin: n2 n1 n3 0 param: p1=300
```

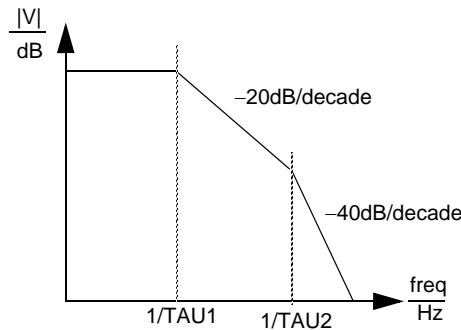
Specifies a linear two-pole operational amplifier **yopa1** of model type **opamp2** having input nodes **n2** (+ve) and **n1** (-ve) with output node **n3**. The op-amp dominant pole frequency parameter is declared explicitly in the instantiation line.

```
.model tdk modfas p1=300
...
yopa4 opamp2d n1 n2 n3 n4 0 model: tdk
```

Specifies a differential linear gain op-amp **yopa4** of type **opamp2d** with input nodes **n1** (+ve) and **n2** (-ve) and output nodes **n3** (-ve) and **n4** (+ve). The dominant pole frequency parameter is declared using the **.MODEL** command.

## Model Equations

**Figure 6-9. Op-amp (Linear 2-pole) Model Characteristic**



## General

$$I(\text{inp}) = \frac{V(\text{inp}, \text{inn}) + V_{\text{OFF}}}{R_{\text{IN}}}$$

$$I(\text{inn}) = - \frac{V(\text{inp}, \text{inn}) + V_{\text{OFF}}}{R_{\text{IN}}}$$

$$\text{TAU1} = \frac{1}{(2\pi \times P1)} + \frac{1}{(2\pi \times P2)}$$

$$\text{TAU2} = \frac{1}{(2\pi \times P1)} \times \frac{1}{(2\pi \times P2)}$$

$$VA = \text{GAIN} \times \left( (V(\text{inp},\text{inn}) + VOFF) + \frac{1}{CMRR} \times \frac{(V(\text{inp}) + V(\text{inn}))}{2} \right)$$

**Note**

If  $CMRR = 0$ , then  $1/CMRR$  is ignored. VA is then calculated as if no value for  $CMRR$  had been specified.

### Single output

$$\frac{V(\text{out,agnd})}{VA} = \frac{1}{1 + TAU1 \times p + TAU2 \times p^2}$$

$p$  is the complex frequency

### Differential output

$$\frac{V(\text{outn,agnd})}{VA} = -\frac{1}{2} \left( \frac{1}{1 + TAU1 \times p + TAU2 \times p^2} \right)$$

$$\frac{V(\text{outp,agnd})}{VA} = \frac{1}{2} \left( \frac{1}{1 + TAU1 \times p + TAU2 \times p^2} \right)$$

## Delay

**DELxx IN OUT VAL**

**Figure 6-10. Delay Macromodel**



This macromodel describes an ideal delay element that transfers its input voltage to its output after a specified time delay, where the reference node is ground. The input impedance is infinite.

### Model Pins

**IN** Name of the input node.

**OUT** Name of the output node.

### Parameters

**VAL** Value of the delay in seconds.

### Example

```
dell a1 a2 2.0e-9
```

Specifies a delay element placed between nodes `a1` and `a2`, with a delay of 2ns.

---

**Note**

This macromodel can not be used in a `.AC` analysis.

---

## Saturating Resistor

**YXX SATR [PIN:] IN OUT [PARAM: PAR=VAL {PAR=VAL}] [MODEL: MNAME]**

**Figure 6-11. Saturating Resistor Macromodel**



An analog saturating resistor macromodel. Current clipping can be specified directly using the parameter **IMAX**. This model is also designed to be used in conjunction with the one- and two-pole op-amp macromodels (**OPAMP1**, **OPAMP2**) and the voltage limiter **SATV**. If this option is used, the saturating current is specified indirectly by the value of the dominant pole frequency **P1** and the Slew Rate **SR**.

### Model Pins

**IN** Name of the input node.

**OUT** Name of the output node.

**Table 6-5. Saturating Resistor Model Parameters**

Nr.	Name	Default	Units	Definition
1	<b>R</b>	1	$\Omega$	Resistance
2	<b>IMAX</b>	1	A	Maximum current
3	<b>SR</b>	0	V/ $\mu$ s	Slew rate
4	<b>P1</b>	$1.0 \times 10^6$	Hz	Dominant pole frequency
5	<b>R1</b>	30	$\Omega$	Resistance of 1st order low pass filter
6	<b>M</b> <sup>a</sup>	1		Device multiplier

a. .OPTION YMFACT must be specified for **M** to work.



See “[General Notes on the Use of FAS Macromodels](#)” on page 6-2 for additional notes on using this model.

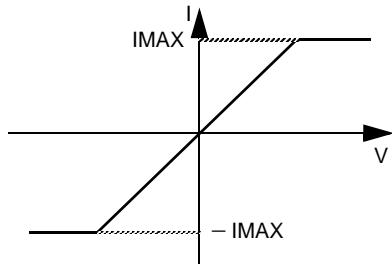
### Example

```
ycl1 satr n1 n2
```

Specifies a current limiter **ycl1** of type **satr** having input node **n1** with output node **n2**. Default model parameters are used.

## Model Equations

**Figure 6-12. Saturating Resistor Model Characteristics**



$$I = \frac{V(\text{in}) - V(\text{out})}{R}$$

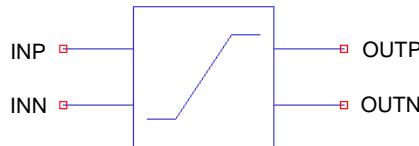
If  $SR$  is specified

$$IMAX = \frac{SR}{(2\pi \times P1 \times R1)}$$

## Voltage Limiter

```
YXX SATV [PIN:] INP INN OUTP OUTN
+ [PARAM: PAR=VAL {PAR=VAL}] [MODEL: MNAME]
```

**Figure 6-13. Voltage Limiter Macromodel**



An analog voltage limiter macromodel, where the input voltage is limited to specified values. The voltage describes exponential behavior in the saturation regions and a 3rd degree polynomial in the working region.

### Model Pins

INP	Name of the positive input node.
INN	Name of the negative input node.
OUTP	Name of the positive output node.
OUTN	Name of the negative output node.

**Table 6-6. Voltage Limiter Model Parameters**

Nr.	Name	Default	Units	Definition
1	<b>VMAX</b>	5	V	Maximum output voltage
2	<b>VMIN</b>	-5	V	Minimum output voltage
3	<b>VSATP</b>	4.75	V	Positive saturation input voltage
4	<b>VSATN</b>	-4.75	V	Negative saturation input voltage
5	<b>NSLOPE</b>	0.25		Slope of Transfer Function at <b>VSATN</b>
6	<b>PSLOPE</b>	0.25		Slope of Transfer Function at <b>VSATP</b>
7	M <sup>a</sup>	1		Device multiplier

a. .OPTION YMFACT must be specified for **M** to work.

- 
- i** See “General Notes on the Use of FAS Macromodels” on page 6-2 for additional notes on using this model.
-

## Examples

```
ysat1 satv n2 n1 n3 0 param: vmin=-3.0
```

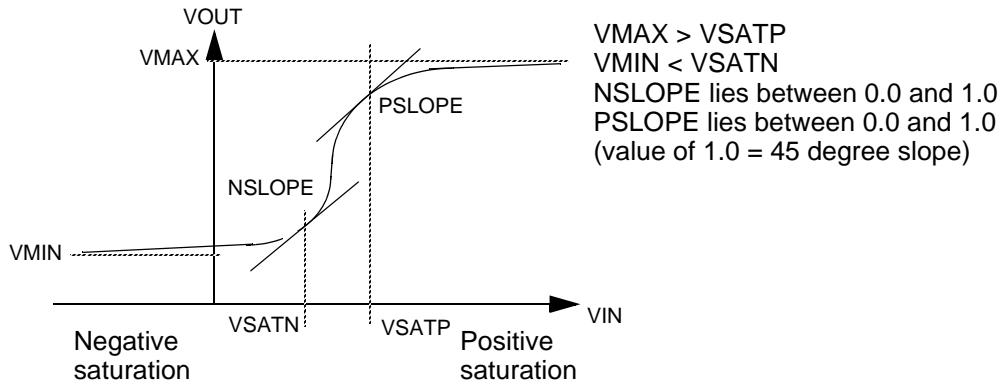
Specifies a voltage limiter **ysat1** of type **satv** with input nodes **n2** (+ve) and **n1** (-ve) and output nodes **n3** (+ve) and ground (-ve). The voltage below which negative saturation occurs is declared explicitly in the instantiation line.

```
.model satur modfas vmax=3.5
...
ys2 satv n1 n2 n3 0 model: satur
```

Specifies a voltage limiter **ys2** of type **satv** having input nodes **n1** (+ve) and **n2** (-ve) with output nodes **n3** (+ve) and ground (-ve). The voltage above which positive saturation occurs is declared in the **.MODEL** command.

## Model Equations

**Figure 6-14. Voltage Limiter Model Characteristics**



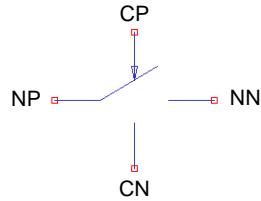
For  $V(\text{inp},\text{inn}) > \text{VSATP} \Rightarrow$  positive saturation.

For  $V(\text{inp},\text{inn}) < \text{VSATN} \Rightarrow$  negative saturation.

## Voltage Controlled Switch

**YXX VSWITCH [PIN:] NP NN CP CN [PARAM: PAR=VAL {PAR=VAL}] [MODEL: MNAME]**

**Figure 6-15. Voltage Controlled Switch Macromodel**



The voltage controlled switch macromodel can be thought of as a voltage controlled resistance, which varies continuously between the ‘ON’ and ‘OFF’ resistances defined by the model parameters **RON** and **ROFF**. The resistance change can be defined to be linear or exponential using the **LEVEL** parameter.

### Model Pins

NP	Name of the input node.
NN	Name of the output node.
CP	Name of the positive controlling node.
CN	Name of the negative controlling node.

**Table 6-7. Voltage Controlled Switch Model Parameters**

Nr.	Name	Default	Units	Definition
1	<b>LEVEL</b>	1		Model level <b>LEVEL</b> =1—Linear interpolation <b>LEVEL</b> =2—Exponential interpolation
2	<b>VON</b>	0.95	V	Control voltage for ‘ON’ state
3	<b>VOFF</b>	0.05	V	Control voltage for ‘OFF’ state
4	<b>RON</b>	$1.0 \times 10^{-2}$	$\Omega$	‘ON’ resistance
5	<b>ROFF</b>	$1.0 \times 10^{10}$	$\Omega$	‘OFF’ resistance
6	<b>M<sup>a</sup></b>	1		Device multiplier

a. **.OPTION YMFACT** must be specified for **M** to work.

- i** See “General Notes on the Use of FAS Macromodels” on page 6-2 for additional notes on using this model.

**Note**



In AC mode, the switch resistance value is equal to that computed in the DC analysis.

### Example

```
ysw1 vswitch n1 n4 n2 n3
```

Specifies a voltage controlled switch **ysw1** of type **vswitch** having input and output nodes **n1** and **n4** respectively with a positive controlling node **n2** and negative controlling node **n3**.

### Model Equations

$$VCTRL = V(cp) - V(cn)$$

$$RC1 = \ln(\sqrt{RON \times ROFF})$$

$$RC2 = \ln\left(\frac{RON}{ROFF}\right)$$

$$VC1 = \frac{(VON + VOFF)}{2}$$

$$VC2 = VOFF - VON$$

When  $VON \geq VOFF$

$$VCTRL \geq VON \Rightarrow RS = RON$$

$VCTRL \leq VOFF \Rightarrow RS = ROFF$  where **RS** is the switch resistance.

$VOFF < VCTRL < VON$

For **LEVEL=1**

$$RS = \frac{ROFF - RON}{VOFF - VON} \times VCTRL + \frac{RON \times VOFF - (ROFF \times VON)}{VOFF - VON}$$

For **LEVEL=2**

$$RS = \exp\left( RC1 - \left( \frac{3 \times RC2 \times (VCTRL - VC1)}{2 \times VC2} \right) + 2 \times \frac{RC2(VCTRL - VC1)^3}{(VC2)^3} \right)$$

When  $VON < VOFF$

$$VCTRL \leq VON \Rightarrow RS = RON$$

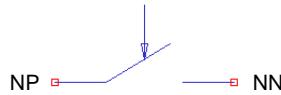
$$VCTRL \geq VOFF \Rightarrow RS = ROFF$$

$VOFF > VCTRL > VON$

## Current Controlled Switch

```
YXX CSWITCH [PIN:] NP NN IC: VNAME
+ [PARAM: PAR=VAL {PAR=VAL}] [MODEL: MNAME]
```

**Figure 6-16. Current Controlled Switch Macromodel**



The current controlled switch macromodel can be thought of as a current controlled resistance, where the resistance varies continuously between the ‘ON’ and ‘OFF’ resistances defined in the model parameters **RON** and **ROFF**. The resistance change can be defined to be linear or exponential using the **LEVEL** parameter.

### Model Pins

NP	Name of the input node.
NN	Name of the output node.
VNAME	Controlling current through voltage source.

**Table 6-8. Current Controlled Switch Model Parameters**

Nr.	Name	Default	Units	Definition
1	<b>LEVEL</b>	1		Model level <b>LEVEL</b> =1—Linear interpolation <b>LEVEL</b> =2—Exponential interpolation
2	<b>I<sub>ON</sub></b>	0.95	A	Control current for ‘ON’ state
3	<b>I<sub>OFF</sub></b>	0.05	A	Control current for ‘OFF’ state
4	<b>R<sub>ON</sub></b>	$1.0 \times 10^{-2}$	$\Omega$	‘ON’ resistance
5	<b>R<sub>OFF</sub></b>	$1.0 \times 10^{10}$	$\Omega$	‘OFF’ resistance
6	M <sup>a</sup>	1		Device multiplier

a. **.OPTION YMFACT** must be specified for **M** to work.



See “[General Notes on the Use of FAS Macromodels](#)” on page 6-2 for additional notes on using this model.

---

### Note

In AC mode, the switch resistance value is equal to that computed in the DC analysis.

## Example

```
ysw1 cswitch n1 n2 ic: v1
```

Specifies a current controlled switch **y**sw1 of type **cswitch** having input and output nodes n1 and n2 respectively. The switch is controlled by the current through the voltage source v1.

## Model Equations

$$ICTRL = I(vname)$$

$$RC1 = \ln(\sqrt{RON \times ROFF})$$

$$RC2 = \ln\left(\frac{RON}{ROFF}\right)$$

$$IC1 = \frac{(ION + IOFF)}{2}$$

$$IC2 = IOFF - ION$$

When  $ION \geq IOFF$

$$ICTRL \geq ION \Rightarrow RS = RON$$

$ICTRL \leq IOFF \Rightarrow RS = ROFF$  where **RS** is the switch resistance.

$IOFF < ICTRL < ION$

For **LEVEL=1**

$$RS = \frac{ROFF - RON}{IOFF - ION} \times ICTRL + \frac{RON \times IOFF - (ROFF \times ION)}{IOFF - ION}$$

For **LEVEL=2**

$$RS = \exp\left(RC1 - \left(\frac{3 \times RC2 \times (ICTRL - IC1)}{2 \times IC2}\right) + 2 \times \frac{RC2(ICTRL - IC1)^3}{(IC2)^3}\right)$$

When  $ION < IOFF$

$$ICTRL \leq ION \Rightarrow RS = RON$$

$$ICTRL \geq IOFF \Rightarrow RS = ROFF$$

$IOFF > ICTRL > ION$

For **LEVEL=1**

$$RS = \frac{ROFF - RON}{IOFF - ION} \times ICTRL + \frac{RON \times IOFF - (ROFF \times ION)}{IOFF - ION}$$

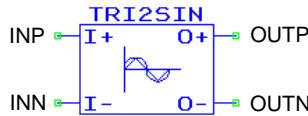
For **LEVEL=2**

$$RS = \exp\left( RC1 - \left( \frac{3 \times RC2 \times (ICTRL - IC1)}{2 \times IC2} \right) + 2 \times \frac{RC2(ICTRL - IC1)^3}{(IC2)^3} \right)$$

## Triangular to Sine Wave Converter

```
YXX TRI2SIN [PIN:] INP INN OUTP OUTN
+ [PARAM: PAR=VAL {PAR=VAL}] [MODEL: MNAME]
```

**Figure 6-17. Triangular to Sine Wave Converter Macromodel**



This analog macromodel converts a triangular wave into a sinusoidal wave.

### Model Pins

INP	Name of the positive input node.
INN	Name of the negative input node.
OUTP	Name of the positive output node.
OUTN	Name of the negative output node.

**Table 6-9. Triangular to Sine Wave Converter Model Parameters**

Nr.	Name	Default	Units	Definition
1	<b>GAIN</b>	1		Gain
2	<b>VOFF</b>	0	V	Offset voltage
3	<b>VU</b>	1	V	Upper limit of input voltage
4	<b>VL</b>	-1	V	Lower limit of input voltage
5	<b>LEVEL</b>	1		Model index
6	M <sup>a</sup>	1		Device multiplier

a. .OPTION YMFACT must be specified for M to work.

**i** See “General Notes on the Use of FAS Macromodels” on page 6-2 for additional notes on using this model.

---

### Examples

```
ytr1 tri2sin pin: n1 0 n2 0 param: vu=0.0 vl=-8
```

Specifies a triangular to sine wave converter **ytr1** of type **tri2sin** having input nodes **n1** (+ve) and ground (-ve) with output nodes **n2** (+ve) and ground (-ve). The upper and lower limits of the input voltage are declared explicitly in the instantiation line.

```
.model sto modfas voff=0.2
...
ytr2 tri2sin n1 n2 n3 n4 param: vu=5 model: sto
```

Specifies a triangular to sine wave converter **ytr2** of type **tri2sin** having input nodes **n1** (+ve) and **n2** (-ve) with output nodes **n3** (+ve) and **n4** (-ve). The offset voltage is declared using the **.MODEL** command.

## Model Characteristics

A mathematical transformation is used to convert the triangular input to the sine output. Input signal frequency, maximum and minimum voltage are all taken into account.

If **LEVEL**=1 is specified (default value), the model assumes that the maximum input voltage is equal to **VU** and that the minimum input voltage is equal to **VL**.

If **LEVEL**=2 is specified, the model itself calculates **VU** and **VL** at runtime.

$$VU = \max(V(inp,inn))$$

$$VL = \min(V(inp,inn))$$

$$V(outp,outn) = VOFF + VS + GAIN \times VM \times \sin(K)$$

where

$$K = (V(inp,inn) - VS) \times \frac{\pi}{VM/2}$$

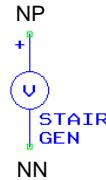
$$VM = \frac{(VU - VL)}{2}$$

$$VS = \frac{(VU + VL)}{2}$$

## Staircase Waveform Generator

**YXX STAIRGEN [PIN:] NP NN [PARAM: PAR=VAL {PAR=VAL}] [MODEL: MNAME]**

**Figure 6-18. Staircase Waveform Generator Macromodel**



A staircase waveform generator macromodel.

### Model Pins

NP Name of the positive input nodes.

NN Name of the negative input nodes.

**Table 6-10. Staircase Waveform Generator Model Parameters**

Nr.	Name	Default	Units	Definition
1	<b>VSTART</b>	0	V	Start voltage
2	<b>VDELTA</b>	0.1	V	Step voltage
3	<b>NSTEP</b>	10		Number of voltage steps
4	<b>TDU</b>	$1.0 \times 10^{-4}$	s	Period duration time
5	<b>SLR</b>	1	V/ $\mu$ s	Slew rate for falling and rising edges
6	<b>M<sup>a</sup></b>	1		Device multiplier

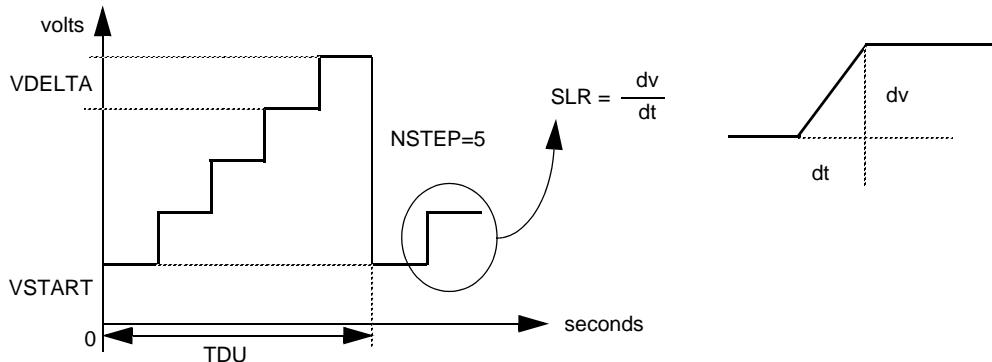
a. .OPTION YMFACT must be specified for **M** to work.



See “General Notes on the Use of FAS Macromodels” on page 6-2 for additional notes on using this model.

---

**Figure 6-19. Staircase Waveform Generator Model Characteristics**



**Note**

The parameter set has to be consistent with the following restriction on Slew Rate:

$$SLR \geq 200 \times (NSTEP - 1) \times \frac{VDELTA}{TDU}$$

**Examples**

```
ytr1 stairgen pin: n1 n2 param: vstart=0.0 nstep=8
```

Specifies a staircase voltage generator **ytr1** of type **stairgen** having input node **n1** with output node **n2**. The start and step voltage are declared explicitly in the instantiation line.

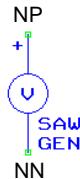
```
.model sto modfas nstep=5.0
...
ytr2 stairgen n1 n2 param: vstart=1.0 vdelta=0.2 model: sto
```

Specifies a staircase voltage generator **ytr2** of type **stairgen** having input nodes **n1** (+ve) and **n2** (-ve). The number of voltage steps is declared using the **.MODEL** command.

## Sawtooth Waveform Generator

**YXX SAWGEN [PIN:] NP NN [PARAM: PAR=VAL {PAR=VAL}] [MODEL: MNAME]**

**Figure 6-20. Sawtooth Waveform Generator Macromodel**



A sawtooth waveform generator macromodel.

### Model Pins

NP Name of the positive input node.

NN Name of the negative input node.

**Table 6-11. Sawtooth Waveform Generator Model Parameters**

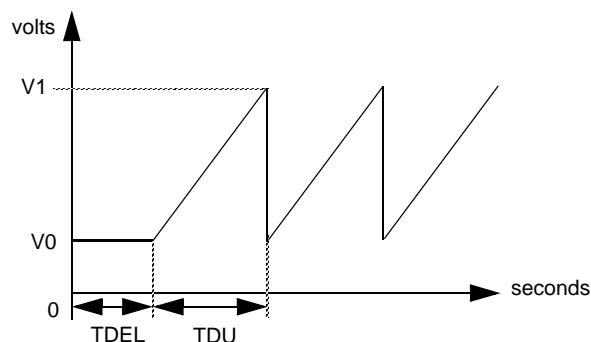
Nr.	Name	Default	Units	Definition
1	v0	0	V	Initial voltage
2	v1	5	V	Voltage magnitude
3	TDU	$1.0 \times 10^{-4}$	s	Duration of sawtooth
4	TDEL	0	s	Delay time
5	M <sup>a</sup>	1		Device multiplier

a. .OPTION YMFACT must be specified for M to work.

**i** See “General Notes on the Use of FAS Macromodels” on page 6-2 for additional notes on using this model.

---

**Figure 6-21. Sawtooth Waveform Generator Model Characteristics**



## Examples

```
ytr1 sawgen pin: n1 n2 param: tdel=0.001
```

Specifies a sawtooth waveform generator **ytr1** of type **sawgen** having input node **n1** with output node **n2**. The delay time is declared explicitly in the instantiation line.

```
.model sto modfas tdel=0.0001 v0=1.0
...
ytr2 sawgen n1 n2 param: tdel=1.0 model: sto
```

Specifies a sawtooth waveform generator **ytr2** of type **sawgen** having input node **n1** and output node **n2**. The delay time and initial voltage are declared using the **.MODEL** command. Note that the delay time declared in the instantiation line overrides the delay time parameter in the **.MODEL** command.

## Triangular Waveform Generator

**YXX TRIGEN [PIN:] NP NN [PARAM: PAR=VAL {PAR=VAL}] [MODEL: MNAME]**

**Figure 6-22. Triangular Waveform Generator Macromodel**



A triangle waveform generator macromodel.

### Model Pins

NP Name of the positive input node.

NN Name of the negative input node.

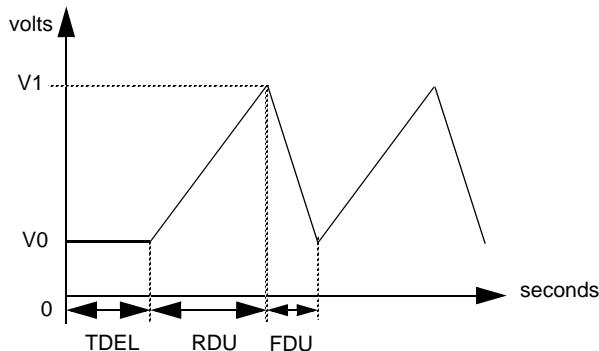
**Table 6-12. Triangular Waveform Generator Model Parameters**

Nr.	Name	Default	Units	Definition
1	v0	0	V	Initial voltage
2	v1	5	V	Voltage magnitude
3	RDU	$1.0 \times 10^{-4}$	s	Duration of first edge
4	FDU	$1.0 \times 10^{-4}$	s	Duration of second edge
5	TDEL	0	s	Delay time
6	M <sup>a</sup>	1		Device multiplier

a. .OPTION YMFACT must be specified for M to work.

- 
- i** See “General Notes on the Use of FAS Macromodels” on page 6-2 for additional notes on using this model.
-

**Figure 6-23. Triangular Waveform Generator Model Characteristics**



## Examples

```
ytr1 trigen pin: n1 n2 param: tdel=0.001
```

Specifies a sawtooth waveform generator **ytr1** of type **trigen** having input node **n1** with output node **n2**. The delay time is declared explicitly in the instantiation line.

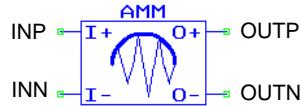
```
.model sto modfas tdel=0.0001 v0=1.0
...
ytr2 trigen n1 n2 param: rdu=1.0e-3 model: sto
```

Specifies a sawtooth waveform generator **ytr2** of type **trigen** having input node **n1** and output node **n2**. The delay time and initial voltage are declared using the **.MODEL** command.

## Amplitude Modulator

```
YXX AMM [PIN:] INP INN OUTP OUTN [PARAM: PAR=VAL {PAR=VAL}] [MODEL: MNAME]
```

**Figure 6-24. Amplitude Modulator Macromodel**



An amplitude modulator macromodel. The carrier frequency is specified with the **FC** parameter and the type of carrier signal is specified by **LEVEL** parameter (**LEVEL**=1—sinusoidal waveform, **LEVEL**=2—pulse waveform). The modulation input at lower frequencies is applied at the **inp** and **inn** terminals. The **VOFF** parameter controls the modulation depth and the final peak-to-peak voltage.

### Model Pins

INP	Name of the positive input node.
INN	Name of the negative input node.
OUTP	Name of the positive output node.
OUTN	Name of the negative output node.

**Table 6-13. Amplitude Modulator Model Parameters**

Nr.	Name	Default	Units	Definition
1	<b>LEVEL</b>	1		Type of carrier signal <b>1</b> —sinusoidal, <b>2</b> —pulse
2	<b>SLR</b>	10	V/ $\mu$ s	Slew rate
3	<b>VOFF</b>	0	V	Offset voltage
4	<b>FC</b>	$1.0 \times 10^6$	Hz	Carrier frequency
5	<b>NSAM</b>	10		Each period of the carrier signal is sampled by at least <b>NSAM</b> points
6	M <sup>a</sup>	1		Device multiplier

a. **.OPTION YMFACT** must be specified for **M** to work.

**i** See “General Notes on the Use of FAS Macromodels” on page 6-2 for additional notes on using this model.

---

### Examples

```
yal amm pin: n1 n2 n3 n4 param: voff=0.5
```

Specifies an amplitude modulator **ya1** of type **amm** having input nodes **n1** (+ve) and **n2** (-ve) with output nodes **n3** (+ve) and **n4** (-ve). The offset voltage is declared explicitly in the instantiation line.

```
.model sto modfas fc=1u
...
ya2 amm n1 n2 n3 n4 model: sto
```

Specifies an amplitude modulator **ya2** of type **amm** having input nodes **n1** (+ve) and **n2** (-ve) with output nodes **n3** (+ve) and **n4** (-ve). The carrier frequency is declared using the **.MODEL** command.

## Model Characteristics

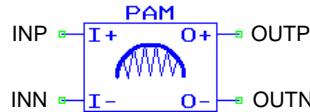
- LEVEL=1

$$V(\text{outp}, \text{outn}) = (V(\text{inp}, \text{inn}) + VOFF) \times \sin(2\pi \times \text{TIME} \times FC)$$

## Pulse Amplitude Modulator

**YXX PAM [PIN:] INP INN OUTP OUTN [PARAM: PAR=VAL {PAR=VAL}] [MODEL: MNAME]**

**Figure 6-25. Pulse Amplitude Modulator Macromodel**



A pulse amplitude modulator macromodel. The carrier frequency is specified with the **FC** parameter and the type of carrier signal is specified by the **LEVEL** parameter (**LEVEL**=1—sinusoidal waveform, **LEVEL**=2—pulse waveform). The modulation input at lower frequencies is applied at the **inp** and **inn** terminals. The modulation input can be centered about zero, or can be set exclusively positive or negative. The **VOFF** parameter controls the modulation depth by offsetting the modulation input.

### Model Pins

<b>INP</b>	Name of the positive input node.
<b>INN</b>	Name of the negative input node.
<b>OUTP</b>	Name of the positive output node.
<b>OUTN</b>	Name of the negative output node.

**Table 6-14. Pulse Amplitude Modulator Model Parameters**

Nr.	Name	Default	Units	Definition
1	<b>LEVEL</b>	1		Type of carrier signal 1—sinusoidal, 2—pulse
2	<b>SLR</b>	10	V/ $\mu$ s	Slew rate
3	<b>VOFF</b>	0	V	Offset voltage
4	<b>FC</b>	$1.0 \times 10^6$	Hz	Carrier frequency
5	<b>NSAM</b>	10		Each period of the carrier signal is sampled by at least <b>NSAM</b> points
6	<b>M<sup>a</sup></b>	1		Device multiplier

a. **.OPTION YMFACT** must be specified for **M** to work.

---

**i** See “General Notes on the Use of FAS Macromodels” on page 6-2 for additional notes on using this model.

---

## Examples

```
yp1 pam pin: n1 n2 n3 n4 param: voff=0.5
```

Specifies a pulse amplitude modulator **yp1** of type **pam** having input nodes **n1** (+ve) and **n2** (-ve) with output nodes **n3** (+ve) and **n4** (-ve). The offset voltage is declared explicitly in the instantiation line.

```
.model sto modfas fc=1u
...
yp2 pam n1 n2 n3 n4 model: sto
```

Specifies a pulse amplitude modulator **yp2** of type **pam** having input nodes **n1** (+ve) and **n2** (-ve) with output nodes **n3** (+ve) and **n4** (-ve). The carrier frequency is declared using the **.MODEL** command.

## Model Characteristics

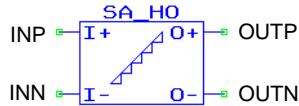
- LEVEL=1

$$V(\text{outp},\text{outn}) = \frac{V(\text{inp},\text{inn}) + VOFF}{2} \times (1 + \sin(2\pi \times \text{TIME} \times FC))$$

## Sample & Hold

```
YXX SA_HO [PIN:] INP INN OUTP OUTN
+ [PARAM: PAR=VAL {PAR=VAL}] [MODEL: MNAME]
```

**Figure 6-26. Sample & Hold Macromodel**



A Sample and Hold circuit macromodel. At each sampling point the output voltage is fixed to the level of the input voltage. After the acquisition time **TACQ** has elapsed, the output voltage is kept at the level of the input voltage for a period of  $1/\text{FS}$  until the next sampling point.

### Model Pins

INP	Name of the positive input node.
INN	Name of the negative input node.
OUTP	Name of the positive output node.
OUTN	Name of the negative output node.

**Table 6-15. Sample & Hold Model Parameters**

Nr.	Name	Default	Units	Definition
1	<b>FS</b>	$1.0 \times 10^6$	Hz	Sampling frequency
2	<b>TACQ</b>	$1.0 \times 10^{-9}$	s	Acquisition time
3	<b>DV</b>	0.02	V	Droop voltage
4	M <sup>a</sup>	1		Device multiplier

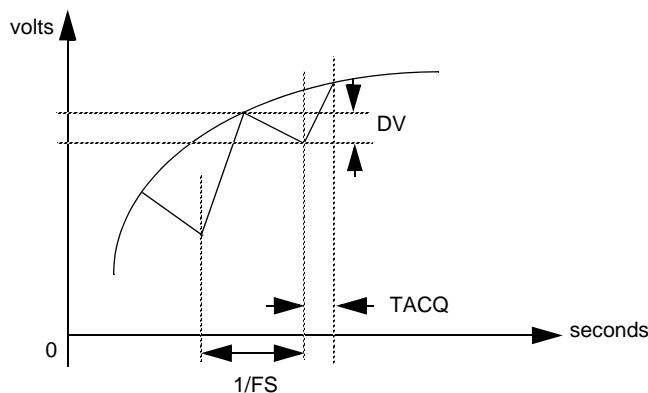
a. .OPTION YMFACT must be specified for **M** to work.



See “General Notes on the Use of FAS Macromodels” on page 6-2 for additional notes on using this model.

---

**Figure 6-27. Sample & Hold Model Characteristics**



## Examples

```
ys1 sa_ho pin: n1 n2 n3 n4 param: fs=5meg
```

Specifies a Sample and Hold macromodel **ys1** of type **sa\_ho** with input nodes **n1** (+ve) and **n2** (-ve) and output nodes **n3** (+ve) and **n4** (-ve). The sampling frequency is declared explicitly in the instantiation line.

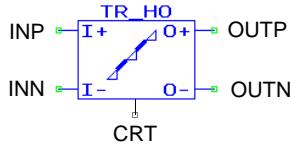
```
.model sto modfas dv=0.05
...
ys2 sa_ho n1 n2 n3 n4 model: sto
```

Specifies a Sample and Hold **ys2** of type **sa\_ho** having input nodes **n1** (+ve) and **n2** (-ve) with output nodes **n3** (+ve) and **n4** (-ve). The droop voltage is declared using the **.MODEL** command.

## Track & Hold

```
YXX TR_HO [PIN:] INP INN OUTP OUTN CRT
+ [PARAM: PAR=VAL {PAR=VAL}] [MODEL: MNAME]
```

**Figure 6-28. Track & Hold Macromodel**



A Track and Hold circuit macromodel. The model has two modes of operation depending on the logic level of the `CRT` voltage. Upon receiving the `CRT` pulse, the model swings the output voltage towards the input voltage and forces the output voltage to follow (or track) the input voltage for the remainder of the pulse. This is called the *Track Mode*. After the S/H pulse is removed, the model holds the output voltage at the value that the input voltage had at the instant of pulse deactivation. This is called the *Hold Mode*.

### Model Pins

<code>INP</code>	Name of the positive input node.
<code>INN</code>	Name of the negative input node.
<code>OUTP</code>	Name of the positive output node.
<code>OUTN</code>	Name of the negative output node.
<code>CRT</code>	Name of the controlling voltage node.

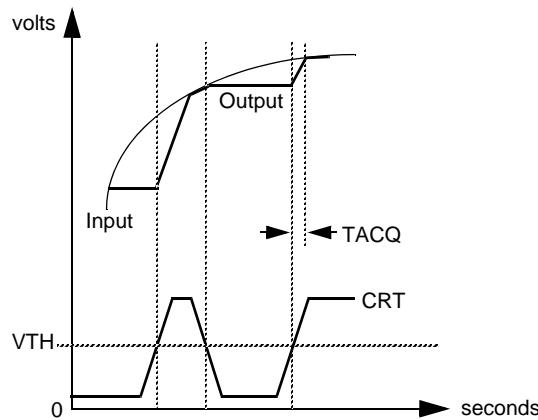
**Table 6-16. Track & Hold Model Parameters**

Nr.	Name	Default	Units	Definition
1	<code>VTH</code>	0.5	V	Threshold voltage for node <code>CRT</code>
2	<code>TACQ</code>	$1.0 \times 10^{-9}$	s	Acquisition time
3	<code>M<sup>a</sup></code>	1		Device multiplier

a. `.OPTION YMFACT` must be specified for `M` to work.

- 
- i** See “General Notes on the Use of FAS Macromodels” on page 6-2 for additional notes on using this model.
-

**Figure 6-29. Track & Hold Model Characteristics**



### Example

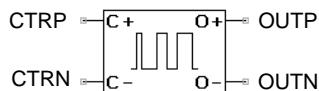
```
yt1 tr_ho pin: n1 n2 n3 n4 param: tacq=1u
```

Specifies a Track and Hold macromodel **yt1** of type **tr\_ho** having input nodes **n1** (+ve) and **n2** (-ve) with output nodes **n3** (+ve) and **n4** (-ve). The acquisition time is declared explicitly in the instantiation line.

## Pulse Width Modulator

```
YXX PWM [PIN:] CTRP CTRN OUTP OUTN
+ [PARAM: PAR=VAL {PAR=VAL}] [MODEL: MNAME]
```

**Figure 6-30. Pulse Width Modulator Macromodel**



This macromodel generates a pulse width modulated signal. The duty cycle of the pulse width modulator signal can be controlled in a specified range by the input voltage.

### Model Pins

CTRP	Name of the positive control node.
CTRN	Name of the negative control node.
OUTP	Name of the positive output node.
OUTN	Name of the negative output node.

**Table 6-17. Pulse Width Modulator Model Parameters**

Nr.	Name	Default	Units	Definition
1	DUTYMIN	0.001		Minimum duty cycle
2	DUTYMAX	0.999		Maximum duty cycle
3	CTRLMIN	0	V	Minimum control voltage
4	CTRLMAX	1	V	Maximum control voltage
5	PFREQ	$1.0 \times 10^3$	Hz	Pulse frequency
6	TR	$1.0 \times 10^{-6}$	s	Pulse rise time
7	TF	$1.0 \times 10^{-6}$	s	Pulse fall time
8	NDELAY	0		Number of delay pulse cycles
9	PVMIN	0	V	Low pulse voltage
10	PVMAX	1	V	High pulse voltage
11	M <sup>a</sup>	1		Device multiplier

a. .OPTION YMFACT must be specified for M to work.



See “General Notes on the Use of FAS Macromodels” on page 6-2 for additional notes on using this model.

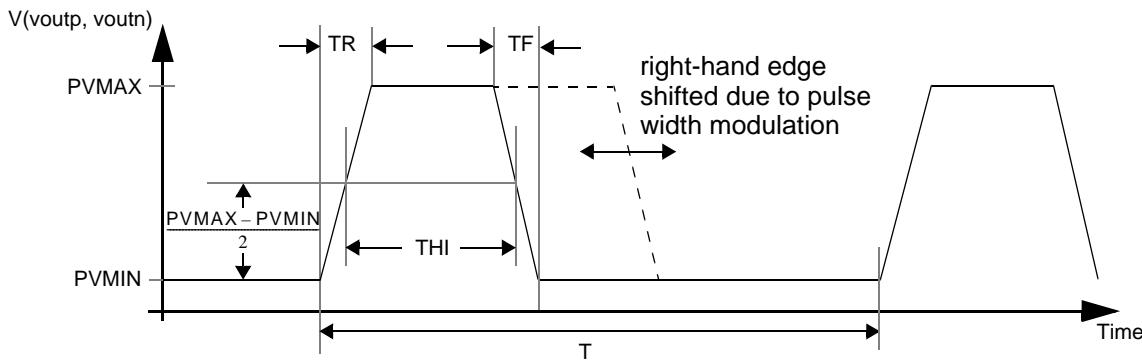
## Example

```
yp1 pwm pin: n1 0 n3 0 param: pvmax=5.0
```

Specifies a pulse width modulator macromodel **yp1** of type **pwm** having input nodes **n1** (+ve control) and ground (-ve control) with output nodes **n3** (+ve output) and ground (-ve output). The high pulse voltage is declared explicitly in the instantiation line.

## Model Characteristics

**Figure 6-31. Pulse Width Modulator Model Characteristics**



### Note



For correct operation of the macromodel, the following constraints must be considered:  
The duty cycle of the output pulse signal can only be in the following range:

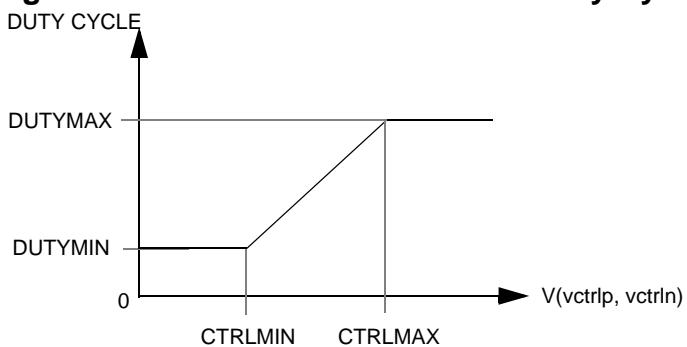
$$\text{DUTYCYCLE} = \frac{((\text{TR})/2 + (\text{TF})/2)}{\text{T}} \dots \frac{(\text{T} - (\text{TR})/2 - (\text{TF})/2)}{\text{T}}$$

with:

$$\text{T} = \frac{1}{\text{PFREQ}}$$

$$\text{DUTYCYCLE} = \frac{\text{THI}}{\text{T}}$$

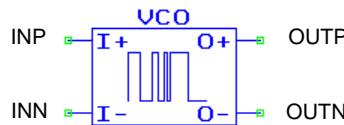
**Figure 6-32. Pulse Width Modulator Duty Cycle**



## Voltage Controlled Oscillator

```
YXX VCO [PIN:] INP INN OUTP OUTN
+ [PARAM: PAR=VAL {PAR=VAL}] [MODEL: MNAME]
```

**Figure 6-33. Voltage Controlled Oscillator Macromodel**



On the output nodes a signal is generated whose frequency is dependent on the input voltage. The **LEVEL** parameter specifies the type for the waveform, either sinusoidal or pulse.

### Model Pins

<b>INP</b>	Name of the positive input node.
<b>INN</b>	Name of the negative input node.
<b>OUTP</b>	Name of the positive output node.
<b>OUTN</b>	Name of the negative output node.

**Table 6-18. Voltage Controlled Oscillator Model Parameters**

Nr.	Name	Default	Units	Definition
1	<b>V1</b>	1	V	Output amplitude
2	<b>VOFF</b>	0	V	Offset voltage
3	<b>FMIN</b>	1	Hz	Minimum allowed frequency—only for <b>LEVEL</b> =2, 3
4	<b>FMAX</b>	$1.0 \times 10^{10}$	Hz	Maximum allowed frequency—only for <b>LEVEL</b> =2, 3
5	<b>LEVEL</b>	1		<b>LEVEL</b> =1—Continuous sinusoidal <b>LEVEL</b> =2—Sampled sinusoidal <b>LEVEL</b> =3—Sampled pulse
6	<b>A</b>	0	Hz	Polynomial parameter
7	<b>B</b>	1	Hz/V	Polynomial parameter
8	<b>C</b>	0	Hz/V <sup>2</sup>	Polynomial parameter
9	<b>D</b>	0	Hz/V <sup>3</sup>	Polynomial parameter
10	<b>M</b> <sup>a</sup>	1		Device multiplier

a. **.OPTION YMFACT** must be specified for **M** to work.

See “[General Notes on the Use of FAS Macromodels](#)” on page 6-2 for additional notes on using this model.

## Examples

```
y1 vco pin: n1 n2 n3 n4 param: voff=0.5
```

Specifies a vco model **y1** having input nodes **n1** (+ve) and **n2** (-ve), with output nodes **n3** (+ve) and **n4** (-ve). The offset voltage parameter is declared explicitly in the instantiation line.

```
.model tdk modfas FMIN=10k
...
ys4 vco n1 n2 n3 0 model: tdk
```

Specifies a vco model having input nodes **n1** (+ve) and **n2** (-ve) with output node **n3** (+ve). The **FMIN** parameter is declared using the **.MODEL** command.

## Model Equations

$$v_{in} = V(inp,inn)$$

## VCO Frequency

$$f = a + b \times v_{in} + c \times v_{in}^2 + d \times v_{in}^3$$

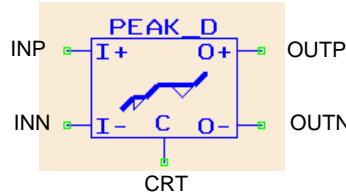
$$V(outp, outn) = voff + v1 \times \sin(2\pi \times TIME \times f)$$

- Level=1
  - The frequency of the sinusoidal output signal is determined at each internal time step.
- Level=2
  - The frequency of the sinusoidal output is determined only once for one period of the output signal.
- Level=3
  - The frequency of the pulse output is determined only once for one period of the output signal.

## Peak Detector

```
YXX PEAK_D [PIN:] INP INN OUTP OUTN CRT
+ [PARAM: PAR=VAL {PAR=VAL}] [MODEL: MNAME]
```

**Figure 6-34. Peak Detector Macromodel**



Positive or negative peak detection is selected using the **LEVEL** parameter. The peak detector tracks the input signal and hold the output at the highest (or lowest) peak found since operation of the **CRT** voltage or the **RES** parameter. The input waveform is continuously compared with the stored peak value to determine whether the stored value should be updated.

### Model Pins

INP	Name of the positive input node.
INN	Name of the negative input node.
OUTP	Name of the positive output node.
OUTN	Name of the negative output node.
CRT	Name of the control node.

**Table 6-19. Peak Detector Model Parameters**

Nr.	Name	Default	Units	Definition
1	<b>VTH</b>	1	V	Threshold voltage—signal on <b>CRT</b> pin
2	<b>RES</b>	0.5	V	If the output voltage crosses the <b>RES</b> value the output is reset to 0
3	<b>SLR</b>	1	V/ $\mu$ s	The output signal follows the input signal with the slewrate <b>SLR</b>
4	<b>RSLR</b>	1	V/ $\mu$ s	Reset to 0 with the slewrate <b>RSLR</b>
5	<b>LEVEL</b>	1		<b>LEVEL</b> =1—positive peak detector <b>LEVEL</b> =2—negative peak detector
6	<b>M</b> <sup>a</sup>	1		Device multiplier

a. .OPTION YMFACT must be specified for **M** to work.



See “[General Notes on the Use of FAS Macromodels](#)” on page 6-2 for additional notes on using this model.

---

## Examples

```
y1 peak_d pin: n1 n2 n3 n4 n5 param: level=2 res=-5.0
```

Specifies a `peak_d` model `y1` having input nodes `n1` (+ve) and `n2` (-ve) with output nodes `n3` (+ve) and `n4` (-ve) and control node `n5`. The `level` and `res` parameters are declared explicitly in the instantiation line.

```
.model tdk modfas SLR=10
...
ys4 peak_d n1 n2 n3 0 0 model:tdk
```

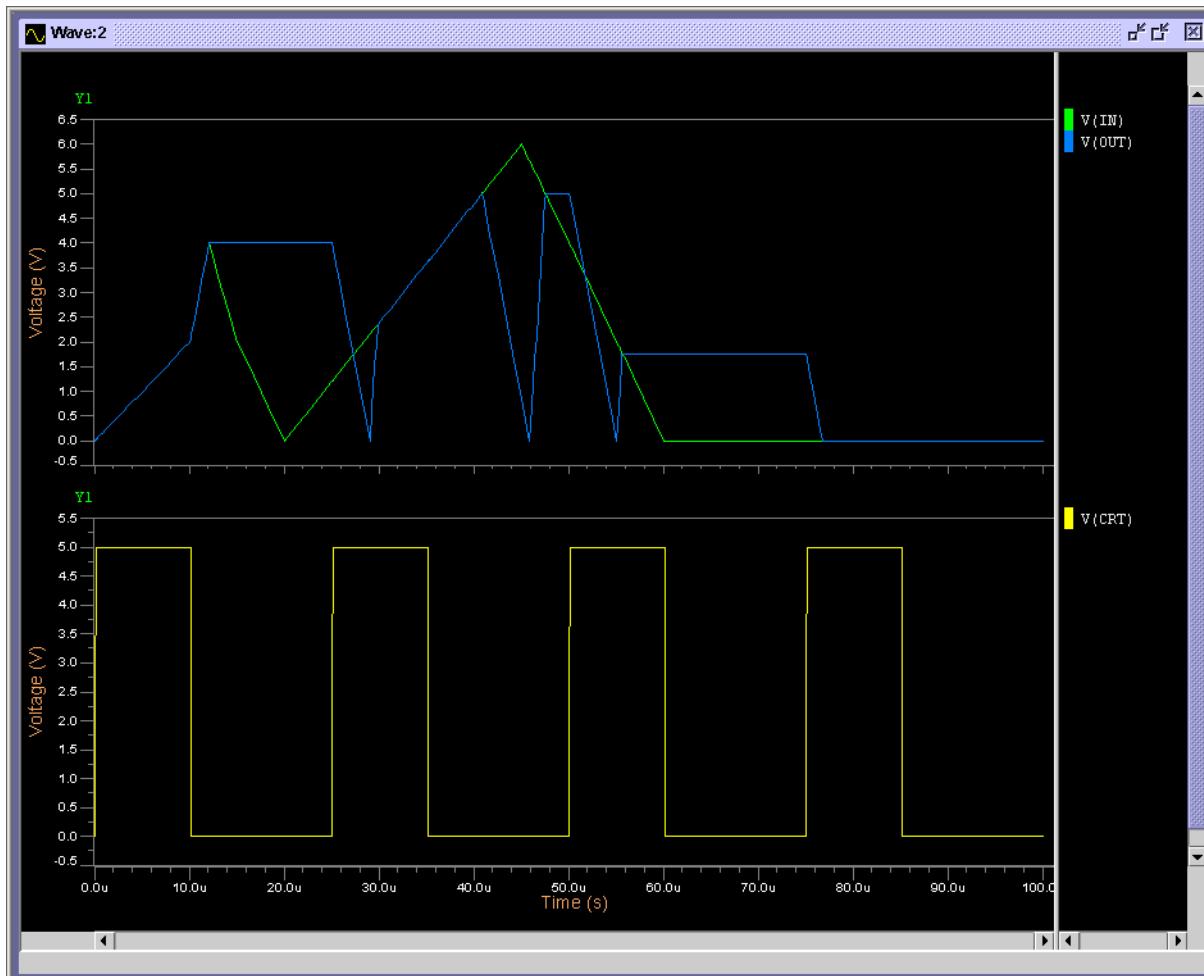
Specifies a `peak_d` model having input nodes `n1` (+ve) and `n2` (-ve) with output node `n3` (+ve). The `SLR` parameter is declared using the `.MODEL` command.

```
.MODEL TDK_PEAK_D MODFAS VTH=1 res=4
Vin in 0 pwl 0 0 10u 2 12u 4 15u 2 20u 0 45u 6 60u 0
Vcrt crt 0 pulse 0 5 0 .1u .1u 10u 25u
*Vcrt crt 0 5
Y26 PEAK_D PIN: IN 0 OUT 0 CRT PARAM: RES=5 SLR=3.0 RSLR=1.0
LEVEL=1.0 MODEL: TDK_PEAK_D
rout out 0 1k
.tran 1u 100u
.plot tran v(in)
.plot tran v(crt)
.plot tran v(out)
.end
```

In this example—see also simulation results over page—the peak detector tracks the input signal `V(IN)` and holds the output `V(OUT)` at the highest value of 4V (seen at 12μs). The rising voltage on the control node `V(CRT)` at 25μs causes the `V(OUT)` voltage to fall by 1V/μs. The output then tracks the input (from 29μs) until it reaches the `RES` value (5V) at 40.8μs and falls back to 0.

At 46 μs `V(OUT)` follows `V(IN)` until 50μs has passed, whereupon the input at `V(CRT)` causes the `V(OUT)` signal to return to 0. Finally, at 55μs, `V(OUT)` stays at the lowest `V(IN)` peak until an impulse on `V(CRT)` at 75μs causes the signal to fall to 0.

Figure 6-35. Peak Detector Simulation Results



## Level Detector

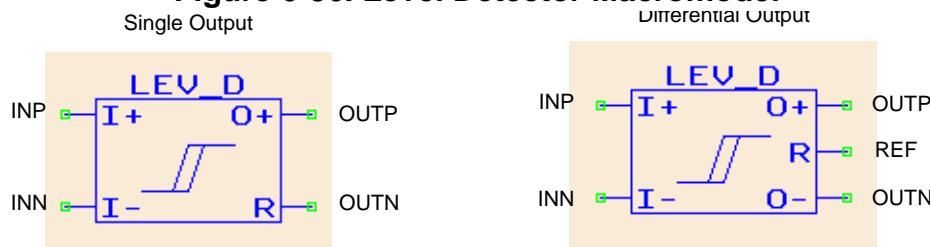
### Single Output

```
YXX LEV_D [PIN:] INP INN OUTP OUTN
+ [PARAM: PAR=VAL {PAR=VAL}] [MODEL: MNAME]
```

### Differential Output

```
YXX LEV_D [PIN:] INP INN OUTP OUTN REF
+ [PARAM: PAR=VAL {PAR=VAL}] [MODEL: MNAME]
```

**Figure 6-36. Level Detector Macromodel**



The model is used to convert analog signals into bi-level signals. This is done by comparing an input signals with a reference value. Depending on the way that the parameters and number of nodes are chosen, the model works as an inverting or non-inverting zero-crossing or level detector with single or differential output and with or without hysteresis.

### Model Pins

INP	Name of the positive input node.
INN	Name of the negative input node.
OUTP	Name of the positive output node.
OUTN	Name of the negative output node.
REF	Name of the reference node. Only used for differential output.

**Table 6-20. Level Detector Model Parameters**

Nr.	Name	Default	Units	Definition
1	TR	1	μs	Time for the output signal switching from <b>v0</b> to <b>v1</b>
2	TF	1	μs	Time for the output signal switching form <b>v1</b> to <b>v0</b>
3	TPD	0	s	Transit time through the comparator
4	v0	0	V	Lower voltage level
5	v1	1	V	Higher voltage level
6	VOFF	0	V	Is added to the potential at the node <b>INN</b>

**Table 6-20. Level Detector Model Parameters**

Nr.	Name	Default	Units	Definition
7	<b>VRL</b>	-0.1	V	Lower reference voltage
8	<b>VRU</b>	0.1	V	Higher reference voltage
9	M <sup>a</sup>	1		Device multiplier

a. **.OPTION YMFACT** must be specified for **M** to work.



See “[General Notes on the Use of FAS Macromodels](#)” on page 6-2 for additional notes on using this model.

## Examples

```
y1 lev_d pin: n1 n2 n3 0 param: v1=5 vrl=2.4 vru=2.6
```

Specifies a `lev_d` model `y1` with single output having input nodes `n1` (+ve) and `n2` (-ve) with output nodes `n3` (+ve). Parameters `v1`, `vrl` and `vru` are declared explicitly in the instantiation line.

```
.model tdk modfas vrl=0.0 vru=0.0
...
y4 lev_d n1 n2 n3 n4 n5 model:tdk
```

Specifies a `lev_d` model with differential output having input nodes `n1` (+ve) and `n2` (-ve), with non-inverting output node `n3` and inverting output node `n4`. As reference for the output node `n5` is used. The `vrl` and `vru` parameters are declared via the **.MODEL** command. The above model works as a differential zero-crossing detector without hysteresis.

## Model Equations

If a rising input voltage  $V_{IN} = V(INP) - V(INN) - VOFF$  passes the lower reference voltage, `VRL`, the output signal switches from `V0` to `V1` during the time `TR`.

If a falling input voltage  $V_{IN} = V(INP) - V(INN) - VOFF$  passes the upper reference voltage, `VRU`, the output signal switches from `V1` to `V0` during the time `TF`.

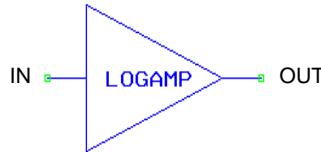
The 4 node model works as a non-inverting comparator, the output signal being applied across `OUTP` and `OUTN`. The 5 node model works as a differential comparator. The non-inverting output signal is applied across `OUTP` and `REF`, and the inverting output signal is applied across `OUTN` and `REF`.

If the parameters `VRL` and `VRU` have the same value the model operates as zero-crossing detector.

## Logarithmic Amplifier

```
YXX LOGAMP [PIN:] IN OUT [PARAM: PAR=VAL {PAR=VAL}] [MODEL: MNAME]
```

**Figure 6-37. Logarithmic Amplifier Macromodel**



The model provides a logarithmic transfer function between the input and the output node. The signal on the output node is limited by user defined values.

### Model Pins

IN	Name of the input node.
OUT	Name of the output node.

**Table 6-21. Logarithmic Amplifier Model Parameters**

Nr.	Name	Default	Units	Definition
1	K	1		Gain
2	E	1	V	Influences the argument of the log function
3	BASE	e—natural logarithm		Base of the logarithm
4	VMAX	5	V	Maximum output voltage
5	VMIN	-5	V	Minimum output voltage
6	VSATP	4.75	V	Positive saturation voltage
7	VSATN	-4.75	V	Negative saturation voltage
8	PSLOPE	0.25		Slope of Transfer Function at VSATP
9	NSLOPE	0.25		Slope of Transfer Function at VSATN
10	M <sup>a</sup>	1		Device multiplier

a. .OPTION YMFACT must be specified for M to work.

### Examples

```
y1 logamp pin: n1 n2 param: K=100.0
```

Specifies a logamp model y1 having input node n1 with output node n2. The gain parameter is declared explicitly in the instantiation line.

```
.model tdk modfas VMAX=10 VMIN=-10 VSATP=9.9 VSATN=-9.9
...
ys4 logamp n1 n2 model:tdk
```

Specifies a logamp model having input node n1 with output node n2. Parameters **VMAX**, **VMIN**, **VSATP** and **VSATN** are declared using the **.MODEL** command.

## Model Equations

$$v_i = v(\text{IN})$$

if:

$$v_i \leq 1.0 \times 10^{-10}$$

then:

$$v_i = 1.0 \times 10^{-10}$$

$$v(\text{OUT}) = \text{LIMITER} \left\{ -K \times \frac{1}{(\log \text{BASE})} \times \log \left( \frac{v_i}{E} \right) \right\}$$

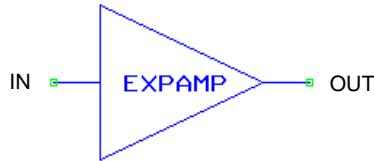


Information on voltage limiting is given in the “[Voltage Limiter](#)” on page 6-17. Non-limited voltages can be specified using `v(yname->lin)`.

## Anti-logarithmic Amplifier

**YXX EXPAMP [PIN:] IN OUT [PARAM: PAR=VAL {PAR=VAL}] [MODEL: MNAME]**

**Figure 6-38. Anti-logarithmic Amplifier Macromodel**



The model provides an anti-logarithmic transfer function between the input and the output node. The signal on the output node is limited by user defined values.

### Model Pins

**IN** Name of the input node.

**OUT** Name of the output node.

**Table 6-22. Anti-logarithmic Amplifier Model Parameters**

Nr.	Name	Default	Units	Definition
1	<b>K</b>	1		Gain
2	<b>E</b>	1	V	Influences the argument of the power function
3	<b>BASE</b>	e exponential		Base of the power function
4	<b>VMAX</b>	5	V	Maximum output voltage
5	<b>VMIN</b>	-5	V	Minimum output voltage
6	<b>VSATP</b>	4.75	V	Positive saturation voltage
7	<b>VSATN</b>	-4.75	V	Negative saturation voltage
8	<b>PSLOPE</b>	0.25		Slope of Transfer Function at <b>VSATP</b>
9	<b>NSLOPE</b>	0.25		Slope of Transfer Function at <b>VSATN</b>
10	<b>M<sup>a</sup></b>	1		Device multiplier

a. **.OPTION YMFACT** must be specified for **M** to work.



See “General Notes on the Use of FAS Macromodels” on page 6-2 for additional notes on using this model.

---

## Examples

```
y1 expamp pin: n1 n2 param: K=100.0
```

Specifies an `expamp` model `y1` having input node `n1` with output node `n2`. The gain parameter `K` is declared explicitly in the instantiation line.

```
.model tdk modfas VMAX=10 VMIN=-10 VSATP=9.9 VSATN=-9.9
...
ys4 expamp n1 n2 model:tdk
```

Specifies an `expamp` model with input node `n1` and output node `n2`. Parameters `VMAX`, `VMIN`, `VSATP` and `VSATN` are declared using the `.MODEL` command.

## Model Equations

$$v(OUT) = \text{LIMITER}\{-K \times \text{BASE}^{v(IN)/E}\}$$



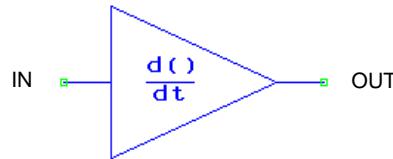
Information on voltage limiting is given in the “[Voltage Limiter](#)” on page 6-17. Non-limited voltages can be specified using `v(yname->lin)`.

---

## Differentiator

```
YXX DIFF [PIN:] IN OUT [PARAM: PAR=VAL {PAR=VAL}] [MODEL: MNAME]
```

**Figure 6-39. Differentiator Macromodel**



The model provides the differentiated input signal at the output node.

### Model Pins

IN Name of the input node.

OUT Name of the output node.

**Table 6-23. Differentiator Model Parameters**

Nr.	Name	Default	Units	Definition
1	K	1	V	Time constant
2	C0	1	V	DC value
3	SLR	$1.0 \times 10^9$	V/s	Limits the slewrate of the signal on the OUT node
4	M <sup>a</sup>	1		Device multiplier

a. .OPTION YMFACT must be specified for M to work.

---

**i** See “General Notes on the Use of FAS Macromodels” on page 6-2 for additional notes on using this model.

---

### Examples

```
y1 diff pin: n1 n2 param: K=100.0
```

Specifies a diff model y1 having input node n1 with output node n2. The K parameter is declared explicitly in the instantiation line.

```
.model tdk modfas C0=-1.0
ys4 diff n1 n2 model:tdk
```

Specifies a diff model having input node n1 with output node n2. The C0 parameter is declared using the .MODEL command.

## Model Equations

### DC Analysis

$$v(OUT) = C_0$$

### Transient Analysis

$$v(OUT) = -K \times \frac{d}{dt}v(IN)$$

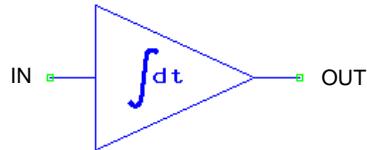
### Frequency Analysis

$$\frac{v(OUT)}{v(IN)} = -j\omega K$$

## Integrator

```
YXX INTEG [PIN:] IN OUT [PARAM: PAR=VAL {PAR=VAL}] [MODEL: MNAME]
```

**Figure 6-40. Integrator Macromodel**



This model provides the integrated input signal at the output node.

### Model Pins

IN Name of the input node.

OUT Name of the output node.

**Table 6-24. Integrator Model Parameters**

Nr.	Name	Default	Units	Definition
1	K	1	V	Time constant
2	C0	1	V	DC value
3	M <sup>a</sup>	1		Device multiplier

a. .OPTION YMFACT must be specified for M to work.

**i** See “General Notes on the Use of FAS Macromodels” on page 6-2 for additional notes on using this model.

---

### Examples

```
y1 integ pin: n1 n2 param: K=100.0
```

Specifies an integ model y1 having input node n1 with output node n2. The K parameter is declared explicitly in the instantiation line.

```
.model tdk modfas C0=-1.0
ys4 integ n1 n2 model:tdk
```

Specifies an integ model having input node n1 with output node n2. The C0 parameter is declared using the .MODEL command.

## Model Equations

### DC Analysis

$$v(OUT) = C_0$$

### Transient Analysis

$$v(OUT) = \frac{-1}{K} \times \int v(IN) dt + C_0$$

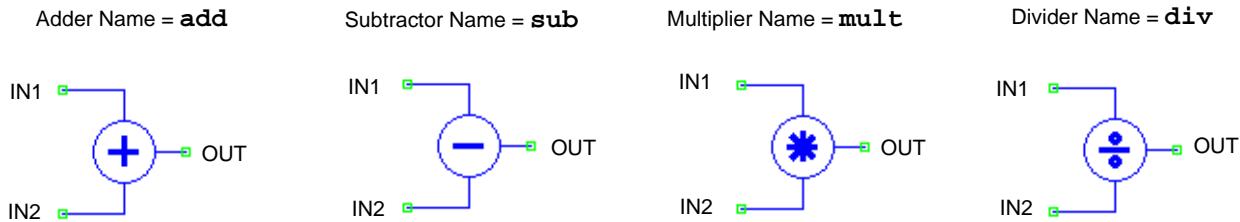
### Frequency Analysis

$$\frac{v(OUT)}{v(IN)} = \frac{-1}{j\omega K}$$

## Adder, Subtractor, Multiplier & Divider

**YXX NAME [PIN:] IN1 IN2 OUT [PARAM: PAR=VAL {PAR=VAL}] [MODEL: MNAME]**

**Figure 6-41. Adder, Subtractor, Multiplier & Divider Macromodels**



This model provides the specified arithmetic operation of the input signals at the output node.

### NAME Parameter

- ADD** Specifies the Adder model.
- SUB** Specifies the Subtractor model.
- MULT** Specifies the Multiplier model.
- DIV** Specifies the Divider model.

### Model Pins

- IN1** Name of the first input node.
- IN2** Name of the second input node.
- OUT** Name of the output node.

**Table 6-25. Adder, Subtractor, Multiplier & Divider Model Parameters**

Nr.	Name	Default	Units	Definition
1	<b>VMAX</b>	5	V	Maximum output voltage
2	<b>VMIN</b>	-5	V	Minimum output voltage
3	<b>VSATP</b>	4.75	V	Positive saturation voltage
4	<b>VSATN</b>	-4.75	V	Negative saturation voltage
5	<b>PSLOPE</b>	1.0		Slope of Transfer Function at <b>VSATP</b>
6	<b>NSLOPE</b>	1.0		Slope of Transfer Function at <b>VSATN</b>
7	<b>M<sup>a</sup></b>	1		Device multiplier

a. .OPTION YMFACT must be specified for **M** to work.



See “[General Notes on the Use of FAS Macromodels](#)” on page 6-2 for additional notes on using this model.

---

## Examples

```
y1 add pin: n1 n2 out param: VMAX=100.0
```

Specifies an adder model **y1** with input nodes **n1** and **n2**, and output node **out**. The **VMAX** parameter is declared explicitly in the instantiation line.

```
.model tdk modfas VMAX=10 VMIN=-10 VSATP=9.9 VSATN=-9.9
...
ys4 mult n1 n2 out model:tdk
```

Specifies a Multiplier model with input nodes **n1** and **n2**, and output node **out**. The **VMAX**, **VMIN**, **VSATP** and **VSATN** parameters are declared using the **.MODEL** command.

## Model Equations

$$v(OUT) = \text{LIMITER}\{v(IN1)[+|-|*|/]v(IN2)\}$$



Information on voltage limiting is given in the “[Voltage Limiter](#)” on page 6-17.

---



# Chapter 7

## Digital Macromodels

---

### Eldo Digital Macromodels

The following digital macromodels are provided in Eldo:

Delay	<b>DEL</b>
Inverter	<b>INV</b>
NAND gate with double, triple, or multiple inputs	<b>NAND</b>
AND gate with double, triple, or multiple inputs	<b>AND</b>
NOR gate with double, triple, or multiple inputs	<b>NOR</b>
OR gate with double, triple, or multiple inputs	<b>OR</b>
XOR gate with double inputs	<b>XOR</b>



For ADC/DAC Mixed Signal Macromodels see [page 7-13](#).

---

Digital macromodels are implemented in Eldo as time variable voltage sources whose output values are computed at execution time.

These macromodels are parameterized with threshold voltages, speed and so forth. For digital gates, parameters can be specified through a **.MODEL** command as is the case for semiconductor devices. Values specified in the macromodel instantiation line supersede values specified in the **.MODEL** command.



For more information on the **.MODEL** command, see the **.MODEL . . . LOGIC** description on [page 7-3](#). The older command **.MODLOGIC** is still supported, but it should no longer be used.

---

The option **DYND2ALOG** can be used to dynamically calculate the threshold values **VTHI** and **VTLO** for digital macromodels using actual values of the high and low bias. The option works by taking the values from two extra pins defined by the user in the macromodel instantiation line. The value specified on the first pin provides the high voltage digital output value (**VHI**) and the value specified on the second pin provides the low voltage digital output value (**VLO**). The active threshold values **VTHI** and **VTLO**, will be computed dynamically using the following equations:

```
VTHI_ACTIVE = VLO + VTHI*DV
VTLO_ACTIVE = VLO + VTLO*DV
```

VTHI and VTLO are the values specified in the **.MODEL** command defining the digital macromodel or in the macromodel instance, VLO is the low output voltage value given on the second extra pin defined by the user, and DV is the voltage difference given by VHI - VLO.



For more information on the **DYND2ALOG** option see [page 11-58](#).

---

## Digital Model Definition

```
.MODEL MNAME LOGIC [VHI=VAL] [VLO=VAL] [VTH=VAL]
+ [VTI=VAL] [VTLO=VAL] [TPD=VAL] [TPDUP=VAL]
+ [TPDOWN=VAL] [CIN=VAL] [DRV1=VAL] [DRVH=VAL]
```

Used for the definition of digital gate models.

### Parameters

MNAME	Name of the model.
LOGIC	Defines the model as a digital gate model.
VHI=VAL	Output voltage for the 1 logical state. Default value is 5V.
VLO=VAL	Output voltage for the 0 logical state. Default value is 0V.
VTH=VAL	Threshold input voltage. Default value is 2.5V.
VTI=VAL	Input threshold voltage for the rising edge.
VTLO=VAL	Input threshold voltage for the falling edge.

#### Note



If only **VTH** is specified, then **VTHI=VTLO=VTH**. If both **VTI** and **VTLO** are specified, **VTH** will be ignored. Ensure that the DC operating point is either above **VTI** or below **VTLO**, otherwise the output may behave unpredictably in the first few cycles of simulation.

**VTI** and **VTLO** can be computed dynamically using the option **DYND2ALOG**. See [page 7-1](#) for more details.

**TPD=VAL** Transit time through the gate (time from input threshold intersection to output threshold intersection). Default value is 1ns.

**TPDUP=VAL** Transit time (See above) for output to reach **VTHI** volts (a value causing a change in the next gate).

**TPDOWN=VAL** Transit time (See above) for output to reach **VTLO** volts (a value causing a change in the next gate).

#### Note



If only **TPD** is specified then **TPDUP=TPDOWN=TPD**. If both **TPDUP** and **TPDOWN** are specified, **TPD** will be ignored. The slopes at the output are determined by the values of **TPD**, **TPDUP** and **TPDOWN** (See the figure below).

The above definitions of transit times and threshold voltages make sure that transit times are additive through a chain of digital operators. It is not realistic, however, to model an Eldo digital gate as a chain of single gates, as the transition at the output of such a gate would not occur immediately.

- CIN=VAL** Capacitance seen at one of the macro inputs, modeling the interconnection capacitance. All inputs are loaded with the same capacitance. **CIN** has no effect when the macros are linked in a chain, as the input of one gate is the output of another, which is modeled as a voltage source. Default value is zero.
- DRV<sub>L</sub>=VAL** Drive resistance for the logic 0 state.
- DRV<sub>H</sub>=VAL** Drive resistance for the logic 1 state.

---

**Note**



These drive resistances are only relevant when current source modeling of Eldo logic primitive outputs is used, as selected using **.OPTION ULOGIC**. By default, if **.OPTION ULOGIC** is not specified, a simple voltage source is used to model logic primitive outputs.

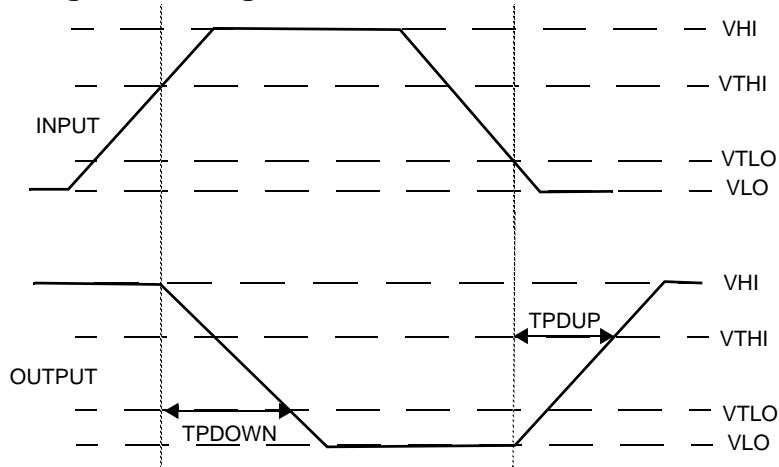
---



See the **.OPTION ULOGIC** command on [page 11-30](#).

---

**Figure 7-1. Digital Model Parameter Thresholds**



**Example**

```
.model nand_1 logic vlo=5 vhi=-5 vth=0
...
nand#1 n1 n2 o1 nand_1 tpd=2.5n cin=0.5p
```

Specifies the two input nand gate **nand#1** of model type **nand\_1**. The parameters of the gate are described using the **.MODEL ... LOGIC** command and indicate that the voltages used for the logical states 0 (**vlo**) and 1 (**vhi**) are 5V and -5V respectively and that the threshold input voltage (**vth**) is set to zero. The time for the output to reach **vth** is 2.5ns and the input capacitance is 0.5pF.

The example below shows how the **DYND2ALOG** option is used to compute dynamic values for VHI and VLO.

```

*DYND2ALOG example
.option dynd2alog
.SUBCKT A2_020 A A0 A1 BIAS VN VP
    AND2UX01 A0 A1 A vp vn 0020
    .MODEL 0020 LOGIC
*   + VHI      = 5
*   + VLO      = 0
*   + VTHI     = (2/3)
*   + VTLO     = (1/3)
*   + TPDUP    = 1ns
*   + TPDOWN   = 1ns
*   + CIN      = 0.05pF
.ENDS A2_020

vn vn 0 1
vp vp 0 3
va a 0 pwl ( 0 0 10n 0 15n 5 )
vb b 0 pwl ( 0 0 5n 5 20n 5 25n 0 )
x1 out a b 0 vn vp a2_020
x2 out2 out a 0 vn vp a2_020
.tran 1n 100n
.plot tran v(a)
.plot tran v(b)
.plot tran v(out)
.plot tran v(out2)
.extract tran yval(v(out),20n)
.extract tran yval(v(out),10n)
.end

```

A 2-input AND gate is instantiated using the model defined in the **.MODEL** statement. In the instantiation line two extra pins are defined as **VP** and **VN**. These are used by the option **DYND2ALOG** which has been defined at the start of the netlist. The difference between calculating the values for **VHI** and **VLO** dynamically and declaring them in the **.MODEL** statement can be seen using the two **.EXTRACT** commands defined in the netlist. These statements will extract the values on the output waveform **v(out)** for x-axis values 20ns and 10ns. The x-axis values define the time when the output voltage is at its high and low states, therefore correspond to **VHI** and **VLO**.

Running the simulation with the **DYND2ALOG** option specified and including the two extra pins will produce the values 3V and 1V for **VHI** and **VLO** respectively. Removing the option and the pins and including the **VHI** and **VLO** parameters in the **.MODEL** statement will produce the values specified on these parameters, in this case they are 5V and 0V.

## Delay

**DELxx IN OUT VAL**

**Figure 7-2. Delay Macromodel**



This macromodel describes an ideal delay element that transfers its input voltage to its output after a specified time delay, where the reference node is ground. The input impedance is infinite.

### Parameters

IN	Name of the input node.
OUT	Name of the output node.
xx	Delay element identifier (ASCII string).
VAL	Value of the delay in seconds.

### Example

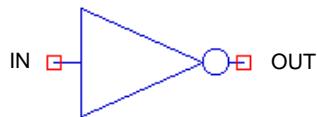
**del1 a1 a2 2.0e-9**

Specifies a delay element placed between nodes `a1` and `a2`, with a delay of 2ns.

## Inverter

**INVxx IN OUT [REF1 REF2] [MNAME] [PAR=VAL]**

**Figure 7-3. Inverter Macromodel**



### Parameters

IN	Name of the input node.
OUT	Name of the output node.
xx	Inverter element identifier (ASCII string).
REF1 REF2	Only to be used when specifying <b>.OPTION DYND2ALOG</b> . Names of the pins used in the dynamic calculation of the threshold values. See <a href="#">page 7-1</a> for more details.
MNAME	Name of a model described with the <b>.MODEL</b> command.
PAR=VAL	A direct assignment of a <b>.MODEL</b> command parameter.

### Example

```
.model inv logic vhi=5 vlo=-5 vthi=1.0 vtlo=1.0
+ tpd=2.5n cin=0.5p
...
inv44 i1 o1 inv
```

Specifies the inverter **inv44** of model type **inv** placed between the nodes **i1** and **o1**. The parameters of the inverter are specified using the **.MODEL** command.

```
inv44 i1 o1 vhi=5 vlo=-5 vthi=1.0 vtlo=1.0
+ tpd=2.5n cin=0.5p
```

Specifies the same inverter as above but with its parameters assigned directly instead of using the **.MODEL** command.



For more information, refer to the “**.MODEL**” on page 10-160.

## Exclusive-OR Gate

**XORxx IN1 IN2 OUT [REF1 REF2] [MNAME] [PAR=VAL]**

### Parameters

IN1, IN2 Names of the input nodes.  
OUT Name of the output node.  
xx Exclusive-OR element identifier (ASCII string).  
REF1 REF2 Only to be used when specifying **.OPTION DYND2ALOG**. Names of the pins used in the dynamic calculation of the threshold values. See [page 7-1](#) for more details.  
MNAME Name of a model described with the **.MODEL** command.  
PAR=VAL A direct assignment of a **.MODEL** command parameter.

### Example

```
.model xor logic vhi=5 vlo=-5 vthi=1.0 vtlo=1.0
+ tpd=2.5n cin=0.5p
...
xor44 i1 i2 o1 xor
```

Specifies the exclusive-OR gate **xor44** of model type **xor** placed between the nodes **i1**, **i2** and **o1**. The parameters of the exclusive-OR are specified using the **.MODEL** command.

```
xor44 i1 i2 o1 vhi=5 vlo=-5 vthi=1.0 vtlo=1.0
+ tpd=2.5n cin=0.5p
```

Specifies the same exclusive-OR as above but with its parameters assigned directly instead of using the **.MODEL** command.



For more information, refer to the “**.MODEL**” on page 10-160.

---

## 2-Input Digital Gates

<dgate><xx> IN1 IN2 OUT [REF1 REF2] [MNAME] [PAR=VAL]

### Parameters

dgate      Digital gate type

**Table 7-1. 2-Input Digital Gate Types**

Gate Type	Function
NAND	NAND gate
AND	AND gate
NOR	NOR gate
OR	OR gate

xx      Digital gate identifier (ASCII string).

#### Note



The first ASCII character of the gate identifier (xx) must not be a 3 since this would indicate a triple-input gate.

IN1, IN2    Names of the input nodes.

OUT        Name of the output node.

REF1 REF2    Only to be used when specifying **.OPTION DYND2ALOG**. Names of the pins used in the dynamic calculation of the threshold values. See [page 7-1](#) for more details.

MNAME        Name of a model described with the **.MODEL** command.

PAR=VAL      A direct assignment of a **.MODEL** command parameter.

### Examples

```
* .MODEL definition
.model nand_1 logic vhi=5 vlo=-5 vth=0 tpd=2.5n cin=0.5p
...
nand4 n1 n2 o1 nand_1
```

Specifies a two input NAND gate **nand4** of model type **nand\_1** with input nodes **n1** and **n2** and output node **o1**. The model parameters of the NAND gate are described using the **.MODEL** command.

```
nand4 n1 n2 o1 vhi=5 vlo=-5 vth=0 tpd=2.5n cin=0.5p
```

Specifies the same NAND gate as above but with its parameters assigned directly instead of with the **.MODEL** command.



For more information, refer to the “[.MODEL](#)” on page 10-160.

## 3-Input Digital Gates

DGATE`xx` IN1 IN2 IN3 OUT [REF1 REF2] [MNAME] [PAR=VAL]

### Parameters

DGATE      Digital gate type

**Table 7-2. 3-Input Digital Gate Types**

Gate Type	Function
NAND3	NAND gate
AND3	AND gate
NOR3	NOR gate
OR3	OR gate

`xx`      Digital gate identifier (ASCII string).

IN1, IN2, IN3  
Names of the input nodes.

OUT      Name of the output node.

REF1 REF2 Only to be used when specifying **.OPTION DYND2ALOG**. Names of the pins used in the dynamic calculation of the threshold values. See [page 7-1](#) for more details.

MNAME      Name of a model described with the **.MODEL** command.

PAR=VAL      A direct assignment of a **.MODEL** command parameter.

### Examples

```
*AND3 .MODEL definition
.model and_1 logic vhi=5 vlo=-5 vth=0 tpd=2.5n cin=0.5p
...
*main circuit
and3_1 n1 n2 n3 o1 and_1
```

Specifies a three input AND gate `and3_1` with input nodes `n1`, `n2` and `n3` and output node `o1`. The parameters of the AND gate are described in the model `and_1` using the **.MODEL** command.

```
and3_1 n1 n2 n3 o1 vhi=5 vlo=-5 vth=0 tpd=2.5n cin=0.5p
```

Specifies the same AND gate as above but with its parameters assigned directly instead of with the **.MODEL** command.



For more information, refer to “[.MODEL](#)” on page 10-160.

## Multiple Input Digital Gates

```
DGATExx IN1 IN2...{INX} OUT [REF1 REF2] MNAME [PAR=VAL]
```

Multiple Input Digital Gates Macromodel.

### Parameters

DGATE      Digital gate type

**Table 7-3. Multiple Input Digital Gate Types**

Gate Type	Function
NAND#	NAND gate
AND#	AND gate
NOR#	NOR gate
OR#	OR gate

xx            Digital gate identifier (ASCII string).

IN1, IN2,...{INX}  
Names of the input nodes.

OUT          Name of the output node.

REF1 REF2 Only to be used when specifying **.OPTION DYND2ALOG**. Names of the pins used in the dynamic calculation of the threshold values. See [page 7-1](#) for more details.

MNAME        Name of a model described with the **.MODEL** command.

PAR=VAL      A direct assignment of a **.MODEL** command parameter.

### Examples

```
*AND# .MODEL definition
.model and_1 logic vhi=5 vlo=-5 vth=0 tpd=2.5n cin=0.5p
...
*main circuit
and#_1 n1 n2 n3 n4 o1 and_1
```

Specifies an AND gate **and#\_1** with four input nodes **n1**, **n2**, **n3** and **n4** and an output node **o1**. The parameters of the AND gate are described in the model **and\_1** using the **.MODEL** command.

```
and#_1 n1 n2 n3 n4 o1 and_1 vhi=5 vlo=0 vth=2.5
+ tpd=2.5n cin=0.5p
```

In this example, the parameters on the instantiation line override the parameters in the **.MODEL** command.



For more information, refer to “[.MODEL](#)” on page 10-160.

---

## Mixed Signal Macromodels

The following mixed signal macromodels are provided in Eldo:

Analog to digital converter [ADC](#)

Digital to analog converter [DAC](#)



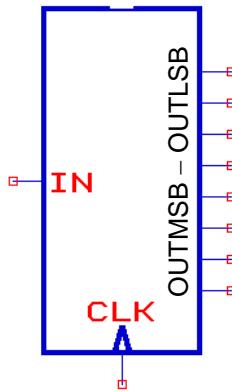
For Digital Macromodels see [page 7-1](#).

---

## Analog to Digital Converter

```
ADCxx CLK IN OUTSB{OUTSB} [EDGE=VAL] [VTH=VAL] [VHI=VAL]
+ [VLO=VAL] [VINF=VAL] [VSUP=VAL] [TCOM=VAL] [TPD=VAL]
```

Figure 7-4. Analog to Digital Converter Macromodel



The analog to digital converter (ADC) is defined by the clock, the analog input and a number of digital outputs. Outputs are computed only when the clock validates the input, i.e. on the rising or falling edge depending on the value of the **EDGE** parameter.

### Parameters

<b>xx</b>	Analog to digital converter identifier (ASCII string).
<b>CLK</b>	Name of the clock node.
<b>IN</b>	Name of the analog input node.
<b>OUTSB</b>	Digital output nodes (MSB to LSB). A maximum of 31 bits can be defined when using this macromodel.
<b>EDGE=VAL</b>	<b>EDGE=1</b> to validate the output on the rising edge of the clock. <b>EDGE=-1</b> to validate the output on the falling edge of the clock. Default value is 1.
<b>VTH=VAL</b>	Threshold voltage for the clock. Default value is 2.5V.
<b>VHI=VAL</b>	Voltage corresponding to the 1 output logical state. Default value is 5V.
<b>VLO=VAL</b>	Voltage corresponding to the 0 output logical state. Default value is 0V.
<b>VINF=VAL</b>	Lower input voltage. If an analog voltage entering the ADC is lower than this value, all outputs remain at <b>VLO</b> . Default value is 0V.
<b>VSUP=VAL</b>	Upper input voltage. If an analog voltage entering the ADC is higher than this value, all outputs remain at <b>VHI</b> . Default value is 5V.
<b>TCOM=VAL</b>	Time for the outputs to change from <b>VHI</b> to <b>VLO</b> or vice versa. Default value is 1ns.

**TPD=VAL**      Transit time through the converter. The output will start changing **TPD** seconds after the clock validates the input. Default value is 10ns.

### Example

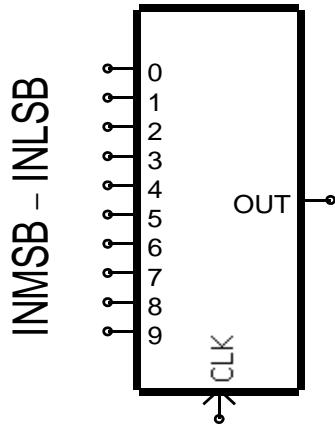
```
adc_1 clk in d4 d3 d2 d1 edge=-1 vth=1 vhi=5
+ vlo=0 vinf=1.0 vsup=4.0 tcom=5n tpd=2n
```

Specifies an ADC named **adc\_1** with clock node **clk**, analog input node **in** and digital output nodes **d4** (MSB), **d3**, **d2** and **d1** (LSB). The output is validated on the falling edge of the clock, with the threshold voltage for the clock being 1V. The voltages corresponding to logical 1 and 0 are 5V and 0V respectively. The upper and lower threshold voltages in order for the output to remain high or low are 4V and 1V respectively with the time for the ADC to change from a high to a low voltage being 5ns. Finally, the transit time through the ADC is 2ns.

## Digital to Analog Converter

```
DACxx CLK INSB{INSB} OUT [EDGE=VAL] [VTH=VAL] [VTIN=VAL]
+ [VHI=VAL] [VLO=VAL] [TPD=VAL] [SL=VAL]
```

Figure 7-5. Digital to Analog Converter Macromodel



The digital to analog converter (DAC) is defined by the clock, the digital inputs and the analog output. The output is computed only when the clock validates the input, i.e. on the rising or falling edge depending on the value of the **EDGE** parameter.

### Parameters

<b>xx</b>	Digital to analog converter identifier (ASCII string).
<b>CLK</b>	Name of the clock node.
<b>INSB</b>	Name of the digital input nodes (MSB to LSB).
<b>OUT</b>	Name of the analog output node. A maximum of 31 bits can be defined when using this macromodel.
<b>EDGE=VAL</b>	<b>EDGE=1</b> validates the output on the rising edge of the clock. <b>EDGE=-1</b> validates the output on the falling edge of the clock. Default value is 1.
<b>VTH=VAL</b>	Threshold voltage for the clock. Default value is 2.5V.
<b>VTIN=VAL</b>	Threshold voltage for the inputs. Default value is 2.5V.
<b>VHI=VAL</b>	Voltage output when all inputs are above <b>VTIN</b> . Default value is 5V.
<b>VLO=VAL</b>	Voltage output when all inputs are below <b>VTIN</b> . Default value is 0V.
<b>TPD=VAL</b>	Transit time through the converter. The output will start changing <b>TPD</b> seconds after the clock validates the output. Default value is 10ns.
<b>SL=VAL</b>	Slope at the output, in $\text{Vs}^{-1}$ . Default value is $0.1 \times 10^9 \text{ Vs}^{-1}$ ( $0.1 \text{ Vns}^{-1}$ ).

## Example

```
dac2 clk s4 s3 s2 s1 out vth=2 vlo=1 tpd=5n s1=1e9
+ vtin=2.2 vth=2.5
```

Specifies a DAC named **dac2** with clock node **clk**, digital inputs **s4** (MSB), **s3**, **s2** and **s1** (LSB) and analog output **out**. The threshold voltages for the clock and inputs are 2.5V and 2.2V respectively. When all inputs are above the input threshold voltage, the output voltage is equal to 2V, and when they are all below it the output voltage is equal to 1V. Finally, the slope of the output is defined as  $1\text{ Vns}^{-1}$ .



# Chapter 8

## Magnetic Macromodels

---

### Eldo Magnetic Macromodels

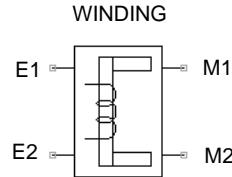
The following magnetic macromodels are provided in Eldo:

Winding Model	<a href="#">WINDING</a>
Non-linear Magnetic Core 1	<a href="#">NLCORE1</a>
Non-linear Magnetic Core 2	<a href="#">NLCORE2</a>
Linear Magnetic Core	<a href="#">LINCORE</a>
Magnetic Air Gap	<a href="#">AIRGAP</a>
Linear Transformer with Variable Number of Windings	<a href="#">LVTRANS</a>
Ideal Transformer	<a href="#">JTRAN</a>

## Transformer Winding

**YXX WINDING [PIN:] E1 E2 M1 M2 [PARAM: PAR=VAL {PAR=VAL}] [MODEL: MNAME]**

**Figure 8-1. Transformer Winding Macromodel**



This is a macromodel for a winding describing the interaction between the electrical and magnetic domain of a wire wrapped around a linear/non-linear material.

### Model Pins

- |    |                                    |
|----|------------------------------------|
| E1 | Name of the first electrical pin.  |
| E2 | Name of the second electrical pin. |
| M1 | Name of the first magnetic pin.    |
| M2 | Name of the second magnetic pin.   |

**Table 8-1. Transformer Winding Model Parameters**

Nr.	Name	Default	Units	Definition
1	<b>N</b>	$1.0 \times 10^3$		Number of turns
2	<b>R</b>	$1.0 \times 10^{-2}$	$\Omega$	Resistance of the winding
3	<b>K</b>	1.0		Coupling coefficient of the winding to the core. May be in the range $0.0 \leq K \leq 1.0$
4	<b>M<sup>a</sup></b>	1		Device multiplier

a. .OPTION YMFFACT must be specified for **M** to work.



See “General Notes on the Use of FAS Macromodels” on page 6-2 for additional notes on using this model.

---

### Example

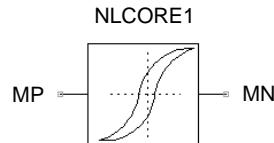
**ymod5 winding n1 n2 p1 p2**

Specifies a winding **ymod5** of type **winding** having electrical pins **n1** and **n2** with magnetic pins **p1** and **p2**. Default model parameters are used.

## Non-linear Magnetic Core 1

**YXX NLCORE1 [PIN:] MP MN [PARAM: PAR=VAL {PAR=VAL}] [MODEL: MNAME]**

**Figure 8-2. Non-linear Magnetic Core 1 Macromodel**



This is a macromodel for a non-linear magnetic core. It is a physically based mathematical model of the ferromagnetic hysteresis, which includes the following effects:

- Mean field approach for domain coupling.
- Domain wall motion.
- Pinning of domain walls on defect sites.
- Frequency dependent domain wall pinning.



The source information for this macromodel can be found in the following technical articles:

D.C. Jiles, D.L. Atherton, “*Theory of Ferromagnetic Hysteresis*”  
“Journal of Magnetism and Magnetic Materials,” Vol. 61, Sept. 1986, pp 48-60.  
R. Brachtendorf, R. Laur, “*Modeling of Magnetic Elements including Frequency Effects.*”  
2.GME/ITG Workshop “*Entwicklung von Analogschaltungen mit CAE-Methoden.*”  
Ilmenau, Germany, March 1993.

### Model Pins

MP Name of the input magnetic node.

MN Name of the output magnetic node.

**Table 8-2. Non-linear Magnetic Core 1 Model Parameters**

Nr.	Name	Default	Units	Definition
1	<b>AREA</b>	$1.0 \times 10^{-4}$	$\text{m}^2$	Core area
2	<b>LEN</b>	$1.0 \times 10^{-3}$	m	Length of the magnetic path
3	<b>MS</b>	$1.7 \times 10^6$	$\text{Am}^{-1}$	Saturation magnetization
4	<b>ALPHA</b>	$1.0 \times 10^{-3}$		Domain coupling coefficient (mean field parameter)
5	<b>A</b>	$1.0 \times 10^3$	$\text{Am}^{-1}$	Domain density

**Table 8-2. Non-linear Magnetic Core 1 Model Parameters**

Nr.	Name	Default	Units	Definition
6	<b>K</b>	$1.0 \times 10^3$	$\text{Am}^{-1}$	Domain wall pinning coefficient
7	<b>C</b>	0.1		Reversible wall motion coefficient
8	<b>KF</b>	$1.0 \times 10^{-6}$		Frequency dependent domain wall pinning coefficient
9	<b>LEVEL</b>	1		Selector for Anhysterisis model <b>LEVEL</b> =1—Langevin function <b>LEVEL</b> =2—Tangens-Hyperbolicus (tanh)
10	<b>MD</b>	$1.0 \times 10^{-5}$		Delay element for irreversible magnetization
11	<b>M<sup>a</sup></b>	1		Device multiplier

a. **.OPTION YMFACT** must be specified for **M** to work.

---

**i** See “[General Notes on the Use of FAS Macromodels](#)” on page 6-2 for additional notes on using this model.

---

## Example

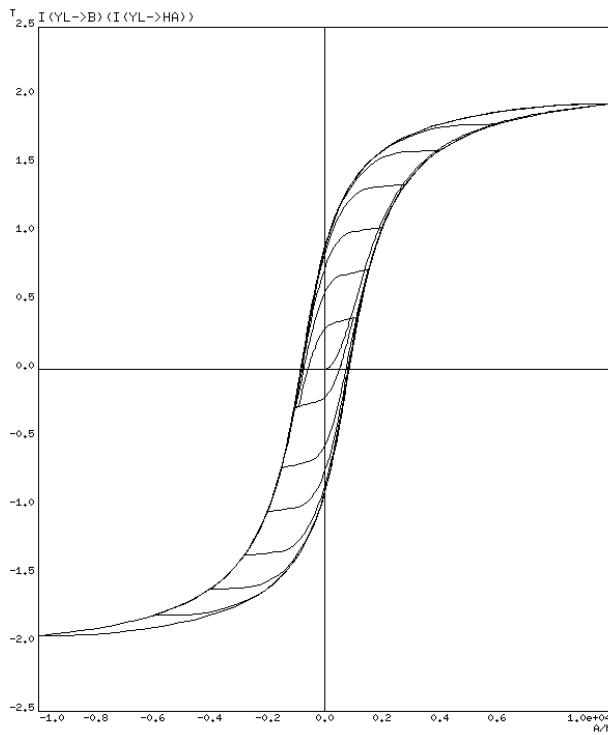
```
ymod1 nlcore1 n1 n2
```

Specifies a non-linear core **ymod1** of type **nlcore1** having input magnetic node **n1** and output magnetic node **n2**. Default model parameters are used.

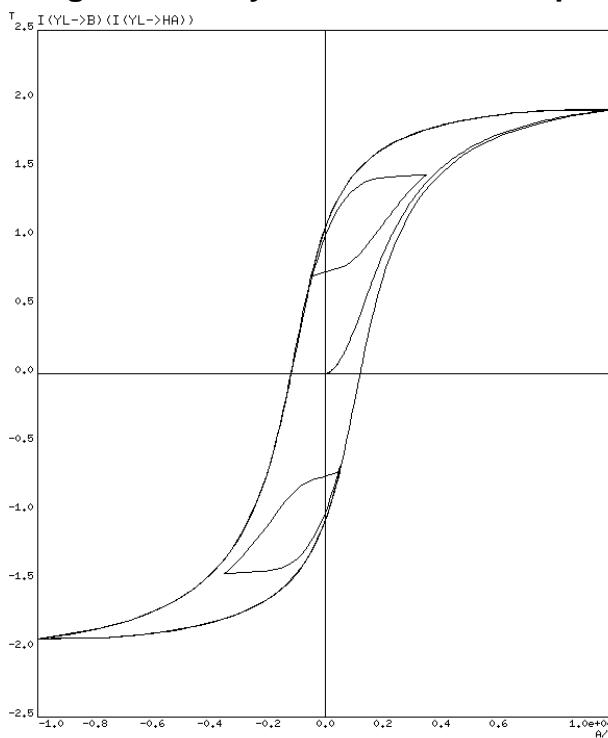
## Model Characteristics

Typical hysteresis curves for this macromodel are shown on the following page:

**Figure 8-3. Symmetric B-H loops with Different Amplitudes**



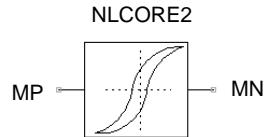
**Figure 8-4. Asymmetric Minor Loops**



## Non-linear Magnetic Core 2

**YXX NLCORE2 [PIN:] MP MN [PARAM: PAR=VAL {PAR=VAL}] [MODEL: MNAME]**

**Figure 8-5. Non-linear Magnetic Core 2 Macromodel**



A non-linear magnetic core macromodel. It describes the hysteretic behavior of a magnetic material including the temperature and frequency dependence of the hysteresis characteristic.



The source information for this macromodel can be found in the following article:  
Chan, Vladimirescu, Gao, Liebmann, Valainis, “*Non-linear Transformer Model for Circuit Simulation*” ‘IEEE Transactions on Computer-Aided Design,’ Vol. 10, No. 4, April 1991.

### Model Pins

MP Name of the input magnetic node.

MN Name of the output magnetic node.

**Table 8-3. Non-linear Magnetic Core 2**

Nr.	Name	Default	Units	Definition
1	<b>AREA</b>	$1.0 \times 10^{-4}$	$\text{m}^2$	Core area
2	<b>LEN</b>	$5.0 \times 10^{-2}$	m	Length of the magnetic path
3	<b>HC</b>	10.0	$\text{Am}^{-1}$	Coercive magnetic field strength
4	<b>BR</b>	0.1	$\text{Vsm}^{-2}$	Remnant magnetic flux density
5	<b>BS</b>	1.0	$\text{Vsm}^{-2}$	Saturation magnetic flux density
6	<b>CEPS</b>	$1.0 \times 10^{-2}$		Coefficient influencing the internal model accuracy
7	<b>TBS</b>	0.0	$\text{K}^{-1}$	Temperature coefficient for <b>BS</b>
8	<b>TBR</b>	0.0	$\text{K}^{-1}$	Temperature coefficient for <b>BR</b>
9	<b>THC</b>	0.0	$\text{K}^{-1}$	Temperature coefficient for <b>HC</b>
10	<b>FNOM</b>	$1.0 \times 10^3$	Hz	Working frequency
11	<b>FC1</b>	1.0		First frequency coefficient
12	<b>FC2</b>	0.0	$\text{Hz}^{-1}$	Second frequency coefficient

**Table 8-3. Non-linear Magnetic Core 2**

Nr.	Name	Default	Units	Definition
13	<b>FC3</b>	0.0		Third frequency coefficient
14	<b>MYI</b>	1000.0		Initial relative permeability of the core material
15	M <sup>a</sup>	1		Device multiplier

a. **.OPTION YMFAC** must be specified for **M** to work.



See “[General Notes on the Use of FAS Macromodels](#)” on page 6-2 for additional notes on using this model.

## Examples

```
ymod1 nlcore2 n1 n2
```

Specifies a non-linear magnetic core **ymod1** of type **nlcore2** having input magnetic node **n1** and output magnetic node **n2**. Default model parameters are used.

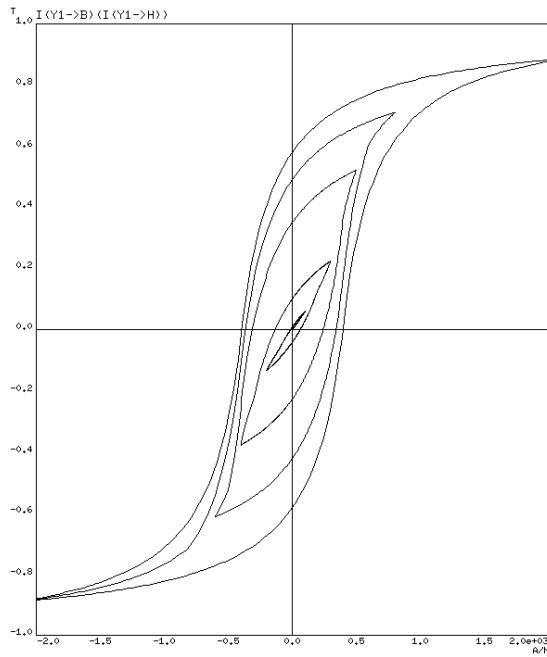
```
.model mod modfas bs=0.5 br=0.2 hc=20.0
...
ycore1 nlcore2 n1 n2 model: mod
```

Specifies a non-linear magnetic core **ycore1** of type **nlcore2** with input magnetic node **n1** and output magnetic node **n2**. Characteristic points of the hysteresis curve are declared using the **.MODEL** command.

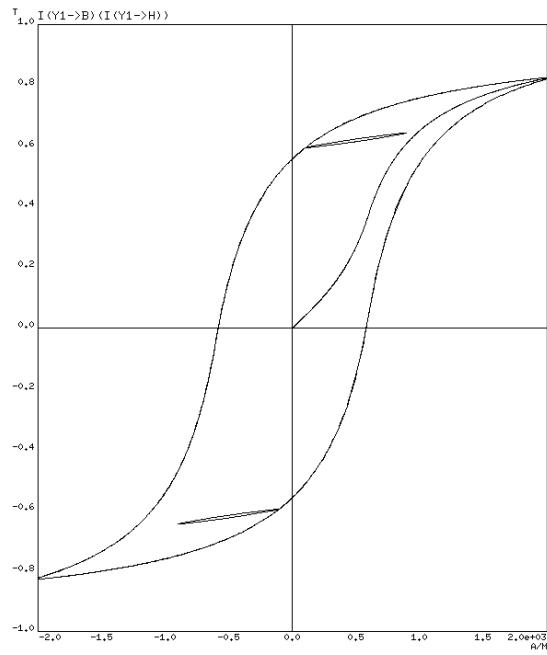
## Model Characteristics

Typical hysteresis curves for this macromodel are shown on the following page:

**Figure 8-6. Symmetric B-H loops with Different Amplitudes**



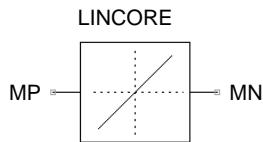
**Figure 8-7. Asymmetric Minor Loops**



## Linear Magnetic Core

**YXX LINCORE [PIN:] MP MN [PARAM: PAR=VAL {PAR=VAL}] [MODEL: MNAME]**

**Figure 8-8. Linear Magnetic Core Macromodel**



This is a macromodel for a magnetic core with linear B-H characteristics.

### Model Pins

- |    |                                   |
|----|-----------------------------------|
| MP | Name of the input magnetic node.  |
| MN | Name of the output magnetic node. |

**Table 8-4. Linear Magnetic Core Model Parameters**

Nr.	Name	Default	Units	Definition
1	<b>AREA</b>	$1.0 \times 10^{-4}$	$\text{m}^2$	Core area
2	<b>LEN</b>	$1.0 \times 10^{-2}$	m	Length of the magnetic path
3	<b>MYR</b>	1		Relative permeability of core material
4	<b>M<sup>a</sup></b>	1		Device multiplier

a. .OPTION YMFACT must be specified for **M** to work.

- 
- i** See “General Notes on the Use of FAS Macromodels” on page 6-2 for additional notes on using this model.
- 

### Example

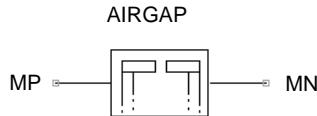
**ymod1 lincore n1 n2**

Specifies a linear core **ymod1** of type **lincore** having input magnetic node **n1** and output magnetic node **n2**. Default model parameters are used.

## Magnetic Air Gap

**YXX AIRGAP [PIN:] MP MN [PARAM: PAR=VAL {PAR=VAL}] [MODEL: MNAME]**

**Figure 8-9. Magnetic Air Gap Macromodel**



A linear resistor macromodel modeling the magnetic resistance of an air gap inside a magnetic core.

### Model Pins

- MP Name of the input magnetic node.  
MN Name of the output magnetic node.

**Table 8-5. Magnetic Air Gap Model Parameters**

Nr.	Name	Default	Units	Definition
1	<b>AGAP</b>	$1.0 \times 10^{-4}$	$m^2$	Cross-sectional area of the air gap
2	<b>LGAP</b>	$1.0 \times 10^{-2}$	m	Length of the air gap
3	M <sup>a</sup>	1		Device multiplier

a. **.OPTION YMFACT** must be specified for **M** to work.

- 
- i** See “General Notes on the Use of FAS Macromodels” on page 6-2 for additional notes on using this model.
- 

### Example

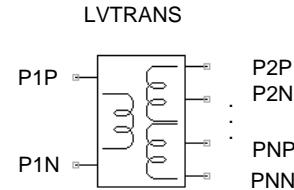
**ymod1 airgap n1 n2**

Specifies a magnetic air gap **ymod1** of type **airgap** having input magnetic node **n1** and output magnetic node **n2**. Default model parameters are used.

## Transformer (Variable # of Windings)

```
YXX LVTRANS [PIN:] P1P P1N P2P P2N {PNP PNN}
+ [PARAM: PAR=VAL {PAR=VAL}] [MODEL: MNAME]
```

**Figure 8-10. Transformer (Variable # of Windings) Macromodel**



A macromodel for a linear transformer with a variable number (maximum is 8) of windings. The number of pins at instantiation determines the number of transformer windings and hence the number of required parameters.

### Model Pins

PNP	Name of the positive pin of the nth transformer winding (dependent on the number of transformer windings declared).
PNN	Name of the negative pin of the nth transformer winding (dependent on the number of transformer windings declared).

**Table 8-6. Transformer Model Parameters**

Nr.	Name	Default	Units	Definition
1	$L_{ij}$	$1.0 \times 10^{-3}$	H	Element ij of inductance matrix
2	$R_i$	$1.0 \times 10^{-3}$	$\Omega$	Element i of winding resistance vector
3	$M^a$	1		Device multiplier

a. **.OPTION YMFACT** must be specified for **M** to work.

Where i=1 to number of windings, j=1 to number of windings.

### Example 1—Transformer with 2 Windings

```
ytr1 lvtrans p1 p2 s1 s2 param: 111=2.0e-3 112=1.0e-3
+ 121=2.0e-3 122=2.0e-3 r1=1 r2=10
```

Specifies a transformer **ytr1** of type **lvtrans** with two windings. The first transformer winding has pins **p1** (+ve) and **p2** (-ve) and the second winding has pins **s1** (+ve) and **s2** (-ve). The model equations given below show the inductance matrix and the resistance vector for the above transformer.

$$\begin{bmatrix} v(p1,p2) \\ v(s1,s2) \end{bmatrix} = \begin{bmatrix} L_{11} & L_{21} \\ L_{12} & L_{22} \end{bmatrix} \cdot \begin{bmatrix} \frac{d}{dt} i_1 \\ \frac{d}{dt} i_2 \end{bmatrix} + \begin{bmatrix} R_1 & 0 \\ 0 & R_2 \end{bmatrix} \cdot \begin{bmatrix} i_1 \\ i_2 \end{bmatrix}$$

where  $i1$  is the current in the first winding and  $i2$  is the current in the second winding.

### Example 2—Transformer with 3 Windings

```
ytr3 lvtrans p1 p2 s1 s2 t1 t2
```

Specifies a transformer **ytr3** of type **lvtrans** with three windings. The first transformer winding has pins **p1** (+ve) and **p2** (-ve), the second winding has pins **s1** (+ve) and **s2** (-ve) and the third winding has pins **t1** (+ve) and **t2** (-ve). Default parameters are used. The model equations given below show the inductance matrix and the resistance vector for the above transformer.

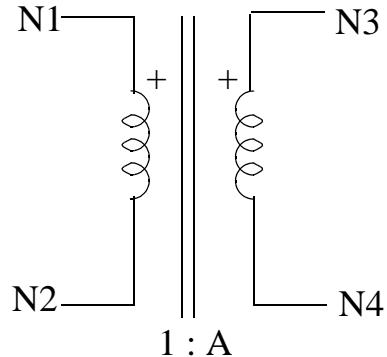
$$\begin{bmatrix} V(p1,p2) \\ V(s1,s2) \\ V(t1,t2) \end{bmatrix} = \begin{bmatrix} L_{11} & L_{21} & L_{31} \\ L_{12} & L_{22} & L_{32} \\ L_{13} & L_{23} & L_{33} \end{bmatrix} \cdot \begin{bmatrix} \frac{d}{dt} i_1 \\ \frac{d}{dt} i_2 \\ \frac{d}{dt} i_3 \end{bmatrix} + \begin{bmatrix} R_1 \\ R_2 \\ R_3 \end{bmatrix} \cdot \begin{bmatrix} i_1 \\ i_2 \\ i_3 \end{bmatrix}$$

where  $i1$  is the current flowing in the first winding,  $i2$  is the current in the second winding and  $i3$  is the current in the third winding.

## Ideal Transformer

**Yxx JTRAN N1 N2 N3 N4 [PARAM: A=VAL]**

**Figure 8-11. Ideal Transformer Macromodel**



A macromodel for an ideal transformer.

### Parameters

N1, N2, N3, N4

Ideal transformer nodes as shown in the figure above.

A Transformer's turns ratio.

### Example

AC simulation of an ideal transformer (1:2).

```
.param tran_ratio=2

Ytran jtran ain 0 aout 0 PARAM: a=tran_ratio

Vin      Xin      0      AC   10
Vdummy1  Xin      ain    AC   0
Vdummy2  aout     Xout   AC   0
Rout     Xout     0      1

.defwave voltage_gain=V(aout)/V(ain)
.defwave current_gain=I(Vdummy2)/I(Vdummy1)
.ac dec 10 1 1g

.plot ac wm(voltage_gain)
.plot ac wp(voltage_gain)
.plot ac wm(current_gain)
.plot ac wp(current_gain)

.end
```



# Chapter 9

## Switched Capacitor Macromodels

---

### Introduction

This chapter is concerned with the area of Switched Capacitor Macromodels. Within the subject area of Switched Capacitor Macromodels, there are two separate types to be distinguished between and they are:

- Switch Level Representation.
- Z-domain Representation.

### Switch Level Representation

The following macromodels have a Charge Conservation characteristic and are the key elements for Switched Capacitor (SC) applications, where the MOS analog switches are represented by an approximate general linearized model of a non-ideal switch.

All these models are Kirchhoff type elements and can be applied as normal components in an electronic circuit in the same way as other models such as bipolar transistors, diodes, noise sources etc.

All switched capacitor networks built using these models can be analyzed in the time domain without any restrictions, but an AC analysis would be meaningless. If an AC analysis is needed, particularly for switched capacitor filter applications, then another type of model representation for the switched capacitor circuit is required.



Refer to “[Z-domain Representation](#)” on page 9-11.

---

### Macromodels

Below is a list of the Eldo Macromodels which are described throughout this chapter:

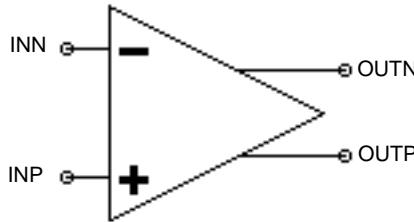
Operational Amplifier	<a href="#">OPAxx</a>
Switch	<a href="#">Sxx</a>
Ideal Operational Amplifier	<a href="#">Yxx SC_IDEAL</a>
Inverting Switched Capacitor	<a href="#">Yxx SC_I</a>

Non-inverting Switched Capacitor	<b>Yxx SC_N</b>
Parallel Switched Capacitor	<b>Yxx SC_P</b>
Serial Switched Capacitor	<b>Yxx SC_S1</b> <b>Yxx SC_S2</b>
Serial-parallel Switched Capacitor	<b>Yxx SC_SP1</b> <b>Yxx SC_SP2</b>
Bi-linear Switched Capacitor	<b>Yxx SC_B</b>
Unswitched Capacitor	<b>Yxx SC_U</b>

## Operational Amplifier

```
OPAXX INP INN OUTP OUTN [MNAME] [LEVEL=VAL] [VOFF=VAL]
+ [SL=VAL] [CIN=VAL] [RS=VAL] [VSAT=VAL] [VSATM=VAL]
+ [GAIN=VAL] [FC=VAL] [FNDP=VAL] [IMAX=VAL] [CMRR=VAL]
```

**Figure 9-1. Operational Amplifier Macromodel**



A macromodel for single- and two-stage operational amplifiers. This syntax supersedes that of all previous Eldo versions. Old syntax is still supported, but no longer recommended.

### Parameters

<b>xx</b>	Amplifier name.
<b>INP</b>	Name of the positive input node.
<b>INN</b>	Name of the negative input node.
<b>OUTP</b>	Name of the positive output node.
<b>OUTN</b>	Name of the negative output node. For single-stage op-amps this must be set to zero.

When specified, the optional parameters listed below override default values set via the **.MODEL** command.

<b>MNAME</b>	The model name, as described in the <b>.MODEL</b> command.
<b>LEVEL=VAL</b>	1 for single-stage, 2 for two-stage op-amps. Default value is 2. The <b>LEVEL</b> parameter cannot be changed in the instantiation line, only in the <b>.model</b> card.
<b>VOFF=VAL</b>	Offset voltage in volts. Default value is 0V.
<b>SL=VAL</b>	Slew rate in volts/second for two-stage op-amps only. Default is $1.0 \times 10^6$ V/s.
<b>CIN=VAL</b>	Input capacitance in farads. Default value is 0F.
<b>RS=VAL</b>	Output resistance in ohms. Default value is $1M\Omega$ for single-stage and $10M\Omega$ for two-stage op-amps.
<b>VSAT=VAL</b>	Symmetrical saturation voltage of $\pm$ <b>VSAT</b> in volts. Default value is 5V.

**VSATM=VAL**

Asymmetrical saturation voltage, with **VSATN** as lower and **VSAT** as upper saturation voltage. Default is -5V.

**Note**



**VSAT** and **VSATM** must be declared together if they are used in the instantiation line.

---

**GAIN=VAL**

Linear or dB scaling factor. Default value is 1000.

**FC=VAL**

Cut-off frequency in Hertz for two-stage amplifiers only.  
Default value is 1kHz.

**FNDP=VAL**

Non-dominant pole frequency in Hertz for single-stage op-amps only. Default value is 1kHz.

**IMAX=VAL**

Saturation current in amps for single-stage op-amps only.  
Default value is 100mA.

**CMRR=VAL**

Common mode rejection ratio, linear or in dB. Default value is zero.

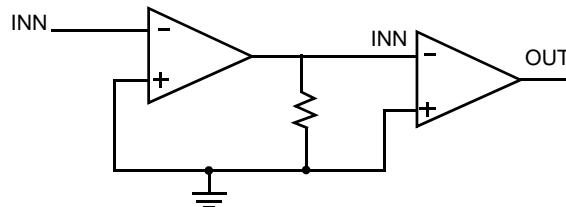
**Note**



When using an operational amplifier macromodel, it is recommended that the default simulator accuracy (**EPS** parameter) is increased from 1mV to 1 $\mu$ V using the **.OPTION** command. The amplifier model calculates a current at its output and expects a voltage at its input. It is recommended, therefore that a resistor be connected between the input and output of cascaded amplifier macromodels, as shown below.

---

**Figure 9-2. Cascaded Amplifier Macromodel**

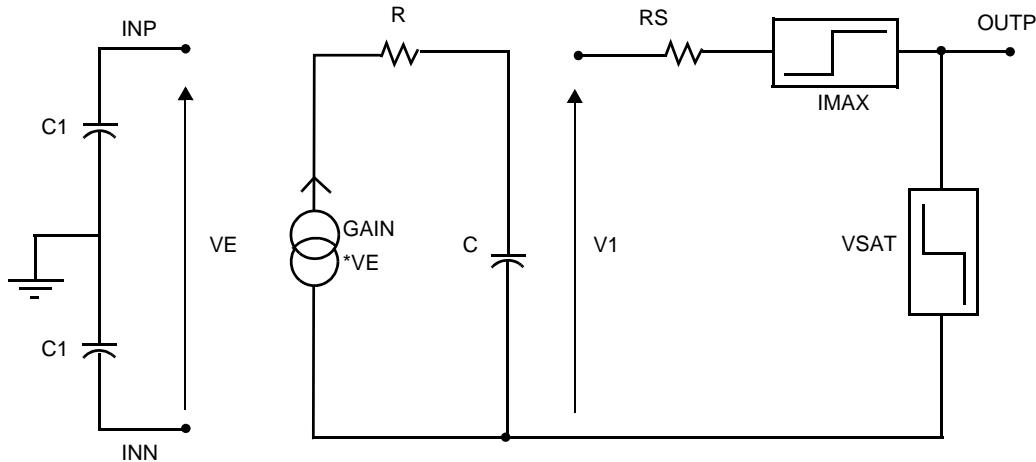


## Operational Amplifier Model

**.MODEL MNAME OPA [ PAR=VAL ]**

### Equivalent Circuit (Single-stage Amplifier)

**Figure 9-3. Equivalent Circuit (Single-stage Amplifier)**



### Model Equations (Single-stage Amplifier)

$R=1\text{k}\Omega$  which cannot be changed.

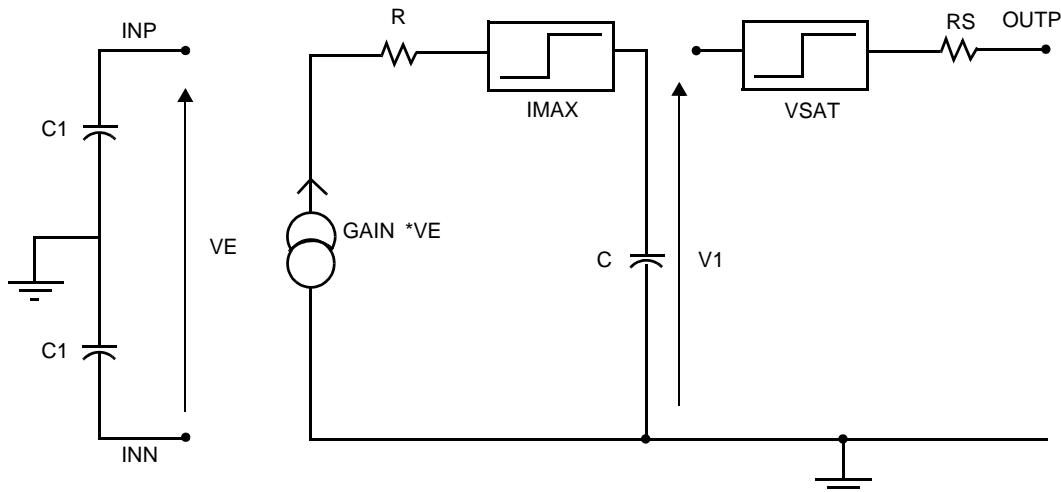
$$C = \frac{1}{2\pi \times R \times \text{FNDP}}$$

For DC analysis,  $C$  is open circuit, therefore:

$$V1 = \text{GAIN} \times VE$$

### Equivalent Circuit (Two-stage Amplifier)

**Figure 9-4. Equivalent Circuit (Two-stage Amplifier)**



### Model Equations (Two-stage Amplifier)

$R=1\text{k}\Omega$  which cannot be changed.

$$C = \frac{1}{2\pi \times R \times FC}$$

$$IMAX = SL \times C$$

For DC analysis,  $C$  is short circuit, therefore:

$$I = \frac{(GAIN \times VE)}{R}$$

If  $I < IMAX$  then:

$$V1 = GAIN \times VE$$

else:  $V1 = IMAX \times R$

**Table 9-1. Operational Amplifier Model Parameters**

Nr.	Name	Default	Units	Definition
1	<b>LEVEL</b>	2		Amplifier index
2	<b>VOFF</b>	0	V	Offset voltage
3	<b>SL</b>	$1.0 \times 10^6$	$\text{Vs}^{-1}$	Slew rate (two-stage)
4	<b>CIN</b>	0	F	Input capacitance
5	<b>RS</b>	$1.0 \times 10^6$	$\Omega$	Output resistance (single-stage)
		1	$\Omega$	Output resistance (two-stage)
6	<b>VSAT</b>	5	V	Symmetrical saturation voltage
7	<b>VSATN</b>	-5	V	Unsymmetrical saturation voltage
8	<b>GAIN</b>	$1.0 \times 10^3$		Scaling factor
9	<b>FC</b>	$1.0 \times 10^3$	Hz	Cut-off frequency (two-stage)
10	<b>FNDP</b>	$1.0 \times 10^8$	Hz	Pole frequency (single-stage)
11	<b>IMAX</b>	0.1	A	Saturation current (single-stage)
10	<b>CMRR</b>	0		Common mode rejection ratio

## Example

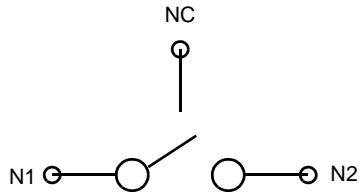
```
*OPAMP model definition
.model amp opa level=2 voff=0 sl=50e06
+ cin=0 rs=10 vsat=5 gain=5000 fc=5000
...
*main circuit
opa1 n2 n1 n3 0 ampop
```

Specifies the operational amplifier **opa1** of model type **ampop** having input nodes **n2** (+ve) and **n1** (-ve) with output nodes **n3** (+ve) and ground (-ve). The electrical parameters of the op-amp are specified using the **.MODEL** command.

## Switch

```
sxx NC N1 N2 [MNAME] [RON [CREC]]
```

**Figure 9-5. Switch Macromodel**



### Parameters

xx	Switch name.
NC	Switch voltage controlling node.
N1	Name of the node 1.
N2	Name of the node 2.
MNAME	Model name, as described in the <b>.MODEL</b> command.
RON	“ON” resistance of the switch in ohms. Default is 1kΩ.
CREC	Overlap capacitance, modeling Charge Injection. Default is zero.

---

**Note**

Switch macromodels may only be used in transient noise or DC simulations.

---

### Example

```
s23 c n2 n6 2000 0.02e-12
```

Specifies a switch named **s23** placed between nodes **n2** and **n6**, with controlling node **c**, having a 2kΩ “ON” resistance and 0.02pF overlap capacitance.

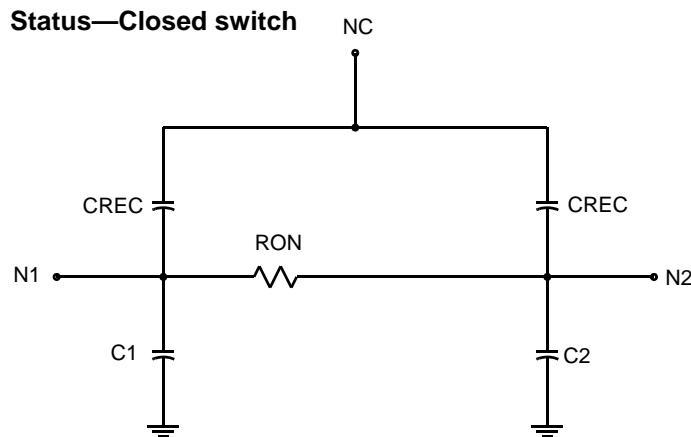
### Switch Model

<b>.MODEL</b> MNAME <b>NSW</b> [ PAR=VAL ]	NMOS
<b>.MODEL</b> MNAME <b>PSW</b> [ PAR=VAL ]	PMOS

## Model Equivalent Circuit

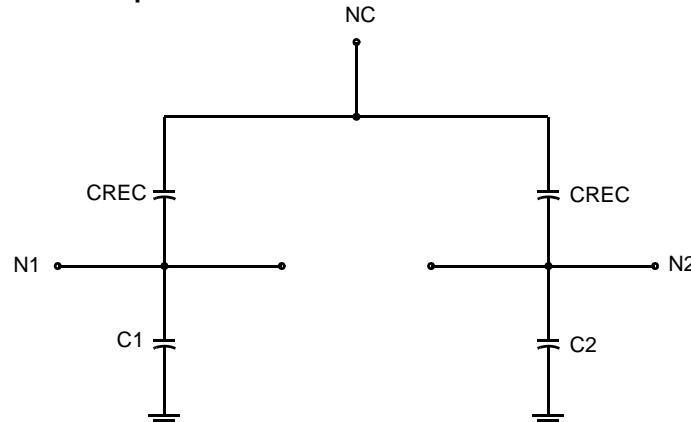
**Figure 9-6. Closed Switch Equivalent Circuit**

Status—Closed switch

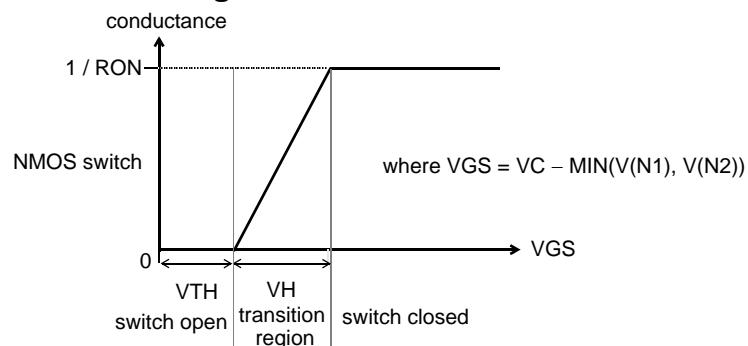


**Figure 9-7. Open Switch Equivalent Circuit**

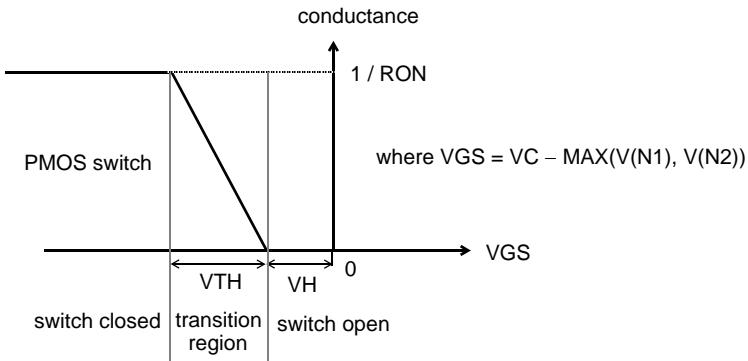
Status—Open switch



**Figure 9-8. NMOS Switch**



**Figure 9-9. PMOS Switch**



**Table 9-2. Switch Model Parameters**

Nr.	Name	Default	Units	Definition
1	<b>VTH</b>	0.47	V	Threshold voltage (for enhanced NMOS)
		-0.47	V	Threshold voltage (for enhanced PMOS)
2	<b>VH</b>	0.5	V	Transition voltage (for enhanced NMOS)
		-0.5	V	Transition voltage (for enhanced PMOS)
3	<b>RON</b>	$1.0 \times 10^3$	$\Omega$	“ON” resistance
4	<b>CREC</b>	0	F	Total overlap capacitance
5	<b>C1</b>	$10 \times 10^{-15}$	F	Switch input capacitance
6	<b>C2</b>	$10 \times 10^{-15}$	F	Switch output capacitance

### Further Explanation of Parameters

**VTH** Threshold voltage. The voltage at which the “OFF” resistance starts changing.

**VH** When the control voltage  $V(NC)$  reaches  $VTH+VH$ , the switch attains the “ON” resistance.

### Example

```
.model styp nsw vh=0.4 vth=0.5 ron=1k crec=50f
...
s7 ck3 5 7 styp
```

Specifies the switch **s7** of model type **styp** (N-type) placed between the nodes 5 and 7 with controlling node **ck3**. The electrical parameters of the switch are specified using the **.MODEL** command.

## Z-domain Representation

### General Notes on the Use of Macromodels

The following macromodels are basic building blocks for switched capacitor filters and sampled data systems, controlled by two clock phases 1 and 2. The behavior of each SC element is characterized by an admittance matrix which is created internally during the transient or the AC analysis as a function of the parameters (capacitance `C`, period `TP` of the controlling clocks and the `LDI` flag). This matrix transforms the vector of all the pin voltages into the vector of all the currents entering the pins of the element. This transformation works in a (discrete) time domain as well as in a frequency domain and allows both the transient and AC analyses of the SC circuit.

---

**Note**

 The controlling clock signals are represented by the parameter `TP`.

---

This modeling approach is linear, but extremely fast. Using this approach, a transient analysis can be performed within a few seconds, whereas a couple of hours would be required if the SC filter was built on a transistor circuit description level!

The following section illustrates the uses of these macromodels.

**LDI Definition:** LDI Transformation flag can equal 0, 1 or 2. It is only important for AC analysis. It has to be set if a switched capacitor representation of a resistor is connected to the inverting input of an op-amp, with a feedback capacitance and if both form an `LDI`, `LDI=1` or `2` identifies the switch branch, which is connected to the inverting input of the op-amp by its equally named controlling clock-phase.



See “SC Integrators & LDI’s” on page 9-13.

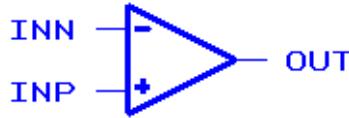
---

The effect is that the current of the corresponding switch branch will be delayed by a factor of  $z-1/2$ . If `LDI=0`, no LDI transformation will be performed.

## Ideal Operational Amplifier

```
YXX SC_IDEAL [PIN:] INP INN OUT [PARAM: M=val]
```

**Figure 9-10. Ideal Operational Amplifier Macromodel**



This is an ideal operational amplifier macromodel implemented for use as a basic building block in SC circuits and sampled data systems.

### Model Pins

INP	Non-inverting input.
INN	Inverting input.
OUT	Output.

#### **PARAM: M=VAL**

Device multiplier. Simulates the effect of multiple devices in parallel. In effect the current value is multiplied by **M**. Default is 1. **.OPTION YMFACT** must be selected in order for this option to work.

---

#### **Note**

A feedback connection between output and input is recommended.

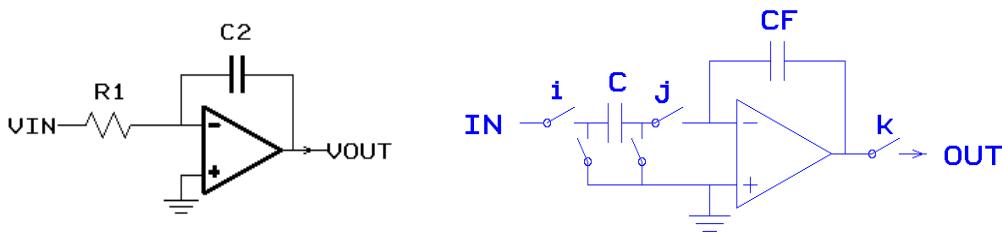
---

## SC Integrators & LDI's

One of the most important applications of Z-domain SC elements is in the construction of SC Integrators & LDI's, the key elements for the design of switch capacitor filters and sampled data systems. In these applications they are basic building blocks, along with others such as delay elements, sample-and-hold elements, summing amplifiers etc.

But even the SC Integrators can be further decomposed into more elementary basic building blocks, namely SC resistors (`sc_i`, `sc_n`, `sc_p` etc.), op-amps (`sc_ideal`, `opa`) and unswitched capacitors (`sc_u`). Most SC Integrators represent the same RC Integrator. They have approximately the same transfer function if the resistor  $R_1$  is substituted by an appropriate switched capacitor (`sc_i`, `sc_n`, `sc_p` etc.). See the figure below:

**Figure 9-11. SC Integrators & LDI's**



**Note**

The transfer function of an SC Integrator depends on the sampling instants of the output signal, i.e. different clock phases  $k$ , which control the switch path at the output of the integrator causing different transfer functions of this integrator.

This is now explained exactly in the formal mathematical language.

Supposing that:

$i$  is the phase which controls the switch branch at the input of the integrator,  
 $j$  is the phase which controls the switch branch at the inverting input of the op-amp,

$k$  is the phase which controls the switch branch at the output of the integrator, and  
 $H_{ijk}$  is the transfer function of the integrator controlled by  $i, j, k$ .

The following relationship will be true:

$$H_{ijk}(z) = H_{ikk}(z) \times z^\alpha$$

where  $H_{ijk}(z)$  and  $H_{ikk}(z)$  are the transfer functions corresponding to the same type  $H$  of the integrator but controlled by the clock phases  $i, j, k$  and  $i, k, k$  respectively and:

$$\alpha = \begin{cases} 0 & \text{if } j=k \\ \text{or} \\ -0.5 & \text{otherwise} \end{cases}$$

i.e.

$$\text{LDI} = \begin{cases} 0 & \text{if } j=k \\ \text{or} \\ j & \text{otherwise} \end{cases}$$



This information is taken from:

Rolf Unbehauen, Andrzej Cichocki, "MOS Switched Capacitor and Continuous-Time Integrated Circuits and Systems."

---



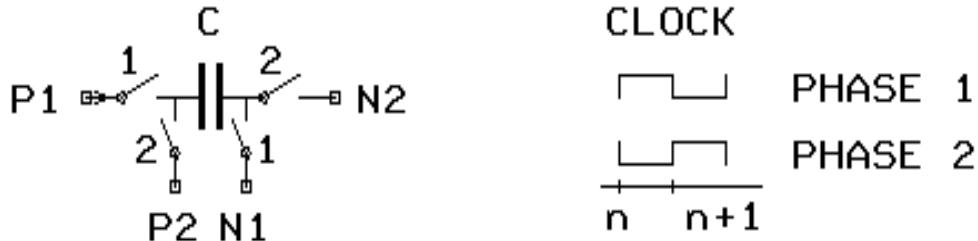
For examples, please see "[Applications](#)" on page 9-30.

---

## Inverting Switched Capacitor

**YXX SC\_I [PIN:] P1 P2 N2 N1 [PARAM: PAR=VAL {PAR=VAL}] [MODEL: MNAME]**

**Figure 9-12. Inverting Switched Capacitor Macromodel**



This is a basic building block for Z-domain modeling of SC circuits and sampled data systems. It represents an SC realization of an analog resistor, the so called inverting switched capacitor.

It has been modeled as a voltage to current four port transfer element with a characteristic transfer or admittance matrix  $Y(z)$ , used for the description of the behavior of this element for both the transient and small signal AC analyses. The transfer function can be symbolically represented as shown above.

It is a Kirchhoff type element having a time-discrete I-U-dependence with charge storage and delay effects controlled by the uniform two-phase clock.

The model assumes that all currents remain constant during each switching phase and that they only change their values at the beginning of every switching sub-interval.

Only one sc\_i model should be connected to an amplifier summing node (minus input). Simulation results will be incorrect if two or more sc\_i macromodels are connected to the minus input node of an amplifier. In this particular case, current (charge) from output nodes of the sc\_i macro is not summed together in the correct way.

### Model Pins

P1	Name of the positive pin to the switch branch controlled by clock phase 1.
P2	Name of the positive pin to the switch branch controlled by clock phase 2.
N1	Name of the negative pin to the switch branch controlled by clock phase 1.
N2	Name of the negative pin to the switch branch controlled by clock phase 2.

**Table 9-3. Inverting Switched Capacitor Model Parameters**

Nr.	Name	Default	Units	Definition
1	C	$1.0 \times 10^{-12}$	F	Capacitance
2	TP	$1.0 \times 10^{-6}$	s	Pulse period

**Table 9-3. Inverting Switched Capacitor Model Parameters**

Nr.	Name	Default	Units	Definition
3	<b>LDI</b>	0		See the note on LDI's in this chapter
4	<b>M</b> <sup>a</sup>	1		Device multiplier

a. **.OPTION YMFACT** must be specified for **M** to work.

**Note**



The above macromodel parameters can be declared in a number of ways, listed below in order of priority.

1. In the instantiation line, using the **.PARAM** keyword.

2. In a model command line. The syntax is as follows:

```
.model mname modfas par=val [par=val]
```

If this syntax is used, a model name must be declared using the **.MODEL** keyword in the instantiation line.

3. If no parameters are explicitly declared, the parameter default values are used.

## Model Equations

This element performs the linear Z-domain transfer function:

$$[I_{p1}(z), I_{n1}(z), I_{p2}(z), I_{n2}(z)] = Y(z, LDI)[V_{p1}(z), V_{n1}(z), V_{p2}(z), V_{n2}(z)]$$

in the time and frequency domain for the transient and small signal AC analyses where:

$I_{p1}, I_{n1}, I_{p2}, I_{n2}$  are the currents contributed to the pins P1, N1, P2, N2.

$V_{p1}, V_{n1}, V_{p2}, V_{n2}$  are the voltages at the nodes P1, N1, P2, N2.

$Y(z, LDI)$  is the characteristic transfer or admittance matrix of the model.

The Z-transfer matrix  $Y(z, \theta)$  is the Z-transform of the difference equations describing the time domain behavior of this element. Therefore the case of **LDI=0** is completely adequate to the time domain behavior.

In some SC Integrators, the pins P2, N1 are grounded, whereas N2 is connected to the virtual ground of an ideal op-amp, in which case the transfer function at node N2 becomes the following form:

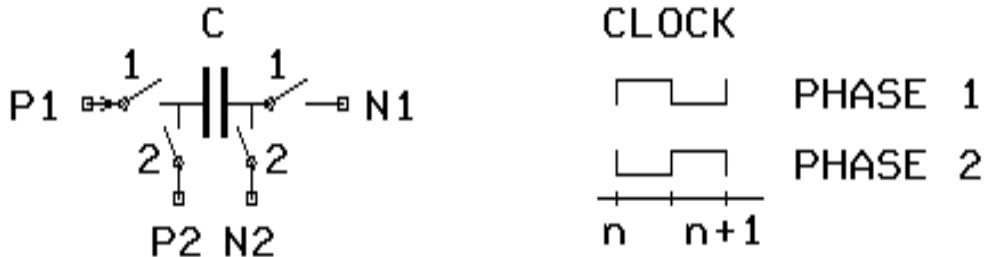
$$I_{n2}(z) = \frac{C}{T_p} \times z^{-1/2} \times V_{p1}(z)$$

where:  $z^k = e^{j\omega(k \times T_p)}$ ,  $k \in \text{Integer}$

## Non-inverting Switched Capacitor

**YXX SC\_N [PIN:] P1 P2 N1 N2 [PARAM: PAR=VAL {PAR=VAL}] [MODEL: MNAME]**

**Figure 9-13. Non-inverting Switched Capacitor Macromodel**



This is a basic building block for Z-domain modeling of SC circuits and sampled data systems. It represents an SC realization of an analog resistor, the so called non-inverting switched capacitor.

It has been modeled as a voltage to current four port transfer element with a characteristic transfer or admittance matrix  $Y(z)$ , used for the description of the behavior of this element for both the transient and small signal AC analyses. The transfer function can be symbolically represented as shown above.

It is a Kirchhoff type element having a time-discrete I-U-dependence with charge storage and delay effects controlled by the uniform two-phase clock.

The model assumes that all currents remain constant during each switching phase and that they only change their values at the beginning of every switching sub-interval.

### Model Pins

- P1 Name of the positive pin to the switch branch controlled by clock phase 1.
- P2 Name of the positive pin to the switch branch controlled by clock phase 2.
- N1 Name of the negative pin to the switch branch controlled by clock phase 1.
- N2 Name of the negative pin to the switch branch controlled by clock phase 2.

**Table 9-4. Non-inverting Switched Capacitor Model Parameters**

Nr.	Name	Default	Units	Definition
1	C	$1.0 \times 10^{-12}$	F	Capacitance
2	TP	$1.0 \times 10^6$	s	Pulse period
3	LDI	0		See the note on LDI's in this chapter
4	M <sup>a</sup>	1		Device multiplier

- a. **.OPTION YMFACT** must be specified for **M** to work.

**Note**



The above macromodel parameters can be declared in a number of ways, listed below in order of priority.

---

1. In the instantiation line, using the **.PARAM** keyword.
2. In a model command line. The syntax is as follows:

```
.model mname modfas par=val [par=val]
```

If this syntax is used, a model name must be declared using the **.MODEL** keyword in the instantiation line.

3. If no parameters are explicitly declared, the parameter default values are used.

## Model Equations

This element performs the linear Z-domain transfer function:

$$[I_{p1}(z), I_{n1}(z), I_{p2}(z), I_{n2}(z)] = Y(z, LDI)[V_{p1}(z), V_{n1}(z), V_{p2}(z), V_{n2}(z)]$$

in the time and frequency domain for the transient and small signal AC analyses where:

$I_{p1}, I_{n1}, I_{p2}, I_{n2}$  are the currents contributed to the pins P1, N1, N2, N2.

$V_{p1}, V_{n1}, V_{p2}, V_{n2}$  are the voltages at the nodes P1, N1, P2, N2.

$Y(z, LDI)$  is the characteristic transfer or admittance matrix of the model.

The Z-transfer matrix  $Y(z, \theta)$  is the Z-transform of the difference equations describing the time domain behavior of this element. Therefore the case of **LDI=0** is completely adequate to the time domain behavior.

In some SC Integrators the pins P2, N2 are grounded whereas N1 is connected to the virtual ground of an ideal op-amp, in which case the transfer function at node N1 becomes the following form:

$$I_{n1}(z) = -\frac{C}{T_p} \times V_{p1}(z)$$

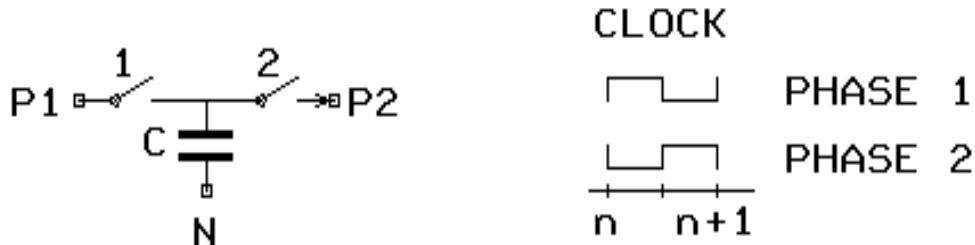
where:

$$z^k = e^{j\omega(k \times T_p)}, k \in \text{Integer}$$

## Parallel Switched Capacitor

**YXX SC\_P [PIN:] P1 P2 N [PARAM: PAR=VAL {PAR=VAL}] [MODEL: MNAME]**

**Figure 9-14. Parallel Switched Capacitor Macromodel**



This is a basic building block for Z-domain modeling of SC circuits and sampled data systems. It represents an SC realization of an analog resistor, the so called parallel switched capacitor.

It has been modeled as a voltage to current three port transfer element with a characteristic transfer or admittance matrix  $Y(z)$ , used for the description of the behavior of this element for both the transient and small signal AC analyses. The transfer function can be symbolically represented as shown above.

It is a Kirchhoff type element having a time-discrete I-U-dependence with charge storage and delay effects controlled by the uniform two-phase clock.

The model assumes that all currents remain constant during each switching phase and that they only change their values at the beginning of every switching sub-interval.

### Model Pins

- P1 Name of the positive pin to the switch branch controlled by clock phase 1.
- P2 Name of the positive pin to the switch branch controlled by clock phase 2.
- N Name of the negative pin at the outer side of the capacitor not connected to the switches of this element.

**Table 9-5. Parallel Switched Capacitor**

Nr.	Name	Default	Units	Definition
1	C	$1.0 \times 10^{-12}$	F	Capacitance
2	TP	$1.0 \times 10^6$	s	Pulse period
3	LDI	0		See the note on LDI's in this chapter
4	M <sup>a</sup>	1		Device multiplier

a. .OPTION YMFACT must be specified for M to work.

 **Note**

The above macromodel parameters can be declared in a number of ways, listed below in order of priority.

---

1. In the instantiation line, using the **.PARAM** keyword.
2. In a model command line. The syntax is as follows:

```
.model mname modfas par=val [par=val]
```

If this syntax is used, a model name must be declared using the **.MODEL** keyword in the instantiation line.

3. If no parameters are explicitly declared, the parameter default values are used.

## Model Equations

This element performs the linear Z-domain transfer function:

$$[I_{p1}(z), I_{p2}(z), I_n(z)] = Y(z, LDI)[V_{p1}(z), V_{p2}(z), V_n(z)]$$

in the time and frequency domain for the transient and small signal AC analyses where:

$I_{p1}, I_{p2}, I_n$  are the currents contributed to the pins P1, P2, N.

$V_{p1}, V_{p2}, V_n$  are the voltages at the nodes P1, P2, N.

$Y(z, LDI)$  is the characteristic transfer or admittance matrix of the model.

The Z-transfer matrix  $Y(z, \theta)$  is the Z-transform of the difference equations describing the time domain behavior of this element. Therefore the case of **LDI=0** is completely adequate to the time domain behavior.

In some SC Integrators the pin N is grounded whereas P2 is connected to the virtual ground of an ideal op-amp, in which case the transfer function becomes the following form:

$$I_{p2}(z) = -\frac{C}{T_p} \times z^{-1/2} \times V_{p1}(z)$$

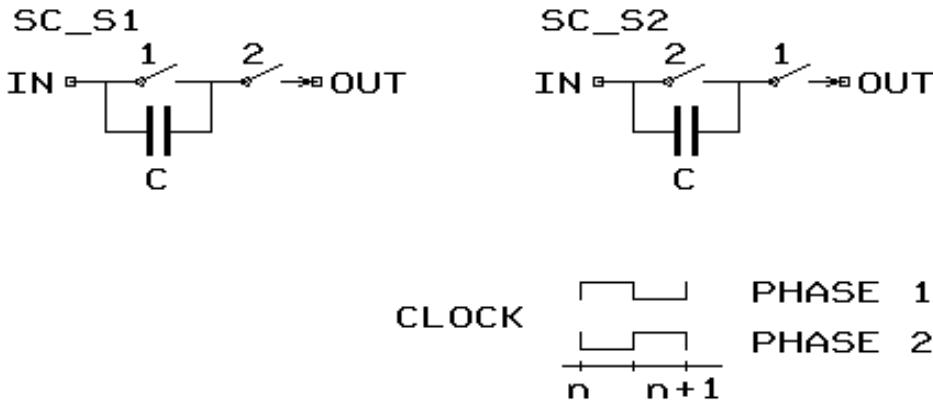
where:

$$z^k = e^{j\omega(k \times T_p)}, k \in \text{Integer}$$

## Serial Switched Capacitor

```
YXX SC_S1 [PIN:] IN OUT [PARAM: PAR=VAL {PAR=VAL}] [MODEL: MNAME]
YXX SC_S2 [PIN:] IN OUT [PARAM: PAR=VAL {PAR=VAL}] [MODEL: MNAME]
```

**Figure 9-15. Serial Switched Capacitor**



These are basic building blocks for the Z-domain modeling of SC circuits and sampled data systems. They represent SC realizations of an analog resistor, the so called serial switched capacitor.

They have been modeled as voltage to current two port transfer elements with a characteristic transfer or admittance matrix  $Y(z)$ , used for the description of the behavior of these elements for both the transient and small signal AC analyses. The transfer function can be symbolically represented as shown above.

They are Kirchhoff type elements having a time-discrete I-U-dependence with charge storage and delay effects controlled by the uniform two-phase clock.

The model assumes, that all currents remain constant during each switching phase and that they only change their values at the beginning of every switching sub-interval.

### Model Pins

IN	Name of the positive pin to the switch branch controlled by clock phase 1 (or 2).
OUT	Name of the pin to the switch branch controlled by clock phase 2 (or 1).

**Table 9-6. Serial Switched Capacitor Model Parameters**

Nr.	Name	Default	Units	Definition
1	C	$1.0 \times 10^{-12}$	F	Capacitance
2	TP	$1.0 \times 10^6$	s	Pulse period
3	LDI	0		See the note on LDI's in this chapter

**Table 9-6. Serial Switched Capacitor Model Parameters**

Nr.	Name	Default	Units	Definition
4	M <sup>a</sup>	1		Device multiplier

a. **.OPTION YMFACT** must be specified for **M** to work.

**Note**



The above macromodel parameters can be declared in a number of ways, listed below in order of priority.

1. In the instantiation line, using the **.PARAM** keyword.
2. In a model command line. The syntax is as follows:

**.model mname modfas par=val [par=val]**

If this syntax is used, a model name must be declared using the **.MODEL** keyword in the instantiation line.

3. If no parameters are explicitly declared, the parameter default values are used.

## Model Equations

This element performs the linear Z-domain transfer function:

$$[I_{in}(z), I_{out}(z)] = Y(z, LDI)[V_{in}(z), V_{out}(z)]$$

in the time and frequency domain for the transient and small signal AC analyses where:

$I_{in}, I_{out}$  are the currents contributed to the pins **IN**, **OUT**.

$V_{in}, V_{out}$  are the voltages at the nodes **IN**, **OUT**.

$Y(z, LDI)$  is the characteristic transfer or admittance matrix of the model.

The Z-transfer matrix  $Y(z, \theta)$  is the Z-transform of the difference equations describing the time domain behavior of this element. Therefore the case of **LDI=0** is completely adequate to the time domain behavior.

In some SC Integrators the pin **OUT** is connected to the virtual ground of an ideal op-amp, in which case the transfer function becomes the following form:

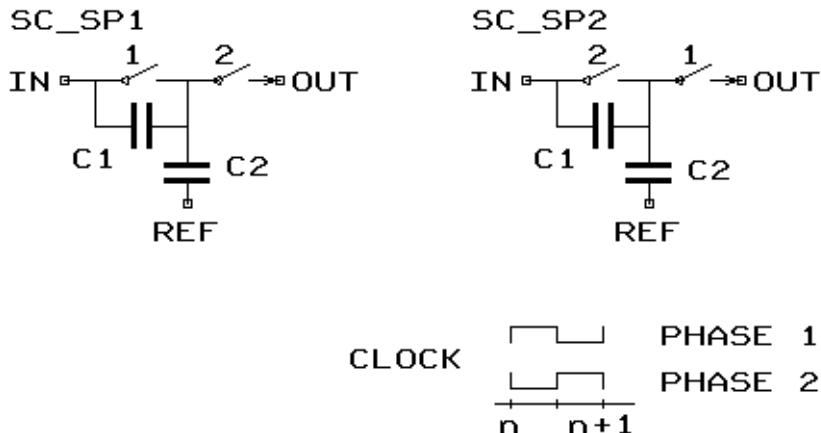
$$I_{out}(z) = -\frac{C}{T_p} \times V_{in}(z)$$

where:  $z^k = e^{j\omega(k \times T_p)}$ ,  $k \in \text{Integer}$

## Serial-parallel Switched Capacitor

```
YXX SC_SP1 [PIN:] IN OUT REF [PARAM: PAR=VAL {PAR=VAL}] [MODEL: MNAME]
YXX SC_SP2 [PIN:] IN OUT REF [PARAM: PAR=VAL {PAR=VAL}][MODEL: MNAME]
```

**Figure 9-16. Serial-parallel Switched Capacitor Macromodel**



These are basic building blocks for the Z-domain modeling of SC circuits and sampled data systems. They represent SC realizations of an analog resistor, the so called serial-parallel switched capacitor.

They have been modeled as voltage to current three port transfer elements with a characteristic transfer or admittance matrix  $Y(z)$ , used for the description of the behavior of these elements for both the transient and small signal AC analyses. The transfer function can be symbolically represented as shown above.

They are Kirchhoff type elements having a time-discrete I-U-dependence with charge storage and delay effects controlled by the uniform two-phase clock.

The model assumes, that all currents remain constant during each switching phase and that they only change their values at the beginning of every switching sub-interval.

### Model Pins

IN	Name of the positive pin to the switch branch controlled by clock phase 1 (or 2).
OUT	Name of the positive pin to the switch branch controlled by clock phase 2 (or 1).
REF	Name of the negative pin.

**Table 9-7. Serial-parallel Switched Capacitor Model Parameters**

Nr.	Name	Default	Units	Definition
1	<b>C1</b>	$1.0 \times 10^{-12}$	F	Capacitance
2	C2	$1.0 \times 10^{-12}$	F	Capacitance
3	<b>TP</b>	$1.0 \times 10^6$	s	Pulse period
4	<b>LDI</b>	0		See the note on LDI's in this chapter
5	M <sup>a</sup>	1		Device multiplier

a. **.OPTION YMFACT** must be specified for **M** to work.

**Note**

The above macromodel parameters can be declared in a number of ways, listed below in order of priority.

1. In the instantiation line, using the **.PARAM** keyword.
2. In a model command line. The syntax is as follows:

```
.model mname modfas par=val [par=val]
```

If this syntax is used, a model name must be declared using the **.MODEL** keyword in the instantiation line.

3. If no parameters are explicitly declared, the parameter default values are used.

## Model Equations

These elements perform the linear Z-domain transfer function:

$$[I_{in}(z), I_{out}(z), I_{ref}(z)] = Y(z, LDI)[V_{in}(z), V_{out}(z), V_{ref}(z)]$$

in the time and frequency domain for the transient and small signal AC analyses where:

$I_{in}, I_{out}, I_{ref}$  are the currents contributed to the pins IN, OUT, REF.

$V_{in}, V_{out}, V_{ref}$  are the voltages at the nodes IN, OUT, REF.

$Y(z, LDI)$  is the characteristic transfer or admittance matrix of the model.

The Z-transfer matrix  $Y(z, \theta)$  is the Z-transform of the difference equations describing the time domain behavior of this element. Therefore the case of **LDI=0** is completely adequate to the time domain behavior.

In some SC Integrators the pin N is connected to the virtual ground of an ideal op-amp, in which case the transfer function becomes the following form:

$$I_{out}(z) = -\frac{1}{Tp}(C_1 + C_2 z^{-1/2}) \times V_{in}$$

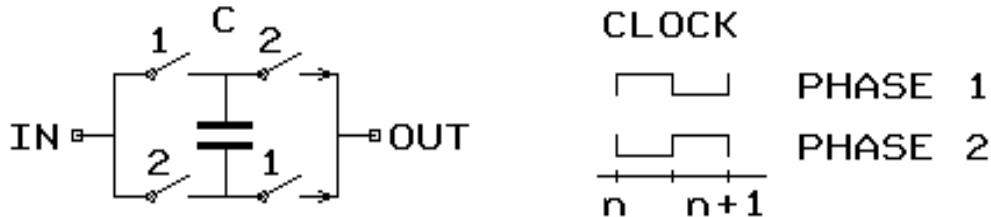
where:

$$z^k = e^{j\omega(k \times Tp)}, k \in \text{Integer}$$

## Bi-linear Switched Capacitor

**YXX SC\_B [PIN:] IN OUT [PARAM: PAR=VAL {PAR=VAL}] [MODEL: MNAME]**

**Figure 9-17. Bi-linear Switched Capacitor Macromodel**



This is a basic building block for Z-domain modeling of SC circuits and sampled data systems. It represents an SC realization of an analog resistor, the so called bi-linear switched capacitor.

It has been modeled as a voltage to current two port transfer element with a characteristic transfer or admittance matrix  $Y(z)$ , used for the description of the behavior of this element for both the transient and small signal AC analyses. The transfer function can be symbolically represented as shown in the above diagram.

It is a Kirchhoff type element having a time-discrete I-U-dependence with charge storage and delay effects controlled by the uniform two-phase clock.

The model assumes that all currents remain constant during each switching phase and that they only change their values at the beginning of every switching sub-interval.

### Model Pins

- |     |                         |
|-----|-------------------------|
| IN  | Name of the first pin.  |
| OUT | Name of the second pin. |

**Table 9-8. Bi-linear Switched Capacitor Model Parameters**

Nr.	Name	Default	Units	Definition
1	C	$1.0 \times 10^{-12}$	F	Capacitance
2	TP	$1.0 \times 10^6$	s	Pulse period
3	M <sup>a</sup>	1		Device multiplier

a. .OPTION YMFACT must be specified for M to work.

### Note

The above macromodel parameters can be declared in a number of ways, listed below in order of priority.

1. In the instantiation line, using the **.PARAM** keyword.
2. In a model command line. The syntax is as follows:

```
.model mname modfas par=val [par=val]
```

If this syntax is used, a model name must be declared using the **.MODEL** keyword in the instantiation line.

3. If no parameters are explicitly declared, the parameter default values are used.

## Model Equations

This element performs the linear Z-domain transfer function:

$$[I_{in}(z), I_{out}(z)] = Y(z, LDI)[V_{in}(z), V_{out}(z)]$$

in the time and frequency domains for the transient and small signal AC analyses where:

$I_{in}, I_{out}$  are the currents contributed to the pins **IN**, **OUT**.

$V_{in}, V_{out}$  are the voltages at the nodes **IN**, **OUT**.

$Y(z, LDI)$  is the characteristic transfer or admittance matrix of the model.

In some SC Integrators, the pin **OUT** is connected to the virtual ground of an ideal op-amp, in which case the transfer function becomes the following form:

$$I_{out}(z) = -\frac{C}{T_p} \times (1 + z^{-1}) \times V_{in}(z)$$

where:

$$z^k = e^{j\omega(k \times T_p)}, k \in \text{Integer}$$

## Unswitched Capacitor

**YXX SC\_U [PIN:] IN OUT [PARAM: PAR=VAL {PAR=VAL}] [MODEL: MNAME]**

**Figure 9-18. Unswitched Capacitor Macromodel**



This is a basic building block for Z-domain modeling of SC circuits and sampled data systems. It represents an SC realization of an unswitched capacitor.

It is a Kirchhoff type element having a time-discrete I-U-dependence with charge storage and delay effects controlled by the uniform two-phase clock.

The model assumes that all currents remain constant during each switching phase and that they only change their values at the beginning of every switching sub-interval.

### Model Pins

IN Name of the input pin.

OUT Name of the output pin.

**Table 9-9. Unswitched Capacitor Model Parameters**

Nr.	Name	Default	Units	Definition
1	C	$1.0 \times 10^{-12}$	F	Capacitance
2	TP	$1.0 \times 10^6$	s	Pulse period
3	M <sup>a</sup>	1		Device multiplier

a. .OPTION YMFACT must be specified for M to work.

The above macromodel parameters can be declared in a number of ways, listed below in order of priority:

1. In the instantiation line, using the **.PARAM** keyword.
2. In a model command line. The syntax is as follows:

```
.model mname modfas par=val [par=val]
```

If this syntax is used, a model name must be declared using the **.MODEL** keyword in the instantiation line.

3. If no parameters are explicitly declared, the parameter default values are used.

## Model Equations

This element performs the linear Z-domain transfer function:

$$[I_{in}(z), I_{out}(z)] = Y(z)[V_{in}(z), V_{out}(z)]$$

in the time and frequency domains for the transient and small signal AC analyses where:

$I_{in}, I_{out}$  are the currents contributed to the pins IN, OUT.

$V_{in}, V_{out}$  are the voltages at the nodes IN, OUT.

$Y(z)$  is the characteristic transfer or admittance matrix of the model.

The Z-transfer matrix  $Y(z)$  is the Z-transform of the difference equations describing the time domain behavior of this element.

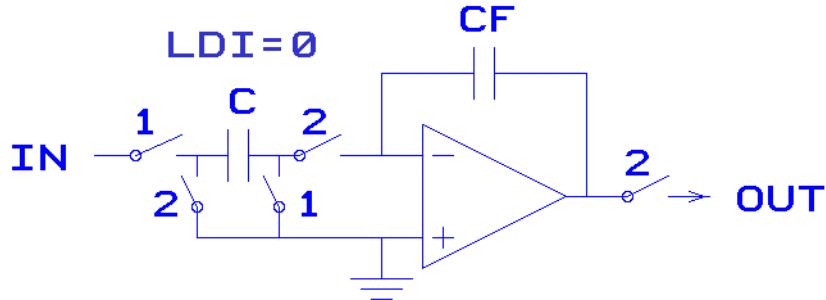
## Applications

The list below gives some of the applications of Switched Capacitor Macromodels which are illustrated throughout the following pages.

- Non-inverting Integrator.
- LDI Phase Control Non-inverting Integrator.
- Euler Forward Integrator.
- LDI Phase Control Euler Forward Integrator.
- Euler Backward Integrator.
- LDI Phase Control Euler Backward Integrator.

## Non-inverting Integrator

**Figure 9-19. Non-inverting Integrator**



The inverting switched capacitor is connected to the input of an op-amp with a feedback capacitor forming a non-inverting Integrator. In this configuration, both the input and output nodes of the op-amp are controlled by the same clock phase, namely clock phase 2, as shown above.

This is the *standard* use of the switched capacitor element and therefore  $LDI=0$ , which means that no additional delay operation has to be performed during an AC analysis.

This leads to the Z-domain transfer function:

$$v_{out}(z) = \frac{C}{CF} \times \frac{z^{-1/2}}{1 - z^{-1}} \times v_{in}(z)$$

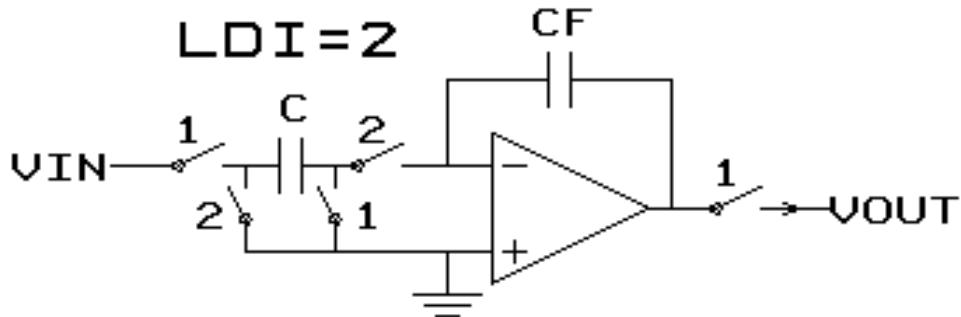
where:

$$z^k = e^{j\omega(k \times T_p)}, k \in \text{Integer}$$

which describes the behavior in the time and frequency domain for the transient and small signal AC analyses.

## LDI Phase Control Non-inverting Integrator

**Figure 9-20. LDI Phase Control Non-inverting Integrator Macromodel**



Here, the inverting switched capacitor is connected to the inverting input of an op-amp with a feedback capacitor forming a non-inverting Integrator. In this configuration the inverting input node and the output node of the op-amp are controlled by different clock phases.

This is the *non-standard* use of the switched capacitor element and therefore the **LDI** flag has to be set properly, which means that during an AC analysis, a delay operation of one half clock period has to be applied to the current through the switch branch of the SC element, which is connected to the input of the op-amp.

The **LDI** flag identifies this switch branch by the name (the number 1 or 2) of the clock phase, through which it is controlled. So **LDI=2** in the case of the above arrangement.

This leads to the transfer function:

$$v_{out}(z) = \frac{C}{CF} \times \frac{z^{-1}}{1 - z^{-1}} \times v_{in}(z)$$

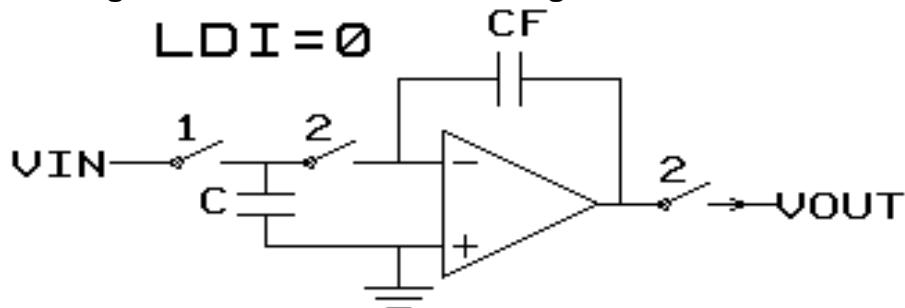
where:

$$z^k = e^{j\omega(k \times T_p)}, k \in \text{Integer}$$

which describes the behavior in the time and frequency domain for the transient and small signal AC analyses.

## Euler Forward Integrator

Figure 9-21. Euler Forward Integrator Macromodel



### Transfer Function

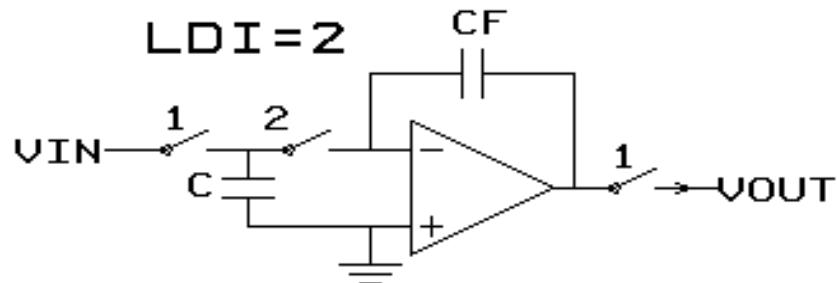
$$v_{out}(z) = -\frac{C}{CF} \times \frac{z^{-1/2}}{1 - z^{-1}} \times v_{in}(z)$$

where:

$$z^k = e^{j\omega(k \times T_p)}, k \in \text{Integer}$$

## LDI Phase Control Euler Forward Integrator

Figure 9-22. LDI Phase Control Euler Forward Integrator Macromodel



### Transfer Function

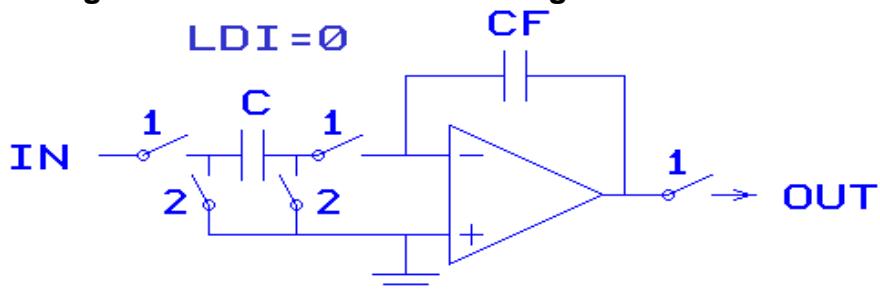
$$v_{out}(z) = -\frac{C}{CF} \times \frac{z^{-1}}{1 - z^{-1}} \times v_{in}(z)$$

where:

$$z^k = e^{j\omega(k \times T_p)}, k \in \text{Integer}$$

## Euler Backward Integrator

**Figure 9-23. Euler Backward Integrator Macromodel**



### Transfer Function

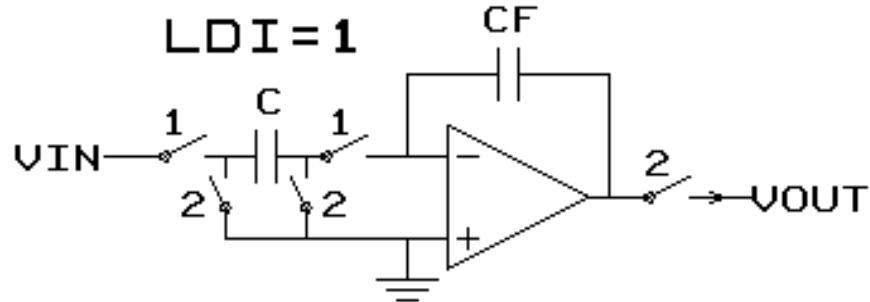
$$v_{out}(z) = -\frac{C}{CF} \times \frac{1}{1 - z^{-1}} \times v_{in}(z)$$

where:

$$z^k = e^{j\omega(k \times T_p)}, k \in \text{Integer}$$

## LDI Phase Control Euler Backward Integrator

**Figure 9-24. LDI Phase Control Euler Backward Integrator Macromodel**



How to deduct the transfer function of an SC Integrator with:

- The transfer function of the SC resistor:

$$Y_{nsc}(z) = C \times v_{in}(z) \times \text{del}_{LDI}$$

- The LDI-delay:

$$\text{del}_{LDI} = z^{-1/2}$$

- The transfer function of the unswitched capacitor in the feedback loop:

$$Y_{usc}(z) = C_F \times (1 - z^{-1})$$

- The KCL at the virtual ground node:

$$Y_{isc}(z) \times V_{in}(z) + Y_{usc}(z) \times V_{out}(z) = 0$$

We get the transfer function of the integrator as follows:

$$V_{out}(z) = -\frac{C}{C_F} \times \frac{z^{-1/2}}{1 - z^{-1}} \times V_{in}(z)$$

## Tutorial—SC Low Pass Filter

For a tutorial on the transient & small signal AC analysis of an SC low pass filter using the Z-domain switched capacitor models, please see “[Tutorial #9—SC Low Pass Filter](#)” on page 24-29.

# Chapter 10

## Simulator Commands

### Introduction

The following table shows a list of simulator commands available.

**Table 10-1. Eldo Commands**

Analysis	Display	Simulation control	Circuit description
.AC	.COMCHAR	.CALL_TCL	.A2D
.AGE	.DEFMAC	.CHECKBUS	.ADDLIB
.CHECKSOA	.DEFPLOTDIG	.CONSO	.AGEMODEL
.DC	.DEFWAVE	.CORREL	.ALTER
.DCMISMATCH	.EQUIV	.DATA	.BIND
.DSP	.EXTRACT	.DISCARD	.CHRENT
.DSPMOD	.EXTMOD	.DISFLAT	.CHRNSIM
.FOUR	.FFILE	.DISTRIB	.CONNECT
.LSTB	.IPROBE	.FORCE	.D2A
.MC	.MEAS	.FUNC	.DEFAULT
.NOISE	.MONITOR	.GUESS	.DEFMOD
.NOISETRAN	.NET	.IC	.DEL
.OP	.NEWPAGE	.INIT	.DSPF_INCLUDE
.OPTFOUR	.NOCOM	.LOAD	.END
.OPTIMIZE	.NOTRC	.LOTGROUP	.ENDL
.OPTNOISE	.PLOT	.MCMOD	.ENDS
.PZ	.PLOTBUS	.MODDUP	.GLOBAL
.RAMP	.PRINTBUS	.MPRUN	.HIER
.SENS	.PRINT	.NODESET	.IGNORE_DSPF_ON_NODE
.SESPARAM	.PRINTFILE	.NWBLOCK	.INCLUDE
.SNF	.PROBE	.OPTION	.LIB

**Table 10-1. Eldo Commands**

Analysis	Display	Simulation control	Circuit description
.SOLVE	.WIDTH	.OPTPWL	.LOOP
.TF		.OPTWIND	.MALIAS
.TRAN		.PARAM	.MAP_DSPF_NODE_NAME
.WCASE		.RESTART	.MODEL
		.SAVE	.MODLOGIC
		.SETBUS	.MSELECT
		.SETSOA	.PART
		.STEP	.PROTECT
		.SUBDUP	.SCALE
		.TABLE	.SETKEY
		.TEMP	.SIGBUS
		.TSAVE	.SINUS
		.USE	.SUBCKT
		.USE_TCL	.TITLE
			.TOPCELL
			.TVINCLUDE
			.UNPROTECT
			.USEKEY

Commands have been categorized according to their function.

- Analysis—Instructs the simulator to perform a specific analysis.
- Display—Used to control the messages and outputs generated from simulation.
- Simulation control—Used to modify general options for simulation.
- Circuit description—Describes the attributes of the circuit.

Clicking on a command will jump to the description of that command. Each command is described in greater detail throughout the chapter.

## Command Line Help

A simple online help can be accessed from the command line with:

```
eldo -help [commands|devices|sources|manual]
```

commands Simulator commands  
devices Device models  
sources Sources and Macromodels  
manual Full *Eldo User's Manual*

Each of the first three help options will open a link document in Acrobat Reader, which will then allow you to select the command, device model, source or macromodel you require information on.

Entering `eldo -help` without any option will display the list of available topics.

## **A2D**

### Analog-to-Digital Converter

```
.A2D [SIM=simulator] eldo_node_name
+ [digital_node_name] [MOD=model_name] [parameters_list]
```

The **.A2D** and **.D2A** statements establish the interface connection between Eldo and digital solvers.



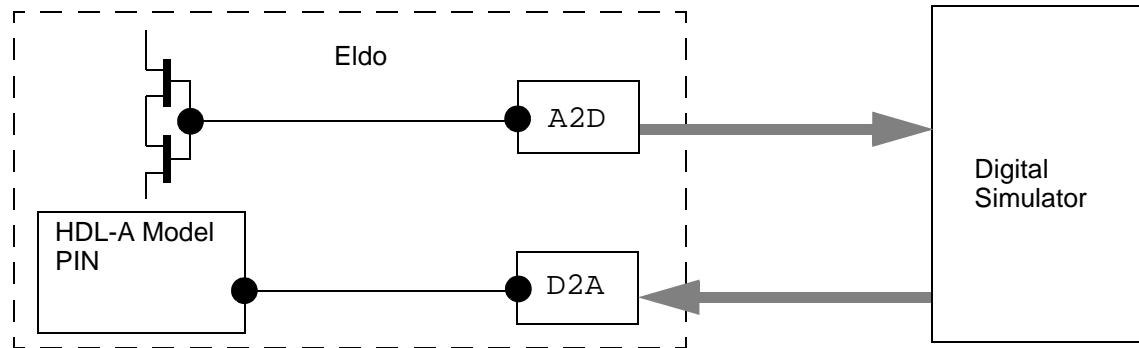
For Digital-to-Analog, please refer to the “[.D2A](#)” on page 10-44.

Currently, the supported digital solvers are HDL-A and Verilog.

- For Eldo/HDL-A, the converters may be used instead of an equivalent user's defined HDL-A model.
- For Eldo/Verilog, communication is done via the Cadence tool Verimix.

**Note**

For Eldo/Verilog communication done via the Cadence tool Verimix, the A2D and D2A syntax follows the grammar specified in the Cadence manual.



These converters transfer ‘analog’ Eldo information to some predefined ‘digital’ types. The following types are implemented: **BIT**, **X01Z**, **STD\_LOGIC** and **REAL**.

**BIT (SLOPE)**

Two logical values are possible: ‘0’ and ‘1’.

**X01**

Three logical values are possible: ‘X’, ‘0’ and ‘1’.

**X01Z (RC)**

Four logical values are possible: ‘X’, ‘0’, ‘1’ and ‘Z’.

**MVL4**

Same values as type **X01Z**. Eldo treats this type as the type **X01Z** for the A2D, but as the type **BIT** (Voltage source) for the D2A.

**STD\_LOGIC (MVL9)** Nine available logical values: ‘U’, ‘X’, ‘0’, ‘1’, ‘Z’, ‘W’, ‘L’ ‘H’ and ‘-’. This type corresponds to the VHDL std\_logic type defined in the STD\_LOGIC\_1164 multi-value logic system package in the IEEE library.

**REAL** Real or float values.

## Global Parameters

The following table shows a global view of the parameters of the .A2D:

**Table 10-2. .A2D - Global Parameters**

Name	Description	Default	Units
Common for all MODEs			
CIN	Capacitance of the digital gate input seen by Eldo	0	F
MODE= BIT   STD_VSRC			
VTH	Voltage threshold value for single threshold mode	2.5	V
MODE= X01   X01Z   MVL4			
VTH1 <sup>a</sup>	Lower voltage threshold value for two threshold mode, corresponding to the logical ‘0’ state	1.5	V
VTH2 <sup>a</sup>	Higher voltage threshold value for two threshold mode, corresponding to the logical ‘1’ state	3.5	V
TX	The ‘X’ state will be sent to the DIGITAL part only if the convertor remains in the ‘X’ state for more than TX seconds	0.0	s
MODE= STD_LOGIC			
STR	Defines the strength of the logic of the digital node. STRONG strength generates ‘X’, ‘0’, ‘1’, and WEAK strength generates ‘W’, ‘H’, ‘L’	STRONG	
VTH1	Lower voltage threshold value for two threshold mode, corresponding to the logical ‘L’ state if STR=WEAK, or ‘0’ if STR=STRONG	1.5	V
VTH2	Higher voltage threshold value for two threshold mode, corresponding to the logical ‘H’ state if STR=WEAK, or ‘1’ if STR=STRONG	3.5	V
TX	The ‘X’ state will be sent to the DIGITAL part only if the boundary element remains in the ‘X’ state for more than TX seconds	0.0	s
R	Resistive part of the impedance of the digital gate	infinity	Ω
C	Capacitive part of the impedance of the digital gate	0.0	F

**Table 10-2. .A2D - Global Parameters**

Name	Description	Default	Units
MODE= REAL			
EPS	Defines the real value threshold	$5 \times 10^{-3}$	V

a. **vth1** has to be specified lower than or equal to **vth2**. If not, the simulator will automatically invert **vth1** and **vth2** values so that **vth1 <= vth2**.

## Parameters

- **SIM=simulator**

The parameter **SIM** can take the value of the digital simulator's name: HDLA or VERILOG. The parameter **SIM** is mandatory only for HDL-A.

- **eldo\_node\_name**

Name of the node in the analog netlist.

- **digital\_node\_name**

Name of the node in the digital description. This parameter is optional. For HDL-A, the syntax is **Yxx:position** where **position** is the index of the port.

- **MOD=model\_name**

Name of the model used for the convertor. If **model\_name** is specified, there must be a corresponding **.MODEL** command:

```
.MODEL model_name A2D|ATOD parameters_list
```

- **parameters\_list**

List of available parameters as described below.

### Note

 For **eldo\_node\_name** connected to a PORT of an HDL-A model, **parameters\_list** is ignored. Default values can depend on simulators (HDL-A, Verilog, etc.).

## Parameters for A2D converters (.A2D)

For **.A2D** the default MODE is **X01** for all simulators when two thresholds are specified. If only one threshold is specified, the default MODE is **BIT**.

## Parameters Common for all Modes

- **CIN=value**

Capacitance of the digital gate input seen by Eldo (in Farads). Default value is 0.0.

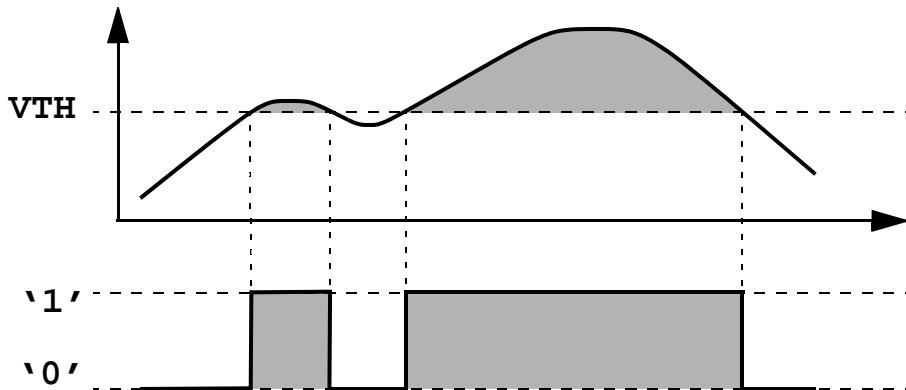
- **TX=value**

If **TX** is positive, then when the convertor mode is *not* of type BIT (i.e. when the convertor can generate logical ‘X’ states) the ‘X’ state will be sent to the DIGITAL part only if the convertor remains in the ‘X’ state for more than **TX** seconds. Default value is 0.0.

### Parameters for MODE=BIT

- **VTH=value**

Voltage threshold value for single threshold mode. Default value is 2.5V.



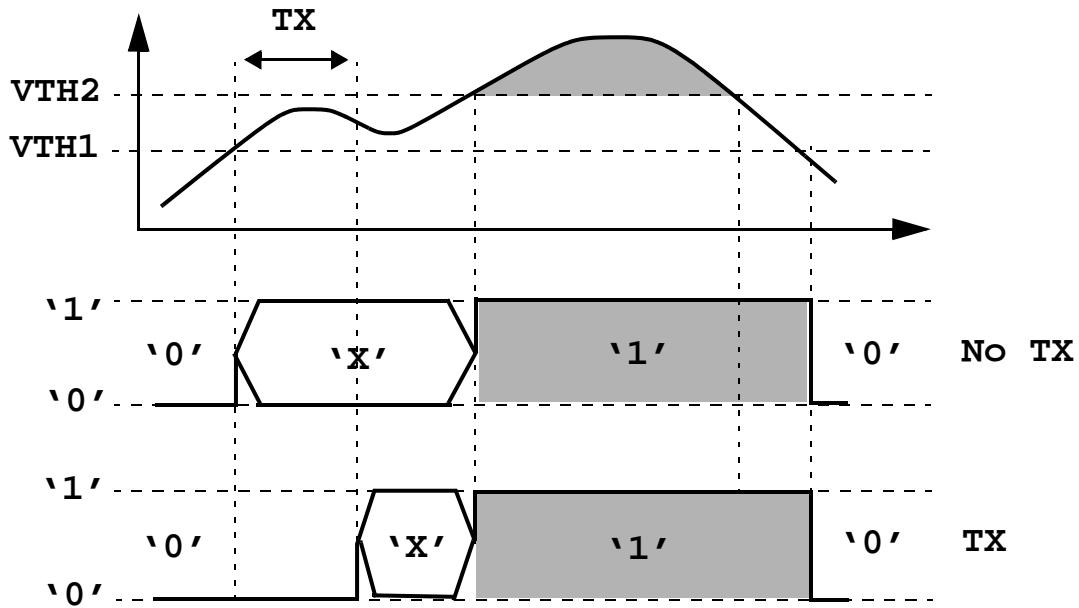
### Parameters for MODE=X01 | X01Z | MVL4

- **VTH1 | VOLTLLOW=value**

Lower voltage threshold value for two threshold mode, corresponding to the logical ‘0’ state. Default value is 1.5V.

- **VTH2 | VOLTHHIGH=value**

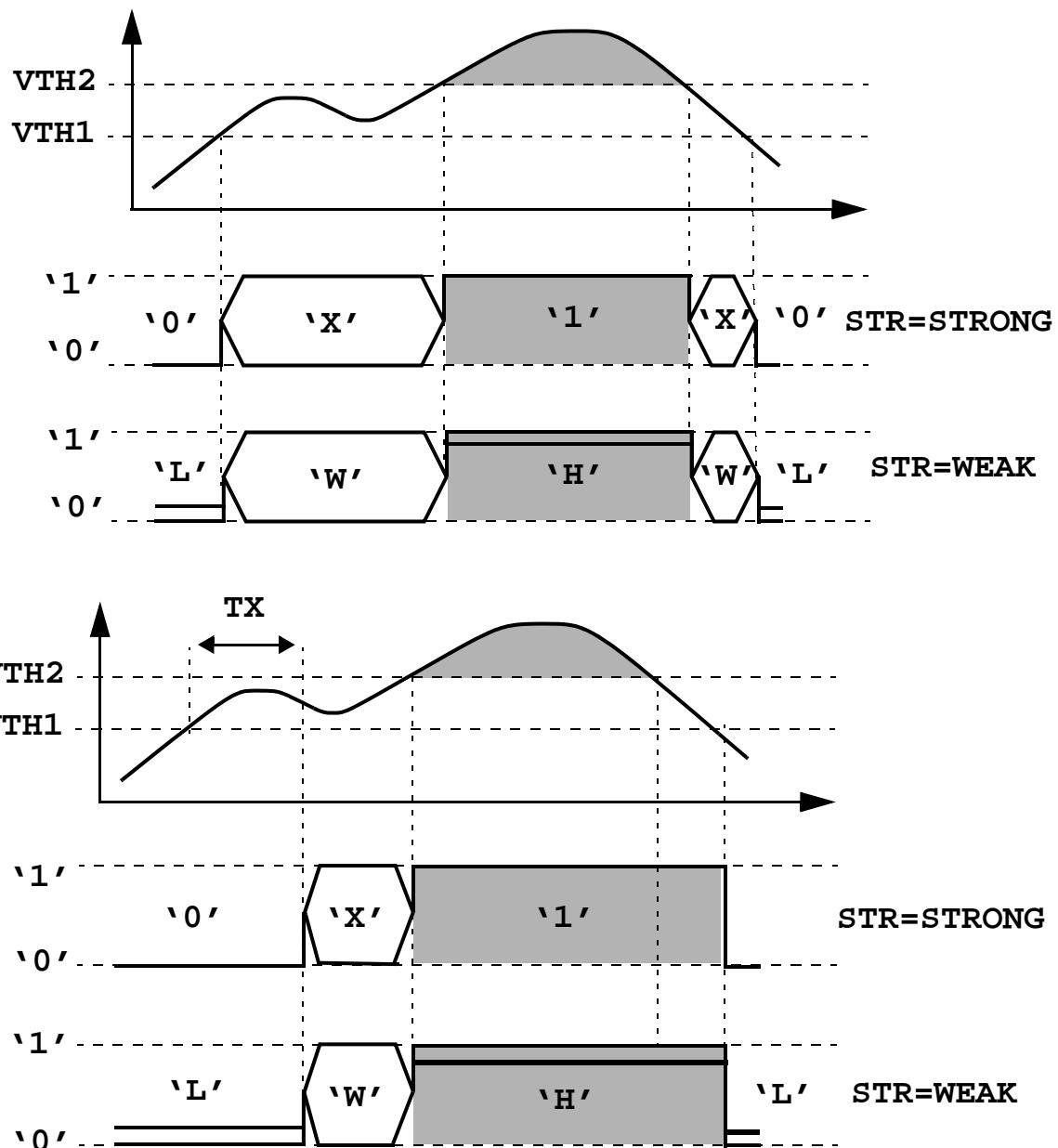
Higher voltage threshold value for two threshold mode, corresponding to the logical ‘1’ state. Default value is 3.5V.

**Note**

 An analog value falling between **VTH1** and **VTH2** gives a logical 'X' state. For **MODE=MVL4**, there is no analog values that can generate a logical 'Z' state.

### Parameters for MODE=STD\_LOGIC

- **STR=WEAK | STRONG**  
Defines the strength of the logic of the digital node. **STRONG** strength generates 'X', '0', '1', and **WEAK** strength generates 'W', 'H', 'L'. Default is **STRONG**.
- **VTH1 | VOLTLLOW=value**  
Lower voltage threshold value for two threshold mode, corresponding to the logical 'L' state if **STR=WEAK**, or '0' if **STR=STRONG**. Default value is 1.5V.
- **VTH2 | VOLTHHIGH=value**  
Higher voltage threshold value for two threshold mode, corresponding to the logical 'H' state if **STR=WEAK**, or '1' if **STR=STRONG**. Default value is 3.5V.
- **R=value**  
Resistive part of the impedance of the digital gate. Default value is infinity.
- **C=value**  
Capacitive part of the impedance (in Farads) of the digital gate. Default value is 0.0.



**Note**

An analog value falling between **VTH1** and **VTH2** generates a logical 'W' state if **STR=WEAK**, and a logical 'X' state if **STR=STRONG**.

### Z state detection on A2D nodes

To enable Eldo to detect an analog 'Z' state on A2D nodes, specify the **.OPTION ZDETECT** command within the netlist. This detection will *only* be performed on the A2D nodes for which an **RZ** value has been specified (either in the A2D card or via the **.model A2D** command). If the *global* value for **RZ** is set using the **.OPTION RZ=val** command then this will

enable Eldo to detect a ‘Z’ state on *all* A2D nodes. A ‘Z’ state is detected when the equivalent impedance of the A2D node exceeds the specified **RZ** value.

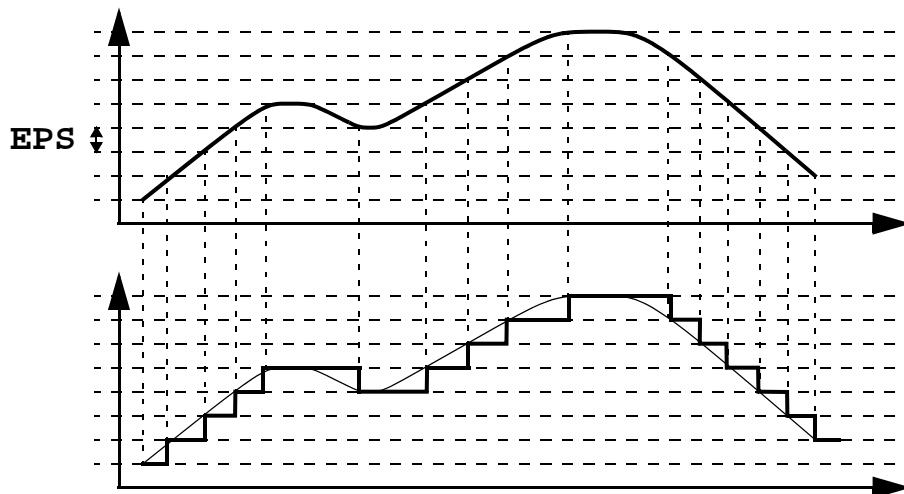
### Notes on usage

- Using the **.OPTION ZDETECT** command to detect the ‘Z’ state will result in extra CPU usage, which is typically an added few percent of the total simulation time.
- The ‘Z’ detection will *not* work if the A2D node is a Mach TA node.
- The detection of the ‘Z’ state is very sensitive to the choice of the time step around the time at which the ‘Z’ state should be detected. In some cases there may be a delay between, when the ‘Z’ state is *detected*, and when the ‘Z’ state *occurs*.

### Parameters for MODE=REAL

- **EPS=value**

Defines the real value threshold. When the real value coming from the digital simulator differs from the previous value by more than **EPS**, Eldo will take this into account. When the difference is less than **EPS**, the digital output is considered to be constant.



### Option DEFA2D

It is normally necessary to specify an explicit A2D between ANALOG nodes and HDL-A ports; however, with the following statement specified:

```
.OPTION DEFA2D=model_name
```

Eldo will automatically insert an implicit A2D convertor when needed. The following must be defined in the netlist:

```
.MODEL model_name A2D parameters
```

Furthermore, it is often necessary to specify **MODE=BIT** in the **.MODEL model\_name** command. This is because the default for **MODE** is X01, and very often HDL-A PORTS are of type **BIT**, hence a resulting error mismatch in convertor type would be printed if **MODE** is not correctly specified. Example:

```
.MODEL model_name A2D mode = BIT
.OPTION DEFA2D = model_name
```

Additionally, these default models must be found in the netlist itself, and cannot be specified via **.LIB** or **.ADDLIB** specifications.

---

#### **Note**

 With the option **DEFA2D**, for any ANALOG node connected directly to an HDL-A **IN** port, an implicit A2D will be automatically inserted.

---

### Option D2DMVL9BIT

Enables direct connection between HDL-A ports of type **BIT** with mixed-mode nodes connected to Eldo via convertors of type **MVL9** (or **std\_logic**).

The type **std\_logic** is an enumerated type. The type **BIT** is a subset of **std\_logic**, so a conversion table is required as follows.

**Table 10-1.**

A2D mode <b>std_logic</b>	HDL-A PORT bit type
‘0’, ‘L’	‘0’
‘1’, ‘H’	‘1’
“others”	ignored

## **.AC**

### AC Analysis

#### Parameter-Driven Analysis

```
.AC TYPE nb fstart fstop [SWEEP DATA=dataname] [UIC] [MONTE=val]
.AC TYPE nb fstart fstop [SWEEP parameter_name TYPE nb start stop]
+ [UIC] [MONTE=val]
.AC TYPE nb fstart fstop [SWEEP parameter_name start stop incr]
+ [UIC] [MONTE=val]
```

#### Data-Driven Analysis

```
.AC DATA=dataname [SWEEP DATA=dataname] [UIC] [MONTE=val]
.AC DATA=dataname [SWEEP parameter_name TYPE nb start stop]
+ [UIC] [MONTE=val]
.AC DATA=dataname [SWEEP parameter_name start stop incr] [UIC] [MONTE=val]
```

#### List-Driven Analysis

```
.AC LIST {list_of_frequency_points}
+ [SWEEP DATA=dataname] [UIC] [MONTE=val]
.AC LIST {list_of_frequency_points}
+ [SWEEP parameter_name TYPE nb start stop] [UIC] [MONTE=val]
.AC LIST {list_of_frequency_points}
+ [SWEEP parameter_name start stop incr] [UIC] [MONTE=val]
```

#### Adaptive Analysis

```
.AC ADAPTIVE tolerance_value fstart fstop
```

The **.AC** command activates the small signal analysis which computes the magnitude and phase of output variables as a function of frequency. The simulator first computes the circuit DC operating point with user specifiable initial conditions; thereafter the behavior of the linearized circuit is computed for a sine input of user specifiable amplitude, phase and frequency range.

Unit amplitude and zero phase gives the circuit transfer function. The inputs are specified via voltage or current sources which have an AC parameter.

The AC results on circuits containing FNS can be altered by the insertion of a **.PZ** statement in the netlist. The reason being that the **.PZ** statement implies the writing of FNS equations in such a way that is not as stable as the Eldo native FNS equation. However, this effect can be seen only at very high frequencies.

Whenever a **.AC** command, a **.TRAN** command, and a **.OP** command containing time specifications are in the *.cir* file, then an AC analysis will be performed at each of these time points. Please see the description of [AC in the middle of a .TRAN](#) for further details.

This command can also be used to perform a sweep on a circuit parameter or device.

The amount of information in the *.chi* file relating to operating point can be limited using the option **PRINT\_ACOP**, see [page 11-49](#).

An adaptive variant of AC analysis can be used when there are ports in the design to generate an accurate touchstone file. Eldo will then adjust the frequency in order that sharp peaks in the AC response are not missed.

---

**Note**

The results of AC analysis runs are output using **.PRINT** and **.PLOT** commands.

---

## Parameters

- **TYPE**

Can be one of the following:

**DEC**

Keyword to select logarithmic variation.

**OCT**

Keyword to select octave variation.

**LIN**

Keyword to select linear variation.

**POI**

Keyword to select a list of frequency points. **POI** is the same as **LIST** except that **POI** expects the number of points (NB) to be specified as it's first argument.

**INCR**

Can only be used when the **SWEEP** keyword is preceding it. See also the **INCR** keyword on [page 10-15](#).

- **DATA=dataname**

Used in conjunction with the **.DATA** command. The **dataname** parameter should be specified using the **.DATA** command. Please refer to the **.DATA** command on [page 10-51](#) for more information.

- **LIST**

Keyword to select a list of frequency points.

- **nb**

Number of points per decade or octave or points over the range from **fstart** to **fstop**. This is determined by the preceding keyword.

- **fstart**

Start frequency in Hertz. This can be specified as a parameter or as an expression.

- **fstop**  
Stop frequency in Hertz. This can be specified as a parameter or as an expression.
- **list\_of\_frequency\_points**  
List of frequency points which can be specified as parameters or as an expression. Values of frequencies must be separated by spaces. When the **LIST** keyword is specified, the values must be in increasing order, the order can be increasing or decreasing with the **POI** keyword specified.
- **UIC**  
If **UIC** is specified, no DC analysis is performed before the AC analysis. Instead the circuit may be initialized using **.RESTART** or **.USE**.  
If the user has in his input deck both of the following commands: **.RESTART file\_name**, where **file\_name** comes from the results of a TRAN simulation, and if there is a **.AC ... UIC** command, then the **.RESTART** will be interpreted as:  

```
.USE file_name OVERWRITE_INPUT
```



Refer to the section on **.RESTART** on [page 10-272](#) for more information.

- **MONTE=val**  
Monte Carlo analysis. Equivalent to **.MC val**. The syntax allows a different **MONTE** value for each run. However, the actual implementation in Eldo does not account for that: it is the last **MONTE** value to be specified in the netlist which will be taken into account.
- **ADAPTIVE**  
Performs an adaptive AC analysis. Useful if there are ports in the design to have an accurate touchstone file generated. Eldo will adjust the frequency in order that sharp peaks in the AC response are not missed. Otherwise, a large number of points would need to be requested in the AC command in order not to miss such transitions, resulting in very large touchstone files, if any.
- **tolerance\_value**  
Specified with an adaptive AC analysis. Tolerance value should be in the range [0.01:0.5]. The smaller the value, the tighter the results. The larger the value, the looser the results.

## Sweep Parameters

This section contains **SWEEP** related parameters that are previously unspecified in the Parameters section.

- **SWEEP**  
Specifies that a sweep should be performed on a parameter or device name.
- **parameter\_name**  
Name of the parameter or device name to sweep.

- TYPE

As defined previously (DEC, OCT, ...). When **INCR** is specified after the **SWEET** keyword, then the functionality is as follows:

#### **INCR**

Increment of the parameter or device name to sweep. When **INCR** is specified as the **TYPE** parameter, the value which directly follows (nb) is the incrementing value.

- nb

Number of points per decade or octave or points over the range from **start** to **stop**. This is determined by the preceding keyword.

- start

Start value of the sweep.

- stop

Stop value of the sweep.

- incr

Increment of the parameter or device name to sweep.

#### **Note**



When **INCR** is specified as the **TYPE** parameter, the value which directly follows (nb) is the incrementing value. If **INCR** is not specified, the incrementing value (incr) must be placed after the **start** and **stop** values.

## Examples

```
.ac dec 10 1 1g sweep r2 INCR 10 50 500
.ac dec 10 1 1g sweep r2 50 500 10
```

The two lines above are equivalent. Either can be used to specify a AC analysis from 1Hz to 1GHz with 10 analysis points per decade. The sweep specification will force Eldo to carry out an AC analysis on each value of **r2** starting at  $50\Omega$  and stopping at  $500\Omega$  with an incrementing value of  $10\Omega$ .

```
vin 2 0 ac 0.5
r1 2 3 5k
c3 3 0 0.1p
.ac dec 10 10e+4 10e+8
.plot ac vdb(3)
```

Specifies an AC analysis from  $10 \times 10^4$ Hz to  $10 \times 10^8$ Hz with 10 analysis points per decade. The AC input parameters are defined using the **ac** parameter in the voltage source definition. The results of the AC analysis are plotted in dB for the voltage at node 3 of the circuit in the limits specified in the **.ac** command.

```
v1 1 0 dc 5 ac 0.2
r1 1 2 1k
c1 2 0 1p
.ac oct 8 10e+6 10e+8
.plot ac vdb(2)
```

Specifies an AC analysis from  $10 \times 10^6$ Hz to  $10 \times 10^8$ Hz with 8 analysis points per octave. The input parameters are defined using the **ac** parameter in the voltage source definition.

The results of the AC analysis are plotted for the voltage in dB at node 2 of the circuit within the limits specified in the **.ac** command.



Examples of this type of analysis can be found in the [Tutorials](#) chapter.

Parameters are allowed in AC analysis commands as shown in the following example:

```
.param p1 = 1e9
.ac dec 10 1 p1
```

The following example shows the use of an adaptive AC analysis:

```
* S-par extraction

l1 1 1x 1e-7
r1 1x 2x 10
c1 2x 0 3e-12

T1 2x 0 3x 0 z0=50 Td=ln

c2 3x 0 6e-12
12 3x 4x 2e-7
r2 4x 2 8

T2 2x 0 5x 0 z0=40 Td=1.5n

Vp 5x 0 0
c11 1 0 10p
c22 2 0 10p

L3 3x 6x 2e-7
R3 6x 5x 2

I1 0 1 iport=1 rport=50
I2 0 2 iport=2 rport=50

.extract ac yval(sdb(1,1),1.45e8)

.Ffile S sb1.s2p Hz RI
.plot ac sdb(1,1)
.ac adaptive 0.05 1e5 1e11
```

You will see from the simulation results that Eldo computes some pole-zeros otherwise not seen with a standard AC analysis.

## AC in the middle of a .TRAN

Whenever a **.AC** command, a **.TRAN** command, and a **.OP** command containing time specifications are in the *.cir* file, then an AC analysis will be performed at each of these time points.

In the following example Eldo will perform AC, NOISE and TRAN analyses at time 0, time 5n, and time 7n:

```
.AC DEC 10 1 1e9
.NOISE v(5) vin 70
.TRAN 1n 20n
.OP 5n 7n
```



Please see “**.NOISE**” on page 10-180, “**.OP**” on page 10-187 and “**.TRAN**” on page 10-318 for more details on these commands.

In case there are some **.EXTRACT** commands related to AC analyses, extract information will be returned for each AC analysis.

When the output is cou format, an extra file, *<circut>\_tac.cou* is created that holds the results of those AC simulations which are performed from within a transient simulation.

When the output is JWDB, a folder TRAN is created with several AC folders inside, for example AC\_1, AC\_2.

Limitations:

1. AC results performed from within a transient simulation will not be dumped in any of the GWI, PSF, and WSF files.
2. If there is a **.TEMP** or a **.STEP** command in the input file, the *.ext* file which normally gets created will not be created in this instance.
3. Information related to **.EXTRACT SWEEP** will not be available.
4. Monte Carlo (**.MC**), Worst-Case (**.WC**), or transient-noise (**.NOISETRAN**) simulations are disabled.

## **.ADDLIB**

### Insert a Model or Subcircuit File

**.ADDLIB N DIR\_NAME**

This command is used to search a directory for files with file extensions *.mod* and *.ckt* (or *.sub*) and include in the circuit description (*.cir*) file the contents of those files whose names correspond to model and subcircuit definitions (respectively) referenced and not defined in the *.cir* file. The **.ADDLIB** command includes a parameter to control the order in which directories are searched.

#### **Note**



When the **.ADDLIB** command is used with **.OPTION MODWL** inside a circuit, Eldo searches for all the files <model\_name>xxxx.mod inside the directories specified inside **.ADDLIB** commands.

The following items need to be stressed for upper and lower case character naming rules conventions:

1. A filename must be in all upper or all lower case and appended with a lower-case *.mod* or *.ckt* or *.sub*.
  - .mod* for models only
  - .sub* or *.ckt* for subcircuits only
2. Files with upper-case are searched first. For example:

My\_2N2222a.mod (not searched)  
 MY\_2N2222A.mod (searched first)  
 my\_2n2222a.mod (searched last)

3. For subcircuits: *.ckt* extension is searched first. *.sub* is searched last. For example:

op11A.ckt (not searched)  
 OP11A.ckt (searched first)  
 op11a.ckt (searched second)  
 OP11A.sub (searched third)  
 op11a.sub (searched last)

### Parameters

- **N**

An integer between 1 and 6, allocating a priority to the directory search command. When several **.ADDLIB** commands are included in a circuit description file, then **N** may be used to determine the order in which the directories are searched. The number 1 allocates the highest priority.

- DIR\_NAME

Name of the directory to be searched. The names of the files have to be the same as the subcircuit or model names specified in the netlist.

### Example

```
mn1 a b c d mna w=w l=l
...
.addlib 2 /users1/examples/
.addlib 1 /users1/models/
```

Specifies that all models and subcircuits missing in the input netlist should be searched for and extracted from *.mod*, *.ckt* or *.sub* files contained in the directories </users1/models/> and </users1/examples/> in this order. The file containing the model *mna* has to be named *mna.mod*.

## **.AGE**

### Age Analysis

```
.AGE [ TAGE=value ] [ NBRUN[S]=value ]
+ [ LIN[={YES|ON|1}|{NO|OFF|0}]] [ LOG[={YES|ON|1}|{NO|OFF|0} ] ]
+ [ TSTART=value ] [ TSTOP=value ]
+ [ MODE=AGESim | AGEload | AGESave ] [ AGELIB=file_name ]
+ [ AGEALL[={YES|ON|1}|{NO|OFF|0} ] ]
+ [ ASCII[={YES|ON|1}|{NO|OFF|0} ] ]
+ [ COMPUTE_LAST[={YES|ON|1}|{NO|OFF|0} ] ]
+ [ PLOT={FRESH_FINAL|ALL} ]
+ [ TUNIT=year|month|day|hour|min]
+ [ AGEDSIM={YES|ON|1}|{NO|OFF|0} ]
+ [ LOGMODE_MINEXP=<VALUE> ]
```

This command activates the Age analysis to test the reliability of a netlist.



For the complete description of Age analysis and information on all reliability commands, see the separate chapter [Reliability Simulation](#).

## .AGEMODEL

### Reliability Model Parameter Declaration

```
.AGEMODEL MODEL=model_name [parameter=value]
```

This reliability analysis command allows the user to specify the model parameters related to reliability calculations.



For the complete description of this command and information on all reliability commands, see the separate chapter [Reliability Simulation](#).

---

# **ALTER**

## Generalized Re-run Facility

```
.ALTER [LABEL]
[ELEMENT]
[SUBCKT]
[COMMAND]
[COMMENT]
.ALTER | .END
```

Used to re-run Eldo with a modified netlist. All Eldo statements between the **.ALTER** command line and either the next **.ALTER** or **.END** command are back-substituted in the original netlist except for certain commands (see below) which are added to the netlist rather than back-substituted.

The system searches for a match of the component, model, subcircuit or command names in the original netlist and when a match is found the new line is substituted for the original. When no match is found, the new line is simply added, allowing modification of the topology of the circuit.

The following Eldo commands are always added to the netlist with no substitution being attempted.

```
.PRINT, .PLOT, .CONSO, .CONNECT, .EXTRACT, .GLOBAL, .INCLUDE, .OP,
.OPTION, .PARAM, .SENS
```

If a **.EXTRACT** command is present inside a **.ALTER** command, a *.ext* file will be created on the condition that the **.EXTRACT** statement remains unchanged for each **.ALTER** file. If this condition is not met, the *.ext* file is not created.

In a series of **.ALTER** commands, all use the initial netlist as the reference to be altered, not the result of the previous **.ALTER** command.

If one of the **.ALTER** commands produces an error of non-convergence, Eldo will proceed with the next **.ALTER** command.

The *circuit\_name.chi* ASCII log file contains a trace of all modifications made. The user may annotate the modification trace label string printed in the *.chi* file using the following syntax:

```
.ALTER <string>
```

Use option **ALTER\_SUFFIX** to switch the naming convention for swept waves used in **.ALTER** statements between: xxx and xxx\_alter:XX.

For JWDB output, the alter index number can be viewed when highlighting waveforms in the EZwave Waves window.

For cou output, the label information is added to the waveform. It can be viewed in Xelga with **Globals > Wave names**; in the dialog box, set **Sim identifier** as **Parameters**; this will display

the label appended to the wave name followed by the alter number (`_ALTER=0`, `_string=1`) instead of the alter number only (`_ALTER=0`, `_ALTER=1`).

**Note**

 The **.ALTER** command may not be used in conjunction with the **.ADDLIB** command. In order to use **.ALTER** with the **.LIB** command, the **KEY** parameter in the **.LIB** command must be used.

**.DEL LIB** can be used in the **.ALTER** section. The library name specified in the **.DEL LIB** command will be removed from the nominal description.

**.MPRUN** can be used to take advantage of multi-processor machines for the **.ALTER** command.



Please see “[.MPRUN](#)” on page 10-166 for further information.

Error handling of multiple run netlists can be managed with option **STOPONFIRSTERROR**. When set to 2, the first error stops the simulation even if the netlist file contains **.ALTER** statements for multiple simulation runs. When set to 1, if the netlist file contains **.ALTER** statements, Eldo will stop on the first **.ALTER** that has an error, but continue with the remaining **.ALTER** statements.

To replace one file by another one when using the Eldo re-run facility, use the option **ALTINC**. This forces Eldo to replace the first **.INCLUDE** statement found in an input netlist by the first **.INCLUDE** statement found in the **.ALTER** section of the netlist.

## Related Options

[ALTER\\_SUFFIX](#), page 11-44, [STOPONFIRSTERROR](#), page 11-12, [ALTINC](#), page 11-7.

## Examples

```
alter command example
r1 1 2 1k
r2 2 0 1k
c1 2 0 1n
.ac dec 10 200 1000meg
vin 1 0 ac 1
.plot ac vdb(2) (-90, 0)
.alter
r1 1 2 10k
c1 2 0 100p
.ac dec 15 200 1500meg
.end
```

Specifies two simulation runs. Both perform an AC analysis but certain component values, the number of points and stop frequency of the AC analysis are changed using the **.alter** command for the second run.

The example below demonstrates the use of the **KEY** parameter in conjunction with the **.ALTER** command.

```
...
.lib key=K1 /work/bip/mymod typ
.lib key=K2 /work/mos/mymod typ
...
.alter
.lib key=K2 /work/mos/mymod best
.alter
.lib key=K2 /work/private/mymod typ
.end
```

This command sequence causes three simulations to be performed always with library /work/bip/mymod typ.

In the first simulation, library /work/mos/mymod typ is used.

In the second simulation, library /work/mos/mymod best is used.

In the third simulation, library /work/private/mymod typ is used.

---

#### **Note**



If there is an error in a **.ALTER** command, it will be skipped and the next **.ALTER** command will be used.

---

The following example shows option **ALTER\_SUFFIX** can be used to switch the naming convention for swept waves used in **.ALTER** statements between: xxx and xxx\_alter:XX. Run the circuit with and without the option and look at the difference in results.

```
*.option alter_suffix=0

.param VTEST1=1
vdc 1 0 dc VTEST1
vin 2 0 dc 0
v2 4 0 dc 1

etest 3 0 value={mult*(2-v(1,2)*v(1,4))}

r1 3 0 rval

.dc vin 0 2 0.1
.step param VTEST1 0.5 1.5 0.1

.plot dc i(r1)

.extract label=imax max(i(r1))
.extract label=imin min(i(r1))

.defwave sweep TEST=meas(imax)

.param RVAL=1k
.param MULT=1
.alter first
.param RVAL=1.3k
.param MULT=1.3
.alter second
```

```

.param RVAL=0.7k
.param MULT=1.3
.alter
.param RVAL=0.7k
.param MULT=0.7
.alter fourth
.param RVAL=1.3k
.param MULT=0.7

.end

```

The following example shows how to replace one included file by another one when using the Eldo re-run facility, with the option **ALTINC**. The *new\_models* file will be included in the re-run simulation instead of *models*.

```

.include models
.option altinc
r1 1 0 rval
v1 1 0 dc 1
.extract dc i(r1)
.dc
.alter
.include new_models

```

### **-compat flag**

Usually in Eldo, **.ALTER** restarts from the original netlist, however, with the **-compat** flag set, **.ALTER** is cumulative. For example:

```

r1 1 0 1
r2 2 0 1
.alter 1
r1 1 0 2
.alter 2
r2 2 0 2

```

If Eldo is running with the **-compat** flag set, the second “alter” simulation will be done with both **r1** set to 2 (inheritance from later number 1), and **r2** set to 2.

If the **-compat** flag is omitted, then the second “alter” simulation will be run with **r1** set to 1 (original netlist) and **r2** set to 2.

## **.BIND**

### Configure Spice Descriptions

```
.BIND inst=inst_name | from_part to_part [mapping=assoc_file_name]
.BIND inst=inst_name from_part to_part [mapping=assoc_file_name]
```

Change one SPICE description (for example, one automatically generated by any schematic tool) to another equivalent description without editing the description itself (or netlisting it again).

The **.BIND** command is used to substitute any SPICE subcircuit by either another SPICE subcircuit or a behavioral model (VHDL, VHDL-AMS, Verilog or Verilog-AMS).

---

 For more information on **.BIND**, see page 5-35 of the *ADVance MS User's Manual*.

---

### Parameters

- **inst=inst\_name**

If the instance name is not provided, it is equivalent to any subcircuit (in this case, only the subckt name is used).

- **from\_part**

**from\_subckt=Eldo\_subckt\_name | from\_model=HDL\_model\_name**

**from\_subckt=Eldo\_subckt\_name**

If the Eldo subckt name is not provided, it is equivalent to any subcircuit (in this case, only the instance name is used). If not specified, all the subcircuits will be replaced by the specified subcircuit or HDL model. This can be restricted by specifying **inst=inst\_name**.

**from\_model=HDL\_model\_name**

If model name is not provided, it is equivalent to any model (in this case, only the instance name is used). If not specified, all the HDL models will be replaced by the specified subcircuit or HDL model. This can be restricted by specifying **inst=inst\_name**.

- **to\_part**

**to\_subckt=new\_Eldo\_subckt\_name [file=file\_name variant=variant\_name]**  
| **to\_model=new\_HDL\_model\_name**

**to\_subckt=new\_Eldo\_subckt\_name**

The Eldo subcircuit name that will replace the existing SPICE or behavioral model(s). The Eldo subcircuit can be replaced with another Eldo subcircuit of the same name from a different library. To do this use the arguments **file=file\_name** and **variant=variant\_name**.

**file=file\_name**

Name of the file that contains the library.

**variant=variant\_name**

Name of the variant.

**to\_model=new\_HDL\_model\_name**

The behavioral model name that will replace the existing SPICE or behavioral model(s).

- **mapping=assoc\_file\_name**

Specifies the interface association file to be applied to the port mapping between the model to replace and the new model. This is optional. By default, a mapping by position is used when this file is not provided.

Wildcards may be used either for the instance name or the subcircuit name that will be replaced (`inst` or `from_subckt`), but not for the `.MODEL` card or subcircuit that is used in replacement (`to_subckt` or `to_model`).

## Example

Suppose that we have a SPICE description called `top.cir`. To substitute all instances of subcircuit `invertor` that are part of the sub-instance called `X1.X2` by the Verilog model `invertor` compiled in the working library, a new file called `new-top.cir` can be written as follows:

```
* file new-top.cir
.INCLUDE top.cir
* substitution model declaration
.MODEL new_invertor macro lang=verilogams
+ mod=invertor param: delay=10ns
* substitution command
.BIND inst=x1.x2.*
+ from_subckt=invertor to_model=new_invertor
```

## **.CALL\_TCL**

### Call Tcl Function

```
.CALL_TCL [ TRAN|AC|DC|... ]
+ WHERE=START|START_OF_RUN|END_OF_RUN|END
+ [ PLOT=[ YES|NO|0|1 ] ] [ LABEL=alias_name ] tcl_function_call
```

This command is used to perform a single call to a Tcl function.

This allows you to dynamically manage User Defined Functions (UDF) written in Tcl language. The functions of the post-processor library and other commands are available through the Tcl interface.



For loading a Tcl file containing functions into Eldo's Tcl interpreter, see the command [“.USE\\_TCL”](#) on page 10-331. For further information on both commands, see the [Post-Processing Library](#) chapter of this manual.

### Parameters

- WHERE

Specify when Eldo must call the tcl function:

START

Only one call at the very beginning of the simulation.

START\_OF\_RUN

One call at the beginning of each run.

END\_OF\_RUN

One call at the end of each run.

END

One call at the very end of the simulation.

- PLOT

If the Tcl function returns a wave, this keyword asks Eldo to dump the wave in the output file.

- LABEL

This name will be used to plot the wave in the output file.

- tcl\_function\_call

The Tcl function. A function can take any kind of arguments: waves, numbers and keywords. These keywords are: IRUN, ICARLO, IALTER which respectively represent the index of the current step, Monte Carlo run, and **.ALTER**. NBRUN, NBCARLO, NBALTER which represent the number of steps, Monte Carlo runs, and **.ALTER**.

## Example

```
.call_tcl tran WHERE=END_OF_RUN
+ MY_FUNCTION(v(1),2.0,irun)
```

## .CHECKBUS

### Check Bus Values

```
.CHECKBUS BNAME [VTH[1]=VAL1 [VTH2=VAL2]]  
+ [BASE=DEC|OCT|BIN|HEX] [LOCK=1] TN VAL {TN VAL} [REPORTX=0|1]  
.CHECKBUS BNAME [VTH[1]=VAL1 [VTH2=VAL2]] [TSAMPLE=VAL3]  
+ [TDELAY=VAL4] [BASE=DEC|OCT|BIN|HEX] [LOCK=1] PATTERN BITS {BITS}  
+ [REPORTX=0|1]
```

Checks that the value of the bus has the value `VAL` at time `TN`. If an error occurs, the list of the errors are printed inside the `.chi` file. It is also possible to specify expected values using the **PATTERN** function:

### Parameters

- **BNAME**  
Bus name.
- **TN**  
Time in seconds at which the bus signal should be equal to `VAL` volts.
- **VAL**  
Bus signal voltage level at time `TN`.
- **PATTERN BITS {BITS}**  
Bus signal voltage levels representing the **PATTERN** source. For instance, the 4<sup>th</sup> `VAL` specified is the voltage level at time `TSAMPLE×4 + TDELAY`. Integer values can be specified, a bus value specifying `base=bin` pattern 101 has the same effect as specifying `base=dec` pattern 5.
- **VTH[1]=VAL1**  
The `VTH` parameters are required by Eldo to compute the HEX (or DEC, OCT, or BIN) value on the bus from the analog value inside Eldo. Then, it compares this value with the value expected and displays the error if they are not the same.
- **VTH2=VAL2**  
This can be used to plot the indeterminate value as shown below:
  - When only `VTH1` is given:  
If value < `VTH` then logic state 0.  
If value > `VTH` then logic state 1.
  - When both `VTH1` and `VTH2` are given:  
If value < `VTH1` then logic state 0.  
If `VTH1` < value < `VTH2` then state X.  
If value > `VTH2` then logic state 1.

- **BASE**

Keyword indicating that the bus signal number system is to be defined. The default number system is decimal.

**OCT**

Keyword indicating that bus signals are defined in octal.

**DEC**

Keyword indicating that bus signals are defined in decimal.

**BIN**

Keyword indicating that bus signals are defined in binary.

**HEX**

Keyword indicating that bus signals are defined in hexadecimal.

- **TSAMPLE=VAL3**

Time spent at 1 or 0 **PATTERN** value.

- **TDELAY=VAL4**

Delay before the **PATTERN** series is started.

- **LOCK=1**

Forces Eldo to compute specified time points. By default, **LOCK** is 0.

- **REPORTX=0 | 1**

Print checkbus warnings for undefined X states. By default, **REPORTX** is 0. Eldo considers that X states taken by the bus or given in a checkbus list mean that every state is correct for this time point. Thus the check is passed if the bus is X and the check value is 0 or 1.

## Related Options

[MAX\\_CHECKBUS, page 11-46.](#)

## Examples

```
.CHECKBUS OUT_BUS base=bin vth=2.5
+5000PS 01111
+10000PS 01011
+20000PS 01011
.CHECKBUS OUT_BUS_2 vth=1 vth2=4 base=bin
+ 1000ps 01
+ 7000ps x
+ 12000ps 11
+ 15000ps 01
+ 19500ps 01
```



Refer to “[PLOTBUS](#)” on page 10-250 for more information.

---

```
.checkbus my_bus vth=2.5 base=hex 15m 15 25m 22 45m 91
```

```
.checkbus my_bus vth=2.5 base=dec 15m 21 25m 34 45m 145
.checkbus my_bus vth=2.5 base=oct 15m 25 25m 42 45m 221
```

These three checkbus lines are equivalent. They use three different bases to specify the expected value on the bus.

The following example shows how it is possible to specify expected values using the **PATTERN** function:

```
.checkbus my_bus vth=2.5 TSAMPLE=10m TDELAY=5m base=hex
+ PATTERN 10 15 22 22 91
```

is equivalent to:

```
.checkbus my_bus vth=2.5 base=hex
+ 5m 10 15m 15 25m 22 35m 22 45m 91
```



Refer to the “[Pattern Function](#)” on page 5-23 for more information.

The following example shows how the **LOCK** parameter can be used:

```
.CHECKBUS S VTH1=0.5 VTH2=4.5 BASE=HEX LOCK=1
+ 15N 0 35N 1 55N 2 75N 3
+ 95N 1 115N 2 135N 3 155N 0
+ 175N 2 195N 3 196.5n 1 215N 0 235N 1
+ 255N 3 275N 0 295N 1 315N 2
```

Eldo will be forced to compute the timepoints 15n, 35n, 55n, etc. and the result of the checkbus will produce the following for any accuracy setting:

```
at time 1.965000e+02 ns bus S is 3 and should be 1
```

Without **LOCK** set, or when **LOCK=0**, the time reported may vary depending on the circuit and tolerance setting, i.e. the nearest point to 1.965000e+02 ns as computed by Eldo. When **LOCK=1**, the time reported will be 1.965000e+02 ns for any accuracy setting.

## .CHECKSOA

### Check Safe Operating Area Limits

```
.CHECKSOA [TRAN] [TSTART=val1 [TSTOP=val2]] [AUTOSTOP]
+ [NOMERGE] [NOLIB] [FILE=filename] [NOXWINDOW]
+ [SUBCKT={list_of_subckt_instances}] [RUNTMSG]
```

Causes Eldo to check for any violations of the circuit safe operating area (SOA) limits specified via the **.SETSOA** command.



See “[.SETSOA](#)” on page 10-284 for more details.

If an entity has more than one set of the same specifications, but with different boundaries, Eldo merges the specifications by taking the more restrictive constraint. This is the default mode.

#### Parameters

- **TRAN**  
Specifies a transient analysis should be used. Optional.
- **TSTART=val**  
Specifies start time value where SOA should be checked. Default is the first time point of the TRAN simulation.
- **TSTOP=val**  
Specifies stop time value where SOA should be checked. Default is the last time point of the TRAN simulation.
- **AUTOSTOP**  
Forces Eldo to quit immediately if an SOA specification is violated.
- **NOMERGE**  
By default, if an entity has more than one set of the same specifications, but with different boundaries, Eldo merges the specifications by taking the more restrictive constraint. Use **NOMERGE** to take the constraints into account as specified in the netlist. e.g.

```
.setsoa label=m1 d m5 vgs=(*,1.36)
.setsoa label=m2 d m5 vgs=(*,1.37)
```

The second constraint will be ignored, since the constraint labelled **m1** implies constraint **m2**. If the keyword **NOMERGE** is specified on the **.CHECKSOA** command, the two constraints will be taken into account as specified in the netlist.

- **NOLIB**  
Eldo will not consider SOA cards which are imported from **.LIB**. Instead, it will only consider those specified in the **.cir** file or from **.INCLUDE** files.

- **FILE=filename**

Specifies a file to which SOA results will be written instead of the standard ASCII output file. If no violations occur the message “No SOA violation detected” will be printed in the file.

- **NOXWINDOW**

Forces Eldo not to write the X-axis window information if the SOA is violated.

- **SUBCKT**

Restricts the scope of **.SETSOA** cards to the subcircuit instances specified by `list_of_subckt_instances`. Only **.SETSOA** commands which are in the subcircuit definition or which explicitly reference the instance will be performed.

- **RUNTMSG**

Specifying this parameter will force Eldo to print the SOA violations to the screen as they are detected during the simulation. This means the user does not have to wait till the end of the simulation to see when the violations occur. The violation information printed to the screen is represented differently to what is printed in the `.chi` file. If the `XAXIS` parameter is specified in the **.SETSOA** command then violations which occur within this time span and continue afterwards will only be printed to the screen when the `XAXIS` time span finishes. Violations which start and finish within the `XAXIS` time span will not be printed.

## Example

```
* Restrict SOA to list subckt instances
.model resis res r= 1k dev/gauss=10%

.subckt xsub a b
r1 a b resis 1k
.setsoa E I(R1)=(1.80u,1.90u) !<= This card will be active
*                                     for X1 only
.ends xsub

v1 1 0 pwl(0 0 10n 10)
x1 1 2 xsub
x2 1 2 xsub
r2 2 0 1k

.setsoa D X1.R1 i=(1.80u,1.90u)      !<= This soa is active
.setsoa D X2.R1 i=(1.80u,1.90u) !<= This one is not active
.checksoa subckt=x1

.tran 1n 10n
.extract tran I(X1.R1)
.end
```

In the example above, two cards will be kept: the one in `xsub` corresponding to `X1`; and the top level card **.setsoa D X1.R1**.

## .CHRENT

### Piece Wise Linear Source

```
.CHRENT NODE TN VN {TN VN} [P|F]
.CHRENT NODE (TN VN {TN VN}) FACTN {(TN VN {TN VN}) FACTN} [P|F]
```

Generates a Piece Wise Linear source using straight lines between specified points as described below.

### Parameters

- **NODE**  
The source is placed between node `NODE` and ground.
- **VN**  
Value of the source at time  $T_i$  in volts. The source value at intermediate times is provided by linear interpolation.
- **TN**  
Time in seconds, at which  $V_i$  is supplied, where  $T_i < T_{i+1}$ .
- **FACTN**  
Factor that indicates how many times the previous block is repeated.
- **P**  
Defines a periodic signal type. Default signal. Optional.
- **F**  
Defines a non-periodic signal type. Optional.

### Notes

The time declared using the first syntax above is an ABSOLUTE time whereas that declared using the second syntax in the parentheses is RELATIVE. This makes life easier when dealing with complex waveforms since calculation of absolute times can be very tedious.

For non-periodic curves, the voltage applied to node `NODE` between time 0 and time `T1` is equal to `V1`, with the voltage applied at time `TN` being `VN`. Time `TN` specifies the last time interval.

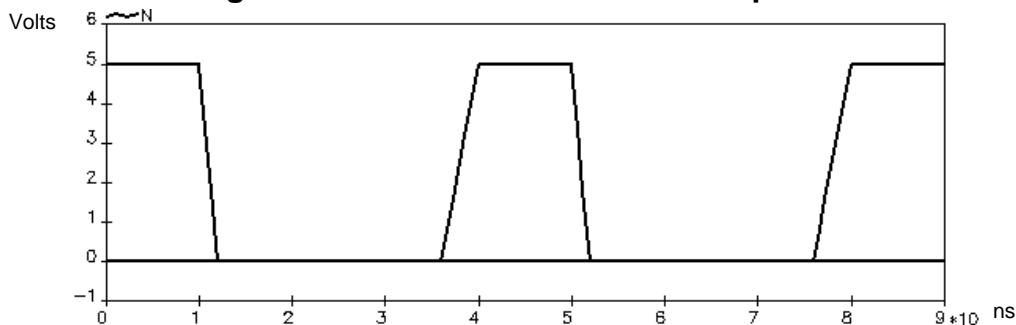
For periodic curves, the voltage applied to node `NODE` between time 0 and `T1` is `V1`. The period however, is specified as `TN` minus `T1` enabling an initialization phase before starting a periodic signal.

A piece-wise linear signal may be declared both by a `.CHRENT` command and a Piece Wise Linear signal on an independent source `Vxx`. The repetition factor, however, may only be declared using the `.CHRENT` command.

**Examples**

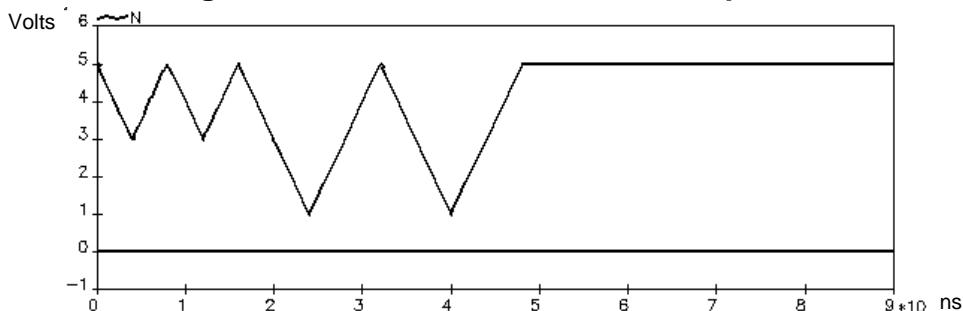
```
.chrent n3 0n 5.0 10n 5.0 12n 0 36n 0 40n 5.0 p
```

Specifies a periodic signal applied between node `n3` and ground. It starts at 0s with 5V, stays there for 5ns, falls in 1ns to 0V, stays there for 32ns, then it rises in 2ns to 5V. This signal repeats with a period of 40ns.

**Figure 10-1. Piece Wise Linear Example 1**

```
.chrent node1(0 5 4n 3 8n 5)2(0 5 8n 1 11n 5)2
```

This example illustrates a way of describing a function which is periodic for a given period of time, then becomes periodic with a different waveform for another period of time etc. The values not in parentheses indicate how often the block is repeated.

**Figure 10-2. Piece Wise Linear Example 2**

## .CHRSIM

### Input from a Prior Simulation

```
.CHRSIM IN OUT FILE [TSTART=V1] [TSTEP=V2] [BP=0|1|2]
+ [ZOOMTIME] [FORMAT=WDB|JWDB] [RUN=VAL]
```

The **.CHRSIM** command enables the user to use the output of previous simulations as input to the current simulation. The previous simulation data must have been generated by a DC sweep or a transient analysis. The simulator assigns the output obtained at node **OUT** of the circuit **FILE** to the node **IN** of the current simulation.

### Parameters

- **IN**  
Name of the input node of the current simulation.
- **OUT**  
Name of the node previously simulated in **FILE**.
- **FILE**  
Name of the circuit previously simulated. **FILE** is the prefix of the file, where *FILE.cou* will be read. This file must be located in the same directory as the file to be simulated, and node **OUT** must have been requested via a **.PLOT** command.
- **TSTART=V1**  
Starting position in the *FILE.cou* file in seconds.
- **TSTEP=V2**  
Selects one point every **TSTEP** points in the *FILE.cou* file.
- **BP=0**  
Input wave will not impose time points on Eldo. Default.
- **BP=2**  
Eldo will make a time point at each time point found in the input wave.
- **BP=1**  
Intermediate between cases 0 and 2. Eldo will impose a time point if the variation of the input wave is significant enough.
- **ZOOMTIME**  
Reuse the results of a simulation with **.CHRSIM** by modifying the time scale. If the simulation read back corresponds to a **TRAN** simulation, and if the current simulation is also a transient simulation for which the simulation duration is known (i.e. **.TRAN** command present in the netlist), then the simulation time as read in the file will be multiplied by **TSTOP/TSTOP\_READ**.

TSTOP is the simulation duration as specified in the **.TRAN** command. TSTOP\_READ is the simulation duration as read in the ‘cou-format’ file specified in the **.CHRSIM** command.

- **FORMAT=WDB | JWDB**

**.CHRSIM** will retrieve data values from a WDB or JWDB file.

- **RUN=VAL**

Specifies which simulation run the results will be used from, for use with multi-run analysis. Default is 1 (first simulation run). The user may experience problems when using WDB format with the **RUN** parameter specified.

## Examples

```
.chrsim n7 a8 adder
```

Specifies the output of the circuit `adder` at node `a8` to be applied at node `n7` of the current simulation. Eldo reads the file `adder.cou` which must be in the same directory.

```
.chrsim i7 o2 mult tstart=4n tstep=1n
```

Specifies the output of the circuit `mult` at node `o2` to be applied at node `i7` of the current simulation. The signal is applied 4ns in from its starting point at 1ns intervals of the waveform. Eldo reads the file `mult.cou` which must be in the same directory.

# .COMCHAR

## Change Comment Character

```
.COMCHAR char {char}
```

This command is used to change the comment character on an input file. Several comment characters can be specified as active at the same time in a single **.COMCHAR** command. Any new **.COMCHAR** command will append the new character to the list of comment characters.

### Parameters

- **char**

A list of comment characters can be specified. The command accepts any character. Usually, only the \$ or the ! characters are used. Unexpected behavior can be obtained when specifying other characters. Default is character '!'.

### Notes

- The first call to **.COMCHAR** overwrites the default comment character, i.e. character ‘!’ in Eldo default mode.
- Specifying option **CUMUL\_COMCHAR= 0** returns to the exclusive mode (pre-v6.4 default scheme) where only the last **.COMCHAR** specified is effective. An alternative is to specify command **.RESETCOMCHAR <list\_of\_char>** which will overwrite the list of comment characters.

### Examples

To tell Eldo that the comment character on an input file is ‘\$’ rather than the default ‘!’, use the command:

```
.comchar $
```

The example below shows how a list of comment characters can be made active in a single command, and then how the list is appended to with a further command:

```
Title
.COMCHAR $#! 
.COMCHAR /
i1 1 0 1 !default comment
r1 1 0 3 $another comment
r2 1 0 3 #another comment
r3 1 0 3 /another comment
.dc
.end
```

## .CONNECT

### Connect Two Nodes

**.CONNECT N1 N2**

The **.CONNECT** command is used to connect the nodes **N1** and **N2** without modifying the circuit description file.

### Parameters

- **N1, N2**

Names of the nodes to be connected.

### Example

**.connect n7 n5**

Specifies the connection of the nodes **n7** and **n5**.

## .CONSO

### Current Used by a Circuit

```
.CONSO VN {VN}
```

The **.CONSO** command computes and displays the average current flowing through the specified voltage source(s) during the simulation period. There is no limit to the number of **.CONSO** commands that may be present in an input netlist.

### Parameters

- **VN**  
Name of the voltage source(s).

---

#### Note



This command may only be used in conjunction with a transient analysis.

---

### Examples

```
vdd 100 101 5v
...
.conso vdd
.tran 1ns 100ns
```

Specifies a printed output of the average current flowing through the voltage source **vdd** over the length of the circuit simulation.

```
v1 n1 0 5
v2 n3 n4 2
...
.conso v1 v2
.tran 1ms 50ms
```

Specifies a printed output of the average current flowing through the voltage sources **v1** and **v2** over the length of the circuit simulation.

# **.CORREL**

## Correlation between Parameters

```
.CORREL [ PARAM= ]param_list cc=VAL
.CORREL DEV[ ICE ]=device_list PARAM=param_list cc=VAL
```

Correlations can be specified between parameters during a Monte Carlo analysis. This is done by specifying the relations between the different parameters. Correlation coefficients can be given for groups of two parameters. It can also be given for a list of parameters, in such a case the correlation coefficient will be the same for each couple. For example, in the list a, b, c the couples will be (a, b), (b, c) and (c, a).

It is also possible to specify correlations for each subcircuit instance via parameters which are assigned **DEVX** variation.



For more information on **DEVX** see [page 10-210](#).

From the correlated parameters specified via **.CORREL** statements, Eldo retrieves a set of uncorrelated parameters and computes a conversion matrix between the uncorrelated parameters and the correlated parameters. This is done using the PCA method (Principal Component Analysis). Then, it is on these uncorrelated parameters that the MC variations will be performed. Eldo will then compute the new values for the correlated parameters, using the transformation matrix.

## Parameters

- **PARAM=param\_list**

List of parameters to be correlated. There is no limit to the number of parameters that can be listed. They can be listed in a number of ways, as follows:

```
.correl a b c cc=val
.correl param=a b c cc=val
.correl param=a, b, c cc=val
.correl param={a b c} cc=val
.correl param={a, b, c} cc=val
```

When used with **DEVICE** it represents a list of parameters which have **DEVX** specifications.

- **cc=VAL**

Correlation coefficient. This can be any real number between -1 and +1.

- **DEVICE=device\_list**

List of devices and subckt instances.

## Examples

The following command:

```
.CORREL param=p1,p2,p3 cc=0.1
```

is equivalent to the following three .CORREL commands:

```
.CORREL p1 p2 cc=0.1
.CORREL p1 p3 cc=0.1
.CORREL p2 p3 cc=0.1
```

The following example shows how parameters with **LOT** variations can be correlated:

```
.param pc=10p LOT=10%
.param rc=1k LOT=5%
.SUBCKT cmod a b
C1 a b pc
R1 a b rc
.ENDS
X1 4 0 cmod
X2 6 8 cmod

.CORREL param=pc,rc cc=0.9
```

Eldo will generate pseudo-random values for **pc** and **rc** using 0.9 as their coefficient of correlation.

The following example shows how parameters with **DEVX** variations can be correlated:

```
.param pc=10p DEVX=10%
.param rc=1k DEVX=5%
.SUBCKT cmod a b
C1 a b pc
R1 a b rc
.ENDS
X1 4 0 cmod 10p
X2 6 8 cmod 10p
.CORREL device=x1,x2 param=pc,rc cc=0.1
```

The .CORREL command will create a relation between X1.pc and X2.pc and also between X1.rc and X2.rc.

---

#### Note



In the case of correlation on devices, and when the parameter list is not specified, then all parameters with **DEVX** variations which are used in the X instances given in the device list will use the same correlation coefficient **cc**.

---

## **.D2A**

### Digital-to-Analog Converter

```
.D2A [SIM=simulator] eldo_node_name
+ [digital_node_name] [MOD=model_name] [parameters_list]
```

The **.D2A** statement establishes the interface connection between digital solvers and Eldo.



For Analog-to-Digital, please refer to “[.A2D](#)” on page 10-4.

When the digital value driving the “mixed-signal” is logic ‘1’, the voltage on the analog side of the signal is driven to the voltage value defined by the parameter **VHI**. When the logic level is ‘0’, the analog voltage is driven to that defined by **VLO**. **TRISE** and **TFALL** are used to specify the duration of the linear rise and fall slopes between the voltages **VHI** and **VLO**. Values of “zero” for **TRISE** and **TFALL** are not allowed since this would represent an infinite energy transition.

The following table shows a global view of the parameters of the **.D2A**:

**Table 10-3. .D2A Global Parameters**

Name	Description	Default	Units
MODE= BIT   X01   MVL4			
<b>VHI</b>	Higher voltage value, corresponding to the logical ‘1’ state	5	V
<b>VLO</b>	Lower voltage value, corresponding to the logical ‘0’ state	0	V
<b>TRISE</b>	Defines the rise time for the voltage source, going from <b>VLO</b> to <b>VHI</b> value	2ns	s
<b>TFALL</b>	Defines the fall time for the voltage source going from <b>VHI</b> to <b>VLO</b> value	2ns	s
MODE= X01Z   STD_LOGIC			
<b>VHI</b>	Higher voltage value, corresponding to the logical ‘1’ state that will be reached if the convertor is applied on a node with infinite input resistance	5	V
<b>VLO</b>	Lower voltage value, corresponding to the logical ‘0’ state that will be reached if the convertor is applied on a node with infinite input resistance	0	V
<b>TRISE</b>	Rise time for the current source. Commutation time ‘0’ $\Rightarrow$ ‘1’ value	2ns	s
<b>TFALL</b>	Fall time for the current source. Commutation time ‘1’ $\Rightarrow$ ‘0’ value	2ns	s
<b>RZ</b>	Effective resistance when input is ‘Z’.	•	$\Omega$

**Table 10-3. .D2A Global Parameters**

Name	Description	Default	Units
RRISE	Impedance for a strong logical ‘1’ state. The conductance is then $G=1/RRISE$ and the current $I=VHI/RRISE$	1	$\Omega$
RFALL	Impedance for a strong logical ‘0’ state. The conductance is then $G=1/RFALL$ and the current $I=VLO/RFALL$	1	$\Omega$
LOWCAP	Capacitance for a logical ‘0’ state (‘L’). This capacitance models the output capacitance of the digital gate	0	F
HIGHCAP	Capacitance for a logical ‘1’ state (‘H’). This capacitance models the output capacitance of the digital gate	0	F
ZCAP	Capacitance for a logical ‘Z’ state. This capacitance models the output capacitance of the digital gate	0	F
XEVAL=PREVIOUS	Forces the corresponding ‘X’ state analog value to the previous analog value before the state changed to ‘X’	-	-
<b>MODE= STD_LOGIC</b>			
WEAHIGHRES	Impedance for a weak (or resistive) logical ‘1’ state (‘H’). The conductance is then $G=1/WEAHIGHRES$ and the current $I=VHI/WEAHIGHRES$	1	$\Omega$
WEALOWRES	Impedance for the weak (or resistive) logical ‘0’ state (‘L’). The conductance is then $G=1/WEALOWRES$ and the current is $I=VLO/WEALOWRES$	1	$\Omega$
<b>MODE= REAL</b>			
TCOM	Defines the commutation time. Eldo switches from the previous value to the new value with a slope of time TCOM	2ns	V

## Parameters

- **SIM=simulator**

The parameter **SIM** can take the value of the digital simulator’s name: **HDLA** or **VERILOG**. The parameter **SIM** is mandatory only for HDL-A.

- **eldo\_node\_name**

Name of the node in the analog netlist.

- **digital\_node\_name**

Name of the node in the digital description. This parameter is optional. For HDL-A, the syntax is **Yxx:position** where **position** is the index of the port.

- **MOD=model\_name**

Name of the model used for the convertor. If **model\_name** is specified, there must be a corresponding **.MODEL** command:

```
.MODEL model_name D2A|DTOA parameters_list
```

- `parameters_list`

List of available parameters as described below.

---

**Note**


For `eldo_node_name` connected to a PORT of an HDL-A model, `parameters_list` is ignored. Default values can depend on simulators (HDL-A, Verilog, etc.).

---

## Parameters for D2A converters (.D2A)

For **.D2A** the default **MODE** is `X01` for all simulators.

### Parameters for MODE=BIT |X01| MVL4

These modes emulate a voltage source for Eldo. Additional parameters:

- `VHI | VOLTHIGH=value`

Higher voltage value, corresponding to the logical ‘1’ state. Default is 5V.

- `VLO | VOLTLOW=value`

Lower voltage value, corresponding to the logical ‘0’ state. Default is 0V.

- `TRISE | TIMERISE=value`

Defines the rise time for the voltage source, going from `VLO` to `VHI` value. Unit is seconds. Default value is 2 ns.

- `TFALL | TIMEFALL=value`

Defines the fall time for the voltage source going from `VHI` to `VLO` value. Unit is seconds. Default value is 2 ns.

---

**Note**


For `MODE=X01`, the logical ‘X’ state is ignored, i.e. the analog value remains the same.

For `MODE=MVL4`, the logical ‘X’ and ‘Z’ states are ignored, i.e. the analog value remains the same.

---

### Parameters for MODE=X01Z | STD\_LOGIC

When the mode of the converter is `X01Z | STD_LOGIC | MVL9` these modes emulate an IRC (or IGC) convertor in Eldo. The convertor consists of a current source in parallel with a conductance in parallel with a capacitor with everything connected between the current node and the ground node. Several IRC convertors can be placed on one node and depending on the strength of the signals applied on each of them, the conflicts can be resolved with one of the modes. The values of this IGC vary depending on the state of the D2A convertor, these different values are computed according to the values specified in the D2A statement. If no values are specified, then the default values are used.

Additional parameters for these modes:

- **VHI | VOLTHIGH**=value

Higher voltage value, corresponding to the logical ‘1’ state that will be reached if the convertor is applied on a node with infinite input resistance. Default value: 5V.

- **VLO | VOLTLLOW**=value

Lower voltage value, corresponding to the logical ‘0’ state that will be reached if the convertor is applied on a node with infinite input resistance. Default value: 0 V.

- **TRISE | TIMERISE**=value

Rise time for the current source. Commutation time ‘0’  $\Rightarrow$  ‘1’ value. Unit is seconds. Default value: 2 ns.

- **TFALL | TIMEFALL**=value

Fall time for the current source. Commutation time ‘1’  $\Rightarrow$  ‘0’ value. Unit is seconds. Default value: 2 ns.

- **RZ | HIGHIMPRES**=value

High impedance to model a high impedance state (‘Z’). In this case, the conductance ( $G=1/RZ$ ) will be small, and the current  $I=0$ . Default is  $1\Omega$ .

- **RRISE | STRHIGHRES**=value

Impedance for a strong logical ‘1’ state. The conductance is then  $G=1/RRISE$  and the current  $I=VHI/RRISE$ . This resistance is usually small. Default is  $1\Omega$ .

- **RFALL | STRLOWRES**=value

Impedance for a strong logical ‘0’ state. The conductance is then  $G=1/RFALL$  and the current  $I=VLO/RFALL$ . This resistance is usually small. Default is  $1\Omega$ .

- **LOWCAP**=value

Capacitance for a logical ‘0’ state (‘L’). This capacitance models the output capacitance of the digital gate. Default is 0 F.

- **HIGHCAP**=value

Capacitance for a logical ‘1’ state (‘H’). This capacitance models the output capacitance of the digital gate. Default is 0 F.

- **ZCAP**=value

Capacitance for a logical ‘Z’ state. This capacitance models the output capacitance of the digital gate. Default is 0 F.

- **XEVAL=PREVIOUS**

When this is specified, the corresponding ‘X’ state analog value will be the previous analog value before the state changed to ‘X’, i.e. if the digital state changed from ‘L’  $\Rightarrow$  ‘X’ then the analog value will stay at **VLO**. The default value is computed from the impedances set in **.D2A** or **.MODEL** statements for the ‘X’ states.

## Parameters for MODE=STD\_LOGIC

- **WEAHIGHRES=value**

Impedance for a weak (or resistive) logical ‘1’ state (‘H’). The conductance is then  $G=1/\text{WEAHIGHRES}$  and the current  $I=\text{VHI}/\text{WEAHIGHRES}$ . This resistance is of course larger than `RFALL` and `RRISE`. Default is  $1\Omega$ .

- **WEALOWRES=value**

Impedance for the weak (or resistive) logical ‘0’ state (‘L’). The conductance is then  $G=1/\text{WEALOWRES}$  and the current is  $I=\text{VHI}/\text{WEALOWRES}$ . This resistance must be larger than `RFALL` and `RRISE`. Default is  $1\Omega$ .

## Logical states

### ‘X’ State

For the logical ‘X’ state which is also referred to as ‘strong X’, the corresponding conductance (`GX`), capacitance (`CX`), transition time (`TX`) and current source (`IX`) values are calculated as shown below:

$$\text{GX} = \frac{\text{RRISE}^{-1} + \text{RFALL}^{-1}}{2}$$

$$\text{CX} = \frac{\text{LOWCAP} + \text{HIGHCAP}}{2}$$

$$\text{TX} = \frac{\text{TRISE} + \text{TFALL}}{2}$$

$$\text{IX} = \frac{(\text{VHI} + \text{VLO})}{2} \times \text{GX}$$

### ‘W’ State

For the logical ‘W’ state which is also referred to as ‘strong W’ the corresponding conductance (`GX`), capacitance (`CX`), transition time (`TX`) and current source (`IX`) values are calculated as shown below:

$$\text{GX} = \frac{\text{WEAHIGHRES}^{-1} + \text{WEALOWRES}^{-1}}{2}$$

$$\text{CX} = \frac{\text{LOWCAP} + \text{HIGHCAP}}{2}$$

$$\text{TX} = \frac{\text{TRISE} + \text{TFALL}}{2}$$

$$\text{IX} = \frac{(\text{VHI} + \text{VLO})}{2} \times \text{GX}$$

### Note

The capacitance (`CX`) and transition time (`TX`) are the same for both ‘X’ and ‘W’ states.

## Parameters for MODE=REAL

Eldo will read from a DIGITAL simulator the real value instead of a logical value. Additional parameters:

- **TCOM=value**

Defines the commutation time. Eldo switches from the previous value to the new value with a slope of time TCOM. Default is 2 ns.

## Example: CMOS model

```
.D2A SIM=HDLA D2A_STD_LOGIC MOD=D2A_B_STD_LOGIC
.MODEL D2A_B_STD_LOGIC D2A MODE=STD_LOGIC
+ VLO=0.0 VHI=5.0 TRISE=3N TFALL=2N
+ RZ=1.0E+12 RRISE=1000 RFALL=1000
+ WEAHIGHRES=10K WEALOWRES=10K
+ LOWCAP=0.1p HIGHCAP=0.1p ZCAP=0.1p
```

## V source converter for STD\_LOGIC

A type of D2A converter, FAST\_STD\_LOGIC, is implemented. This FAST\_STD\_LOGIC converter can improve simulation speed, especially when used in ADMS and Mach TA.

The STD\_LOGIC D2A models are very flexible; they can handle high-impedance states and signals of different strengths. However, simulation can be longer compared with a D2A acting as a voltage source, as is the case for the BIT model. The FAST\_STD\_LOGIC converter overcomes this limitation.

When interfacing a Verilog or VHDL Std\_logic signal with analog nets, it is possible to manage conflicts of a ‘Z’ state with a V source built-in boundary element of type FAST\_STD\_LOGIC which can handle only 0, 1, X and Z states.

The V source built-in boundary element can be selected by setting the **MODE** of the converter to **FAST\_STD\_LOGIC** or **STD\_VSRC** (they are synonymous). It has the same set of parameters as **MODE=BIT** or **MODE=X01** Eldo converters.

Eldo checks whether there are one or more D2A boundary elements present, which are not in a high-impedance (‘Z’) state. If there are, Eldo will take into account all the D2A signals on the node and will compute the resulting voltage value. This value will be imposed on the analog part as a voltage source would. This means that the voltage on that node is not an *unknown* to the system of equations, and this can make simulation faster. If, however, there are only high-impedance states on the D2A node, Eldo will solve for that node.

## Option DEF D2A

It is normally necessary to specify an explicit D2A between ANALOG nodes and HDL-A ports; however, with the following statement specified:

```
.OPTION DEF D2A=model_name
```

Eldo will automatically insert an implicit D2A convertor when needed. The following must be defined in the netlist:

```
.MODEL model_name D2A parameters
```

Furthermore, it is often necessary to specify **MODE=BIT** in the **.MODEL** `model_name` command. This is because the default for **MODE** is `X01`, and very often HDL-A PORTS are of type **BIT**, hence a resulting error mismatch in convertor type would be printed if **MODE** is not correctly specified.

## Example

```
.MODEL model_name D2A mode = BIT
.OPTION DEFDEF2A = model_name
```

Additionally, these default models must be found in the netlist itself, and cannot be specified via **.LIB** or **.ADDLIB** specifications.

### Note



With the option **DEFDEF2A**, for any ANALOG node connected directly to an HDL-A **OUT** or **INOUT** port, an implicit D2A will be automatically inserted.

## Option D2DMVL9BIT

Enables direct connection between HDL-A ports of type **BIT** with mixed-mode nodes connected to Eldo via convertors of type **MVL9** (or `std_logic`).

The **BIT** type is a subset of `std_logic` so there is no conversion: ‘0’ -> ‘0’ and ‘1’ -> ‘1’.

# **.DATA**

## Parameter Sweep

```
.DATA dataname parameter_list
+ val_list1
+ val_list2
+ ...
[ .ENDDATA ]
.DATA dataname MER[GE] | LAM[INATED]
+ file=filename1 param=column ... param=column
+ file=filename2 param=column ...
+ ...
+ [out=outfile]
[ .ENDDATA ]
```

Two forms of **.DATA** statement definition are allowed. One is the inline form, and the other is where the statements are defined using external files. The data from multiple external files can be processed sequentially or in parallel.

## Parameters

- **dataname**

The name assigned to the **.DATA** statement, this name could be used in **.TRAN**, **.DC** or **.AC** commands using the format: **SWEEP DATA=dataname**

- **parameter\_list**

List of n parameter names.

- **val\_list**

Set of n double values corresponding to the values of the **parameter\_list**.

- **MER[GE]|LAM[INATED]**

Forces Eldo to process the data statements from multiple external files sequentially (**MER[GE]**) or in parallel (**LAM[INATED]**).

- **file=filename**

Definition of **.DATA** statements using external files. The data from multiple files can be processed sequentially or in parallel.

- **param=column**

Defines the column number from which the parameter values are taken.

- **out=outfile**

Specifies the output file in which the resulting **.DATA** statements can be saved.

## Notes

1. It is important to note that the “+” continuation characters must be present for the list of values when using this command. This may differ from other simulators where the “+” continuation characters are not required.

- 
2. Implicit **.DATA** creation is also possible:

```
.TRAN 1n 10n SWEEP p2 3.0 4.0 1.0
```

Eldo will make two TRAN runs: one with `p2` set to 3.0, another with `p2` set to 4.0.

3. **.ENDDATA** is optional.
4. Eldo will generate an ASCII output file with extension `.mt#` (for transient), `.ma#` (for AC) and `.md#` (for a DC sweep). The `#` character stands for the index of the file and a new file is created for each new **.TEMP** or **.ALTER** command.
5. It is possible to have the input signal being read from **.DATA** commands. An example is shown below:

```
.DATA srcname
+ time pv1
+ 0 0
+ 5n 0
+ 20n 5
.ENDDATA
V1 1 0 PWL (time,pv1)
.TRAN data=srcname
```

Here, the signal on `V1` will be read from the **.DATA** command. This works only if the following two conditions apply:

- a. The **.TRAN** command is defined via a **.DATA** command. Time will be assumed to be the first parameter name in the **.DATA** command, and simulation will proceed until the last value specified in the **.DATA** command.
- b. Both parameters which appear in the **PWL** must be in the same **.DATA** command.
6. **.MPRUN** can be used to take advantage of multi-processor machines for the **.DATA** command. Please see [.MPRUN](#) for further information.
7. SOA limits can be defined using **.DATA** statements. Please see [.SETSOA](#) for further information.
8. A wave defined with a **.DATA** statement can be plotted with the **DATA(dataname, parameter)** plot specification.

## Examples

```
.TRAN 1n 10n SWEEP DATA=datatran
.DATA datatran p1 p2 p3
+ 1.0 3.0 4.0
+ 2.0 5.0 7.0
.ENDDATA
```

Eldo will make two TRAN runs:

The first with `p1=1.0`, `p2=3.0`, and `p3=4.0`, and the second with `p1=2.0`, `p2=5.0`, and `p3=7.0`.

Here follow some examples of specifying multiple external files to be processed either sequentially or in parallel:

```
.data example1 MERGE
+ file=data1.inc col1=3 col2=2
+ file=data2.inc colA=1 colB=3
+ out=example1.inc
.enddata
```

After the first file name, the names of the columns to be created and the index where the data comes from are specified. The resulting **.DATA** is created using the merge of the two files:

1	2	3
4	5	6
7	8	9
10	20	30
40	50	60
70	80	90

The specification col1=3 col2=2 and colA=1 colB=3 literally means: the first column is named col1 and its values come from the third column of merged files; the second column is named col2 and its values come from the second column of merged files. The resulting **.DATA** will be:

```
.DATA example1
+ col1  col2  colA  colB
+ 3    2    1    3
+ 6    5    4    6
+ 9    8    7    9
+ 30   20   10   30
+ 60   50   40   60
+ 90   80   70   90
.enddata
```

The following example shows the **LAMINATED** specification for processing in parallel:

```
.data example2 LAM
+ file=data1.inc col1=3 col2=2
+ file=data2.inc colA=1 colB=3
+ out=example2.inc
.enddata
```

As before, the names of the columns to be created and the index where the data comes from are specified. However, unlike the merge form, values come from the original files so in this case you will notice that the resulting **.DATA** will be different as follows:

```
.DATA example2
+ col1  col2  colA  colB
+ 3    2    10   30
+ 6    5    40   60
+ 9    8    70   90
.enddata
```

## **.DC**

### DC Analysis

#### Single Analysis

**.DC**

#### Component Analysis

```
.DC CNAM [L|W] [TYPE nb] START STOP INCR [SWEET DATA=dataname] [MONTE=val]
.DC CNAM [L|W] [TYPE nb] START STOP INCR
+ [SWEET parameter_name TYPE nb start stop] [MONTE=val]
.DC CNAM [L|W] [TYPE nb] START STOP INCR
+ [SWEET parameter_name start stop incr] [MONTE=val]
```

#### Voltage or Current Source Analysis

```
.DC SNAM [TYPE nb] START STOP INCR [SNAM2 START2 STOP2 INCR2]
+ [SWEET DATA=dataname] [MONTE=val]
.DC SNAM [TYPE nb] START STOP INCR [SNAM2 START2 STOP2 INCR2]
+ [SWEET parameter_name TYPE nb start stop] [MONTE=val]
.DC SNAM [TYPE nb] START STOP INCR [SNAM2 START2 STOP2 INCR2]
+ [SWEET parameter_name start stop] incr [MONTE=val]
```

#### Temperature Analysis

```
.DC TEMP START STOP INCR [SWEET DATA=dataname] [MONTE=val]
.DC TEMP START STOP INCR [SWEET parameter_name TYPE nb start stop]
+ [MONTE=val]
.DC TEMP START STOP INCR [SWEET parameter_name start stop incr]
+ [MONTE=val]
```

#### Parameter Analysis

```
.DC PARAM PARAM_NAME START STOP INCR [SWEET DATA=dataname] [MONTE=val]
.DC PARAM PARAM_NAME START STOP INCR
+ [SWEET parameter_name TYPE nb start stop] [MONTE=val]
.DC PARAM PARAM_NAME START STOP INCR
+ [SWEET parameter_name start stop incr] [MONTE=val]
```

#### Data-Driven Analysis

```
.DC DATA=dataname [SWEET DATA=dataname] [MONTE=val]
.DC DATA=dataname [SWEET parameter_name TYPE nb start stop] [MONTE=val]
.DC DATA=dataname [SWEET parameter_name start stop incr] [MONTE=val]
```

The **.DC** command activates a DC analysis, and is used to determine the quiescent state or operating point of the circuit. The operating point of the circuit is computed with capacitances opened and inductances short-circuited. A DC analysis may be requested to determine the stable initial condition of an analog circuit, prior to a transient or AC analysis. Three levels of nesting are allowed.

There are six different types of DC analysis available as shown below:

1. **DC** with no further parameters results in a single analysis of the circuits' quiescent state.

2. **DC** followed by a component name results in a variation of the element size or value. The variation sweeps from the value `START` to the value `STOP` with the incremental step `INCR`. The quiescent state is calculated for each incremental step.
3. **DC** followed by a voltage or current source, results in a voltage or current sweep of the specified source from `START` to `STOP` in increments `INCR`. The quiescent state is calculated for each incremental step. A second source `SNAM2` may optionally be specified with associated sweep parameters. In this case, the first source is swept over its range for each value of the second source.
4. **DC** followed by the **TEMP** keyword results in a variation of temperature. The temperature sweeps from the value of `START` to the value of `STOP` with the incremental step `INCR`. The quiescent state is calculated for each incremental step.
5. **DC** followed by the **PARAM** keyword results in a variation of the value of a globally declared parameter `PARAM_NAME`. The variation sweeps from the value of `START` to the value of `STOP` with the incremental step `INCR`. The quiescent state is calculated for each incremental step.
6. **DC** followed by the parameter `DATA=dataname` results in a sweep of the values pre-defined in the `.DATA` command. The quiescent state is calculated for each incremental step.

Multi-threading can be activated for a single DC simulation, Eldo will share computer resources on a multi-processor machine. Alternatives are:

- command line flag **-mthread** (see [page 2-9](#)) at Eldo invocation, or option **MTHREAD** in the netlist. Eldo will make use of all the possible CPUs on the machine.
- command line flag **-usethread #** (see [page 2-10](#)) at Eldo invocation, or option **USETHREAD=val**. This forces Eldo to use at maximum the specified (#) number of CPU. The number specified can exceed the number of CPUs available, but this is not recommended, even though Unix will allow it.

Statistics, generated at the end of simulation, show how many CPUs have been used for the current simulation.

## Parameters

- **CNAM**  
Name of component on which geometrical or value variations are performed. Acceptable components are inductors, resistors, MOS transistors and controlled sources (VCVS, CCVS, CCCS, VCCS). For the control sources, only the linear case is supported and only the gain of these sources may be swept.
- **START**  
Start value of the component `CNAM`, voltage, temperature or current sweep. This can be specified as a parameter or as an expression.

- **STOP**  
Stop value of the component CNAM, voltage, temperature or current sweep. This can be specified as a parameter or as an expression.
- **INCR**  
Increment of the component, voltage, temperature, or current sweep. This can be specified as a parameter or as an expression.
- **SNAM**  
Name of the voltage or current source which performs the DC sweep.
- **TEMP**  
Keyword indicating that the temperature is to be varied.
- **PARAM**  
Keyword indicating that a parameter is to be varied.
- **DATA=dataname**  
Used in conjunction with the **.DATA** command. The **dataname** parameter should be specified using the **.DATA** command. Please refer to the **.DATA** command on [page 10-51](#) for more information.
- **PARAM\_NAME**
- Name of the globally declared parameter to be varied. Both primitive and non-primitive parameters are allowed.
- **L, W**  
Keywords which determine if the length **L** or the width **W** of the MOS component CNAM is to be varied.
- **TYPE**  
Type name of the first level of variation for DC component analysis and voltage/current source analysis. Can be one of the following:

**DEC**

Keyword to select logarithmic variation.

**OCT**

Keyword to select octave variation.

**LIN**

Keyword to select linear variation.

**POI**

Keyword to select a list of frequency points. **POI** is the same as **LIST** except that **POI** expects the number of points **nb** to be specified as its first argument.

- **SNAM2**  
Name of the secondary voltage or current source for DC sweep.
- **START2**  
Start value of the secondary voltage or current sweep. This can be specified as a parameter or as an expression.
- **STOP2**  
Stop value of the secondary voltage or current sweep. This can be specified as a parameter or as an expression.
- **INCR2**  
Increment of the secondary voltage or current sweep. This can be specified as a parameter or as an expression.
- **MONTE=val**  
Monte Carlo analysis. Equivalent to **.MC val**. The syntax allows a different **MONTE** value for each run. However, the actual implementation in Eldo does not account for that: it is the last **MONTE** value to be specified in the netlist which will be taken into account.

## Sweep Parameters

This section contains **SWEEP** related parameters that are previously unspecified in the Parameters section.

- **SWEEP**  
Specifies that a sweep should be performed on a parameter or device name.
- **parameter\_name**  
Name of the parameter or device name to sweep.
- **TYPE**  
Can be one of the following:

### **DEC**

Keyword to select logarithmic variation.

### **OCT**

Keyword to select octave variation.

### **LIN**

Keyword to select linear variation.

### **POI**

Keyword to select a list of frequency points. **POI** is the same as **LIST** except that **POI** expects the number of points **nb** to be specified as its first argument.

**INCR**

Increment of the parameter or device name to sweep. (**INCR** can only be used when the **SWEET** keyword is preceding it. When **INCR** is specified, nb is the incrementing value.)

- **nb**  
Number of points required.
- **start**  
Start value of the parameter or device.
- **stop**  
Stop value of the parameter or device.
- **incr**  
Increment of the parameter or device name to sweep.

**Note**

When **INCR** is specified as the **TYPE** parameter, the value which directly follows (nb) is the incrementing value. If **INCR** is not specified, the incrementing value (incr) must be placed after the **start** and **stop** values.

**Examples**

```
vin 1 0 10
.dc vin 0 5 0.2
```

Specifies a DC analysis with voltage sweep of the voltage source **vin** from 0 to 5V with an increment of 0.2V.

```
r7 3 4 100k
.dc r7 10k 100k 10k
```

Specifies a DC analysis with resistor value variation of **r7** from 10k $\Omega$  to 100k $\Omega$  with increments of 10k $\Omega$ .

```
r1 1 2 p1
.param p1=1k
.dc param p1 1k 10k 1k
```

Specifies a DC analysis with resistor parameter **p1** variation from 1k $\Omega$  to 10k $\Omega$  with an increment of 1k $\Omega$ .

```
VIN 1 0 10
.param p1=3
.param pend=10
.DC VIN p1 pend 0.1
```

Specifies a DC analysis with voltage sweep of the voltage source **vin** from 3 to 10V with an increment of 0.1 V. This example shows how the **.DC** command accepts parameters as arguments.

```
.dc v1 dec 10 2 12 sweep r2 INCR 10 1k 100k
.dc v1 dec 10 2 12 sweep r2 1k 100k 10
```

The two lines above are equivalent. Either can be used to specify a DC analysis at each of the values for **v1** starting at 2V and stopping at 12V with 10 analysis points per decade. The sweep specification will force Eldo to carry out a DC analysis on each value of **r2** starting at 1k $\Omega$  and stopping at 100k $\Omega$  with an incrementing value of 10 $\Omega$ .

```
.dc E1 3 5 0.5
```

In the sweep of gain for linear controlled sources example above, the gain of the VCVS E1 will be swept from 3 to 5 in increments of 0.5.

The next example shows that the DC sweep can be nested up to three levels deep. For each value of the third level variable, the second level variable is swept through its specified range. For each value of the second level variable, the first level variable is swept through its specified range. To show the order of sweepings, the below shows three independent voltage sources swept with the .DC statement.

```
v1      1      0 0v
v2      2      0 0v
v3      3      0 0v
.dc v1 1 2 0.5    v2 2 3 0.5      v3 3 4 0.5
.print dc v(1) v(2) v(3)
```

The .PRINT statement results in the following output in the *.chi* file:

V1	V(1)	V(2)	V(3)
1.0000E+00	1.0000E+00	2.0000E+00	3.0000E+00
1.5000E+00	1.5000E+00	2.0000E+00	3.0000E+00
2.0000E+00	2.0000E+00	2.0000E+00	3.0000E+00
1.0000E+00	1.0000E+00	2.5000E+00	3.0000E+00
1.5000E+00	1.5000E+00	2.5000E+00	3.0000E+00
2.0000E+00	2.0000E+00	2.5000E+00	3.0000E+00
1.0000E+00	1.0000E+00	3.0000E+00	3.0000E+00
1.5000E+00	1.5000E+00	3.0000E+00	3.0000E+00
2.0000E+00	2.0000E+00	3.0000E+00	3.0000E+00
1.0000E+00	1.0000E+00	2.0000E+00	3.5000E+00
1.5000E+00	1.5000E+00	2.0000E+00	3.5000E+00
2.0000E+00	2.0000E+00	2.0000E+00	3.5000E+00
1.0000E+00	1.0000E+00	2.5000E+00	3.5000E+00
1.5000E+00	1.5000E+00	2.5000E+00	3.5000E+00
2.0000E+00	2.0000E+00	2.5000E+00	3.5000E+00
1.0000E+00	1.0000E+00	3.0000E+00	3.5000E+00
1.5000E+00	1.5000E+00	3.0000E+00	3.5000E+00
2.0000E+00	2.0000E+00	3.0000E+00	3.5000E+00
1.0000E+00	1.0000E+00	2.0000E+00	4.0000E+00
1.5000E+00	1.5000E+00	2.0000E+00	4.0000E+00
2.0000E+00	2.0000E+00	2.0000E+00	4.0000E+00
1.0000E+00	1.0000E+00	2.5000E+00	4.0000E+00
1.5000E+00	1.5000E+00	2.5000E+00	4.0000E+00
2.0000E+00	2.0000E+00	2.5000E+00	4.0000E+00
1.0000E+00	1.0000E+00	3.0000E+00	4.0000E+00

---

1.5000E+00	1.5000E+00	3.0000E+00	4.0000E+00
2.0000E+00	2.0000E+00	3.0000E+00	4.0000E+00

The following examples show how the keywords **DEC**, **OCT**, **LIN**, and **POI** can be specified as the type name of the first level of variation for DC component analysis and voltage/current source analysis.

```
.dc r1 DEC 8 2k 9k
.dc r1 OCT 8 2k 9k
.dc r1 LIN 100 2k 9k
.dc r1 POI 8 2k 3k 4k 5k 6k 7k 8k 9k
```

## **.DCMISMATCH**

### DC Mismatch Analysis

```
.DCMISMATCH [output]
+ [ SORT_REL=value ] [ SORT_ABS=value ] [ SORT_NBMAX=value ] [ NSIGMA=value ]
```

This command is a special form of sensitivity analysis based on statistical deviation properties of device model parameters. It computes the sensitivity of node voltages and of currents though voltage sources. All devices using a device model with statistical deviations potentially contribute to the output deviation. The aim of the **.DCMISMATCH** analysis is to quantify these contributions and to identify the biggest contributors to the total output deviation. It has some degree of similarity with the worst-case analysis (**.WCASE**). However, it will usually provide (much) less pessimistic results than worst-case analysis.

**.DCMISMATCH** will consider exclusively the **DEV** or **DEVX** variations specified for device model parameters (in **.MODEL** statements) and for global parameters in the *.cir* file (**.PARAM** statements specified at the top-level, outside any subcircuit definition). The **LOT** specifications are not taken into account.

Mismatch models try to model the inherent differences which exist between two identically drawn transistors. Each transistor behavior departs (statistically) from the nominal behavior. The deviation depends strongly on the geometry (W, L) of the transistor. Small transistors tend to be more random than large ones.

**.DCMISMATCH** produces an ASCII report in the main output (*.chi*) file, which lists the output deviations due to the deviations of the sensitive devices and parameters. Specifically, deviation is applied independently to the sensitive devices, and the resulting deviation of the output is computed from the RMS sum of the resulting deviations.

The **SORT\_** parameters are used to limit the amount of output information.

Multiple sensitivity and statistical analyses cannot be used simultaneously. Specifically, **.WCASE**, **.MC** and **.DCMISMATCH** are exclusive. Only one of them can be specified in a netlist.

DC mismatch information can also be extracted with the **.EXTRACT DCM** function, which returns the result for a specific **label\_name**. It extracts the value which is dumped in the *.chi* file.

There are two methodologies when using the **.DCMISMATCH** command:

- **.DCMISMATCH** with no output specified, the analysis is performed automatically for all DC extracts (associated **.EXTRACT** statements required) to obtain the result.
- **.DCMISMATCH** with V or I output specified, in this case, **.EXTRACT DCM** statements should not be specified (they are not taken into account).

Extractions using general purpose functions, such as `yval`, `xycond`, cannot be used for **.DCMISMATCH** analysis. Only extractions using a combination of `I(device)` or `V(node)` quantities are allowed.

The **.DCMISMATCH** analysis will be performed only for the first DC of a DCSWEEP.

## Parameters

- `output`

Optional. The output can be `V(net_name1[,net_name2])`, `I(Vsrc)`, `EXTRACT(label)`, or `MEAS(label)`. If no output is specified, the analysis is performed for all DC extracts.

- `SORT_REL`

Only devices with a contribution to the output greater than the value: `SORT_REL*<total_variation_on_output>` will be listed. The default value of `SORT_REL` is 0.001, which means that all devices contributing to more than 1/1000 of the total output will be listed.

- `SORT_ABS`

Used to specify the absolute threshold, below which contributors will not be listed. Default value is 0.

- `SORT_NBMAX`

Allows the user to limit the list of contributors to a certain value. By default, all contributors are listed.

### **Note**



The sorting parameters are additive, that is, their respective effects are cumulative. For example, to create a report with only devices contributing to 10% or more of the output deviation and has 15 contributors at most the user would need to specify:

```
SORT_REL=0.1
SORT_NBMAX=15.
```

- `NSIGMA`

Parameter used to change the default sigma value used for Gaussian distributions in Eldo statistical analysis.

The total variation on the output is computed from the 4-sigma variation for Gaussian distribution, or from the max variation for other distributions (UNIFORM distribution or distribution defined with a **.DISTRIB** command). 4- sigma is the default used for Gaussian distributions in Eldo statistical analyses. This can be changed with the `NSIGMA` value of the **.DCMISMATCH** command, and if not specified, from the global option `SIGTAIL=val`, which defaults to 4.

**Note**

With gaussian distributions, 4-sigma deviations may cause unrealistic situations if the standard deviation is too large, so this should be handled with care. For example specifying **DEV=10%** for a 0.35V threshold voltage with a Gaussian distribution will most probably cause a problem when a -4 sigma variation is applied.

## Examples

These first two examples below show how **.DCMISMATCH** will consider the **DEV** variations specified for device model parameters and for global parameters:

```
.model nch nmos level=53 vt0=0.45 dev=1%...
```

All transistors using the **nch** model will be considered.

```
.param vdd=1.2V dev=5%
```

The **vdd** parameter will be considered.

```
.extract dc label=VS V(s)
.extract dc label=dcmmm_vs dcm(VS)
.dcmismatch nsigma=1 sort_nbmax=5
```

Here Eldo will return the variation of all the valid DC extracts, namely the EXTRACT labelled VS (the second extract dcmmm\_vs is just for output, it will not be seen by the mismatch). The above is equivalent to the following:

```
.extract dc label=dcmmm_vs dcm(VS)
.dcmismatch V(s) nsigma=1 sort_nbmax=5
```

or:

```
.extract dc label=VS V(s)
.extract dc label=dcmmm_vs dcm(VS)
.dcmismatch meas(VS) nsigma=1 sort_nbmax=5
```

Below is a typical DC mismatch analysis output showing N-sigma deviation of the outputs, and a table of sorted contributors. In this example, XM2.M1 is the main contributor, and the deviation of the U0 mobility parameter of its associated .model (XM2.MOD2) is the main contributor to the XM2.M1 contribution.

```
#.DCMISMATCH V(S) SORT_REL = 1.000000e-03 NSIGMA = 1.000000e+00
```

Analysis results:

Output DC value	:	1.7892U	Volt
Total output deviation	:	+/- 14.9166M	Volt
Output deviation due to the contributors in the report table	:	+/- 14.9166M	Volt

Report table

Output deviation	Contributor	Parameter
7.9308M	XM2.M1	

Simulator Commands  
.DCMISMATCH

---

	5.1779M	M( XM2.MOD2,U0 )
	6.0051M	M( XM2.MOD2,VTH0 )
	163.0689U	M( XM2.MOD2,TOX )
7.9294M	XM1.M1	
	5.1769M	M( XM1.MOD2,U0 )
	6.0040M	M( XM1.MOD2,VTH0 )
	163.0386U	M( XM1.MOD2,TOX )
6.1906M	XM7.M1	
	2.5507M	M( XM7.MOD1,U0 )
	5.6326M	M( XM7.MOD1,VTH0 )
	302.4733U	M( XM7.MOD1,TOX )
6.1906M	XM6.M1	
	2.5507M	M( XM6.MOD1,U0 )
	5.6326M	M( XM6.MOD1,VTH0 )
	302.4703U	M( XM6.MOD1,TOX )
...		

# .DEFAULT

## Set Default Conditions

```
.DE[FAULT] TYPE VALUE
.DE[FAULT] TYPE {KEYWORD [VALUE]}
```

This command resets the default values for elements, device initial conditions and model parameters.

The first syntax shown above is the general form for resistors, capacitors, and inductors. The second syntax is the general form for other types.

You may specify only one element or model type per **.DEFAULT** statement. However, in your circuit netlist file you may include as many **.DEFAULT** statements as required to set all required default conditions.

## Parameters

- **TYPE**  
Type of element or model.
- **KEYWORD**  
Any valid element or model parameter or keyword.
- **VALUE**  
Optional value given to keywords.

## Limitation

Eldo implementation of **.DEFAULT** does not support the Lossy Transmission Line (LDTL) model and IC for active devices are ignored.

## Example

# **.DEFMAC**

## Macro Definition

**.DEFMAC MAC\_NAME(ARG{, ARG})=EXPRESSION**

The **.DEFMAC** command is used to define a parameterized macro which may be instantiated (used) in the circuit netlist. The macro may contain a combination of arithmetic expressions or pre-defined Eldo functions. The macro is instantiated using the **.EXTRACT** command and results are listed to the ASCII output (*.chi*) file.

## Parameters

- **MAC\_NAME**  
Macro name.
- **ARG**  
Argument names passed to the macro at instantiation time.
- **EXPRESSION**  
A combination of arithmetic expressions and pre-defined function calls that may use the arguments passed to the macro.

### Note



Argument names cannot contain arithmetic operators.

Only one macro may be defined per **.DEFMAC** statement.

TRISE, TFALL, TPDxx, SLEWRATE, RMS, INTEG, DW\_A cannot be used inside **.DEFMAC**. Only the so called “general purpose extraction language” terms (END, MAX, MIN, START, XAXIS, XYCOND, XUP, XDOWN, YVAL) can be used.

A macro instantiation is made by preceding the macro name with the \$ character.

Keywords **XAXIS**, **END** and **START** are recognized by the **.DEFMAC** statement. This simplifies the writing of macros.

## Examples

```
.defmac phmag(a, b)=xycond(a, b<0.0)-yval(a, 0.001)
...
.extract $phmag(vp(s), vdb(s))
```

This example defines a macro called **phmag** with arguments **a** and **b**. This macro is then instantiated via the **.extract** command with the arguments **a** and **b** being replaced by the phase voltage **vp(s)** and magnitude **vdb(s)** on node **s** respectively. The results are listed in the ASCII output file. The **phmag** macro uses the pre-defined functions **xycond** and **yval**.



For more information on these functions, refer to “[.EXTRACT](#)” on page 10-95.

```
.defmac sett(out,ratio) =
+ XYCOND(XAXIS,(out > (yval(out,END) * (1 + ratio))) ||
+ (out < (yval(out,END) * (1 - ratio))),END,START)
.extract sett(v(s),0.1)
```

Returns the settling time of `v(s)`, i.e. the time at which `v(s)` does not vary outside the range  $0.9 \times \text{final\_value}$ ,  $1.1 \times \text{final\_value}$ , where `final_value` is the value of `v(s)` at the end of the simulation.

```
.defmac sett(x,out,tx,ratio) =
+ XYCOND(x,(out > (yval(out,tx)*(1 + ratio))) ||
+ (out < (yval(out,tx) * (1 - ratio))),tx,0)
.extract $sett(xaxis,v(s),50n,0.1)
```

This is for finding the settling time on wave `out`.

## **.DEFMOD**

### Model Name Mapping

```
.DEFMOD alias_model_name actual_model_name
```

This command can be used to map model names in a netlist to model names specified in **.MODEL** cards. This works if the **.DEFMOD** is placed before any use of the string `alias_model_name`.

### Parameters

- `alias_model_name`  
Model name given in a netlist.
- `actual_model_name`  
Model name defined in **.MODEL** card.

### Example

```
.model modell r tc1=2 tc2=1
.defmod modalias modell
r1 1 0 modalias r=1k
```

## .DEFPLOTDIG

### Plotting an Analog Signal as a Digital Bus

```
.DEFPLOTDIG [VTH[1]=VAL1 [VTH2=VAL2]]
```

The **.DEFPLOTDIG** command enables the user to plot an analog signal as a digital bus. This command can *only* be used as a precursor to **VDIG** of the **.PLOT** command.

#### Parameters

- **VTH[1]=VAL1**

If a voltage threshold is specified, the bus of an analog signal is plotted as a bus (hexadecimal format), else all the different signals of the bus are plotted separately in the wave viewer as analog waves. (**VTH** and **VTH1** are synonymous to ensure backwards compatibility.)

- **VTH2=VAL2**

Can be used to plot the indeterminate value, as shown below:

When only **VTH1** is given:

If value < **VTH** then logic state 0.

If value > **VTH** then logic state 1.

When both **VTH1** and **VTH2** are given:

If value < **VTH1** then logic state 0.

If **VTH1** < value < **VTH2** then state X.

If value > **VTH2** then logic state 1.

#### Example

```
.DEFPLOTDIG VTH1=2.2 VTH2=2.7
.PLOT TRAN VDIG(n2)
```

When **.DEFPLOTDIG** is used in conjunction with **.PLOT VDIG**, the signal node **n2** is plotted as a digital curve and will only have values of “1” or “0”.

## **.DEFWAVE**

### Waveform Definition

```
.DEFWAVE [SWEEP] [ANALYSIS] WAVE_NAME=WAVE_EXPR
```

The **.DEFWAVE** command is used to define a new waveform by relating previously defined waveforms and nodes. The waveform definition may contain a combination of arithmetic expressions or pre-defined functions available in Eldo. The waveform may be instantiated using the **.EXTRACT** command with the results being listed to the ASCII output (*.chi*) file or may be printed or plotted using the **.PLOT** and **.PRINT** commands.

### Parameters

- **SWEEP**

When a circuit contains some **.EXTRACT** or **.MEAS** commands, and in the case of **.STEP** cards, it applies a wave operator on the curves **EXTRACT = F(STEP)**, i.e. those waves dumped in the *.ext* file.

- **ANALYSIS**

Type of analysis to be used.

- **WAVE\_NAME**

New waveform name.

- **WAVE\_EXPR**

Arithmetic expression relating previously defined waveforms and nodes. See “[Arithmetic Functions](#)” on page 3-7 for further details.

The function types **PWL**, **PWL\_CTE** and **PWL\_LIN** are also available for **.DEFWAVE**. They create piece-wise linear waves which can be plotted in the binary output (*.wdb* or *.cou*) file. These function are used for creating an ideal waveform which can then be displayed in EZwave or Xelga and matched against actual simulation results.



For more information on **PWL** see “[Piece Wise Linear Function](#)” on page 5-27.

The syntax for **PWL**, **PWL\_CTE** and **PWL\_LIN** is the same but their representation is different. The standard **PWL** function will plot all specified points. **PWL\_CTE** will plot the specified points but will hold the first specified value if the simulation start time is less than the first time point specified in the **PWL\_CTE** function. Similarly it will hold the last value until the simulation is complete. **PWL\_LIN** will linearly extrapolate first value using the first two points of the **PWL\_LIN** function. The last value will also be linearly extrapolated using the last two points. This will only happen if the simulation start time is less than the first time value given in the **PWL\_LIN** function and the simulation end time is greater than the last time value of the function.

## Notes

- Waveform names cannot contain arithmetic operators.
- Only one waveform may be defined per **.DEFWAVE** statement.
- A wave is instantiated by preceding the wave name `w`, which can be followed by a format suffix for complex waveforms, e.g. `wm`, `wdb`, `wp`, `wi`, `wr` are enclosed in parentheses `( )`.
- A waveform expression cannot contain a division by zero. If this is the case,  $1.0 \times 10^{-15}$  is automatically assigned to a value.
- The **.DEFWAVE** command is order dependent in that all components of the waveform expression must have already been defined earlier in the netlist. Therefore, the following is illegal:

```
.defwave power_vdd=i(v1)*v(v1)
...
v1 in out ...
```

However, the following is allowed:

```
v1 in out ...
...
.defwave power_vdd=i(v1)*v(v1)
```

- The waveform expression has to consist of waves and functions which are related to the type of analysis being performed as described in the **.PRINT** section later in this chapter. Thus, the following is illegal and will produce false results:

```
.defwave amplification=v(2)/v(1)
...
.plot ac w(amplification)
```

However, the following is allowed:

```
.defwave amplification=vm(2)/vm(1)
...
.plot ac wdb(amplification)
```

## Examples

```
.defwave power_vdd=v(a)*i(vdd)
...
.extract max(w(power_vdd))
```

The example above defines a waveform `power_vdd` as the product of the waveforms `v(a)` and `i(vdd)`. The maximum value of the waveform is then extracted and listed to the ASCII output file using the **.extract** command.

```
.defwave w1=v(a)/v(b)
...
.extract integ(w(w1))
.tran 1n 100n
.plot tran w(w1)
```

The example above defines a waveform `w1` as waveform `v(a)` divided by waveform `v(b)`. The integral value of the waveform is extracted and listed to the ASCII output file using the `.extract` command.



For more information, see “[.EXTRACT](#)” on page 10-95.

```
.defwave powf = 0.5*(vdip(vxx) * CONJ(I(vxx)))
.plot ac wr(powf)
```

Plots the AC power of the `vxx` voltage source.

Example of applying a wave operator:

```
.EXTRACT LABEL = A MAX(V(1))
.EXTRACT LABEL = B MAX(V(2))
.STEP PARAM P1 1 3 1
.DEFWAVE SWEEP my_wave = MEAS(a) + MEAS(b)
.end
```

Example of using the `PWL` function:

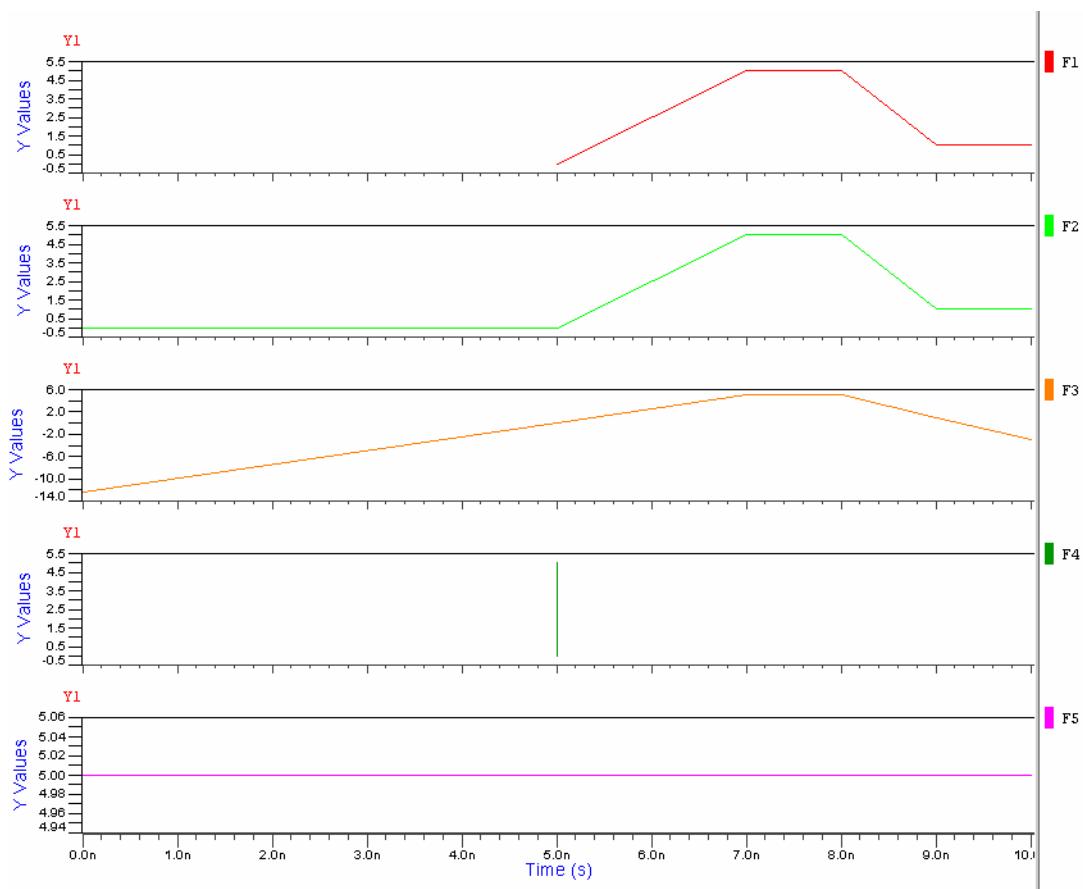
```
v1 1 0 pwl ( 0 0 10n 10)
r1 1 0 1
.defwave tran foo = pwl(0,0,20n, meas(ex))
.extract label = ex yval(v(1),5n)
.tran 1n 10n
.plot tran w(foo)
.end
```

Example of using the `PWL`, `PWL_CTE` and `PWL_LIN` functions.

```
v1 1 0 pwl ( 0 0 10n 10)
r1 1 0 1
.defwave tran f1 = pwl(5n,0,7n,5,8n,5,9n,1)
.defwave tran f2 = pwl_cte(5n,0,7n,5,8n,5,9n,1)
.defwave tran f3 = pwl_lin(5n,0,7n,5,8n,5,9n,1)
.defwave tran f4 = pwl_lin(5n,0, 5n, 5)
.defwave tran f5 = pwl_lin(0n, 5)
.extract label = ex yval(v(1),5n)
.tran 1n 10n
.plot tran w(f1) w(f2) w(f3) w(f4) w(f5)
.end
```

The plot obtained from the netlist is shown in [Figure 10-3](#). The wave `f4` demonstrates how to plot vertical waves with the `.DEFWAVE` command.

**Figure 10-3. PWL, PWL\_CTE and PWL\_LIN Functions**



## **.DEL**

### Remove library name

```
.DEL LIB LIB_NAME
```

This command removes a library name from the nominal description. It can be used in conjunction with the **.ALTER** command to create modifications within the netlist before re-runs.



Please refer to “[.ALTER](#)” on page 10-22.

### Example

```
v1 1 0 1
x1 1 2 r
r2 2 0 1
.dc
.extract dc v(2)
.lib delib1.lib
.alter
.del lib delib1.lib
.lib delib2.lib
.end
```

This example demonstrates how a library file can be replaced after a first run. *delib1.lib* is removed and *delib2.lib* is put in its place.

## .DISCARD

### Ignore Instances or Subckt Definitions

```
.DISCARD INST | SUBCKT | DEV =(name {,name})
```

This command is used to specify whether one or more subcircuits/instances/devices should be ignored during an Eldo simulation. The **.DISCARD** command cannot remove *hierarchical* instances and cannot contain wildcards.

### Parameters

- **INST**  
Specifies that one or more subcircuit instances will be ignored.
- **SUBCKT**  
Specifies that all instances of the subcircuit(s) will be ignored.
- **DEV**  
Specifies that all instances of the primitive element(s) will be ignored.
- **name**  
The name of the instance or subcircuit that will be ignored. A list of names can be specified, in this case the brackets must be used. See [Example](#).

### Example

In the example below the instances `X1` and `X2` and the subcircuit `RESI` will be ignored.

```
.SUBCKT RESI a b
R1 a b 1
.ENDS RESI

X1 1 0 RESI
X2 1 0 RESI
X3 1 0 RESI

.DISCARD INST=(X1, X2)
.DISCARD SUBCKT=RESI
```

In the example below the primitive element `r2` will be ignored.

```
*SEARCH extracted value should be -2
i1 1 0 1
r1 1 2 1
r2 1 2 1
r3 2 0 1
.discard dev = r2
.dc
.op
.extract dc v(1)
.end
```

## .DISFLAT

### Disable Flat Netlist Mode

**.DISFLAT**

This command is used to disable the flat netlist mode.

In the case of a large netlist, which does not use any subcircuit instances, Eldo will assume that the netlist is flat and will automatically enter into a mode where it consumes less memory for parsing the netlist. However, this mode requires that all **.MODEL** statements are placed at the top of the netlist.

A message is displayed when Eldo enters this mode. Use the command **.DISFLAT** to disable this mode.

## **.DISTRIB**

### User Defined Distributions (Monte Carlo)

```
.DISTRIB DIST_NAME (DEV1 PROB1) [{(DEVn PROBn)}]
```

This command is used to specify user defined distributions for the device tolerances **DEV** and **LOT** used in Monte Carlo analysis. As such, **.DISTRIB** commands are only active when the **.MC** command is present in the netlist causing a Monte Carlo analysis to be performed.

Different entities are able to share the same distribution. Anywhere Eldo accepts **LOT/DEV** specifications, you can specify **LOTGROUP=group\_name**.



Please refer to “[.LOTGROUP](#)” on page 10-142.

### Parameters

- **DIST\_NAME**  
Name of the user defined distribution being specified.
- **DEVxx**  
A deviance value in the range [-1, 1]. **DEVxx** values must be ordered from lower to upper values. Two successive values of **DEVxx** may be equal in the case of steep distributions.
- **PROBxx**  
A probability value in the range [0, 1].

## **.DSP**

### DSP (Digital Signal Processing) Computation

```
.DSP LABEL=label_name MODEL=model_name waveform_name
```

Selects the waveform on which the user requires DSP to be applied.

#### Parameters

- **label\_name**

It is via this label name that Eldo will select which wave has to be plotted (**.PLOT**) or on which wave extraction of useful quantities has to be done (**.EXTRACT**). Required.

- **model\_name**

It is via **model\_name** that Eldo will know which DSP to apply on the wave specified by **waveform\_name**. **model\_name** must be a valid **.DSPMOD** label. Required.

- **waveform\_name**

Any regular Eldo waveform name: there can be any number of **.DSP** commands specified in the *.cir* file.

#### Examples

```
.DSP LABEL=LBL1 MODEL=PSD_CORRELO V(1)
.DSP LABEL=LBL2 MODEL=PSD_PERIODO W('v(2)+v(4)')
```

These DSP commands define two new PSD entities named LBL1 and LBL2. LBL1 is obtained by applying a DSP model PSD\_CORRELO on wave V(1). LBL2 is obtained by applying a DSP model PSD\_PERIODO on wave V(2)+V(4). PSD\_CORRELO and PSD\_PERIODO must be defined by a **.DSPMOD** command.

Display and **.EXTRACT** of DSP results:

```
[.PLOT | .PRINT | .PROBE] DSP DSPxx(label_name)
.EXTRACT DSP <expression>
```

where xx stands for DB, R, I, P, M.

```
.DSP LABEL=LBL2 MODEL=PSD_PERIODO V(1)
.PLOT DSP DSPDB(LBL2)
```

It is also possible to extract values from a DSP waveform:

```
.DSP LABEL=LBL1 MODEL=PSD_CORRELO V(1)
.PLOT DSP DSPDB(LBL1)
.EXTRACT DSP YVAL(DSPDB(LBL1),5MEG)
```

Eldo will print out the value (in dB) at 5 Meg for the DSP done on V(1).

## .DSPF\_INCLUDE

### Load DSPF File

```
.DSPF_INCLUDE [FILE=]DSPF_FILENAME [INST={list_of_subckt_inst}]  
+ [LEVEL=C|RC|RCC] [DEV=DSPF|SCH[EMATIC]] [RMINVAL=val] [CMINVAL=val]  
+ [CCMINVAL=val] [ADDXNET] [ADDX]  
.DSPF_INCLUDE [FILE=]DSPF_FILENAME [LEVEL=C|RC|RCC]  
+ [DEV=DSPF|SCH[EMATIC]] DEDICATEDDX=subckt_name [RMINVAL=val]  
+ [CMINVAL=val] [CCMINVAL=val] [ADDXNET] [ADDX]
```

By specifying this command inside a netlist, Eldo will add the parasitic elements described inside the specified DSPF (Detailed Standard Parasitic Format) file.

To include DSPF files containing the complete netlist (active elements and parasitics) use the **.INCLUDE** command. Subcircuits can be instantiated using either the **.TOPCELL** command or an **X** instance.

Use the option **MAX\_DSPF\_PLOT**=val ([page 11-46](#)) to override the maximum number of DSPF interface nodes plotted. By default, this limit is 100.

The **INST** and **DEDICATEDDX** specifications are mutually exclusive.

### Parameters

- **DSPF\_FILENAME**  
Name of the DSPF file. This parameter must be the first specified on the command line.  
**FILE=** is optional.
- **INST**  
Specifying this parameter will force Eldo to only include the parasitic elements for the specified subcircuit instances given in **list\_of\_subckt\_instances**. It will assume that the DSPF file is hierarchical.
- **LEVEL=C | RC | RCC**  
Defines the level of parasitics to use from the specified DSPF file. This can also be specified using **.OPTION DSPF\_LEVEL** (see [page 11-25](#) for more information).
- **DEV=DSPF | SCH[EMATIC]**  
**DSPF** specifies the DSPF file contains only the parasitics.  
**SCH[EMATIC]** specifies the DSPF file contains both the parasitics and the intentional devices. Sometimes it is more accurate to have the extractor generating a file containing both.  
When **DEV=DSPF** is specified, Eldo will take three actions:
  - Eldo will include the file specified in the command as if it was a **.LIB** command. Eldo will therefore have two variants of the same subcircuit in its database, one corresponding to the intentional devices which should be present in the circuit, and

another one included by the **.DSPF\_INCLUDE** command, corresponding to the version with its parasitics.

- Eldo will emulate a **.BIND** command on the instances/subcircuits which are specified on the **.DSPF\_INCLUDE** command.
- Eldo will apply filters, if any, to the DSPF devices.

Generally, the number of pins and the pin order given in the **.SUBCKT** found in the DSPF file differs from the list in the intentional design. Eldo will check for pin names in both intentional and DSPF subcircuit to ensure proper connection, based on node name.

In case the DSPF version contains more pins than the intentional, then it is assumed that these pins correspond to global nets, and Eldo will map as such. Errors will be issued if no such global name can be found. Note that the index of the pin in the IX or VX plot command refers to the index of the intentional design.

- **RMINVAL**

Specifies the minimum resistance of a resistor, in ohms. Resistors with a value less than the specified minimum value are removed from the netlist and replaced by short circuits. The default value is -infinity.

- **CMINVAL**

Specifies the minimum value of a grounded capacitor, in Farads. Grounded capacitors with a capacitance of less than the specified minimum value are removed from the netlist. The default value is infinity.

- **CCMINVAL**

Specifies the minimum value of a floating capacitor, in Farads. Floating capacitors with a capacitance of less than the specified minimum value are removed from the netlist. The default value is infinity.

- **ADDXNET**

Keyword instructing Eldo to insert an X character on hierarchical names specified in the DSPF file for retrieving the node in the pre-layout design (where X is added as necessary). This addition of the X character is used only for the \*| NET instructions. For example see [page 10-84](#).

- **ADDX**

Keyword instructing Eldo to insert an X character on hierarchical names specified in the DSPF file for retrieving the node in the pre-layout design (where X is added as necessary). This addition of the X character is used only for both \*| NET and \*| I instructions. There is no flag specifically for just \*| I instructions.

- **DEDICATEDX=subckt\_name**

The specified DSPF file, will *only* be used for the dedicated instance `subckt_name`. Eldo will assume that the **.SUBCKT** command is not specified in the DSPF file and that the node name in the DSPF file refers to the node names local to the dedicated instance name.

Without the option Eldo will assume that the nodes/objects in the DSPF are the top level nodes/objects.

In DSPF files, it is common not to specify the `X` on subcircuit names that are referenced in the file. When Eldo cannot find a NET name (in `* | NET` command), Eldo will check whether the NET exists with the letter `X` specified. If the node is found, Eldo will assume that the `X` was meant for all node names in the DSPF file.

If the contents of the DSPF file are:

```
.subckt top IN OUT
* | NET TOP/net1 0.827ff
* | I (TOP/1/M1:D TOP/1/M1 D O 0ff)
* | I (TOP/1/M2:D TOP/1/M2 D O 0ff)
* | I (TOP/2/M1:G TOP/2/M1 G I 0.070ff)
* | I (TOP/2/M2:G TOP/2/M2 G I 0.105ff)
R1 TOP/1/M1:D TOP/net1:1 100
```

and the contents of the `.cir` file are:

```
.subckt top in out
XTOP in out inv2
.ends
V1 in 0 pulse( 0 2 1p 100p 100p 5n 10n)
C1 out 0 1f
X1 in out top
```

Eldo will not find the node `X1.TOP.NET1`. Therefore, Eldo will look for `X1.XTOP.NET1`, which will be found. Eldo will now *assume* that the letter `X` should be used for all the nodes in the DSPF. Eldo will now assume that the instruction:

```
R1 TOP/1/M1:D TOP/net1:1 100
```

should be:

```
X1.RTOP.NET1_R1 X1.XTOP.X1.M1:D X1.XTOP.NET1:1 100
```

## Connecting a source to a DSPF back-annotated node

It is possible to connect a source to a node that is back-annotated with DSPF. This might be useful to insert a voltage or current source to a specific node to check a critical path. However, after DSPF extension, the original node might be expanded in several nodes, and there might be no path remaining between the extra source and the network, i.e. the extra source no longer has a connection to the network. This is because the extra source was not in the design when DSPF information was created. The solution is that nodes connected to that extra source are not expanded by DSPF. Eldo can detect such situations when the extra source is instantiated from the top of the design and connected to a node at a lower level of hierarchy, using the full node pathname. For example:

```
.SUBCKT INV in out
...
...
```

---

```
R1 in internal 1k
...
.ends
X1 in out INV
v1 X1.internal 0 5      !full pathname usage
```

Eldo will detect this configuration, and DSPF instructions will be ignored on node X1.internal. This is similar to the command: **.IGNORE\_DSPF\_ON\_NODE** X1.internal.

However if the extra source is added at the same level as the nodes (for example all at the top), Eldo will only be able to issue a message that the node is connected to a single device only, and will indicate that there is a possible DSPF connection problem. The user must then manually add the command **.IGNORE\_DSPF\_ON\_NODE** on that node in order to deal with that specific situation.

## Related Command

**.IGNORE\_DSPF\_ON\_NODE**, [page 10-127](#).

## Restrictions

This command does not work for Mach TA blocks when using Eldo Mach.

## Example

This example (see netlist on next page) can be run with or without the following line to compare the simulation results with or without the parasitics effects:

```
.dspf_include testspf2.spf
```

Inside this example, when the DSPF is included, the node XTOP.net1 is replaced by a network of resistors, so, according to the DSPF file, the node itself does not exist anymore, but is replaced by three different nodes: XTOP.net1:1, XTOP.net1:2, XTOP.net1:3. In this case, the plot statement: **.plot tran v(in) v(out)** must be replaced by **.probe tran v(XTOP.net1\*)** to plot all the nodes “composing” the original node XTOP.net1.

File *test\_dspf.cir*:

```
* test DSPF
.dspf_include testspf2.spf

.global vdd gnd

.subckt inv a y
+ln=2e-07 wn=2.8e-07 lp=2e-07 wp=2.8e-07
M1 Y A vdd vdd EPLLMM9JU w=wp l=lp m=1
M2 Y A gnd gnd ENLLMM9JU w=wn l=ln m=1
.ends

.subckt inv2 a y
X1 a net1 inv ln=0.18u wn=2.0u lp=0.18u wp=4.0u
X2 net1 y inv ln=0.18u wn=2.0u lp=0.18u wp=4.0u
```

```

.ends

V1 in 0 pulse( 0 2 1p 100p 100p 5n 10n)
C1 out 0 1f
XTOP in out inv2

VDD vdd 0 2
VGND gnd 0 0

.plot tran v(in) v(out)
*.plot tran v(XTOP.net1)
.probe tran v(XTOP.net1*)

.tran 1n 20n

.model EPLLMM9JU PMOS level=53 rd=100 rs=100
.model ENLLMM9JU NMOS level=53 rd=80 rs=80

.end

```

File *testspf2.spf*:

```

* | DSPF 1.0
* | DIVIDER /
* | DELIMITER :
* | GROUND_NET VSS
* | NET XTOP/net1 0.827ff
* | I (XTOP/X1/M1:D XTOP/X1/M1 D O 0ff)
* | I (XTOP/X1/M2:D XTOP/X1/M2 D O 0ff)
* | I (XTOP/X2/M1:G XTOP/X2/M1 G I 0.070ff)
* | I (XTOP/X2/M2:G XTOP/X2/M2 G I 0.105ff)
R1 XTOP/X1/M1:D XTOP/net1:1 100
R2 XTOP/X1/M2:D XTOP/net1:1 100
R3 XTOP/net1:1 XTOP/net1:2 2000
R4 XTOP/net1:2 XTOP/net1:3 2000
R5 XTOP/X2/M1:G XTOP/net1:3 100
R6 XTOP/X2/M2:G XTOP/net1:3 100
C1 XTOP/X2/M1:G VSS 1.53ff
C2 XTOP/X2/M2:G VSS 1.71ff
C3 XTOP/net1:1 VSS 2.80ff
C4 XTOP/net1:2 VSS 2.80ff
C5 XTOP/net1:3 VSS 2.80ff
.END

```

Parasitic resistor values have multiplied by 1000 in order to display a real DSPF loading impact on the results.

## Example 2

When the DSPF file does not contain the **.SUBCKT** command, for example:

```

dedicated_example.spf
* | NET TOP/net1 0.827ff
...

```

and the *.cir* file contains:

---

```

.SUBCKT top in out
XTOP in out inv2
.ENDS
V1 in 0 pulse( 0 2 1p 100p 100p 5n 10n)
C1 out 0 1f
X1 in out top

.DSPF_INCLUDE dedicated_example.spf DEDICATEDX=X1

```

Specifying **DEDICATEDX=X1**, the node **TOP** in the DSPF file will refer to the node **X1.TOP** in the *.cir* file, i.e. node **IN**.

### Example 3

Naming in DSPF files can vary from one extractor to the next. In particular, node names specified in the \*|NET instructions and device names specified in the \*|I instructions may or may not contain the letter X in names.

For instance, inside the DSPF could be:

```

* | NET I117/net18 0.00415015PF
* | I (XI117/XM26:D XI117/XM26 D   B 0 62.255 143.84)
* | I (XI117/XM31:D XI117/XM31 D   B 0 67.595 146.295)

```

This means that on \*|NET is XI117/net18, while on the \*|I instance naming is correct from the Eldo point of view.

Setting keyword parameter **ADDXNET** instructs Eldo to insert an X character on hierarchical names specified in the DSPF file for retrieving the node in the pre-layout design (where X is added as necessary).

## **.DSPMOD**

### PSD (Power Spectral Density) Computation

```
.DSPMOD DSP=CORRELO | PERIODO LABEL=label_name
+ [TSTART=val] [TSTOP=val] [FS=val] [NBPT=val]
+ [PADDING=val] [WINDOW=name] [ALPHA=val] [BETA=val]
+ [NORMALIZED=val] [INTERPOLATE=val] [DISPLAY_INPUT=val]
+ [FNORMAL=val] [FMIN=val] [FMAX=val]
+ [NAUTO=val] [NCORR=val] [NPSD=val] [NSECT=val]
```

-  For details on the principles of PSD computation, please see [page 2-53](#) of the *Xelga User's Manual*.

### Histogram Computation

```
.DSPMOD DSP=HISTOGRAM LABEL=label_name NBINTERVAL=val
+ [XSTART=val XSTOP=val SAMPLE=YES | NO FS=val]
```

Generates a histogram of the input wave showing the wave's magnitude probability density distribution. The input wave is first sampled to equidistant points if the optional argument **SAMPLE** was set.

-  For Histogram parameter descriptions, please see [page 10-88](#) of this manual.

There can be any number of **.DSPMOD** commands specified in the *.cir* file.

### PSD Computation Parameters

- **DSP=CORRELO | PERIODO**  
Specifies either the correlogram or periodogram method.
  - **LABEL**  
It is via this label name that Eldo will select which model has to be applied to a specific input signal. Required.
  - **TSTART**  
Time value from which the time window will start.
  - **TSTOP**  
Time value that specifies the end of the time window.
  - **FS**  
Sampling frequency.
  - **NBPT**  
Number of sampling points to be used in the time window.
- These parameters satisfy the following equation:

$$\frac{Nbpt - 1}{Fs} = Tstop - Tstart$$

- **PADDING**

Specifies that padding zeros before or after the signal can be added.

- 0: No padding
- 1: Zeros are added at the beginning of the FFT input window
- 2: Zeros are added at the end of the FFT input window (default)
- 3: Zeros are added evenly at the beginning and at the end of the FFT input window.

- **WINDOW**

Specifies the Sampling Window to be used. The following parameters are allowed:

**RECTANGULAR** (default), **PARZEN**, **BARTLETT**, **WELCH**, **BLACKMAN**, **BLACKMAN7**, **KLEIN**, **HAMMING**, **HANNING**, **KAISER**, **DOLPH\_CHEBYCHEV**.



- **ALPHA**

Used when the **WINDOW** is **DOLPH\_CHEBYCHEV** or **HANNING**.

- **BETA**

Used when the **WINDOW** is **KAISER**. Constant which specifies a frequency trade-off between the peak height of the side lobe ripples and the width of energy in the main lobe.

- **NORMALIZED**

Specifying 1 means that all the points of the result are multiplied by 2/**NBPT**. Default. Specifying 0 means no normalization of the results.

- **INTERPOLATE**

Sets type of interpolation:

- 0 = No interpolation (default)
- 1 = Linear interpolation
- 2 = Cubic Spline interpolation
- 3 = Blocker Sampler interpolation

- **DISPLAY\_INPUT**

Allows visualization of the transient window used as input of the PSD: this waveform contains equally-spaced time value points.

- 0 = do not display PSD input (default)
- 1 = display PSD input (if output is displayed)

- **FNORMAL**  
Adjusts the results around the Y-axis so that the point for the specified frequency is 0.0.
- **FMIN**  
Starting frequency used inside the result window.
- **FMAX**  
Last frequency used inside the result window.
- **NAUTO**  
Number of points for the autocorrelation result.
- **NCORR**  
Number of autocorrelation points used for the PSD computation.
- **NPSD**  
Number of points for the PSD result.
- **NSECT**  
Number of points for each section. Can only be specified if the periodogram method is specified (**DSP=PERIODO**).

## PSD Computation Examples

```
.DSPMOD DSP=CORRELO LABEL=PSD_CORRELO
+ TSTART = 200n
+ TSTOP = 400n
+ NORMALIZED = 0
+ INTERPOLATE=0
+ DISPLAY_INPUT=1
```

This DSP model command defines a set of parameters for a PSD computation using the correlogram method. Input values will be sampled from 200n to 400n. Default value for NBPT is 1024. Output signals will not be normalized and the PSD input signal will be dumped in the output file.

```
.DSPMOD DSP=PERIODO LABEL=PSD_PERIODO
+ TSTART = 0
+ TSTOP = 1.93e-8
+ NBPT = 100
+ NORMALIZED = 0
+ INTERPOLATE=0
+ DISPLAY_INPUT=1
+ FS=5.12e9
+ ALPHA=0.5
```

This DSP model command defines a set of parameters for a PSD computation using the periodogram method. The input signal has 100 points, sampled between 0 and 19.3n.

## Histogram Computation Parameters

- **DSP=HISTOGRAM**  
Specifies the histogram method.
- **LABEL**  
It is via this label name that Eldo will select which model has to be applied to a specific input signal. Required.
- **NBINTERVAL**  
Specifies the number of intervals which will be used to compute the distribution.
- **XSTART**  
X value from which the measurement window will start.
- **XSTOP**  
X value from which the measurement window will stop.
- **SAMPLE**  
If YES, perform a sampling of the input wave (default is NO).
- **FS**  
Sampling frequency.

## Histogram Computation Example

```
.DSPMOD DSP=HISTOGRAM LABEL=HISTO1 NBINTERVAL=10
+ XSTART=3n XSTOP=END SAMPLE=YES
.DSP label=s_histo model=HISTO1 v(1)
.PLOT dsp dsp(s_histo)
```

## .END

### End Eldo Netlist

.END

This command terminates the simulator input description. Any text which follows it is ignored, unless there is another **.END** statement; in which case, all lines between the preceding **.END** and the current **.END** will be considered as another circuit to simulate. This enables the possibility to have several circuits in the same netlist file.

For multi-circuit simulations, Eldo simulates all the circuits contained inside the input files; the **.END** statement is optional inside the first circuit, but it must be specified inside the subsequent circuits.

### Example

```
first title
...
...
.end
second title
...
...
.end
```

The line following the **.end** statement is the title of the next circuit.

## .ENDL

### End Eldo Library Variant Description

.ENDI

This command terminates a library variant description. Any text which follows it is ignored.



See “[Library variant management](#)” on page 10-135 for details.

---

## .ENDS

### End Eldo Subcircuit Description

.ENDS

This command terminates an Eldo subcircuit description. Any text which follows it is ignored.

---

**Note**

---



.EOM is equivalent to .ENDS.

---

## **.EQUIV**

### Replace Node Name for Display

```
.EQUIV new_name=netlist_name
```

This command is used to change the name of a netlist node to a new display name. The new name can then be used in display output statements such as **.PLOT** and **.EXTRACT**. Both the original name and the new name will be valid. The name can be a hierarchical name.

### Parameters

- `new_name`  
New name of node for display.
- `netlist_name`  
Node name to be changed.

### Examples

This is a full netlist using the **.EQUIV** command:

```
equiv

.sigbus B[3:0] vhi=2.5 vlo=0 trise=0.1n tfall=0.1n base=hexa
+ 0n A
+ 10n 3
+ 20n 4
+ 30n 8
+ 50n F
r0 B[0] 0 1k
r1 B[1] 0 1k
r2 B[2] 0 1k
r3 B[3] 0 1k

.tran 1n 60n
.equiv enable=b[0]
.equiv xin.out=b[1]

.probe
.plot tran v(enable)
.plot tran v(xin.out)
```

From the above netlist, the following two lines change the names of the nodes `b[0]` and `b[1]` to `enable` and `xin.out`. The name can be anything the user chooses.

```
.equiv enable=b[0]
.equiv xin.out=b[1]
```

From the above netlist, the **.PLOT** command uses the new node names, `v(enable)` and `v(xin.out)`. These names will be the names displayed in the waveform viewer after the simulation is carried out.

```
.plot tran v(enable)
.plot tran v(xin.out)
```

The following example uses the **.EQUIV** command with a voltage source node:

```
v1 vss 0 5
.tran 10n 100n
.equiv vin=vss
.plot tran v(vin)
```

The following example show how the new name can be a hierarchical name:

```
.EQUIV XBUF.IN = NET$34
.PLOT TRAN V(XBUF.IN)
```

## **.EXTMOD**

### Extract Mode

```
.EXTMOD [FILE=filename]
```

The **.EXTRACT** commands are merged with the simulation itself. In some cases, this causes a lot of wasted CPU time. If the user makes a mistake when specifying the **.EXTRACT** command, the command must be fixed, then the simulation itself re-run to obtain the correct results. When the simulation takes a few seconds, this is not a problem. When the simulation takes a few hours of CPU, this is of course a big problem.

Use the **.EXTMOD** command to perform only the **.EXTRACT** commands, without performing the simulation(s). Eldo will decorrelate the ‘simulation’ from the ‘extraction’. See [page 10-95](#) for more information on the **.EXTRACT** command.

This may cause large output files. Eldo automatically identifies the waveforms used in **.EXTRACT/.MEAS** statements and save them to the file. This is preferable to being forced to re-run hours of CPU because the wrong **.EXTRACT** formula was entered.

This functionality can also be specified with the command line argument **-extract [filename]**. See [page 2-6](#).

- **filename**

Name of an Xelga **.cou** or EZwave **.wdb** output file. The extractions in the netlist will be solved using the waves contained in this file.

## .EXTRACT

### Extract Waveform Characteristics

```
.EXTRACT [EXTRACT_INFO] [LABEL=NAME] [FILE=FNAME] [UNIT=UNAME] [VECT]
+ [CATVECT] $MACRO|FUNCTION [OPTIMIZER_INFO] [MC_INFO]
```

This command extracts waveform information using a combination of arithmetic expressions or pre-defined functions. A flexible language exists in Eldo to extract characteristics from raw simulation results (for example the maximum value of a waveform, or the time at which a given threshold is crossed by a waveform, etc.). The type of analysis for which the specified extraction is carried out may be defined. This is useful when different types of analyses are performed in the same simulation run. The results are listed to the ASCII output (*.chi*) file. They can also be written to a specified file *fname.aex* when **.OPTION AEX** is specified in the netlist. The command also creates a *.ext* file with the extraction results which can be read by EZwave or Xelga when **.OPTION EXTFILE** is specified in the netlist. This file will not always be created as it depends on the type of simulation and the specification of the **.EXTRACT** command. By default, extraction results are saved inside the *EXT* folder in the main *.wdb* file.

The **.PLOT** command may also be used to plot *extraction* results. It is used to specify which simulation results have to be kept by the simulator for graphical viewing and post-processing. See the **.PLOT** command on [page 10-216](#).

When extraction statements are combined with parameter sweeping statements (see the **.STEP** command on [page 10-298](#)), Eldo automatically creates waveforms showing the extraction results versus the swept parameter. For example, a user may extract the width of an output pulse from a transient simulation, and sweep the power supply level. In this case Eldo will automatically create a waveform showing the width of the pulse versus the power supply level. Similarly, if extractions and **.ALTER** statements are combined, Eldo will automatically create waveforms showing the extraction results versus the index of the **.ALTER** runs (see **.ALTER** command on [page 10-22](#)). In this case, the X axis will contain 1, 2, 3, etc. The initial display in the viewer can also be prepared using **.PLOT** commands (see the **.PLOT EXTRACT...** description [page 10-223](#)). Example:

```
.EXTRACT TRAN label=VMAX MAX(V(out))
.PARAM powersupply=1.2
VDD VDD 0 'powersupply'
.STEP PARAM powersupply list 1.2V 1.3V 1.4V 1.6 2V
.PLOT EXTRACT meas(VMAX) ! this will create a waveform
* showing VMAX(powersupply)
```

By default, using the *.wdb* format, the **.EXTRACT** waveforms are saved inside the *EXT* folder in the main *.wdb* file.

If using the *.cou* format, these **.EXTRACT** waveforms are created in a separate file with the *.ext* extension. This file can be loaded by Xelga. The command-line switch **-couext (eldo -couext -i <netlist.cir>)** can be used to merge the extract waveforms into the *.cou* file, when

possible. In this case, a single *.cou* file will contain the *raw* waveforms and also the extraction waveforms.

Use the option **ALIGNEXT** to specify Eldo to write tabulated **.EXTRACT** results in the *.aex* file. By default the results are not tabulated (aligned). Remember, the *.aex* file is only created if the option **AEX** is activated.

It is also possible to extract values from an FFT waveform. Additionally, in the case where a set of simulation runs were done with a parameter (P) being varied, and a given output (O) extracted, users may wish to extract a value based on the obtained P/O curve.

It is always preferable to insert any **.EXTRACT** commands at the end of the netlist, since they can refer to objects (such as voltage sources) which need to be defined beforehand.

The **.EXTRACT** command has been extended for the purpose of optimization.



For more information see the [Optimizer in Eldo](#) chapter.

The keywords **EXTRACT( )** or **MEAS( )** can be used in **.EXTRACT** statements to perform an “EXTRACT of EXTRACT” such as in the example below:

```
.TEMP 10 20 30
.EXTRACT LABEL = T1 MAX(v(s))
.EXTRACT YVAL(EXTRACT(T1),20) ! extract T1 for T=20Celcius
```

Here, the second **.EXTRACT** is by default not evaluated after a single analysis, but only on completion of a SWEEP analysis, i.e. on completion of the three temperature analyses in the example above.

To compare an extracted wave to a reference specify the **UNIT** parameter to define the unit of the extract. By default EZwave displays separate axis. This can be used when plotting the extract wave during a sweep analysis.

If you want to use results of EXTRACT in another **.EXTRACT** command that applies to a single analysis, i.e. not to SWEEP analysis, then the keyword **AC**, **TRAN**, or **DCSWEEP** should be specified, otherwise the value is not extracted. However, when there are no ambiguities, i.e. only one analysis being performed, without **.STEP** or **.TEMP** commands, then the keyword is optional.

If a **.EXTRACT** command is present inside a **.ALTER** command, a *.ext* file will be created on the condition that the **.EXTRACT** statement remains unchanged for each **.ALTER** file. If this condition is not met, the *.ext* file is not created. For further details regarding the **.ALTER** command, see [page 10-22](#).

If the user makes a mistake when specifying a **.EXTRACT** command then they can correct the offending command and re-invoke Eldo using the **-extract** flag. This will then only re-run

the **.EXTRACT** commands in the netlist without performing the simulation(s), see [page 2-6](#) for more information. This is especially useful for simulations that take a considerable amount of time. This functionality can also be specified using the command **.EXTMOD** see [page 10-94](#) for more details.

It is possible to use wildcards in **.EXTRACT** commands. For example:

```
.extract tran max(v(*))
.extract tran yval(v(x*.1-9]*),20n)
.extract tran yval(id(M*o),20n)
.extract tran min(i(X*.M*.d),20n)
.extract tran yval(v(x*.ti*),20n)
```

Quantities which allow wildcards are the same as those that can be used in **.PROBE**, see [page 10-258](#).

## Parameters

To ensure that compatibility with previous versions of Eldo is maintained, it is preferable to select one of the optional parameters from the following list:

- **EXTRACT\_INFO**

should be replaced with one of the following parameters;

**AC**

Extraction during AC analysis.

**DC**

Extraction during DC analysis.

**DCTRAN**

Extraction after the DC analysis performed prior to a TRAN analysis.

**DCAC**

Extraction after the DC analysis performed prior to an AC analysis.

**DCSWEET**

DCSWEET extraction.

**TRAN**

Extraction during TRAN analysis.

**NOISETRAN**

Extraction during **NOISETRAN** analysis.

**FOUR**

Extract values from an FFT waveform.

**DSP**

Extract values from a DSP waveform.

**SWEET**

Used in conjunction with **.STEP** to extract values from a curve (EXTRACT = FUNCTION(SWEEP\_parameter)).

**SSTAC|SSTXF|SSTNOISE|SSTJITTER|TMODSST|FMODSST|FOURMODSST|TSST|FSST|SSTSTABIL**

Please refer to the [Display Command Syntax](#) chapter of the *Eldo RF User's Manual* for more information regarding these RF options.

**OPT**

Extraction during optimization analysis. Can only be used with wave type **WOPT**. This wave type is used for optimization analysis only, i.e. it may only be used with **OPT**. For more information on wave type **WOPT** see [page 20-42](#).

- **LABEL=NAME**

Label name of the extraction result. This can also be a string as shown below:

```
.EXTRACT [AC|DC|DCAC|DCTRAN|DCSWEEP|TRAN|
+ NOISETRAN|OPT] [LABEL=NAME] <expression>
.EXTRACT [AC|DC|DCAC|DCTRAN|DCSWEEP|TRAN|
+ NOISETRAN|OPT] [LABEL=" string "] <expression>
```

If a **LABEL** is specified, then you will obtain " **LABEL** "=result inside the output file and not <expression>=results.

- **FILE=FNAME**

Results will be dumped into the specified file. The values are also still written into the *.chi* file, i.e. the same values and information are dumped in both files.

- **UNIT=UNAME**

Defines the unit of the extract. It is used when plotting the extract wave during a sweep analysis. This could be useful when you want to compare an extracted wave to a reference because by default EZwave displays separate axis.

- **VECT**

By default, **.EXTRACT** returns the first value which matches the expression. If keyword **VECT** is set on the **.EXTRACT** statement, then all values will be returned.

---

**Note**

 Extraction with **VECT** does not work when it is done in reverse. In the instance below, the message "VECT keyword cannot be used when extracting backwards" will be printed:

```
.EXTRACT VECT xup(v(1),2.5,END,START)
```

---

- **CATVECT**

Works in the same way as **VECT** but in addition all measurements corresponding to all analyses (**.STEP/.TEMP**) will be combined. This functionality is usually used in

conjunction with the **CONTOUR** function in the **.PLOT** command. For more information on the **CONTOUR** function see [page 10-223](#).

- **OPTIMIZER\_INFO**

Represents additional arguments for optimization. Note that these additional arguments (for example GOAL, LBOUND) have no effect when the **.OPTIMIZE** command is not specified in the netlist. For more information, please refer to [page 20-29](#).

- **MC\_INFO**

Represents additional arguments for Monte Carlo analysis. For more information, please refer to [page 19-7](#).

- **\$MACRO**

Instantiation of a macro previously defined using the **.DEFMAC** command. The macro name must be preceded by the **\$** character.

- **FUNCTION**

Pre-defined function. The functions listed on [page 10-100](#) are available. Many of these functions use the optional parameters listed in the table below:

**Table 10-4. Extract Function Parameters**

BEFORE=VAL	Causes the function to be performed only if TIME < val.
AFTER=VAL	Causes the function to be performed only if TIME > val.
OCCUR=VAL	Computes the function for the VALth occurrence of the event.
VTH=VAL	Voltage defining a threshold or starting point.
VTHIN=VAL	Voltage defining a threshold or starting point.
VTHOUT=VAL	Voltage defining a threshold or starting point.
VH=VAL	Voltage defining a threshold or starting point.
VL=VAL	Voltage defining a threshold or starting point.
WAVE	Valid waveform name or keyword <b>XAXIS</b> . <b>XAXIS</b> extracts an x-axis value when a condition becomes true, i.e. refers to the TIME if current simulation is transient, or frequency if current simulation is AC analysis. It is possible to use wildcards in wave names. For more information see <a href="#">page 10-96</a> .
MIN	Minimum value on x-axis or keyword <b>START</b> . <b>START</b> corresponds to the first value of the <b>XAXIS</b> , i.e. the starting point of the simulation.
MAX	Maximum value on x-axis or keyword <b>END</b> . <b>END</b> refers to the last point of the simulation.

In all cases below where **VDD** and **VSS** are used in function definitions, **VDD** represents the larger, and **VSS** the smaller voltage input found in the netlist at **TIME=0**. These values are computed only once, even when a sweep is performed on **VDD** or **VSS**.

For all function arguments requiring a value, a parameter name may be passed to the function, the parameter value being specified via the following command:

```
.PARAM PARAM_NAME=VAL
```

For wave arguments used in the functions below, either the name of a waveform created using the **.DEFWAVE** command or a waveform expression may be used. For example:

```
.extract max(W('V(s)-V(a)'))
```

which is equivalent to:

```
.defwave my_wave=V(s)-V(a)
.extract max(W(my_wave))
```

## Two-port Noise Parameters

Any noisy two-ports can be represented by the equivalent noiseless two-port with the two equivalent noise sources ( $e_n$  and  $i_n$ ) or the corresponding noise correlation matrix  $C^A$ .



For more information on these parameters see [page 10-223](#).

It is also possible to use the center and radius of RF Gain/Noise circles in an extract command. The quantities are in the form:

```
<noise_circle>_R    !(real part)
<noise_circle>_I    !(imaginary part)
<noise_circle>_RAD  !(radius)
```

where `<noise_circle>` can be any of the following: **GAC**, **GPC**, **LSC**, **SSC**, **NC**.

## FUNCTION

There are two types of Extraction Language:

- Transient Extraction Language:  
Faster to execute, and consumes less memory. Transient extract language is based on the following functions:

**Table 10-5. Transient Extraction Language Functions**

D_WA	DTC	SLEWRATE	SLOPE	TCROSS
TINTEG	TPD	TPDUU	TPDUD	TPDDU
TPDDD	TRISE	TFALL	VALAT	

- General Purpose Extraction Language:  
More expensive in terms of memory and CPU, but is much more general in the sense that arguments can be expressions. It is based on the following functions:

**Table 10-6. General Extraction Language Functions**

AVERAGE	COMPRESS	DCM	DISTO	EVAL
INTEG	KFACTOR	MAX	MIN	MODPAR
OPMODE	POW	POWER	PVAL	RMS
WFREQ	WINTEG	XCOMPRESS	XDOWN	XMAX
XMIN	XTHRES	XUP	XYCOND	YVAL

General purpose extraction language functions do not accept **EXTRACT( )** and **MEAS( )** keywords as parameters when they reference results from other **.EXTRACT** commands.

If **MIN** and **MAX** are not specified in a function call, information is returned when the **CONDITION** is true for the first time.

The x-axis values **MIN** and **MAX** may be replaced by the keywords **START** and **END** to specify the beginning and end of the simulation interval respectively. The x-axis value **MIN** may be made greater than the **MAX** value. This causes Eldo to look backwards when **CONDITION** is true for the first time. This is useful for extracting waveform settling time data.

## AVERAGE

**AVERAGE(WAVE [, MIN, MAX])**

Returns the average value of the waveform **WAVE** in the x-axis range **MIN** to **MAX**. This is calculated as follows:

$$\text{AVERAGE} = \frac{1}{\max - \min} \times \int_{\min}^{\max} \text{wavedx}$$

## COMPRESS

**COMPRESS(WAVE, VALUE [, SLOPE])**

Extracts the Y-axis value of the wave at the point where the difference between the actual value of **wave** and the linear extrapolation of **wave** based on the computed slope value becomes greater than **value**.

**SLOPE** corresponds to the minimum slope which must be between the first two points of the wave. If the actual slope on the signal is not high enough, a message will be displayed. Default for this 3rd argument is 0.99 if the **COMPRESS** is performed on a **SWEEP** extract. Otherwise, it is undefined.



See also [XCOMPRESS](#) and for further information see “[.EXTRACT—Compression Point Values](#)” on page 10-112.

## D\_WA

**D\_WA(WAVE1, WAVE2 [, AFTER=VAL])**

Returns the distance between two waveforms. **D\_WA** has the units of the waveforms, and is computed by dividing the area between the two waves by the x-axis range. **D\_WA** may also be used in DCSWEEP to compare two characteristics of currents, e.g.

```
.extract d_wa(v(a), v(b))
.extract d_wa(i(r1), id(m1))
```

## DCM

**DCM[ (label\_name) ]**

Returns the DC mismatch information for a specific *label\_name* (see the [.DCMISMATCH](#) command). It extracts the value which is dumped in the *.chi* file, so you do not have to browse the *.chi* file to obtain that value. **DCM** can be specified without any argument (*label\_name*) if there is no ambiguity about the name.

## DTC

**DTC(WAVE [, VTH=VAL] [, AFTER=VAL] [, BEFORE=VAL])**

Returns the duty-cycle of the waveform *WAVE* in transient analysis.

This is equivalent to the three extracts:

```
.extract tran label=period abs(xup(wave,vth,2) - xup(wave,vth,1))
.extract tran label=cycle abs(xdown(wave,vth,1) - xup(wave,vth,1))
.extract label=duty_cycle {extract(cycle)/extract(period)}
```

## DISTO

**DISTO(WAVE, FUND\_FREQ, FMIN, FMAX)**

Returns the total harmonic distortion of a signal. *FUND\_FREQ* is the fundamental frequency and *FMIN* and *FMAX* specify the window in which you require the harmonic distortion to be calculated.

The harmonics inside the interval *[FMIN, FMAX]* are then computed as follows:

$$\text{harmonic}(i) = \left| \frac{A(i)}{A_0} \right|$$

where:

*A(i)* = amplitudes of the multiples of the fundamental frequency

*A0* = amplitude of the fundamental frequency

The Total Harmonic Distortion is given by the following:

$$\text{tot\_harm} = \sqrt{\sum \text{harmonic}(i)^2}$$

where the sum is computed over all multiples ( $\geq 2$ ) of the fundamental frequency in the specified band. If these values are not identical to the sampled data values, then they are computed by interpolation.

The results of the harmonic distortion are given as a percentage ( $100 \times \text{tot\_harm}$ ).

```
.extract four disto(fourdb(fft_label),5meg,0,2.5e8)
.extract fsst disto(vm(node),75meg,10meg,600meg)
```

## EVAL

```
E[VAL]( INSTANCE, W|L|AD|AS|PD|PS|AREA)
E[VAL]( INSTANCE, Weff|Leff|ADeff|ASEff|PDeff|PSeff|Geo|RDeff|RSeff)
E[VAL] (dipole_name)
```

Returns the value of the requested parameter for the circuit element `INSTANCE` or the `dipole_name`. The instance must be declared before the `.EXTRACT` command, e.g.

```
.extract eval(m1, w)
```

The second syntax shows additional parameters which are only available for MOSFETs.

## INTEG

```
INTEG(WAVE [, MIN, MAX])
```

Returns the integral of the waveform `WAVE` in the x-axis range `min` to `max`. This is calculated as follows:

$$\text{INTEG} = \int_{\text{min}}^{\text{max}} \text{wavedx}$$

## KFACTOR

Computes the stability factor for 2-ports. The keyword returns:

$$\frac{(1.0 - |S11|^2 - (|S22|^2 + |S11 \times S22 - S12 \times S21|^2))}{(2 \times |S12 \times S21|)}$$

`S11`, `S22`, and so on, are the S parameters. This is available with `AC` and `FSST` results.

## MAX

```
MAX(WAVE [, MIN, MAX])
```

Returns the maximum value of the waveform `WAVE` in the x-axis range `MIN` to `MAX`.

## MIN

```
MIN(WAVE [, MIN, MAX])
```

Returns the minimum value of the waveform `WAVE` in the x-axis range `MIN` to `MAX`.

**MODPAR****MODPAR(MODNAME, PARAM\_NAME)**

Returns the value of the parameter, `PARAM_NAME`, of the model, `MODNAME`.

To extract the value of a parameter of a model defined inside a subcircuit:

- When no model parameters are an expression of an instance parameter, to extract a model parameter defined inside a subcircuit use the syntax  
`modpar(subckt_name.model_name, parameter_name)`, for example:

```
.extract dc modpar(toto.tutu, vt0)
```

- When some model parameters are an expression of an instance parameter, to reach a model parameter defined inside a subcircuit use the syntax  
`modpar(hierarchical_instance_name.model_name, parameter_name)`, for example:

```
.extract dc modpar(x1.tutu, vt0)
```

**OPMODE****OPMODE(DEVICE\_NAME)**

Returns the DC working mode of the device model, `DEVICE_NAME`.

Eldo extracts the DC working mode of a device as a string instead of a real value. Strings returned are:

- for MOS devices:  
**LINEAR**, if  $VGS > VT$  and  $VDS < VDSAT$   
**SATURATION**, if  $VGS > VT$  and  $VDS > VDSAT$   
**SUBTHRESHOLD**, if  $VGS < VT$
- for BJT devices:  
**SATURATION**, if  $VBE > 0$  and  $VBC > 0$   
**ON**, if  $VBE > 0$  and  $VBC < 0$   
**OFF**, if  $VBE < 0$  and  $VBC < 0$   
**INVERSE**, if  $VBE < 0$  and  $VBC > 0$

**POW****POW(Xinstance\_name)**

Returns the power curve dissipated in the `Xinstance_name`.

**POWER**

Returns the power dissipated in the entire design. This number is the sum for the power dissipated in R, E, H, I, M, Q, D, J elements exclusively.

In all cases, the power stored in capacitances is ignored, only dissipated power is taken into account. Power is measured only in DC and TRANSIENT analyses, e.g.

```
.extract min(power)
```

## PVAL

**PVAL**(parameter\_name)

Returns the value of the requested parameter.

## RMS

**RMS**(WAVE [, MIN, MAX])

Returns the root mean square value of the waveform WAVE in the x-axis range MIN to MAX. This is calculated as follows:

$$\text{RMS} = \sqrt{\left(\int_{\min}^{\max} \text{wave}^2\right) dx / (\max - \min)}$$

For noise analysis, RMS calculations can not be performed in the same way as for other analyses, because of the specific nature of noise signals. RMS for noise analysis can be achieved with the following syntax:

```
.extract RMS(inoise)
.extract RMS(onoise)
.extract RMS(noise(Q1))
```

**RMS(inoise)** is calculated as:

$$\text{RMS}(inoise) = \sqrt{\text{INTEG}(INOISE \times INOISE)}$$

**RMS(onoise)** is calculated as:

$$\text{RMS}(onoise) = \sqrt{\text{INTEG}(ONOISE \times ONOISE)}$$

**RMS(noise(object))** is calculated as:

$$\text{RMS}(noise(object)) = \sqrt{\text{INTEG}(NOISE(OBJECT))}$$

## SLEWRATE

**SLEWRATE**(WAVE [, VTH=VAL | BEFORE=VAL | AFTER=VAL OCCUR=VAL])

See the descriptions of these parameters on [page 10-99](#).

Returns the slope of the waveform WAVE at v(wave) = vth. Default is vth =  $\frac{(VDD + VSS)}{2}$

## SLOPE

**SLOPE**(WAVE, VTH [, MIN, MAX], n)

Returns the slope of the waveform WAVE at the nth occurrence of it crossing the y-axis value VTH in the x-axis range MIN to MAX.

## TCROSS

**TCROSS**(WAVE, VTH=VAL [, OCCUR=VAL] [, AFTER=VAL] [, BEFORE=VAL])

Equivalent to the **XTHRES** function. **TCROSS** is part of the ‘Transient-extraction language’, while **XTHRES** is part of the ‘general-purpose extraction language’. The former

mode is faster to execute, and consumes less memory, while the latter is more expensive in terms of memory and CPU, but is much more general in the sense that arguments can be expressions.

## TINTEG

```
TINTEG(WAVE, [AFTER=VAL] [, BEFORE=VAL])
```

Equivalent to the **INTEG** function. **TINTEG** is part of the ‘Transient-extraction language’, while **INTEG** is part of the ‘general-purpose extraction language’. The former mode is faster to execute, and consumes less memory, while the latter is more expensive in terms of memory and CPU, but is much more general in the sense that arguments can be expressions.

## TPD

```
TPD(WAVE1, WAVE2[, VTH=VAL|VTHIN=VAL|VTHOUT=VAL  
+ BEFORE=VAL|AFTER=VAL|OCCUR=VAL])
```

See the descriptions of these parameters on [page 10-99](#).

Returns the propagation delay between WAVE1 and WAVE2. The thresholds **VTHIN** and **VTHOUT** are defined for WAVE1 and WAVE2 respectively.

**VTHIN** is the threshold voltage to be used to detect transition on the first node given in the **TPD** command (e.g. **TPD (V(IN), V(OUT))**), while **VTHOUT** is the threshold voltage for the second node (**V(OUT)** in this example). If **VTHIN** equals **VTHOUT**, it is possible to specify only **VTH**.

Default is  $v_{th} = \frac{(VDD + VSS)}{2}$ . Default value of **OCCUR** is 1.

If there is more than one transition, the average of the first and second is used. To measure specific signal transitions (e.g. rising/falling on the input/output), the following functions may be used:

**OCCUR=1** would force Eldo to return the **TPD** value for the first occurrence of thresholds crossing, **OCCUR=2** for the second occurrence and so on.

Alternatively, one can use **AFTER=val** and **BEFORE=val** in order to tell Eldo to return **TPD** values only if both threshold are crossed in the specified time interval.

## TPD Examples

Assume **V(IN)** crosses **VTHIN** at 10n 50n and 100n, assume **V(OUT)** crosses **VTHOUT** at 5n 75n and 110n:

```
TPD(V(IN), V(OUT), OCCUR=1)
```

would return 65n (75n - 10n), while:

```
TPD(V(IN), V(OUT), OCCUR=2)
```

would return 10n (110n - 100n) (second occurrence).

However, **TPD(V(IN), V(OUT))** would return 37.5n (average between the first **TPD** in one direction and the first **TPD** in the other direction).

## TPDUU

```
TPDUU(WAVE1, WAVE2 [ ,VTH=VAL|VTHIN=VAL|VTHOUT=VAL
+ BEFORE=VAL|AFTER=VAL|OCCUR=VAL ] )
```

WAVE1 and WAVE2 are both rising.

## TPDUD

```
TPDUD(WAVE1, WAVE2 [ ,VTH=VAL|VTHIN=VAL|VTHOUT=VAL
+ BEFORE=VAL|AFTER=VAL|OCCUR=VAL ] )
```

WAVE1 is rising, WAVE2 is falling.

## TPDDU

```
TPDDU(WAVE1, WAVE2 [ ,VTH=VAL|VTHIN=VAL|VTHOUT=VAL
+ BEFORE=VAL|AFTER=VAL|OCCUR=VAL ] )
```

WAVE1 is falling and WAVE2 is rising.

## TPDDD

```
TPDDD(WAVE1, WAVE2 [ ,VTH=VAL|VTHIN=VAL|VTHOUT=VAL
+ BEFORE=VAL|AFTER=VAL|OCCUR=VAL ] )
```

WAVE1 is falling and WAVE2 is falling.

## TRISE

```
TRISE(WAVE [ , VH=VAL|VL=VAL|BEFORE=VAL|AFTER=VAL|OCCUR=VAL ] )
```

Returns the rise time of the next rising edge on WAVE.

## TFALL

```
TFALL(WAVE [ , VH=VAL|VL=VAL|BEFORE=VAL|AFTER=VAL|OCCUR=VAL ] )
```

Returns the fall time of the next falling edge on WAVE.

See the descriptions of these parameters on [page 10-99](#). **VH** and **VL** are the high and low voltages defining the start and end points of the rise/fall. Defaults are:

$$vl = VSS + 0.1 \times (VDD - VSS)$$

$$vh = VSS + 0.9 \times (VDD - VSS)$$

## VALAT

```
VALAT(WAVE, AT=VAL)
```

Returns the value of the waveform WAVE extracted at time VAL, e.g.

```
.EXTRACT VALAT(v(s),AT=10n)
```

Returns the value of `v(s)` extracted at time 10ns.

## WFREQ

**WFREQ(WAVE [, START, END])**

Returns the average frequency of the waveform `WAVE` between the times `START` and `END`, e.g.

```
.EXTRACT WFREQ(V(OUTPUT), 10n, 100n)
```

Returns the average frequency of the voltage on node `OUTPUT` between 10n and 100n.

## WINTEG

**WINTEG(WAVE [, T])**

Returns the time integral of the waveform `WAVE` from 0 to `T`. This is calculated as follows:

$$\text{WINTEG} = \int_0^t \text{wave}(u)du$$

When `WINTEG` is used in `.DEFWAVE` cards, the corresponding waves are dumped in `.meas` rather than in the binary output file.

## XCOMPRESS

**XCOMPRESS(WAVE, VALUE [, SLOPE])**

Extracts the X-axis value of the wave at the point where the difference between the actual value of `wave` and the linear extrapolation of `wave` based on the computed slope value becomes greater than `value`.

`SLOPE` corresponds to the minimum slope which must be between the first two points of the wave. If the actual slope on the signal is not high enough, a message will be displayed. Default for this 3rd argument is 0.99 if the `XCOMPRESS` is performed on a `SWEEP` extract. Otherwise, it is undefined.

Using the `COMPRESS/XCOMPRESS` function is especially useful in the case of parametrized simulations (`.STEP`). It works in the following way:

1. compute the slope of `wave` on the first two points
2. when the difference between the actual value of `wave` and the linear extrapolation of `wave` based on the slope value computed at step 1 becomes greater than `value`, then Eldo will return the value of the wave at that point (`COMPRESS` function), or the X-axis value at which the difference occurs (`XCOMPRESS` function).



See also `COMPRESS` and for further information see “[.EXTRACT—Compression Point Values](#)” on page 10-112.

## XDOWN

**XDOWN(WAVE, VTH [, MIN, MAX], n)**

Returns the x-axis value of the waveform WAVE at the nth occurrence of it falling below a y-axis value VTH in the x-axis range MIN to MAX.

## XMAX

**XMAX(WAVE [, MIN, MAX])**

Returns the value of the x-axis when MAX value is reached, e.g.

**.EXTRACT TRAN XMAX (v(out))**

Returns the time at which v(out) reaches its maximum value.

## XMIN

**XMIN(WAVE [, MIN, MAX])**

Returns the value of the x-axis when MIN value is reached.

## XTHRES

**XTHRES(WAVE, VTH [, MIN, MAX], n)**

Returns the x-axis value of the waveform WAVE at the nth occurrence of it crossing a y-axis value VTH in the x-axis range MIN to MAX.

## XUP

**XUP(WAVE, VTH [, MIN, MAX], n)**

Returns the x-axis value of the waveform WAVE at the nth occurrence of it rising above a y-axis value VTH in the x-axis range MIN to MAX, e.g.

**.EXTRACT XUP(V(S),2.5,100n,300n,5)**

Returns the x-axis value when V(S) rises above 2.5 on the y-axis for the 5th time between 100n and 300n.

---

### Note

For SLOPE, XDOWN, XTHRES, and XUP functions: occurrence is always assumed to be 1 if the search occurs in the reverse direction, e.g.

**XUP(V(S),2.5,end,start)**

**XUP(V(S),2.5,end,start,5)**

Both will return the last x-axis value when V(S) crosses above 2.5 on the y-axis.

---

## XYCOND

**XYCOND(WAVE, CONDITION [, MIN, MAX])**

Returns the value of the waveform WAVE when the expression represented by CONDITION is true for the first time, evaluated between the limits MIN to MAX.

Parameter `WAVE` may be a valid waveform name or the keyword `XAXIS` which extracts an x-axis value when a condition becomes true. The parameter `CONDITION` may contain any of the available arithmetic expressions.

**Table 10-7. XYCOND Arithmetic Expressions**

<code>==</code>	Equal
<code>!=</code>	Not equal
<code>  </code>	Or
<code>&amp;&amp;</code>	And
<code>&lt;</code>	Less than
<code>&gt;</code>	Greater than

**Note**

`xycond(wave, wave2=vth)` is also valid.



See “[Arithmetic Functions](#)” on page 3-7.

**YVAL**

`YVAL(WAVE, X_VALUE)`

Returns the y-axis value of waveform `WAVE` when an x-axis value `X_VALUE` is reached. Complex results information instead of real values are enabled by default. For example, `.extract ac label=EX1 yval(v(out),10k)` would deliver the result:

\* EX1 = 10.011 , -1.8

The general form is: real\_part, imaginary\_part.

Setting option `NOEXTRACTCOMPLEX` ([page 11-48](#)) disables this complex result information.

**Examples**

This example extracts the time when the voltage on node 1 rises to a value of 2.5V between the range 0 and 100ns and lists the results to the `.chi` file:

```
.extract xup(v(1), 2.5, 0, 100n)
```

The example below extracts the maximum value of the user defined waveform `power_vdd` and lists the results to the ASCII output file:

```
.extract max(w(power_vdd))
```

The example below extracts the average value of the power waveform of voltage source V1 (equivalent to `.CONSO` command) and lists the results to the ASCII output file:

```
.extract tran average(pow(V1))
```

The example below extracts the value of the power waveform of voltage source V1 at T=0 and lists the results to the ASCII output file:

```
.extract tran pow(V1)
```

The example below extracts the first occurrence of the phase of the voltage on node s when the magnitude of the voltage on node s is less than 0dB in the range of the simulation interval and lists the results to the ASCII output file:

```
.extract xycond(vp(s), vdb(s)<0.0, start, end)
```

The next, more complex, example extracts the delay between the voltages on nodes 1 and 2 of the circuit and lists the results in the .chi file:

```
.extract xup(v(1), 2.5, 0, 100n) -  
+ xdown(v(2), 2.5, 0, 100n)
```

The example below defines a macro called phmag with arguments a and b. This macro is then instantiated via the **.extract** command with the arguments a and b being replaced by the phase voltage **vp(s)** and magnitude **vdb(s)** on node s respectively. The results are listed in the ASCII output file. The phmag macro uses the pre-defined functions **xycond** and **yval**:

```
.defmac phmag(a, b) = xycond(a, b<0.0) - yval(a, 0.001)  
...  
.extract $phmag(vp(s), vdb(s))
```

It is also possible to extract values from a FFT waveform. Eldo will print out the value (in dB) at 5 Meg for the FFT done on **v(A)**:

```
.FOUR LABEL = t1 V(a)  
.EXTRACT FOUR YVAL(FOURDB(t1),5MEG)
```

In the example below, 10 simulations will be made with P1 varying from 1 to 10. There will be an extracted value for each of the 10 values of P1. The **.ext** file contains the information **EXTRACTED\_VALUES = f(stepped\_value)**; this file can be read by EZwave or Xelga:

```
.STEP P1 1 10 1  
.EXTRACT MIN(V(OUT))
```

In this next example, eleven analyses are performed for P1 varying from 1p to 10p. After each analysis, **MAX(VDB(S))** is extracted; at that step, the second **.EXTRACT** command is ignored. Once the 11 simulations are complete, Eldo will interpret the **.EXTRACT SWEEP...** commands, and will then extract toto, i.e. **MAX(VDB(S))** when P1 was 5p:

```
.PARAM p1 = ..  
C1 S 0 p1  
.STEP PARAM P1 1p 10p 1p  
.EXTRACT LABEL=toto MAX(VDB(S))  
.EXTRACT SWEEP YVAL(MEAS(toto),5p)
```

The next example shows how to use the results of EXTRACT in another **.EXTRACT** command that applies to a single analysis, with the keyword **TRAN** specified. When there are no

ambiguities, i.e. only one analysis being performed, without **.STEP** or **.TEMP** commands, then the keyword is optional:

```
.TRAN 1n 10n
.TEMP 10 20 30
.EXTRACT LABEL = T1 MAX(v(s))
.EXTRACT LABEL = T2 MIN(v(a))
.EXTRACT TRAN EXTRACT(T1)-EXTRACT(T2) ! keyword TRAN expected
```

The following is an example of the results of an **.EXTRACT** command dumped into a specified file *file.aex*:

```
EXTRACT for AC ANALYSIS
  TEMPERATURE = 2.700000e+01 Celcius
    *YVAL(VDB(NET4),100K) = -1.01E+02
EXTRACT for TRANSIENT ANALYSIS
  TEMPERATURE = 2.700000e+01 Celcius
    *V(NET4) = 1.93E-09 Volts
    *YVAL(V(NET4),50N) = 1.40E+00
```

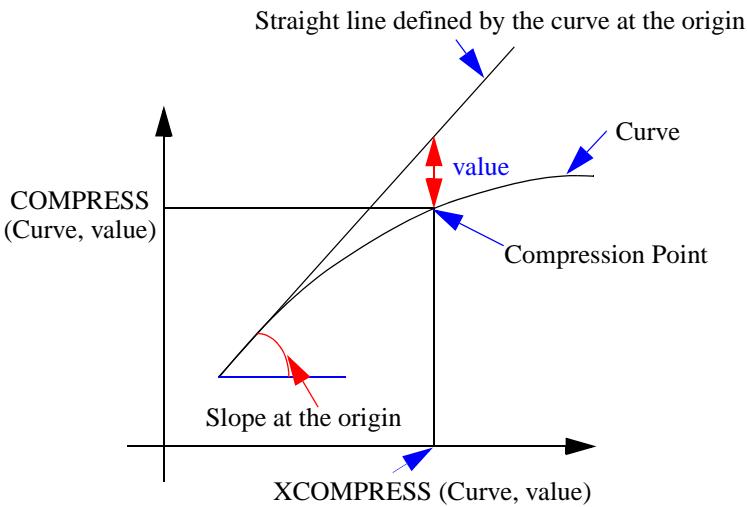
The following example shows how the **VECT** keyword can be used to return all the values which match the expression. By default, **.EXTRACT** returns the first value which matches the expression.

```
v1 1 0 pulse(0 5 0 1n 1n 8n 20n)
r1 1 0 1
.extract vect xup(v(1),2.5)
.meas vect tran memes trig at 0 targ v(1) val=2.5 rise=1
.tran 1n 100n
.end
```

## **.EXTRACT—Compression Point Values**

The **COMPRESS/XCOMPRESS** functions extract the co-ordinates of a compression point in the curve specified in the first parameter of the function. **XCOMPRESS** provides the X-value and **COMPRESS** the Y-value.

This compression point is the point where the curve is **value** below the straight line defined by the slope of the curve at the origin. (**value** is the second parameter of the function.)



In the illustration above, the curve is `meas(POUTdBm)`, and the compression (`value`) is 1.0 dBm.

### **Caution**



When these functions are used to extract compression points, the swept parameter (X-axis) and the curve (Y-axis) must be in dB (or dBm).

### **Example**

```
* 1dB compression point
.param fund=900Meg
.extract fsst label=YVAL(PdBm(RL), fund)
.extract sweep xcompress(meas(POUTdBm), 1.0)
.extract sweep compress(meas(POUTdBm), 1.0)
```

### **.EXTRACT—Used with Monte Carlo Analysis**

At the end of a Monte Carlo simulation, Eldo prints a histogram for each `.EXTRACT` command in the `.chi` file. The example below shows the `.chi` file entry for a Monte Carlo analysis with the following command:

```
.EXTRACT YVAL(V(2), 10n)
Distribution of YVAL(V(2), 10N)
      Range [6.4544 6.9066]
      Average value: 6.6665
      Standard deviation: 7.659184e+01
      6.4400 NB = 1 FREQ = 9.091e+00% | *****
      6.4880 NB = 0 FREQ = 0.000e+00% |
      6.5360 NB = 2 FREQ = 1.818e+01% | *****
      6.5840 NB = 1 FREQ = 9.091e+00% | ***
      6.6320 NB = 4 FREQ = 3.636e+01% | *****
      6.6800 NB = 0 FREQ = 0.000e+00% |
      6.7280 NB = 1 FREQ = 9.091e+00% | *****
      6.7760 NB = 1 FREQ = 9.091e+00% | *****
      6.8240 NB = 0 FREQ = 0.000e+00% |
      6.8720 NB = 1 FREQ = 9.091e+00% | *****
```

```
6.9200 NB = 0 FREQ = 0.000e+00% |
```

To produce a histogram such as the one shown above, a Monte Carlo Analysis ([.MC](#)) with at least 9 runs must be performed.

---

**Note**



The histogram is also output in the binary output file and can be displayed with EZwave or Xelga.

---



For more information about Monte Carlo Analysis, see [“.MC”](#) on page 10-147.

---

## .FFILE

### S, Y, Z Parameter Output File Specification

.FFILE S|Z|Y|G|H|T|A [SINGLELINE] FILENAME [HZ|KHZ|MHZ|GHZ] [RI|MA|DB]

#### Parameters

- **S**  
Specifies S (Scattering) frequency parameters tabulation.
- **Y**  
Specifies Y (Admittance) frequency parameters tabulation.
- **Z**  
Specifies Z (Impedance) frequency parameters tabulation.
- **G**  
Specifies G (Hybrid-G) matrix parameters tabulation.
- **H**  
Specifies H (Hybrid-H) matrix parameters tabulation.
- **T**  
Specifies T (transfer scattering) matrix parameters tabulation.
- **A**  
Specifies A (chain or ABCD) matrix parameters tabulation.
- **FILENAME**  
Name of the file where the S, Y or Z parameters will be stored.
- **SINGLELINE**  
This enables the user to obtain the S-parameter file in single line format as shown below.

Freq S11 S21 S12 S22

- **HZ**  
Specifies the units to be Hz. This is the default.
- **KHZ**  
Specifies the units to be kHz.
- **MHZ**  
Specifies the units to be MHz.
- **GHZ**  
Specifies the units to be GHz.

- **RI**  
Specifies Real Imaginary storage format. This is the default.
- **MA**  
Specifies Magnitude Angle storage format.
- **DB**  
**MA** with magnitude in dB.

Two-port noise parameters NFMIN\_MAG, GAMMA\_OPT\_MAG, PHI\_OPT and RNEQ are automatically written to the specified output file when a **.NOISE** command is specified in the netlist and the circuit to be analyzed is a two-port circuit.

## Examples

```
r1 1 2 100k
c1 2 0 10pf
v1 1 0 iport=1 rport=100
v2 2 0 iport=2 rport=20
.ac dec 10 1 100meg
.plot ac sdb(2,1)
.ffile S sb1.par khz ri
```

In this example, the S parameters of an RC circuit are extracted between 1 Hz and 100 MHz with 10 points per decade. The reference impedance is 100 for port1 and 20 for port2. The magnitude of S21 is plotted in dB, and the extracted S parameters are stored in the file *sb1.par* with the frequency in kHz. The data is stored in the form of the Real and Imaginary parts.

```
v1 1 0 iport=1 rport=50

R1 1 n1 1k
C1 n1 0 100p
R2 n1 2 1k
.Rc1 n1 0 100k

v2 2 0 iport=2 rport=50

.ac lin 21 1meg 21meg
.noise v(n1) V1 3

.plot noise rneq gopt bopt nfmin_mag
.ffile Z Z.par kHz ma
```

In this example the Z parameters are being extracted between 1 MHz and 21 MHz with 21 analysis points. The extracted Z parameters are stored in the file *Z.par* with the frequency in kHz. The data is stored in the form of the Magnitude Angle. As the circuit is a two-port circuit and there is a **.NOISE** command specified in the netlist then the two-port noise parameters are also stored in the output file. The output file is shown below:

```
! Data from ffile_test
# KHZ Z MA  R 5.000000E+01
!
```

```
1.000000000000001E+03 1.8928908821086993E+03 -5.7202544106307627E+01
1.5913478969070338E+03 -8.9088186330215706E+01

...
7.5788046363417195E+01 -8.9956576643609125E+01 7.5788046363417209E+01
-8.9956576643609125E+01 1.0029250742358629E+03 -4.3338006361865542E+00

! Noise Data: Nfmin(dB) GammaOpt PhiOpt Rneq/R0
1.000000000000001E+03 5.2661113010485536E+00 9.6890047604419338E-01
1.7851510536508837E+02 4.8497683522933400E+01

...
2.100000000000000E+04 2.8441528902446137E+01 9.0554147314407418E-01
1.7956971588917614E+02 3.5225984336136335E+03
```

## **.FORCE**

### Initial Transient Analysis Conditions

```
.FORCE [NODE] {node_name value}
+ [IND | OBJ] {object_name value}
```

Forces one or more nodes to specified voltage(s) with respect to ground for the initial transient solution. This is similar to the **.IC** command, values will be used only for the DC performed prior to TRAN analysis. Both commands are used to give values replacing the DC solution. With the **.FORCE** syntax, initial values of inductors current can be given while with **.IC** only node voltages can be initialized including nodes inside subcircuits.

If the **UIC** parameter is also present (in the **.TRAN** command) no DC analysis is performed and the voltages are initialized as defined in the **.FORCE** command. All other voltages on nodes not initialized in the **.FORCE** command are determined by Eldo itself. During subsequent analysis (transient), the node voltages are freed of their initial values, and may therefore assume different values.

### Parameters

- **node\_name**  
Node name.
- **value**  
Value at node.
- **IND | OBJ**  
Inductor or object. For objects, only voltage source components can be specified to set the current on.
- **object\_name**  
Inductor or voltage source component.

### Example

```
V1 1 0 pw1(0 0 10n 5v)
R1 1 2 1k
C1 3 0 100p
L1 2 3 1u
.force ind L1 10mA
.tran 1n 0.1u uic
.plot tran i(L1)
.plot tran v(3)
.end
```

# .FOUR

## FFT Select Waveform

```
.FOUR LABEL = label_name waveform_name
```

Selects the waveform on which the user requires FFT to be done. This command also accepts signals, e.g. **.FOUR sg(ycore->A(0))**. See also the **.OPTFOUR** command on [page 10-192](#).

### Parameters

- **label\_name**

It is via this label name that Eldo will select which wave has to be plotted (**.PLOT**) or on which FFT wave extraction of useful quantities has to be done (**.EXTRACT**).  
**label\_name** is required.

- **waveform\_name**

Any regular Eldo waveform name: there can be any number of **.FOUR** commands specified in the *.cir* file.

### Examples

```
.FOUR LABEL = t1 V(a)
.FOUR LABEL = t2 W('V(a) - v(b)/2.0')
```

## Display and .EXTRACT of FFT results

```
[.PLOT | .PRINT | .PROBE] FOUR FOURxx(label_name)
.EXTRACT FOUR <expression>
```

**xx** stands for **DB**, **R**, **I**, **P**, **M**.

### Examples

```
.FOUR LABEL = t1 V(a)
.PLOT FOUR FOURDB(t1)
```

It is also possible to extract values from a FFT waveform:

```
.FOUR LABEL = t1 V(a)
.EXTRACT FOUR YVAL(FOURDB(t1),5MEG)
```

Eldo will print out the value (in dB) at 5 Meg for the FFT done on **v(A)**.

## **.FUNC**

### User Defined Function

```
.FUNC P(a,b,...) EXPR
```

This command is used to define functions used in expressions. They are flexible and useful when there are several similar sub-expressions in a circuit file.

This command provides compatibility with PSpice. It is equivalent to the .PARAM user-defined function specification.

### Parameters

- `P(a,b)` EXPR

Specifies a user-defined function in order to define a parameter using an expression. The parameter may then be called when required.

### Example

```
.func P(a,b) expression  
R1 1 2 'P(a,b)'
```

# **.GLOBAL**

## Global Node Allocation

**.GLOBAL** NN {NN}

Declare global node(s), making them known throughout a circuit without having to declare them in each subcircuit. The number of such nodes is unlimited. **.GLOBAL** is active when placed anywhere in the netlist. This applies to any launch mode.

By default, nodes which appear in subcircuit definitions have priority over the nodes defined with **.GLOBAL** statements. It is exactly the opposite when option **DSCGLOB=GLOBAL** is specified (must be at the beginning of the netlist): nodes defined with **.GLOBAL** statements take priority over nodes which appear in subcircuit definitions.

## Parameters

- NN

Name(s) of the node(s) to be declared as global.

## Example

```
.global vdd vss
```

Specifies vdd and vss as global nodes throughout the circuit.

```
.global vdd mid

.subckt div1 mid
r1 vdd mid 1k
r2 mid 0 1k
.ends

x1 net1 div1
r1 net1 0 1k
```

By default, Eldo will connect X1.r1 and X1.r2 to node net1. If option **DSCGLOB=GLOBAL** is set X1.r1 and X1.r2 will be connected to global node mid.

## Related Option

**DSCGLOB**, [page 11-25](#).

## **.GUESS**

### Initial DC Analysis Conditions

**.GUESS V(NN)=VAL [SUBCKT=subckt\_name] {V(NN)=VAL [SUBCKT=subckt\_name]}**

Helps to calculate the DC operating point by setting voltage values at selected nodes for the first iteration of a DC operating point calculation.

This command differs from the **.NODESET** command in so far as when using **.GUESS** the node voltages are only fixed for the first iteration of a DC operating point calculation, whereas when using **.NODESET** the node voltages are fixed for the duration of the first DC operating point calculation.

This command is useful when the approximate whereabouts of the DC operating point is known, enabling the simulator to converge more quickly.

#### **Note**



By default, the first **.GUESS** specification has precedence over subsequent **.GUESS** specifications. Setting **.OPTION LICN**, the last **.GUESS** specification will have precedence.



See [page 11-28](#) of the *Eldo User's Manual* for further information.

### Parameters

- **V(NN)=VAL**  
Voltage at node **NN** in volts.
- **SUBCKT=subckt\_name**  
If specified it will fix the voltage of the preceding node in all instances of the subcircuit **subckt\_name**.

### Example

```
.guess V(n4)=6v V(n5)=2v V(n6)=-5v
```

Specifies that for the first iteration of a DC operating point calculation the voltages at the nodes n4, n5 and n6 be set to 6V, 2V and -5V respectively.

```
.guess V(2)=3v SUBCKT=sub1 V(4)=-2v SUBCKT=sub2
```

Specifies that for the first iteration of a DC operating point calculation the voltages at node 2 of subcircuit sub1 and node 4 of subcircuit sub2 will be set to 3 and -2V respectively.

## .HIER

### Changing the Hierarchy Separator

**.HIER** . | / |<char>

By default the hierarchical separator in Eldo is the '.' character. The Verimix interface imposes the use of the '/' character for the hierarchical separator. Therefore, for Verimix integration, '/' is allowed as the hierarchical separator.



Please refer to “[.A2D](#)” on page 10-4 and “[.D2A](#)” on page 10-44.

The rule defined in “[Hierarchical Management](#)” on page 10-124 applies, however, the hierarchical separator differs:

There is no impact on the Eldo naming convention. In the Eldo description part, all the hierarchical names use the default hierarchical separator (by default '.'). To change the hierarchical separator in Eldo, use the **.HIER** command, for example the line:

**.HIER** /

changes the default hierarchical separator '.' to '/'.

- **char**

It is possible to specify any character as the hierarchical separator. However, care must be taken as there may be possible side effects. One restriction is that this character should not appear elsewhere in instance or net names.

Example for the **.A2D** command with Verimix:

in the eldo description part	in the verilog description part
XIC1 OUT1 OUT5 My_Subckt	
...	
.SUBCKT My_Subckt IN OUT	
...	
.A2D digital_name eldo_name	\$vmx_define_export(test.top.ic1.out2,
+MOD=model_a2d	"XIC1/DIGITAL_NAME");
...	
.ENDS My_Subckt	

The hierarchical name of the eldo node `eldo_name` is `XIC1.DIGITAL_NAME`. In this case the name sent from Eldo to the Verimix interface is `XIC1/DIGITAL_NAME`.

Example for the **.D2A** command in Verimix:

in the eldo description part	in the verilog description part

```

XIC1 OUT1 OUT5 My_Subckt
...
.SUBCKT My_Subckt IN OUT
...
.D2A digital_name
+VAnalogSource
+MOD=model_d2a
...
.ENDS My_Subckt

```

```

...
$vmx_define_import(test.top.ic1.out4,
                    "XIC1/DIGITAL_NAME");
...

```

In this case the name sent from Eldo to the Verimix interface is XIC1/DIGITAL\_NAME.

---

**Note**


---

If the subcircuit My\_Subckt is instantiated in another subcircuit, the hierarchical name will look like <...>/XIC1/DIGITAL\_NAME.

---

## Hierarchical Management

The hierarchical management for the digital\_name which is implicitly generated from the eldo\_name by Eldo is allowed. To allow hierarchical netlist generation, the implicit digital\_name generated by Eldo for the synchronizer takes into account the hierarchical context, e.g.

```

XIC1 OUT1 OUT5 My_Subckt
...
.SUBCKT My_Subckt IN OUT
...
.A2D eldo_name
+MOD=model_a2d
...
.ENDS My_Subckt

```

The hierarchical name of the Eldo node eldo\_name is XIC1.ELDO\_NAME. However, the implicit digital\_name sent to the synchronizer is XIC1.ELDO\_NAME and *not* eldo\_name.

## **.IC**

### Initial Transient Analysis Conditions

```
.IC V(NN)=VAL [SUBCKT=subckt_name] {V(NN)=VAL [SUBCKT=subckt_name]}
```

Used to fix node voltages for the duration of a DC analysis. If the **UIC** parameter is also present (in the **.TRAN** command) no DC analysis is performed and the voltages are initialized as defined in the **.IC** command. All other voltages on nodes not initialized in the **.IC** command are determined by Eldo itself. During subsequent analysis (transient), the node voltages are freed of their initial values, and may therefore assume different values.

The **.IC** command on devices is interpreted only on C and L devices. For all other devices, the **.IC** command is ignored.

#### **Note**



By default, the first IC specification has precedence over subsequent IC specifications. Setting **.OPTION LICN**, the last IC specification will have precedence.

### Parameters

- **V(NN)**  
Voltage at the node **NN** in volts.
- **SUBCKT=subckt\_name**  
If specified it will fix the voltage of the preceding node in all instances of the subcircuit **subckt\_name**.

### Examples

```
.dc  
.ic v(2)=3v v(4)=-2v
```

Specifies that for the duration of the DC analysis, the voltages at the nodes 2 and 4 be fixed to 3 and -2V respectively. Other circuit node voltage values are computed by Eldo.

```
.tran 1ns 100ns uic  
.ic v(n3)=7v v(n4)=2v v(n5)=-3v v(x1.222)=0v
```

Specifies that no DC analysis should be performed and that at the beginning of the transient analysis the voltages at the nodes n3, n4 and n5 be set to 7V, 2V and -3V respectively, while node 222 of subcircuit x1 is set to 0V. Other node voltages are calculated by Eldo.

```
.dc  
.ic v(2)=3v SUBCKT=sub1 v(4)=-2v SUBCKT=sub2
```

Specifies that for the duration of the DC analysis, the voltages at node 2 of subcircuit sub1 and node 4 of subcircuit sub2 will be fixed to 3 and -2V respectively. Other circuit node voltage values are computed by Eldo.

## Related options

[ICDC](#) and [ICDEV](#), page 11-28; [LICN](#) page 11-28.

## **.IGNORE\_DSPF\_ON\_NODE**

### Ignore DSPF on Specified Node

```
.IGNORE_DSPF_ON_NODE {NODE}
```

Used to force Eldo to ignore parasitic elements on a specified node when a DSPF file has been included using the **.DSPF\_INCLUDE** command.

#### Parameters

- NODE

Specifies node(s) for which the parasitics given in the DSPF file will be ignored.

#### Example

```
.DSPF_INCLUDE testspf.spf
.IGNORE_DSPF_ON_NODE n1 n2
```

For nodes n1 and n2 parasitics as specified in DSPF file *testspf.spf* will be ignored.

## **.INCLUDE**

### Include a File in an Input Netlist

**.INC [LUDE] FNAME**

Inserts the contents of a file into a circuit description file. Including a file is the same as typing the included file's text directly into the circuit description file. Included files may not contain title lines (comments may be used). A **.END** statement in an included file marks only the end of the included file.

Eldo searches for the specified file in a specific order. First it checks the absolute path, if it is specified. Then it checks the directory the netlist is in. This is followed by the SEARCH\_PATH if no absolute path was specified, and then in the file parent directory. This works in the same way as the **.LIB** command.



For more information see “[Searchpath priorities](#)” on page 10-135.

Nesting of **.INCLUDE** commands is allowed. **.INCLUDE** files are included only once even if the **.INCLUDE** statement appears several times. However, when the **.INCLUDE** is located inside a SUBCKT, the inclusion will occur.

The **.INCLUDE** command can also be used to include DSPF files that contain the complete netlist (active elements and parasitics). Subcircuits can be instantiated using either the [\*\*.TOPCELL\*\*](#) command or an **X** instance.

Use option **CONTINUE\_INCLUDE** to specify that continuation lines with + as the first character apply to the **.INCLUDE** command in the file.

To replace one file by another one when using the Eldo re-run facility, use the option **ALTINC**. This forces Eldo to replace the first **.INCLUDE** statement found in an input netlist by the first **.INCLUDE** statement found in the **.ALTER** section of the netlist.

### Related Options

[ALTINC, page 11-7](#), [CONTINUE\\_INCLUDE, page 11-8](#).

### Parameters

- **FNAME**

Name of the file to be included. This name may be any character string which is a legal path name. The filename can be written in either upper or lower case letters. It is possible to have a mixture of both.

### Examples

```
.INCLUDE circuit_add_on.cir
```

Specifies that the contents of the *circuit\_add\_on.cir* file be included in the circuit description file. In this case, the included file is resident in the same directory as the circuit description file.

```
.INCLUDE /users1/examples/circuit2.cir
```

Specifies that the */users1/examples/circuit2.cir* file is to be included in the circuit description file.

```
TITLE
.INCLUDE foo
.INCLUDE foo !this include will be ignored.
...
.END
```

In the example above, the second **.INCLUDE** statement is ignored.

```
TITLE
.SUBCKT S1..
    .INCLUDE foo
.ends
.SUBCKT S2
    .INCLUDE foo
.ends
.end
```

In the example above, the file named *foo* will be included twice. The **.INCLUDE** statements have been located inside subcircuits.

The following example shows how to specify that continuation lines with + as the first character apply to the **.INCLUDE** command in the file. If the main netlist has the two lines:

```
.include file.inc
+ b=2
```

With the **CONTINUE\_INCLUDE** option specified, if file *file.inc* contains as its last line:

```
.param a=1
```

Eldo would interpret this as:

```
.param a=1
+ b=2
```

The following example shows how to replace one included file by another one when using the Eldo re-run facility, with the option **ALTINC**. The *new\_models* file will be included in the re-run simulation instead of *models*.

```
.include models
.option altinc
r1 1 0 rval
v1 1 0 dc 1
.extract dc i(r1)
.dc
.alter
.include new_models
```

## **.INIT**

### Initial Digital Circuit Conditions

```
.INIT NODE [DC=VAL] TI TS VALI {TI TS VALI}
```

This command is used when initializing digital circuits. Between the times **TI** and **TS**, Eldo forces the node **NODE** to the voltage level specified by **VALI**. If a DC value is also specified, the simulation is started with the specified value assigned to the node. Outside the specified time limits the node is computed as a normal node.

### Parameters

- **NODE**  
The name of the node to be initialized with a voltage level.
- **TI**  
The time from which the node is to be initialized to the specified voltage level.
- **TS**  
The time beyond which the voltage initialization on the node should be stopped.
- **VALI**  
The voltage level to which the node should be initialized, in volts.
- **DC=VAL**  
Voltage level to which the node should be initialized at the start of the simulation period.  
Optional.

### Example

```
.init n1 dc=2 5u 10u 5
```

Specifies that the node **n1** be initialized to 5V between the time period 5 and 10μs. The node is initialized to 2V at the start of the simulation.

# .IPROBE

## Probe Current Between Pins

```
.IPROBE NAME=vname [DIRECTION=POS|NEG] pinref1 {pinrefn}
```

Use this command to probe current between pins. This is equivalent to inserting a zero voltage source between a node and a specified pin of a device. However it does not require netlist modification, which means for users working in a UI schematic environment, the netlister does not have to be run again.

Inserting a zero voltage source can be useful to measure current entering objects, or for some extra commands such as **.LSTB**.

In addition, it is possible to access the probed current with the expression `IPROBE(vname)`.

## Parameters

- **vname**  
Name of the voltage source being added. Current through this voltage source can be read using `IPROBE(vname)`.
- **DIRECTION=POS | NEG**  
This is **POS** by default. The voltage source is added between the original nodes referred to by the `pinref` list and a new node which will be created by Eldo, the name of which will be constructed from the prefix `IPRB#` followed by `vname`. If direction is **NEG**, the voltage source is reversed.
- **pinref**  
List of pin references. Syntax is `device_name.pin_index`. The number of pin references is not limited. But all pin references must refer to the same original netlist, otherwise an error will be displayed. The pin references cannot make reference to ADMS devices, because such ADMS objects (Y instances) can be hierarchical devices, and this situation is not handled.

## Notes

**.IPROBE** is used to insert a zero voltage source between a node and a selected number of pin objects. This pin list does *not* correspond to the pin list of the new zero voltage source. Imagine you have:

```
R1 A B 1k
R2 B C 1K
```

If you want to insert a zero voltage source between R1 and R2, the syntax will be:

```
.IPROBE name = I R1.2
```

and not:

```
.IPROBE name = I R1.2 R2.1
```

The first statement works OK, the second does not.

In both cases, Eldo will first create a new node named, for example, NEW. Eldo will then create a zero voltage source named, for example, VI. Eldo will connect node “B” to the first pin of VI, the second pin of ‘I’ will be connected to NEW, and Eldo will then connect the pin list specified to the node NEW. With the first syntax above, only pin 2 of device R1 will be reconnected to NEW, while for the second syntax, both the second pin of R2 and the first pin of R2 will be reconnected to node NEW. In that second case, the original net B will therefore be connected only to pin NEW, which is not what was required.

The equivalent netlist would be in case 1:

```
R1 A B 1k
VI B NEW 0
R2 NEW C 1K
```

The equivalent netlist would be in case 2:

```
R1 A NEW 1k
VI B NEW 0
R2 NEW C 1K
```

and B is here a dangling node.

## Example

```
V1 1 0 1
r1 1 2 1
r2 3 1 1
r3 2 0 1
r4 3 0 1
r5 1 0 1

.dc
.IPROBE name = ux r1.1 r2.2
.print dc iprobe(ux)
.extract dc iprobe(ux)
```

The actual pin indicated by the pin reference list here is the node named “1”. It is the first pin of device R1 or the second pin of device R2. Eldo will insert a zero voltage source between node 1 and a new node named `IIPRB#UX`, and the first pin of R1 and the second pin of R2 will be disconnected from node 1 and reconnected to the new node `IIPRB#UX`. The `IPROBE(UX)` statement will therefore return the sum of the current inside R1 and R2.

## .LIB

### Insert Circuit Information from a Library File

```
.LIB [KEY=KNAME] FNAME [LIBTYPE]
.LIB LIBTYPE
```

This command is used to insert model or subcircuit definitions into an input netlist from a library file. Eldo includes the whole contents of the file specified. Although many subcircuits, models and parameters may be defined but not used, it is quicker than picking out the individual instances required.

If there are several definitions of the same instance defined in a block, Eldo will stop and produce an error message.

Nesting of **.LIB** commands is allowed. For more information on nested **.LIB** commands see “[Library nesting](#)” on page 10-136.

Eldo searches for the specified library file in a specific order. First it checks the directory the netlist is in, followed by the directory Eldo was started in, and then in the library file parent directory. For more information, see “[Searchpath priorities](#)” on page 10-135.

If a library file contains corners, the **LIBTYPE** parameter is required in the netlist (see second syntax definition), otherwise Eldo will exit with an error. If a library file has been referenced without a corner, Eldo does not expect the syntax **.LIB LIBTYPE** inside the library file, and interprets all the **.LIB** commands as **.LIB FILENAME**. Avoid this by referencing any library file containing corners with the **LIBTYPE** parameter.

Libraries can be accessed directly from model and subcircuit description lines. This allows the properties of the model, or subcircuit, to be called from another file or directory. The syntax is as follows:

```
.MODEL LIB FNAME MNAME [LIBTYPE]
.SUBCKT LIB FNAME SNAME [LIBTYPE]
```

where **MNAME** and **SNAME** are the model or subcircuit names respectively.

It is also possible to use the **.LIB** command in interactive mode in SimPilot. See the “[Interactive mode](#)” on page 10-136 for details.

If a parameter **P** is referred to in a netlist but not defined, Eldo searches for **P** in the **.LIB** files. Only global **.PARAM** definitions are considered. Parameter declarations within a **.SUBCKT** definition will not be considered outside that subcircuit.

### Parameters

- **FNAME**

Name of the library file to be searched. This name may be any character string which is a legal path name. **FNAME** must not end with *<circuit\_name>.lib* or

*<circuit\_name>.eldo.lib* since Eldo uses this type of filename for temporary files which are later deleted automatically by the Eldo script, i.e. in the circuit file *mycircuit.cir*, the file *mycircuit.eldo.lib* must not be used.

**Note**

A filename specified in the **.LIB** statement may be enclosed in single quotes to preserve compatibility with other simulators. When a character string is enclosed in single quotes, the case of the string remains unchanged, i.e. no conversion to upper or lower case, allowing a mixture of the two.

- **KEY**

Used in conjunction with the **.ALTER** command. For more information on the key parameter see “[Key parameter example](#)” on page 10-137.

- **KNAME**

Key name used to specify the library files and variants (used in subsequent simulations) that may be replaced with the **.ALTER** command.

- **LIBTYPE**

Name of a library variant to be used. If a library file contains corners, this parameter is required for the **.LIB** statement in the netlist. For more information on library variants see “[Library variant management](#)” on page 10-135.

## Library management mechanism

A number of different mechanisms are available to manage libraries.

The library management changed in v5.8 compared to previous versions, in the way that the entire content of the selected library is included in the database. In pre-v5.8 versions, only the missing **.SUBCKT**, **.PARAM** or **.MODEL** definitions were extracted from the library.

In v5.9 and upwards, the change for v5.8 remains, but the **.LIB** command is acted upon immediately, i.e. the content of the library is incorporated as soon as the command is parsed. In v5.8, the contents of the different libraries were incorporated only upon completion of reading the input file. In pre-v5.8 versions, libraries were re-read repeatedly to load missing definitions.

**Note**

The pre-v5.8 version of library management is no longer supported.

An option is available to use the alternate mechanism:

- option **INCLIB** causes Eldo to use the v5.8 mechanism

## v5.8 mechanism

To use Eldo’s v5.8 mechanism of library management, specify the option **INCLIB**.

With libraries containing thousands of definitions which are all likely to be used in a design, the procedure used in pre-v5.8 versions of Eldo, was far too slow. Specifying the v5.8 mechanism means Eldo will include the whole content of the **.LIB** (or the full content of what is inside **.LIB .ENDL** blocks when variants are specified on the **.LIB** card). This procedure is faster, even if some models, subcircuits or parameters are defined but not used.

However, there are four side effects to this mechanism:

1. When there are several definitions defined in the block that Eldo will include, the simulation will stop on error (previous versions accepted this but used only the first definition—it was however not safe to potentially have two definitions for the same entity).
2. Eldo includes the full content of the **.LIB** card, which means Eldo interprets any commands/options which are specified in **.LIB**. With the pre-v5.8 implementation, these commands were not acted upon because they were not looked for.
3. Parameters and models are defined in the order which corresponds to their order in the library. With the pre-v5.8 implementation, the order was the order of usage (first one used was the first one defined). Therefore, the Monte Carlo series associated to each entity will not be the same in both cases.
4. If the filename specified in the **.LIB** does not exist, Eldo exits with an error.

In all cases, the content of what is actually included by Eldo appears in the ASCII output file.

## Searchpath priorities

This is the order of directories in which library files are searched for:

1. Absolute path
2. Parent directory  
Directory of the library file contained in the **.LIB** statement
3. Current directory  
Directory from which Eldo was started
4. Search path  
Specified using the **-searchpath** flag or **.OPTION SEARCH=path** within the netlist. If both are specified, then the contents of **-searchpath** is searched first.

For more information on **-searchpath** see [page 2-2](#) or **.OPTION SEARCH** see [page 11-60](#). For an example, see the “[Searchpath priorities example](#)” on page 10-137.

## Library variant management

Library variants such as **best**, **worst**, and **typical** process variation, may be handled using the following command:

**.LIB FNAME LIBTYPE**

Within the library `FNAME`, you must then have sections defined by:

```
.LIB <libtype>
...
.ENDL
```

For an example see “[Library variant management example](#)” on page 10-138.

## Interactive mode

It is possible to use the `.LIB` mechanism in SimPilot. The netlist must contain one or several lines of type:

```
.LIB KEY=kname FNAME [LIBTYPE]
```

In interactive mode, the following commands can be issued:

```
.LIB key=toto mymod.tech nominal
.LIB key=toto mymod.tech worst
.LIB key=toto mymod2.tech
```

Eldo will search for the `kname` to identify which library must be replaced. Limitations of the `.LIB` command when issued in interactive mode are as follows:

1. Subcircuits are not replaced.  
When `.LIB` is specified in the input file, it can be used to select a subcircuit to be included in the design. Since taking a subcircuit from another library than the library specified in the input file could result in a change of topology (and changing topology of the current design is strictly impossible), Eldo would issue an error whenever the user attempts to substitute one subcircuit for another.
2. Models which were used by Eldo-XL cannot be substituted.
3. Switching between GUDM and non-GUDM models is prohibited. Similarly, switching between GUDM models is prohibited.
4. Only MOS, BJT, DIODE, JFET, R, L and C `.model` commands can be substituted. Attempting to substitute other kinds of models will lead to an error.

## Library nesting

Eldo libraries may themselves contain other libraries. For example, consider a library `lib.lib`:

```
.lib best
.lib mos.lib best
.lib bip.lib best
.endl best
.lib typ
.lib mos.lib typ
.lib bip.lib typ
.endl typ
```

If a netlist contains the command:

```
.LIB lib.lib best
```

then Eldo will browse the *mos.lib* and *bip.lib* which are surrounded by **.lib** *best* and **.endl** *best*.

## Examples

The following example specifies that any model and subcircuit information missing in the input netlist should be taken from the file *circuit1.cir*.

```
.LIB circuit1.cir
```

The following example specifies that any model and subcircuit information missing in the input netlist should be used from the file *circuit2.cir* in the directory */users1/examples*.

```
.LIB /users1/examples/circuit2.cir
```

## Searchpath priorities example

The following example is related to searchpath priorities and shows the order in which directories are searched.

If you have the following command in a netlist file:

```
.LIB ./foo/myfile.lib
```

and if in the library file *myfile.lib*, you have:

```
.LIB toto
```

then if file *toto* is not found in the netlist directory or parent directory, it will be searched for in the current directory from which Eldo was started.

In the case of a nested **.LIB/ .INCLUDE**, it would search the directory of the library file containing the **.LIB/ .INCLUDE** statement.

## Key parameter example

The example below demonstrates the use of the **KEY** parameter in conjunction with the **.ALTER** command.

```
...
.lib key=K1 /work/bip/mymod typ
.lib key=K2 /work/mos/mymod typ
...
.alter
.lib key=K2 /work/mos/mymod best
.alter
.lib key=K2 /work/private/mymod typ
.end
```

This command sequence causes three simulations to be performed always with library */work/bip/mymod typ*. In the first simulation, library */work/mos/mymod typ* is

---

used. In the second simulation, library /work/mos/mymod best is used. In the third simulation, library /work/private/mymod typ is used.

### Library variant management example

The next example relates to Library variant management.

Library *mos.lib*:

```
.lib best
.model MN nmos level=3 vt0=0.5
.endl best
.lib typ
.model MN nmos level=3 vt0=0.75
.endl typ
.lib worst
.model MN nmos level=3 vt0=1.0
.endl worst
```

Circuit Netlist:

```
lib case management
.lib mos.lib typ
m1 vdd g 0 0 MN l=1.2U w=5U
vdd vdd 0 5
vg g 0 0.8
.op
.end
```

## **.LOAD**

### Use Previously Simulated Results

**.LOAD [FILE=]filename**

This command takes a set of voltages previously saved using the **.SAVE** command, and inserts these as **.NODESET**, **.GUESS**, or **.IC** commands depending on how the LOAD file was created. It is also possible to have multiple occurrences of the **.LOAD** command in an input netlist.

This command works in a similar way to the **.USE** command. Simply, the attribute **NODESET/IC/GUESS** need not be specified in the command since that information was dumped when creating the file with the **.SAVE** command.



See also “[.USE](#)” on page 10-328.

### Parameters

- **[FILE=]filename**

Filename into which the DC values were saved via the **.SAVE file\_name DC** command.

## **.LOOP**

### Insert a Feedback Loop

```
.LOOP INPUT OUTPUT [R|C|I|V VALUE] [DISCONNECT=DEV_NAME] [KEEPINPUT]
```

Inserts a feedback loop between the input and output nodes of an op-amp during transient analysis. This is, therefore, useful in obtaining both the Open and Closed loop characteristics of a circuit in the same simulation run. The following devices can be inserted in the feedback loop:

- Resistor.
- Capacitor.
- Independent Voltage Source.
- Independent Current Source.

If no device is specified, Eldo will insert a zero-voltage source between **INPUT** and **OUTPUT**. A second operation which is performed in the Closed-loop mode is to disconnect the voltage source (**V** or **E** element) which is applied between node **INPUT** and **GND**, if present. This feature can be disabled using the keyword **KEEPINPUT** in the **.LOOP** command.

In addition to the above operation, a device which is active in Open-loop mode may be disconnected during simulation in Closed-loop mode using the **DISCONNECT** keyword.

### Parameters

- **INPUT**  
Input node of the operational amplifier which is to be connected in closed loop.
- **OUTPUT**  
Output node of the operational amplifier which is to be connected in closed loop.
- **R**  
Keyword indicating a feedback loop with a resistive load.
- **C**  
Keyword indicating a feedback loop with a capacitive load.
- **I**  
Keyword indicating a feedback loop with an independent current source.
- **V**  
Keyword indicating a feedback loop with an independent voltage source.
- **VALUE**  
Value of the feedback device.

- **DISCONNECT**

Keyword indicating that a device in the feedback loop is to be disconnected.

- **DEV\_NAME**

Instance name of the device in the feedback loop which is to be disconnected.

- **KEEPINPUT**

If specified, the voltage source (**V** or **E** element) which is applied between node **INPUT** and **GND** will not be disconnected, if present.

---

**Note**

The AC analysis conditions defined in a netlist are always simulated prior to any transient analysis conditions in that same netlist.

---

### Example

```
r1 1 2 1k
c1 2 0 10p
c2 4 0 1p
r2 3 0 100
.model ampop opa level=2 voff=0 sl=50e06
+ cin=0 rs=10 vsat=5 gain=5000 fc=5000
v1 1 0 ac 1 pwl(4n 5 10n 0 20n 0 30n 5)
opal 2 3 4 0 ampop
.loop 2 4 r 100
.ac dec 10 10e+4 10e+8
.tran 1m 100m
.plot ac vdb(4)
.plot tran v(4)
```

Specifies that during transient analysis of the circuit, a resistive load of  $100\Omega$  is inserted between the input node 2 and output node 4 of **opal**. During AC analysis, no feedback loop is inserted between these nodes.

---

**Note**

The definition of two voltage sources **v1** in the netlist. Eldo uses the AC voltage source definition for the AC circuit analysis and the transient voltage source definition for the transient circuit analysis.

---

## **.LOTGROUP**

### Share Distributions

```
.LOTGROUP group_name[ /distrib_type]=val[%]
```

This syntax allows different entities to share the same distribution.

Anywhere Eldo accepts **LOT** or **DEV** specification, one can specify **LOTGROUP=group\_name**.

### Parameters

- **group\_name**  
Name of group.
- **distrib\_type**

This can be one of the following options:

#### **UNIFORM**

Uses a uniform distribution (default).

#### **GAUSS**

Uses a Gaussian distribution.

#### <dist\_name>

Selects a user defined distribution named **dist\_name**.

If no distribution type is selected, it will default to **UNIFORM**.

### Examples

```
.LOTGROUP my_lot_group=14%
.LOTGROUP my_lot_group/uniform=14%
.LOTGROUP my_lot_group/gauss=12%
.LOTGROUP my_lot_group=(nor,-12%,10%)
.LOTGROUP my_lot_group/my_distrib_name=8%
```

This works together with:

```
.DISTRIB my_distrib_name (-1,0) (-1,0.5)
+
(-0.4,0.5) (-0.4,0)
+
(0.4 0) (0.4 0.5)
+
(1 0.5) (1,0)
```



For more information see “[.DISTRIB](#)” on page 10-77.

All entities which refer to the same “lotgroup” will share the same distribution.

```
.LOTGROUP my_lot_group=14
.MCMOD MOD1 TOX LOTGROUP=my_lot_group
.PARAM P1=1 LOTGROUP=my_lot_group
```

---

In this example, the same random number, between +14 and -14, will be used for updating **P1** and **TOX** of model **MOD1**.

## **.LSTB**

### **Loop Stability Analysis**

**.LSTB SOURCE\_NAME**

This command improves the analysis of circuit stability. The classical method for stability analysis is to break the feedback loop at an appropriate point on AC analysis, while maintaining correct DC conditions. This means that the loop must be terminated with the appropriate impedance it ‘sees’ looking at the loop input. Obtaining this impedance value is not always a simple task.

The **.LSTB** command measures the loop gain by successive injection (Middlebrook Technique). A zero voltage source is placed in series in the loop: the first pin of the voltage loop must be connected to the loop input, the other pin to the loop output. The name of this voltage source is given in the **.LSTB** card, and the loop gain can be displayed using keywords **LSTB\_DB**, **LSTB\_P**, **LSTB\_R**, **LSTB\_I**, **LSTB\_M** (see meanings in table below) in any **.PLOT/.PRINT/.PROBE/.EXTRACT** commands.

**Table 10-8. LSTB Output Formats**

Format	Meaning
<b>DB</b>	Magnitude in dB
<b>M</b>	Magnitude
<b>P</b>	Phase
<b>R</b>	Real part
<b>I</b>	Imaginary part

### **Example**

*aopstb.cir* (provided in: \$MGC\_AMS\_HOME/eldo/\$eldover/examples/ eldo/)—this circuit is derived from *aopbou.cir* and *aopalt.cir* which are also provided.

- Split node S into node SL and S, and insert the VSTB source:

```
M7 B SL C VSS mod1 W=130U L=4U
VSTB SL S
```

- Invoke LSTB analysis:

```
.LSTB VSTB
```

- Output the LSTB analysis:

```
.plot ac lstb_db
.plot ac lstb_p
```

# .MALIAS

## Model Name Mapping

```
.MALIAS alias_model_name actual_model_name
```

This command can be used to map model names in a netlist to model names specified in **.MODEL** cards. This works if the **.DEFMOD** is placed before any use of the string `alias_model_name`.

### Parameters

- `alias_model_name`  
Model name given in a netlist.
- `actual_model_name`  
Model name defined in **.MODEL** card.

### Example

```
.model modell r tc1=2 tc2=1
.defmod modalias modell
r1 1 0 modalias r=1k
```

## **.MAP\_DSPF\_NODE\_NAME**

### Map Eldo Node to DSPF Node

```
.MAP_DSPF_NODE_NAME LOGICAL=ELDONAME DSPF=NEWNAME
```

This command can be used if the DSPF file specifies for a node to be replaced causing references to nodes in commands such as **.PLOT**, **.PROBE** and **.EXTRACT** to become unusable. Eldo will map all of the node references to the new DSPF node name. For information on loading DSPF files see “[.DSPF\\_INCLUDE](#)” on page 10-79.

### Parameters

- **LOGICAL=ELDONAME**  
Name of node to be mapped.
- **DSPF=NEWNAME**  
DSPF node name to replace all references to **ELDONAME**.

## **.MC**

### Monte Carlo Analysis

```
.MC RUNNO [OUTER] [OV] [SEED=integer_value] [NONOM] [ALL]
+ [VARY=LOT|DEV] [IRUN=val] [NBBINS=val] [ORDMCS] [MCLIMIT]
+ [PRINT_EXTRACT=NOMINAL|ALL|run_number]
```



Please refer to the [Monte Carlo Analysis](#) chapter for further information.

The Monte Carlo system may be implemented for DC, AC and transient analysis and is useful to obtain statistical information derived from estimates of the random variability of all circuit components. Model parameters may be specified with nominal and tolerance values. The Monte Carlo analysis system carries out multiple simulation runs, each run using model and device values differing from the nominal one within the specified tolerance limit, the variation being a simulated random variable satisfying a specified distribution (uniform, Gaussian, or user-defined).

This kind of analysis is useful in yield prediction and synthesis. When Monte Carlo analysis has been requested, information is added to each item plotted or printed regarding minimum and maximum values.

The standard deviation is calculated for the item specified as parameter to the **.MC** command for each Monte Carlo run. At the end of the Monte Carlo analysis, a print-out is made of the worst case value. Furthermore, all the Worst Case model parameters, together with the values of the dipoles are printed.

#### Note



The **.MC**, Monte Carlo Analysis command may not be used in a netlist together with worst case **.WCASE** analysis and with transient noise **.NOISETRAN** analysis.

The results of Monte Carlo analysis runs are output using the **.PRINT** and **.PLOT** commands.



**.MPRUN** can be used to take advantage of multi-processor machines for the **.MC** command. Please see “[.MPRUN](#)” on page 10-166 for further information.

### Parameters

- **RUNNO**  
Number of simulation runs.
- **OUTER**  
When there are both **.STEP** and **.MC** commands, Eldo performs a full Monte Carlo analysis for each point of the **.STEP** command. If the keyword **OUTER** is specified on

the **.MC** command the nesting of the simulations will be inverted. Instead, a **.STEP** will be performed at each Monte Carlo run (the **OUTER** keyword must be placed after the specification of the number of Monte Carlo runs).

- **OV**

Requests the output of a node voltage or current through a voltage source. These output values are used as reference for the Worst Case analysis.

The syntax for the voltage or current output is as follows:

**I(Vxx[ , Vyy ])**

Specifies the current difference between the voltage sources **Vxx** and **Vyy**. If **Vyy** and the comma are omitted, the current through **Vxx** will be printed.

**V(N1[ , N2 ])**

Specifies the voltage difference between nodes **N1** and **N2**. If **N2** and the preceding comma is omitted, ground is assumed.

The following AC analysis output commands are also available:

<b>VDB(N1[ , N2 ])</b>	<b>IDB(Vxx[ , Vyy ])</b>	<b>IGD(v_source)</b>
<b>VI(N1[ , N2 ])</b>	<b>II(Vxx[ , Vyy ])</b>	<b>VGD(node_name)</b>
<b>VM(N1[ , N2 ])</b>	<b>IM(Vxx[ , Vyy ])</b>	
<b>VP(N1[ , N2 ])</b>	<b>IP(Vxx[ , Vyy ])</b>	
<b>VR(N1[ , N2 ])</b>	<b>IR(Vxx[ , Vyy ])</b>	



For more details on the above AC analysis output formats, refer to “[.PRINT](#)” on page 10-254.

---

For each run, Eldo computes the standard deviation on the **OV** quantity and outputs it in the ASCII output file. The standard deviation (sigma) is computed in the following manner:

For transient simulation:

**sigma** = **SUM(delta\*delta\*h) / TMAX**

where **delta** is the difference between **OV** for nominal run and current run at current step, **h** is the time step for current step, and **TMAX** is the transient simulation duration.

For other analyses:

**sigma** = **SUM(delta\*delta) / nbpt**

where **nbpt** is the total number of points in the simulation.

At the end of the MC simulation, Eldo indicates the run that generated the worst standard deviation as the worst case conditions in the ASCII output file. Here is an example of the output in the **.chi** file:

```
Standard Deviation for run    1: 1.621303E+00
Standard Deviation for run    2: 2.016042E+00
Standard Deviation for run    3: 8.805810E-02
...
Worst Case Conditions: Run Number   30
```

- **SEED=integer\_value**

This number is used to initialize the pseudo-random sequence of numbers. Running the **.MC** analysis twice with the same seed specified will provide the same simulation results.

- **NONOM**

Nominal run for Monte Carlo analysis is bypassed. This option is used by Accusim. When **NONOM** is active, the **ALL** option of **.MC** is enabled as well.

- **ALL**

If this optional parameter is declared, the results of each simulation run are stored in the output files in contrast to the usual nominal, maximum and minimum results.

- **VARY=LOT | DEV**

Specifies that only a **LOT** or **DEV** specification is taken into account. When specifying **DEV** then **DEV** and **DEVX** variation is taken into account. Only one **VARY** specification can be set. By default, Eldo applies **LOT**, **DEV** and **DEVX** variation.

- **IRUN=VAL**

Can be specified if a single, specific run, of a Monte Carlo analysis is required. **RUNNO** is still required to be specified, even though it is not used in that mode. Example:

```
.MC 10 IRUN=3
```

Eldo will then perform a single analysis which corresponds to the 3rd Monte Carlo analysis.

- **NBBINS=VAL**

Specifies the number of bins for the histogram produced when Monte Carlo analysis is used with **.EXTRACT** statements. Default is 10.

- **ORDMCS**

Determines whether multiple MC parameters in the simulation share the same pseudo-random probability values or not. See [page 19-2](#) for usage examples.

- **MCLIMIT**

Specifies that all parameters with statistical distribution (**DEV**, **DEVX**, or **LOT**) will have their distribution modified to one of two deviation values. These values correspond to the maximum deviation of the original distribution as defined by the option **SIGTAIL**. For example, a parameter with a nominal value of 1.0, a statistical deviation of **DEV/GAUSS=5%**, and with option **SIGTAIL** at its default value of 4, the two values will be calculated as follows:

$$1 - (4 * 0.05) = 0.8, \quad 1 + (4 * 0.05) = 1.2$$

This functionality can be useful to force Monte Carlo runs to use maximum deviation combinations. **MCLIMIT** affects all statistical parameters whatever their original distribution.

- **PRINT\_EXTRACT=NOMINAL | ALL | run\_number**

Specifies for which run extracted values should be printed and written to output files.

**NOMINAL**

Only the nominal extracted value is printed (default).

**ALL**

Extracted values are printed for all runs.

**run\_number**

Extracted values are printed only for the specified run\_number (0 is the nominal run).

## Related options

[CARLO\\_GAUSS \(page 11-24\)](#), [SIGTAIL \(page 11-30\)](#), [DISPLAY\\_CARLO \(page 11-45\)](#),  
[EXTMKSA \(page 11-45\)](#), [DUMP\\_MCINFO \(page 11-45\)](#)

## Tolerance Setting Using **DEV**, **DEVX** or **LOT**

The size of the tolerance appears in a **.MODEL** command after the parameter keyword and value. The tolerance may be specified either as a percentage or an absolute quantity. To denote a percentage, the % sign must be used. Parametric expressions are allowed when specifying **DEV** or **LOT**.

The **DEV** tolerance parameter causes devices which use the same **.MODEL** statement to vary independently of each other, as illustrated in the following example:

```
c1 4 0 cmod 10p
c2 6 8 cmod 10p
.model cmod cap dev=10%
```

In the above example, both **c1** and **c2** use the model **cmod**. Their nominal values are both 10pF. The **DEV** declaration placed immediately after the **CAP** keyword indicates that during a Monte Carlo analysis, their values may vary independently of each other by at most  $\pm 10\%$ . So, **c1** could have a value of 9.9pF while **c2** has the value of 10.1pF during a simulation run.

The **DEV** tolerance is appropriate for situations where the variation of parameters is un-correlated. The devices on a printed circuit board are such an example.

The **DEVX** specification forces Eldo to use a new random value for each instance of a subcircuit. The difference with **DEV** is that even if a parameter is used several times in the same subcircuit, only one value will be used for that particular instance.

### Note



**DEVX** can only be used in a **.PARAM** statement and not in a **.MODEL** statement. It is impossible to have **DEV** and **DEVX** specified on the same parameter. If **DEV** and **DEVX** are specified on the same parameter, the last specification will be retained.

The **LOT** tolerance setting causes devices which use the same **.MODEL** statement to vary with each other, as illustrated in the following example:

```
c1 4 0 cmod 10p
c2 6 8 cmod 10p
.model cmod cap lot=10%
```

This is the same as the previous example with the exception that the tolerance has been changed from **DEV** to **LOT**. Now **c1** and **c2** will always have the same value.

They may be both equal to 9.9 pF during one run and 10.1 pF during another run, but **c1** will not have a value of 9.9 pF while **c2** has the value of 10.1 pF during the same run.

The **LOT** tolerance is appropriate for situations where there is a variation of the parameters track. Devices in an integrated circuit are such examples.

## Multiple Runs

It is necessary to specify the number of simulations to be run in the implementation of a Monte Carlo test. The computation time increases linearly with this number, but in practice it is rarely necessary to go beyond 15 runs. The number of runs are specified in one of the parameters of the **.MC** command. Multiple runs of the selected analysis are carried out whereby the first one uses nominal component values, with subsequent runs varying model parameters according to the specifications given via the **LOT** and **DEV** tolerances on each **.MODEL** parameter.

## Examples

```
.model rmod res dev=2%
...
.mc 5 v(n5)
.tran 1n 100n
.plot tran v(n5)
```

Specifies five Monte Carlo runs of the transient analysis at the output node **n5**. All devices with the model name **rmod** have a **DEV** tolerance attached to their nominal value.

```
.model cmod cap lot/gauss=2%
...
.mc 5 v(n5)
.tran 1n 50n
.plot tran v(n5)
```

Specifies five Monte Carlo runs of the transient analysis at the output node **n5**. All devices with the model name **cmod** have a **LOT** tolerance with a Gaussian distribution attached to their nominal value.

```
*MODEL definition
.model mod1 nmos niv=6 eox=25.0n mu0=600
+ nb=2.0e+16 k1=2.24u lot=0.5% gw=3.91u
+ gl=0.7u dev=0.07e-6 rec=0.15u vt0=0.55
...
.mc 7 vdb(n6)
.ac dec 10 1.0e3 1.0e9
.plot ac vdb(n6) vp(n6)
```

---

Specifies seven Monte Carlo runs of the AC analysis at the output node `n6`. All devices with the model name `mod1` have a `LOT` tolerance attached to the parameter `KL` and a `DEV` tolerance attached to the parameter `GL`.



An example of this type of analysis can also be found in “[Tutorial #6—High Voltage Cascade](#)” on page 24-19.

```
*MODEL definition
.model mod1 nmos niv=6 eox=25.0n mu0=600
+ nb=2.0e+16 kl=2.24u gw=3.91u
+ gl=0.7u rec=0.15u vt0=0.55
...
.mcmod mod1 kl lot=0.5% gl dev=0.07e-6
.mc 7 vdb(n6)
```

Specifies the same Monte Carlo run as the previous example, but using the `.mcmod` command.



For more information, see “[.MCMOD](#)” on page 10-153.

---

DEV variation specified with `.MODEL` statements (or `.MCMOD`) can refer to dimensions of the current object directly, without any need to encapsulate models into subcircuits, for example `E(*,<instance_parameter_name>)`.

```
.MODEL M nmos VTH=1 DEV=sqrt(E(*,l)*(E(*,w))*1.0e-5
M1 p1 p2 p3 p4 m w=10u l = 3u
```

The two lines above are equivalent to all those below:

```
.SUBCKT foo d g s b param: w=lu l=lu
.model m nmos VTH=1 DEV=sqrt(l*w)*1.0e-5
M1 d g s b m w=w l=l
.ENDS
X1 p1 p2 p3 p4 foo w=10u l=3u
```

## .MCMOD

### LOT & DEV Variation Specification on Model Parameters (Monte Carlo)

```
.MCMOD MNAME [(list_of_instances)] PAR LOT|DEV=VAL {PAR LOT|DEV=VAL}
.MCMOD MNAME PAR LOTGROUP=my_lot_group
```

This command is used to specify the amount of variation on a given model parameter using the **LOT** and/or **DEV** parameters. By specifying the variation in this way, no modification of the **.MODEL** definition concerned is required. This command is used in combination with Monte Carlo and Worst Case analyses.

#### Note



If you specify a Lot/Dev variation in a **.MCMOD** file, the variation will be around whatever value is input for that model. This may seem obvious but the point here is that the worst case models might be used and the Monte Carlo analysis be run around that. Engineers not clear on “models” might do this without realizing.

Different entities are able to share the same distribution. Anywhere Eldo accepts **LOT/DEV** specifications, you can specify **LOTGROUP=group\_name**.

**.MCMOD** expects the name of a **.MODEL** command as the first argument. However, when **.OPTION MODWL** is used, there are usually several models with the same prefix (**.MODEL MYMOD.1...**, **.MODEL MYMOD.2...**) and none matching exactly the name specified in the **.MCMOD** command (**.MCMOD MYMOD**). Consequently, **.MCMOD** accepts only the prefix name as the argument.

If a parameter is not a primitive, but depends on parameters with no LOT/DEV specification, then a LOT /DEV specification can be set on that parameter.

#### Parameters

- **MNAME**  
Model name.
- **PAR**  
Parameter to be varied by **LOT**, **DEV** or **LOTGROUP**.
- **VAL**  
Value of **PAR LOT | DEV**.
- **list\_of\_instances**  
List of instances; the subsequent parameter list and variation will only apply to the specified devices. The list should be separated by spaces, and enclosed in brackets. e.g.

```
.mcmmod mod1 (m1 m2 m3) vt0=...
```

- **LOTGROUP=my\_lot\_group**

Specifies a “lotgroup” to enable different entities to share the same distribution. Anywhere Eldo accepts **LOT/DEV** specifications, you can specify **LOTGROUP=my\_lot\_group**.



Please refer to “[.LOTGROUP](#)” on page 10-142 for more information.

## Examples

```
*MODEL definition1
.model mod1 nmos niv=6 eox=25.0n mu0=600
+ nb=2.0e+16 k1=2.24u gw=3.91u
+ gl=0.7u rec=0.15u vt0=0.55
...
.mcmod mod1 k1 lot=0.5% gl dev=0.07e-6
.mc 7 vdb(n6)

*MODEL definition2
.model mod1 nmos niv=6 eox=25.0n mu0=600
+ nb=2.0e+16 k1=2.24u lot=0.5% gw=3.91u
+ gl=0.7u dev=0.07e-6 rec=0.15u vt0=0.55
...
.mc 7 vdb(n6)
```

The above two netlist samples produce the same results.

```
.MODEL R R rho=25
.MCMOD R rho=20%
.MCMOD R r=20%
```

The example above shows the limitation of the command for R, L, and C models. The 2nd line is not valid and an error message is issued. The 3rd line is a valid command, parameter name is R.

```
.MODEL NMOS.1 VT0=... WMIN=1u WMAX=10u LMIN=1u LMAX=10U
.MODEL NMOS.2 VT0=... WMIN=11u WMAX=100u LMIN=1u
+ LMAX=10U
M1 ... NMOS W=9U l=9u
M2 ... NMOS W=50u l=9u
.MCMOD NMOS VT0 lot=10%
.option modwl
.end
```

In the above example, model NMOS.1 will be attached to M1, model NMOS.2 will be attached to M2. The **.MCMOD** command will apply to both NMOS.1 and NMOS.2, i.e. the same random number will be used for NMOS.1 and NMOS.2.

```
.param p1 = 1
.param p2 = p1 lot = 5%
.param p3 = p2 dev = 5%
```

In the above example, MC variation on parameter p2 is accepted, MC variation on parameter p3 is ignored because its value depends on the value of parameter p2. The nominal value of p3 will be p2.

# **.MEAS**

## Measure Waveform Characteristics

```

.MEAS [ANALYSIS_INFO] [VECT] [CATVECT] label_name
+ TRIG trig_spec TARG targ_spec
.MEAS [ANALYSIS_INFO] [VECT] [CATVECT] label_name WHEN when_spec AT val
.MEAS [ANALYSIS_INFO] [VECT] [CATVECT] label_name FIND wave WHEN when_spec
.MEAS [ANALYSIS_INFO] [VECT] [CATVECT] label_name FIND wave AT val
.MEAS [ANALYSIS_INFO] [VECT] [CATVECT] label_name FIND W('wave')
+ WHEN when_spec [FROM=val] [TO=val]
.MEAS [ANALYSIS_INFO] [VECT] [CATVECT] label_name DERIVATIVE wave
+ WHEN when_spec
.MEAS [ANALYSIS_INFO] [VECT] [CATVECT] label_name DERIVATIVE wave AT val
.MEAS [ANALYSIS_INFO] [VECT] [CATVECT] label_name meas_k wave
+ [FROM=val] [TO=val]
.MEAS [ANALYSIS_INFO] [VECT] [CATVECT] label_name PARAM='expression'

```

This command can be used as an alternative to the **.EXTRACT** command. **.MEAS** has the same capabilities as **.EXTRACT**, however the syntax of **.MEAS** is occasionally preferred by some users. **.MEAS** can be used in three ways:

1. Measuring a time interval or wave values

Measurement starts when the **TRIG** conditions, as defined by the **trig\_spec** parameters, are matched.

2. Measurement on a wave

3. Combination of measurements

For compatibility with other simulators, no asterisk \* character is printed in the ASCII output (.chi) file before the **.MEAS** result.

As a comparison between **.MEAS** and **.EXTRACT**: **.EXTRACT** offers greater flexibility, however it requires that each entity appearing in a **.EXTRACT** must be saved in memory. **.MEAS** does not make this requirement.

## Parameters

- **ANALYSIS\_INFO**

should be replaced with one of the following parameters;

### **AC**

Measurement during AC analysis.

### **DC**

Measurement during DC analysis.

### **DCTRAN**

Measurement after the DC analysis performed prior to a TRAN analysis.

### **DCAC**

Measurement after the DC analysis performed prior to an AC analysis.

**DCSWEET**

DCSWEET measurement.

**TRAN**

Measurement during TRAN analysis.

**NOISETRAN**

Measurement during NOISETRAN analysis.

- **VECT**

By default, **.MEAS** returns the first value which matches the expression. But if keyword **VECT** is set on the **.MEAS** statement, then all values will be returned.

- **CATVECT**

Works in the same way as **VECT** but in addition all measurements corresponding to all analyses (**.STEP** / **.TEMP**) will be combined. This functionality is usually used in conjunction with the **CONTOUR** function in the **.PLOT** command.



For more information on the **CONTOUR** function see [page 10-223](#).

- **label\_name**

Identifies the **.MEAS** command in all output files. **label\_name** must be defined.

- **TRIG**

Measurement starts when the **TRIG** conditions, as defined by the **trig\_spec** parameters below, are matched:

- **trig\_spec:**

**WAVE VAL=val [TD=val] [CROSS=index] [RISE=index] [FALL=index]**  
+ [**SIG\_H=val**] [**SIG\_L=val**]

or:

**AT[=]val**

**WAVE**

Name of wave.

**VAL**

Threshold value.

**TD**

Time delay until measurement commences.

**CROSS**

Number of times the threshold must be crossed (in whichever direction) before measurement starts.

**RISE**

Number of times the wave values rise above the threshold before measurement starts.

**FALL**

Number of times the wave values fall below the threshold before measurement starts.

### **SIG\_H, SIG\_L**

High and Low signal values respectively. Default high and low values are taken from the threshold value **VAL**. These high and low values are used to validate a transition before incrementing the cross, rise or fall counter.

- **AT**

Measurement starts/stops at **val**. The **val** depends on the analysis type: time for TRAN analysis, frequency for AC analysis, or the parameter (x-axis value) for DC analysis.

- **TARG**

Measurement stops when the **TARG** conditions, as defined by the **targ\_spec** parameters, are matched.

- **targ\_spec**

The arguments are the same as for **trig\_spec**, except that the keyword **LAST** can be specified in place of a value for CROSS, RISE and FALL. If the time delay **TD** is not specified, the time delay specified in **trig\_spec** will be used.

- **WHEN**

Measurement stops when the **WHEN** conditions, as defined by the **when\_spec** parameters, are matched.

- **when\_spec**

The arguments are the same as for **targ\_spec**, except that both **WAVE** and **VAL** specifications are replaced by either of the following:

```
WAVE=VAL
WAVE1=WAVE2
```

- **DERIVATIVE**

Keyword which can be used in place of **FIND**, to specify that it is the derivative of the wave that must be returned when **WHEN** specification is matched.

- **meas\_k**

Used for measurement on a wave. This can be one of the following keywords:

#### **AVG**

Average value of the waveform in the range [ FROM , TO ].

#### **RMS**

RMS value of the waveform in the range [ FROM , TO ].

#### **MIN**

Minimum value of the waveform in the range [ FROM , TO ].

#### **MAX**

Maximum value of the waveform in the range [ FROM , TO ].

#### **PP**

Peak-to-Peak value of the waveform in the range [ FROM , TO ].

- **PARAM= 'expression'**

Regular expression that combines the label names of the **.MEAS** commands.  
**expression** cannot be a wave name. Plot quantities such as **i()**, **v()**, **lv9()** are only accepted in a **.MEAS DC** analysis. For example:

```
.MEAS DC vth0 param='lv9(m)'
```

## Examples

The example below will return the last time that **v(in1)** crossed value 2.0 while falling:

```
.MEAS TRAN mymo TRIG AT=0 TARG v(in1) val=2 FALL=LAST
```

The example below will return the value of **v(1)** when **v(2)** becomes equal to the **v(5)** measurement, starting after a delay of 3n:

```
.MEAS TRAN foo FIND v(1) WHEN v(2)=v(5) TD=3n
```

The example below will return the average of **v(in1)** between 3n and 5n:

```
.MEAS TRAN my_avg avg AVG v(in1) FROM=3n TO=5n
```

The example below will return the average of **V(s)**:

```
.PARAM p1=1
.MEAS TRAN avg_name AVG V(s)
```

The example below will return the value of measurement named **avg\_name** + value of parameter **P1**:

```
.MEAS TRAN f2 PARAM='avg_name+p1'
```

The following example shows how the **VECT** keyword can be used to return all the values which match the expression. By default, **.MEAS** returns the first value which matches the expression.

```
v1 1 0 pulse(0 5 0 1n 1n 8n 20n)
r1 1 0 1
.extract vect xup(v(1),2.5)
.meas vect tran memes trig at 0 targ v(1) val=2.5 rise=1
.tran 1n 100n
.end
```

The following example shows how the **W** notation can be used with the **.MEAS** command. This will search the value of wave **vp(1)** when wave **vdb(1)** will cross zero between 10Hz and 1000Hz.

```
.MEAS AC ph1 FIND W('vp(1)') WHEN vdb(1)=0 FROM=10 TO=1000
```

## Related option

**FROM\_TO** ([page 11-14](#))

## .MODDUP

### Aspire/SimPilot Command

```
.MODDUP device_name [.... device_name]  
.MODDUP element_name
```

This command is useful in connection with SimPilot/Aspire only. It tells Eldo that for each `device_name` a private `.MODEL` command be created. Then, it will be possible for the SimPilot/Aspire engine to alter the content of this new model command via commands.

When the netlist is sourced, SimPilot/Aspire stores the `.MODDUP` arguments (device instance names) and relates these to the associated `.MODEL` commands. Therefore, the user does not need to manually duplicate `.MODEL` commands inside the netlist. These copies reside inside the Eldo simulator.

With `element_name` specified, the effect of this command is that the model attached to `element_name` is duplicated, and becomes private to that element. Parameters of the associated `.MODEL` command can be modified/displayed in interactive mode via the commands:

```
SET EM(Element_name,Model_parameter) = value  
PRINT EM(Element_name,Model_parameter)
```

- `device_name`  
Device instance name.
  - `element_name`  
Element name.
- `.MODDUP` ‘carries’ parameter dependencies, e.g.

```
.PARAM P1 = 1.0  
.MODEL FOO NMOS VT0 = P1  
M1 .... FOO ..  
M2 .... FOO ..  
.MODDUP M1  
.STEP PARAM P1 1 2 1  
...  
.END
```

Here, `.STEP P1` will update the `VT0` parameter of `FOO` as well as updating the `VT0` parameter of the newly created model attached to `M1`.



For an example, please see the *Aspire User’s Manual* document [on page 9-34](#).

---

## **.MODEL**

### Device Model Description

```
.MODEL MNAME TYPE [ PAR=VAL ]
.MODEL LIB FILENAME MODNAME [ LIBTYPE ]
```

This command groups sets of pre-defined parameters which may be used by one or more devices. Models may be described directly in the input file or may be read from a library file using either the second syntax above or the **.LIB** and **.ADDLIB** commands. The second syntax allows no continuation lines.

It is also possible to specify model parameters on the instance command. The effect is that a private model will be created for that instance.

### Parameters

- **MNAME**  
The model name. It must not start with a number.
- **TYPE**  
Defines the model used. The following models are available:

**Table 10-9. Model Types**

<b>RES</b>	Resistor
<b>R</b>	RC wire
<b>CAP</b>	Capacitor
<b>IND</b>	Inductor
<b>NPN</b>	NPN bipolar junction transistor
<b>PNP</b>	PNP bipolar junction transistor
<b>LPNP</b>	Lateral PNP bipolar junction transistor
<b>D</b>	Diode
<b>NMOS</b>	N-channel metal oxide field effect transistor
<b>PMOS</b>	P-channel metal oxide field effect transistor
<b>NJF</b>	N-channel junction field effect transistor
<b>PJF</b>	P-channel junction field effect transistor
<b>NSW</b>	N-type switch (SC)
<b>PSW</b>	P-type switch (SC)
<b>OPA</b>	Operational amplifier (SC)
<b>MODFAS</b>	Analog macromodel

**Table 10-9. Model Types**

<b>LOGIC</b>	Digital Gate
<b>A2D</b>	Analog-to-Digital converter
<b>D2A</b>	Digital-to-Analog converter
<b>LDTL</b>	Lossy transmission line
<b>FBLOCK</b>	S parameter blocks

- **.LIB**

Keyword indicating a model library file is to be used.

- **FILENAME**

Name of the library file that contains the model description.

- **MODNAME**

Name of the model stored in library file FILENAME. See the [Library variant management](#) in the **.LIB** command description for details.

- **PAR=VAL**

Name and value of a model parameter. Model parameters not given values are assigned default values. Model parameters may also be declared via the **.PARAM** command. See the [“.PARAM”](#) on page 10-205 for further details.

The parameter **BULK=node\_name** can be specified on a **.MODEL NMOS | PMOS** command. This will connect the bulk node to node\_name if it is not specified in the instantiation of the model. By default node\_name is 0.

- **LIBTYPE**

Name of a library variant to be used.



For more information on electrical parameters of models, refer to the [Device Models](#) chapter.

## Examples

```
*MODEL definition
.model rmodel res tc1=0.001 tc2=0.005
...
*main circuit
r2 n1 n19 rmodel 2.5k
```

Specifies the resistor **r2** of model type **rmodel**.

```
*BJT model definition
.model qmod npn bf=160 rb=100 cjs=2p
+ tf=0.3n tr=6n cje=3p cjc=2p vaf=100
...
*main circuit
q23 10 24 13 qmod
```

Specifies the bipolar transistor q23 of model type qmod.

```
*OPAMP model definition
.model ampop modfas voff=100e-6
...
*main circuit
yopal opamp1 n2 n1 n3 0 model: ampop
```

Specifies the single-stage 1-pole op-amp yopal with the electrical parameters specified in the model ampop.

```
*NOR# .MODEL definition
.model nor_1 logic vhi=5 vlo=-5 vth=0
+ tpd=2.5n cin=0.5p
...
*main circuit
nor#_1 n1 n2 n3 n4 o1 nor_1
```

Specifies a NOR gate nor#\_1 with four input nodes n1, n2, n3 and n4 and an output node o1, the parameters of which are described in the model nor\_1.

```
M1 ... MOD1 w=1u l=1u M(DW)=0.5u
M2 ... MOD1 w=1u l=1u
.model MOD1 NMOS DW=3u LOT=3%
```

In this example, a private model will be created for M1. Changing the DW parameter value of model MOD1 (via **.STEP** for instance) will not affect the value of parameter DW for the model attached to device M1.

#### **Note**



This feature only works for MOS, BJT, Diodes and JFET. Care must be taken when using this feature, since a private model is created for each instance. The memory requirement on circuits that contain many such model parameter specifications on instance commands can be very high. Monte Carlo specifications are not propagated to the new model. In the example above, the DW value of instance M1 will not be changed during a Monte Carlo analysis.

In the following example the bulk node will be connected to node 4. This is because a node has not been specified in the instantiation of the MOS model and the parameter BULK has been specified in the **.MODEL** command.

```
M1 1 2 3 N W=10u l = 3U
.MODEL N NMOS BULK = 4
```

If **BULK** was omitted, by default the bulk node would be connected to node 0 (ground node).

## Monte Carlo and Models

The Monte Carlo command is used in combination with extra parameters placed in the **.MODEL** command. Each **.MODEL** parameter to be subjected to statistical variation, may

have two extra related parameters added, **DEV** and **LOT**. The significance of these parameters is explained in the following paragraphs.

An example of a **.MODEL** command, modified to include Monte Carlo analysis parameters, is shown below:

```
.model mmod nmos vto=0.65v dev=0.4v
+ tox=1.5e-7 dev=0.2e-7 lot=5%
```

The first **DEV** declaration refers to the **VTO** parameter, and the second (combined) **DEV** & **LOT** declaration refers to the **TOX** parameter.

## .MODLOGIC

### Digital Model Definition

**Caution**



**Obsolete Syntax!**

Kept for compatibility reasons only and should no longer be used

---



For the new syntax, see the **.MODEL ... LOGIC** description on [page 7-3](#).

---

```
.MODLOGIC MNAME [VHI=VAL1] [VLO=VAL2] [VTH=VAL3] [VTHI=VAL4]
+ [VTLO=VAL5] [TPD=VAL6] [TPDUP=VAL7] [TPDOWN=VAL8]
+ [CIN=VAL9] [DRV1=VAL10] [DRVH=VAL11]
```

Used for the definition of digital gate models.

### Parameters



For detailed information on the parameters, see the **.MODEL ... LOGIC** description on [page 7-3](#).

---

## .MONITOR

### Monitor Simulation Steps

**.MONITOR ANALYSIS [=] [modulo]**

This command can be used to display the steps taken by the simulator when doing an AC, TRAN or DCSWEEP simulation. Current TIME/FREQ or DCSWEEP value is also displayed. This command can be used for debugging purposes, i.e. to see how the simulation proceeds.

Eldo will display the current time and time step every modulo steps.

### Parameters

- **ANALYSIS**

Analysis type for which you request simulation steps to be monitored. Can be one of the following:

**DC**

Specifies that a DC analysis is monitored.

**AC**

Specifies that a AC analysis is monitored.

**TRAN**

Specifies that a transient analysis is monitored. analysis.

- **modulo**

Number of steps at which the information is displayed. Default is zero, meaning all values are printed out.

### Examples

**.MONITOR AC**

Eldo will display all the frequency points in an AC analysis.

## **.MPRUN**

### Multi-Processor Simulation

```
.MPRUN [ALL|HOST={host[(nbjobs)]}|FILE=filename] [NBLICENSES=val]
+ [MAX_NBJOBS=val] [CLEAN=YES|NO] [QUEUE=YES|NO] [SETENV=YES|NO]
+ [VIEW_COMMAND=YES|NO] [CHECK_DELAY=val] [INIT_FILE=filename]
+ [DEFAULT_INIT=YES|NO] [CD_WORKDIR=YES|NO] [LOGFILE=YES|NO]
+ [SHELL_SYNTAX=(source_cmd, setenv_cmd, setenv_sep, init_anacad_ext)]
+ [USE_LOCAL_HOST=YES|NO] [FLAT=YES|NO]
+ [FILE_PREFIX=(name1, name2, ..., nameX)]
```

This command is used to run multi-threading simulations on one multi-processor machine, or on many machines. **.ALTER**, **.MC**, **.TEMP**, **.STEP**, **.DATA** and **.OPTIMIZE** are distributed by this command.

The child processes are launched through a *rsh* call which by default inherits the environment variables used in **.INCLUDE** or **.LIB** statements. Temporary results are stored in subdirectories named **<NETLIST\_NAME>.part<X>** where **X** is an incrementing counter. By default—and unless they are in use—temporary directories are removed once the simulation is complete.

#### **Note**



All *boolean* keywords, i.e. **CLEAN**, **SETENV**, **QUEUE**, **VIEW\_COMMAND**, **CD\_WORKDIR**, **LOGFILE** can be set using: **<keyword>[=YES|NO|1|0]**. For example, specifying the **SETENV** keyword is equivalent to **SETENV=YES**, which is equivalent to **SETENV=1**. Additionally, when the **SETENV** keyword is not specified it will use its default value, in this case it would be **SETENV=NO**, which is equivalent to **SETENV=0**.

### Parameters

- **ALL**

This is the default. Keyword specifies Eldo to run the simulation on all the processors of the machine. Eldo will find the number of processors of the machine and distribute the tasks between them. If the machine has one processor, Eldo will run the simulation normally without taking this command into consideration.

- **HOST={host1[(nbproc1)] host2[(nbproc2)]... hostN[(nbprocN)]}**

Keyword specifies Eldo to run the simulation on the list of machines: host1, host2, ... hostN (commas are optional). Eldo will distribute the tasks on the list of machines specified.

**nbjobs** is an optional parameter that explicitly tells Eldo the maximum number of jobs that can be submitted on this machine. On multi-processor stations, this number should be the number of processors.

- **FILE=filename**

Specifies the name of a file which contains a list of machine names (with a number of processors if needed). The file can have any extension or no extension at all. The first line of the file is read. Example file contents:

```
pluton kebra(3)
morkai(2) nao
cochise
```

- **NBLICENSES=val**

Specifies the maximum number of licenses that this job can use. It must be greater than 1 to be taken into account, since the parent process always takes its own license.

- **MAX\_NBJOBS=val**

Specifies the maximum number of jobs that can be submitted for all machines.

- **CLEAN[ =YES | NO ]**

Default value is YES. This specifies Eldo to remove temporary files created in any child process subdirectories, together with removing the subdirectories themselves. If keyword is set to NO, the temporary files are not removed.

- **QUEUE[ =YES | NO ]**

Instructs the system to wait for the release of a license if one is not immediately available. A consequence is that the parent process will hang until its child process has finished. Default value is NO.

- **SETENV[ =YES | NO ]**

Keyword specifies Eldo to reuse all environment variables in child processes, and not only those given on **.LIB** and/or **.INCLUDE** statements. Default value is NO.

- **VIEW\_COMMAND[ =YES | NO ]**

Prints the command Eldo submits to the remote host and the contents of the command file. Default value is NO.

- **CHECK\_DELAY=val**

Forces Eldo to wait `val` seconds while checking host connections. This is useful when using both Linux and Unix networks, since a delay can appear between the time when a remote command is executed on Unix and the time when the result of the command is effective on Linux.

- **INIT\_FILE=filename**

This allows the user to define a script file which is sourced before running child processes.

- **DEFAULT\_INIT[ =YES | NO ]**

Default value is YES. If keyword is set to NO, it tells Eldo that the script specified in **INIT\_FILE** will replace any other default. This should always be used in conjunction with **INIT\_FILE=filename**. If using this, it is the user's responsibility to insure that the script file correctly sets the path and required variables for Eldo to run.

- **CD\_WORKDIR** [ =YES | NO ]

This tells Eldo to change the working directory in the remote environment to the directory where the netlist is located. Default value is YES.

- **LOGFILE** [ =YES | NO ]

This controls the redirection of the standard output of sub-processes. The default value is YES, which means that Eldo dumps the standard output in <NETLIST\_NAME>.log in the temporary directories.

- **SHELL\_SYNTAX**= ( source\_cmd, setenv\_cmd, setenv\_sep, init\_anacad\_ext )

Before performing a simulation on a remote host, Eldo uses shell commands to define environment variables and execute initialization scripts. By default, Eldo automatically recognize C-Shell and Korn-shell syntaxes. But if the shell is unknown or the commands need to be redefined, it can be done using the **SHELL\_SYNTAX** statement.

source\_cmd

command used to execute shell scripts (“source” for *csh* and “.” for *ksh*).

setenv\_cmd

command used to define environment variables (“setenv” for *csh* and “export” for *ksh*).

setenv\_sep

character used to set the value of an environment variable (a blank for *csh* and “=” for *ksh*).

init\_anacad\_ext

specific extension of *init\_anacad* script for this shell (none for *csh* and “.ksh” for *ksh*).

- **USE\_LOCAL\_HOST** [ =YES | NO ]

Setting this option to NO tells Eldo that it cannot use the local host to perform some simulations. Default value is YES (the local host is the machine on which the main process has been launched).

- **FILE\_PREFIX**= ( name1, name2, . . . , nameX )

In case of splitting **.ALTER** statements, a user may want to rename the output files of each run. If a flag is given after **.ALTER**, it is used as the name. Output names can also be redefined with this **FILE\_PREFIX** option. For example if a netlist contains two **.ALTER** statements and the following **.MPRUN** option **FILE\_PREFIX**= (first, second, third), output files for each run will be:

- i. for the netlist before **.ALTER** statements: *first.chi*, *first.wdb*, etc.
- ii. for the netlist after the first **.ALTER** statement: *second.chi*, *second.wdb*
- iii. for the netlist after the second **.ALTER** statement: *third.chi*, *third.wdb*

- **FLAT** [=YES | NO]

Usual **.MPRUN** mechanism uses temporary directories to store the results of intermediate runs. If this option is set to YES, Eldo will use different names for each intermediate run instead of using directories. Thus it is highly recommended to use this option in conjunction with **FILE\_PREFIX** to define the name of the runs. Default value is NO.

## Usage

Before performing a run on a remote host, the **.MPRUN** command must initialize the remote environment. By default, Eldo does the following before running a simulation:

```
cd <working_directory>
setenv anacad ...
source $anacad/com/init_anacad
setenv eldover ...
setenv LM_LICENSE_FILE ...
```

If you specify **INIT\_FILE=<filename>**, the sequence becomes:

```
cd <working_directory>
source <filename>
setenv anacad ...
source $anacad/com/init_anacad
setenv eldover ...
setenv LM_LICENSE_FILE ...
```

and if you also specify **DEFAULT\_INIT=NO**, the sequence becomes:

```
cd <working_directory>
source <filename>
```

and if you also specify **CD\_WORKDIR=NO**, the sequence becomes:

```
source <filename>
```

and if you remove the **INIT\_FILE=<filename>**, absolutely no initializations will be performed before Eldo is run.

## Notes

1. In all cases, Eldo will warn you if a host cannot be used. This will happen when:
  - You don't have the permissions to see the machine and/or to write in the working directory. This error must be fixed by the system administrator before the **.MPRUN** command can be used.
  - Too many hosts are given in the command line compared to the number of tasks that can be distributed.
2. To use the **.MPRUN** command with both Linux and Sun/HP platforms, you must provide an initialization script and use the **DEFAULT\_INIT=NO** keyword. This script must give paths for both Linux and Sun/HP installation directories. It should look like:

```
set AMS_OStype='uname -a | grep -ic "Linux"'
switch ($AMS_OStype)
```

```

case 1:
    setenv anacad /product/AMS_linux
    breaksw
default:
    setenv anacad /product/AMS
endsw

source $anacad/com/init_anacad
setenv LM_LICENSE_FILE ...

unset AMS_OStype

```

## Examples

```
.MPRUN HOST=host1, host2 NBLICENSES=2 QUEUE=YES
```

Specifies Eldo to run the simulation on both the host1 and host2 machines. A maximum of two licenses can be used by this simulation. If a license is not immediately available, the system will wait for the release of a license.

```
.MPRUN HOST=host3(2) host4
+ INIT_FILE=script.shell DEFAULT_INIT=NO
```

Specifies Eldo to run the simulation on both the host3 and host4 machines, also with two processors of host3 specified as running the simulation. A script file *script.shell* will be sourced before running child processes. This script will replace any other default initialization file.

## .MPRUN and external dispatchers

```

.MPRUN DISPATCHER= [LSF |
+ (dispatcher_name, install_check_cmd, submission_cmd]
+ [DISPATCHER_OPTIONS=(options)]
+ [NBLICENSES=val] [MAX_NBJOBS=val] [CLEAN=YES|NO]
+ [QUEUE] [SETENV] [VIEW_COMMAND] [CHECK_DELAY=val]
+ [INIT_FILE=filename [DEFAULT_INIT=YES|NO]]
+ [USE_LOCAL_HOST=YES|NO] [FLAT=YES|NO]
+ [FILE_PREFIX=(name1,name2,...,nameX)]]

.MPRUN
+ DISPATCHER_TEMPLATE=command_line
+ [USE_LOCAL_HOST=YES|NO] [FLAT=YES|NO]
+ [FILE_PREFIX=(name1,name2,...,nameX)]

```

A dispatcher is software which shares and manages computer resources across a network. Instead of a basic remote shell command, the dispatcher uses various criteria (memory usage, CPU charge) to determine which machine is suitable to run the simulation on. The second syntax above is useful for **.ALTER** dispatching only. For all other options listed above, please refer to the descriptions on the previous pages.

- **DISPATCHER=LSF**

Specifies Eldo to use LSF as external dispatcher. This syntax is a shortcut to **DISPATCHER=( "LSF" , "bsub -V" , "bsub" )**.



For more information on LSF, please visit the Platform web site:  
<http://www.platform.com>

- **DISPATCHER=(dispatcher\_name, install\_check\_cmd, submission\_cmd)**

The general syntax to configure a dispatcher.

dispatcher\_name

name of the software (required for print purpose only).

install\_check\_cmd

a command that can prove the software is accessible.

submission\_cmd

the command that submits a job to the engine.

- **DISPATCHER\_OPTIONS=(options)**

This keyword allows to send extra options to the dispatcher submission command. The dispatcher options must be enclosed in ( ) or {}.

- **DISPATCHER\_TEMPLATE**

This option allows to completely override the usual mprun mechanism. It is useful for .ALTER dispatching only. The user is in charge of providing a valid shell command which will dispatch jobs. This command can use predefined variables which are substituted by Eldo before executing the command. These variables are:

%NETLIST\_NAME%

stands for the name of the netlist %RUN\_NAME% stands for the name of each run (may be redefined with FILE\_PREFIX)

%RUN\_NUMBER%

stands for an absolute run counter (starting from 1) %MPRUN\_OPTIONS% is a mandatory variable which represents internal options added by Eldo.

## Example with LSF

```
.MPRUN DISPATCHER=LSF
+ DISPATCHER_OPTIONS=(-q myqueue -m "host1 host2")
+ MAX_NBJOBS=5
```

Specifies Eldo to use the LSF management system as a dispatcher for the remote jobs. The simulation will be run on both the host1 and host2 machines. A maximum of five jobs will be submitted to LSF. -q myqueue is the LSF option which controls the Batch Queue to which the jobs will be submitted.

## Example with .ALTER dispatching

Suppose file *mplex.cir* contains the following command in conjunction with two .ALTER statements:

```
.mprun
+ flat=yes
+ use_local_host=no
+ dispatcher_template="eldo %NETLIST_NAME% -queue %MPRUN_OPTIONS% >
%RUN_NAME%.log"
+ file_prefix=(first, second, third)
```

After parsing the netlist, Eldo will immediately execute the following commands:

```
eldo mpex.cir -queue ... -out first > first.log eldo mpex.cir -queue ...
-out second > second.log eldo mpex.cir -queue ... -out third > third.log
```

This example only demonstrates the syntax since it does no actual dispatching.

## .MSELECT

### Automatic Model Selection

```
.MSEL[LECT] dummy [MODELS] mod1 [mod2 [mod3 [...]]]
```

This command allows the user to select models automatically for MOS devices. The selection is based on:

- the size and temperature of the specific device (W, L, TEMP)
- the size and temperature constraints of each model in the list provided (WMIN, WMAX, LMIN, LMAX, TEMPMIN, TEMPMAX)

It is not allowed to have a model statement with the same name as an mselect dummy model name. If a model statement has the same name as an mselect dummy model name, Eldo will display an error message, for example:

```
ERROR 953:Dummy model name MOD1 on .MSELECT statement is also defined on
a .MODEL statement
```

### Parameters

- dummy  
Dummy model name that is used on the devices for which you want automatic model selection.
- MODELS  
Optional keyword used only to enhance **.MSELECT** statement readability.
- mod1 ... modn  
List of model names from which a new model is selected.

### Additional information

- Device temperature can also be specified. Eldo will check against model parameters TEMPMIN/TEMPMAX to select the right device
- If a value is not specified for any of the models parameters (TEMPMIN/TEMPMAX/LMIN/LMAX/WMIN/WMAX) the checks versus the limits are not done.
- Models are searched in the order there are given on the mselect statement
- Many mselect with the same name can be defined. The previous definitions are automatically overwritten. Eldo will use the last one, for example:

```
.mselect dummy2 models mo1 mo2.
.mselect dummy2 models mo1 mo2 mo3 mo4
```

The last one will be used.

- When none of the models defined on a mselect fit with the device parameter an error message is displayed, for example:

```
ERROR 845: OBJECT "M1": None of the models in .MSELECT fits this instance
```

- If the one the models specified on the mselect model list does not exist a simple warning is emitted, for example:

```
Warning 488: COMMAND .MSELECT: Model MOD7 is not defined
```

- When 2 different types of models (example NMOS and PMOS) an error is emitted, for example:

```
ERROR 945: Wrong Model type for MOD2 on .MSELECT statement - mixing models type is not allowed
```

## Limitation

Eldo implementation of mselect only work for MOS devices and models.

## Example

An example with mselect is below.

```
.MODEL mod11 ...
.MODEL mod12 ...
.MODEL mod21 ...
.MODEL mod22 ...

.mselect mod2 MODELS mod21 mod22
.mselect mod1 MODELS mod11 mod12

M1 A G VDD VDD MOD2 W=120U L=5.5U
M2 B G VDD VDD MOD2 W=120U L=5.5U
M3 D K A VDD MOD2 W=116U L=3.5U
M4 S K B VDD MOD2 W=116U L=3.5U
M5 C I VSS VSS MOD1 W=63U L=6U
M6 A EP C VSS MOD1 W=130U L=4U
M7 B EN C VSS MOD1 W=130U L=4U
M8 D D FF VSS MOD1 W=5.5U L=4.5U
M9 S D E VSS MOD1 W=5.5U L=4.5U
M10 FF E VSS VSS MOD1 W=42U L=4U
M11 E E VSS VSS MOD1 W=42U L=4U
M12 G G VDD VDD MOD2 W=14.5U L=5.5U
M13 G G H VSS MOD1 W=9U L=5.5U
M14 I I H VDD MOD2 W=19U L=4.5U
M15 I I VSS VSS MOD1 W=6U L=6U
M16 J G VDD VDD MOD2 W=20U L=5.5U
M17 J J K VSS MOD1 W=26U L=3.5U
M18 NL I K VDD MOD2 W=3U L=3.5U
M19 NL NL VSS VSS MOD1 W=4U L=3.5U
```

There are two mselect statements and 4 models definitions. The MOS devices are instantiated using the mselect dummy names instead of models real names. In this example only W and L

---

instance parameters are used. The actual models used on the instances are selected when these dimensions are within the model parameters LMIN/LMAX and WMIN/WMAX

As the results of the selection you will have:

* M1 , M2 M3 M4	mapped to MOD21
* M12 M14 M16 M18	mapped to MOD22
* M5 M8 M9 M10 M11 M13 M15 M17 M19	mapped to MOD11
* M6 M7	mapped to MOD12

## .NET

### Network Analysis

#### 2-port network

```
.NET output input RIN=val ROUT=val
```

#### 1-port network

```
.NET input RIN=val
```

The **.NET** command is another approach to extract the S parameters (Scattering parameters), the Y parameters (Admittance), the Z parameters (Impedance) or the H parameters (Hybrid) in the frequency domain for a specified circuit.

The circuit can only have one or two ports.

### Parameters

- **input**  
Can be V source or I source.
- **output**  
Can be V source, I source or V(NP,NN).
- **RIN ROUT**  
Specify the values of access resistors of the input and output port.

### Example

```
.ac dec 500 1e6 10e6
.net v(outputnode) vinput_source RIN=50 ROUT=50
```

The following example shows a 1-port network extraction on voltage source **v1** with input access resistance of 50 ohms.

```
v1 1 0 ac 1 0
r1 1 2 1
c1 2 0 10p
.ac dec 10 1 10G
.plot ac sdb(1,1)
.plot ac sp(1,1)
.net v1 rin=50
```

# **.NEWPAGE**

## Control Page Layout

### **.NEWPAGE**

This command allows the control of the default page composition layout (Xelga) or the saved windows file (EZwave) in the waveform viewers. It allows lines to be inserted into an Eldo netlist, with the effect that **.PLOT** commands located between two **.NEWPAGE** commands will be plotted in the same EZwave window or Xelga page. **.NEWPAGE** only acts on *.wdb* and *.cou* files.

---

### Note



The number of items per **.PLOT** is not limited. It is possible to have any number of waves in the same plot, although reading the ASCII plot may be difficult.

---



See “**.PLOT**” on page 10-216 for more information.

---

## Example

```
.PLOT TRAN (v1) (v2)
.PLOT TRAN (v3) (v4)
...
.PLOT TRAN (v15)
.NEWPAGE
.PLOT TRAN (v15)
...
```

This allows the user to control page layout, Eldo will plot the graphs following the **.NEWPAGE** command in a new composition page.

## .NOCOM

### Suppress Comment Lines from Output File

.NOCOM

This command suppresses any comment lines in the ASCII output (*.chi*) file which come after it in the netlist. Saves disk space.

### Example

```
example title
.nocom
*This is a sample comment line
r1 1 2 5
*here is another
...
.end
```

# **.NODESET**

## DC Analysis Conditions

```
.NODESET V(NN)=VAL [ SUBCKT=subckt_name ] {V(NN)=VAL [ SUBCKT=subckt_name ] }
```

This command is used to help calculate the DC operating point by initializing selected nodes during the first DC operating point calculation. After the first calculation has been completed the node values are “released” and a second DC operating point calculation is started. This command is useful when the whereabouts of the DC operating point is known, enabling the simulator to converge directly to it and also for bistable circuits or circuits with more than one operating point.

The **.NODESET** command differs from the **.GUESS** command in so far as when using **.NODESET** node voltages are fixed for the duration of the first DC calculation, whereas the node voltages are only initialized for the first iteration of a DC operating point calculation when using **.GUESS**. It is very important to specify realistic **.NODESET** values as convergence problems may occur when this command is not used properly.

### Note

By default, the first **.NODESET** specification has precedence over subsequent **.NODESET** specifications. Setting **.OPTION LICN**, the last **.NODESET** specification will have precedence.



See [page 11-28](#) of the *Eldo User’s Manual* for further information.

## Parameters

- **V(NN)=VAL**  
Voltage at node **NN** in volts.
- **SUBCKT=subckt\_name**  
If specified it will fix the voltage of the preceding node in all instances of the subcircuit **subckt\_name**.

## Examples

```
.nodeset v(n4)=6v v(n5)=2v v(n6)=-5v
```

Specifies that during the first DC operating point calculation, the initial values for the voltages at the nodes **n4**, **n5** and **n6** be initialized to 6V, 2V and -5V respectively.

```
.nodeset v(2)=3v SUBCKT=sub1 v(4)=-2v SUBCKT=sub2
```

Specifies that during the first DC operating point calculation, the initial values for the voltages at node **2** of subcircuit **sub1** and node **4** of subcircuit **sub2** will be initialized to 3 and -2V respectively.

## **.NOISE**

### Noise Analysis

```
.NOISE OUTV INSRC NUMS (X<subckt_instance_name>)
```

The **.NOISE** command controls the noise analysis of the circuit and must be used in conjunction with an AC analysis.

The results of noise analysis runs are output using the **.PRINT** and **.PLOT** commands.



It is possible to control the output of the noise information via the **NOXTABNOISE** option on [page 11-48](#).

### Parameters

- OUTV

Name of the output voltage node for which the equivalent output noise is to be calculated.  
The syntax is as follows:

**V(N1[ , N2 ])**

Specifies the voltage difference between nodes N1 and N2. If N2 and the preceding comma are omitted, ground is assumed.

**I(Vxx)**

Specifies the first argument as a voltage source.

- INSRC

Name of the input voltage or current source for which the equivalent input noise is to be calculated.

- NUMS

Indicates that only every NUM<sup>th</sup> frequency point is stored for print-out. The contribution of every noise generator in the circuit is printed at every NUM<sup>th</sup> frequency point. If NUMS is zero, no print-out is made. NUMS can be specified as a parameter or as an expression.

- X<subckt\_instance\_name>

Returns the total noise contribution of a subcircuit instance. Its value is the sum of the noise of all elements that are part of the specified subcircuit instance.

### Example

```
.ac dec 70 100k 10meg
.noise v(5) vin 70
...
*output control
.plot noise inoise onoise
.plot noise db(inoise) db(onoise)
```

Specifies `vin` as input noise reference and `v(5)` as the voltage at the summing point. The noise is averaged over seventy frequency points and the input and output results are to be plotted on the same graph, the limits of which are controlled by the `.AC` command.



---

An example of this type of analysis can be found in “[Tutorials](#)” on page 24-1.

---

---

**Note**

---



NOISE analysis may also be run from within a `.tran` command, see the [AC in the middle of a .TRAN](#) section for more details.

---

## **.NOISETRAN**

### Transient Noise Analysis

```
.NOISETRAN FMIN=VAL FMAX=VAL NBRUN=VAL [NBF=VAL] [AMP=VAL]
+ [SEED=VAL] [NOMOD=VAL] [NONOM] [TSTART=VAL] [TSTOP=VAL]
+ [MRUN] [ALL] [NBBINS=VAL] [FMIN_FLICKER=VAL]
```

This command is used to control the transient noise analysis of a circuit and must be used in conjunction with a transient (**.TRAN**) analysis, and not with a Monte Carlo (**.MC**) analysis.

It is possible to define the three parameters **FMIN**, **FMAX** and **NBF** for each noisy component; [Resistor](#), [Junction Diode](#), [BJT—Bipolar Junction Transistor](#), [JFET—Junction Field Effect Transistor](#), [MESFET—Metal Semiconductor Field Effect Transistor](#), [MOSFET](#), [Independent Voltage Source](#) and [Independent Current Source](#).



Please refer to the appropriate sections.

To reduce the CPU time, parallel noise runs are performed during a single transient analysis to compute the RMS noise results, instead of several runs. This allows larger circuits to be handled, larger number of runs and the ability to analyze a circuit with a CPU time close to a normal (noiseless) transient analysis.

### Parameters

- **FMIN=VAL**

Lower limit of the noise frequency band.

- **FMAX=VAL**

Upper limit of the noise frequency band.

**FMIN** and **FMAX** define the frequency band of the noise sources. This frequency range may sometimes not correspond to the noise frequency band at the output of the circuit. For instance, the band (**FMIN**, **FMAX**) does not correspond to the output noise frequency band in the case of filters or oscillators and mixers that exhibit frequency conversion. **FMIN** is also used to specify the algorithm used to generate the noise sources in the time domain.

When **FMIN>0** the noise sources are generated as a sum of **NBF** sinusoids. When **FMIN=0** another algorithm is used, generating noise sources with a continuous spectrum between 0 and **FMAX**.

- **NBRUN=VAL**

Defines the number of simulations which are performed with the noise sources included. If **NBRUN** is 1, the output voltages or currents stored in the binary output file are stored as simple voltages or currents, otherwise an RMS curve will be displayed in Xelga for the noise. The noise voltages and currents of the **NBRUN** simulations are stored together in a file with the suffix **.hmp.wdb** which may be visualized with the waveform viewer (EZwave). The **.hmp.wdb** file contains results for each of the NOISE analysis. The **.hmp.wdb** file is

created only if the **MRUN** flag is set on the **.NOISETRAN** command, or if this flag has been automatically activated by Eldo in the case it could not complete a **.NOISETRAN** in single run. If **NBRUN** is greater than 1, the binary output file output will be the RMS value of these curves. Otherwise, the binary output file will simply contain the noise voltage(s) or current(s).

If not required for the simulation results, the *.hmp.wdb* file should be removed in order to save disk space.

- **NBF=VAL**

Specifies the number of sinusoidal sources with appropriated amplitude and frequency and with randomly distributed phase from which the noise source is composed. The default value is 50. This parameter has no effect when **FMIN** is set to 0.

- **AMP=VAL**

This parameter is the noise source amplification factor and only affects internal noise computations. The noise is internally multiplied by this factor in order to differentiate electrical noise from numerical noise and afterwards, when RMS noise values are calculated, the values are divided by this factor again to provide correct results. This factor should only be used if the noise level is very low (e.g. below 1  $\mu$ V). Care must be taken if this feature is used as some circuits are non-linear, even with small signals, which may cause incorrect results. The default value of **AMP** is 1.

- **SEED=VAL**

Used to initialize the random number generator. Must be an integer between 0 and  $2^{31}-1$ . Performing 2 noise simulations will provide the same RMS noise results. The default value is 0.

- **NOMOD=VAL**

Switch giving the following results:

- 0 No effect. Default.
- 1 Noise simulation without thermal noise.
- 2 Noise simulation without flicker noise.

---

**Note**

The **NOMOD** parameter only concerns the MOSFET device.

---

- **NONOM**

Specifies that only one noisy simulation should be performed and correspondingly, the nominal simulation is suppressed.

- **TSTART=VAL**

Specifies the first time point from which the circuit generates noise (before **TSTART** the circuit is noiseless). Default value is 0.

- **TSTOP=VAL**

Specifies the last time point of the transient noise analysis. After TSTOP the circuit no longer generates noise. Default value is TSTOP of the **.TRAN** command.

- **MRUN**

Forces the algorithm to perform several runs sequentially to compute the RMS noise results. This should be specified if you do not want the algorithm to perform parallel noise runs. Under the JWDB output format the **.hmp** output file will not be generated, this can be overridden using the option **KEEP\_HMPFILE** on [page 11-46](#). When specified, extracted results for noisy simulations are printed to the output file. Histograms will be written to the output file if the number of simulation runs is greater than the number specified.

- **ALL**

Forces Eldo to dump the results of the simulation in the ASCII output files (by default these waves are not dumped, since they are normally unusable).

- **NBBINS=VAL**

Specifies the number of bins for the histogram produced when the transient noise is used with the **.EXTRACT** command. Default is 10.

- **FMIN\_FLICKER=VAL**

Specifies the lower limit of the noise frequency band for the flicker noise source when **FMIN=0**. If **FMIN>0** this parameter will be ignored. Optional.

---

**Note**

 The default values of **eps**, **vntol** and **reltol** are changed in transient noise analysis from the values used in other analyses. See “[.OPTION](#)” on page 10-198.

---

In this particular case, the defaults are **eps**= $1.0 \times 10^{-6}$ , **vntol**= $1 \times 10^{-6}$  and **reltol**= $1 \times 10^{-6}$ . Thus by default, the simulator is able to compute noise voltages down to about  $1 \mu\text{V}$ . As most of the applications create higher levels of noise, this should be sufficient in most cases. In all other cases, it is advisable to increase the simulator accuracy.

## **.NOTRC**

### Suppress Netlist from an Output File

**.NOTRC**

This command, when inserted immediately after the title line of a circuit description file, suppresses the rewriting of the circuit description file in the ASCII output (*.chi*) file.

### Example

```
example title
.notrc
r1 1 2 5
...
.end
```

## .NWBLOCK

### Partition Netlist into Newton Blocks

```
.NWBLOCK [RELTOL=value] [VNTOL=value] list_of_nodes
```

This command allows the user to control the way Eldo will partition the netlist into several Newton Blocks. Additionally, blocks can have different accuracies.

There can be several **.NWBLOCK** commands. Each **.NWBLOCK** will result in the creation of a Newton Block.

### Parameters

- **list\_of\_nodes**

The list of nodes to be assigned to a Newton Block. The wildcard character '\*' is allowed as shown in the following example:

```
.NWBLOCK x1.*
```

- **RELTOL=value**

Controls the accuracy of the Newton Block. Optional. Same meaning as the **RELTOL** global parameter that can be specified with:

```
.OPTION RELTOL=val
```

The scope of visibility is limited here to the node/instances referred to in the **.NWBLOCK** command.

- **VNTOL=value**

Controls the voltage accuracy of the simulator. Optional. Same meaning as the **VNTOL** global parameter that can be specified with:

```
.OPTION VNTOL=val
```

The scope of visibility is limited here to the node/instances referred to in the **.NWBLOCK** command.

### Notes

- If a node is referred to in several **.NWBLOCK** commands, the node will belong to the first block created via the **.NWBLOCK** command.
- **.NWBLOCK** commands are used in transient analysis (TRAN) only. **.NWBLOCK** is ignored for DC analysis.
- For nodes which do not appear in any **.NWBLOCK** commands: if the node cannot be solved by OSR, they will be grouped into another newton block.
- Once nodes have been assigned to a Newton Block, Eldo may decide to group together some or all of the newton blocks just created, for the sake of simulation efficiency.

## .OP

### DC Operating Point Calculation

```
.OP [[KEYWORD] T1 {[KEYWORD] TN}]  
.OP TIME=VAL|END [STEP=VAL] [TEMP=VAL]  
.OP DC=VAL [DC2=VAL] [STEP=VAL] [TEMP=VAL]
```

This command forces Eldo to determine the DC operating point of the circuit with inductors short-circuited and capacitors opened. If either the specified simulation time is reached or one of the conditions described in the “optional parameters” below is fulfilled, the operating point is saved to the *.chi* file.

If no parameter is specified, the operating point information is saved for DC prior to AC or first DC analysis in the case of a DC sweep.

Additional information concerning the operating points such as power dissipation, node voltages and source currents are written to the *.chi* output file.

The result table generated by the **OP** command includes all **OP** results in addition to node voltages and device currents. These terms (BETADC, BETAAC, CXS, VTH\_D) are printed out in the **OP** table, and can also be plotted/printed.

### Parameters

- **KEYWORD**  
Can be one of the strings: **CURRENT**, **VOLT**, **BRIEF**, **ALL**. Default is **ALL**. When keyword is **BRIEF** or **CURRENT** (they are synonymous), then OP information for each element is displayed, but contains only a limited set of information compared to the **ALL** case. When **VOLT** is set, only the voltage nodes are displayed.
- **T1**, **TN**  
Simulation times at which operating point information will be recorded. The parameter **END** can be specified as a simulation time.
- **TIME=VAL**  
Simulation time for which **.OP** results are written.
- **END**  
Forces Eldo to output the **.OP** information after a transient analysis (**.TRAN**) if specified.
- **STEP=VAL**  
Step value for which **.OP** results are written. This is the case for when the **.STEP** command is used.
- **TEMP=VAL**  
Current temperature for which **.OP** results are written.

- **DC=VAL**  
DC sweep value for which **.OP** results are written.
- **DC2=VAL**  
Second DC value for which **.OP** results are written in cases where a double DCSWEEP analysis is performed.

**Note**

A **.OP** command is always automatically performed prior to an **AC** analysis. If no other analysis is specified, a **.OP** command forces a **DC** analysis to be performed after the operating point has been calculated.

For MOSFET and BJT, Eldo prints in the DCOP point table the operating region of the device. The following messages are written:

- For MOSFET:

```
if ((vgs - vgt) < 0) "SUBTHRESHOLD"
else if ((vds - vdss) < 0) "LINEAR"
else "SATURATION"
```

- For BJT:

```
if (VBC > VBCSAT)
  if (VBE > 0) "SATURATION"
  else "INVERSE"
else
  if (VBE > 0) "ON"
  else "OFF"
```

VBCSAT can be specified in the **.OPTION** command. Default is 0.

**Note**

When displayed in the ‘operating point table’, the gain **GMB** of a MOS transistor could sometimes be negative. This occurs whenever topological Source and Drain are different from electrical Source and Drain. Now, **GMB** is always positive.

**Options**

- |                          |  |
|--------------------------|--|
| <b>.OPTION OPTYP=VAL</b> | Used to change the way Operating Point information related to MOSFETs is displayed in the ASCII output file.   |
| <b>.OPTION OPTYP=1</b>   | Full operating point table is displayed. This is the default.  |
| <b>.OPTION OPTYP=2</b>   | Can be specified only when either the <b>-st</b> flag or the <b>STVER</b> option are set. Used to print the reduced operating point information for the MOSFET models (see the Spice documentation for more details). Otherwise equivalent to <b>optyp=1</b> . This can be used with UDM and BSIM4 models. |
| <b>.OPTION OPTYP=3</b>   | Output compatible with Spice3e2.   |

The table below gives the character string that appears in the Operating Point table for the different **optyp** values. Elements on the same line are synonymous: that is, the same value would be printed if **optyp** changes.

**Table 10-10. Operating Point—optyp values**

Default	optyp=3	When -st flag or STVER option is set	When -st flag or STVER option is set & optyp=2
ID	ID	ID	ID
		IS	
		IB	
VGS	VGS	VGS	VGS
VDS	VDS	VDS	VDS
VBS	VBS	VBS	VBS
VTH	VTH	VTH	VTH
VDSAT	VDSAT	VDSAT	VDSAT
			gm
			gm
GM	GM		
GDS	GDS		
GMB	GMB		gmbs
		Ibd	Ibd
		Ibs	Ibs
		Gdd	Gbd
		Gdg	Gbs
		Gds	
		Gsd	
		Gsg	
		Gss	

An explanation of some of these parameters is provided below:

- Ibd* Bulk-drain current through the parasitic *BD* diode
- Ibs* Bulk-source current through the parasitic *BS* diode
- VTH* Internal threshold OP dependent; see the *Eldo Device Equations Manual* for how it is computed

VSS or *VDSAT*Saturation voltage at DCOP

*G<sub>xy</sub>* Derivative of static current on node *x* with respect to *V<sub>y</sub>*; *x* and *y* can be one of *D, G, S, B*

*C<sub>xy</sub>* Derivative of charge on node *x* with respect to *V<sub>y</sub>*; *x* and *y* can be one of *D, G, S, B*

*I<sub>sub</sub>* Subthreshold current component

In the OP table for BSIM3v3, Eldo also displays *VBI* and *PHI*:

*PHI* Surface potential at strong inversion

*VBI* Built-in voltage for the PN junction between the substrate and the source

**Table 10-11. Operating Point—optyp values  
Dynamic Part for Charge Control Model**

Default	optyp=3	When -st flag or STVER option is set	When -st flag or STVER option is set & optyp=2
Cdd	Cddb	Cdd	
Cdg	Cdgb	Cdg	
Cds	Cdsb	Cds	
Cdb		Cdb	Cdb+CJDB
Cgd	Cgdb	Cgd	Cgd+CVGD
Cgg	Cggb	Cgg	
Cgs	Cgsb	Cgs	Cgs+CVGS
Cgb		Cgb	Cgb+CVGB
Csd		Csd	
Csg		Csg	
Css		Css	
Csb		Csb	Csb+CJSB
Cbd	Cbdb	Cbd	
Cbg	Cbgb	Cbg	
Cbs	Cbsb	Cbs	
Cbb		Cbb	

**Table 10-12. Spice3e Capacitance / Eldo Capacitance**

<b>Spice3e capacitance</b>	<b>Relationship with Eldo capacitance</b>
Cbsb	$-(C_{bg} + C_{bd} + C_{bb})$
Cbdb	$C_{bd}$
Cbgb	$C_{bg}$
Cdsb	$(C_{gg} + C_{gd} + C_{gb} + C_{bg} + C_{bd} + C_{bb} + C_{sg} + C_{sd} + C_{sb})$
Cddb	$-(C_{gd} + C_{bd} + C_{sd})$
Cdgb	$-(C_{gg} + C_{bg} + C_{sg})$
Cgsb	$-(C_{gg} + C_{gd} + C_{gb})$
Cgdb	$C_{gd}$
Cggb	$C_{gg}$

In Spice operating point display the **Cbs** and **Cbd** values are bulk-source and bulk-drain currents correspondingly. These results don't correspond to Eldo **Cbs** and **Cbd** since these last two items correspond to capacitances.

### Example

```
R1 N1 N2 R1
.PARAM R1=1k
.STEP PARAM R1 1k 10k 1k
.TRAN 1n 100n
.OP
.OP TIME=9n STEP=5k
.END
```

In the above example, an Operating Point will be determined for a resistance of  $5\text{k}\Omega$  at time 9ns.

### Note



The **.op** command may be used to run AC and NOISE analyses from within a **.tran** command at specified times, see “[AC in the middle of a .TRAN](#)” on page 10-17 for more details.

## .OPTFOUR

### FFT Post-processor Options

```
.OPTFOUR [ TSTART=VAL|EXPR ] [ TSTOP=VAL|EXPR ] [ NBPT=VAL ] [ FS=VAL ]
+ [ INTERPOLATE=0|1|2|3 ] [ NOROUNDING[=1] ] [ WINDOW=name ] [ ALPHA=VAL ]
+ [ BETA=VAL ] [ FMIN=VAL ] [ FMAX=VAL ] [ FNORMAL=freq ] [ PADDING=1|2|3 ]
+ [ NORMALIZED=0|1 ] [ DISPLAY_INPUT=0|1 ]
```

Used to supply the options for the FFT post-processor. (The FFT post-processor inside Eldo is exactly the same as that in Xelga.) See also the [.FOUR command](#) on page 10-119.

### Setting Parameters

The following four parameters are used to specify the time window on which the FFT will be performed. This can be summarized by [Table 10-13](#), which provides the values of TSTART, TSTOP, FS, NBPT for all possible cases.

- **TSTART**

Time value from which the time window will start. Default value is option STARTSMP if specified, or TSTART from .TRAN, or TSTOP-NBPT/FS if all three other parameters specified.

- **TSTOP**

Time value that specifies the end of the time window. Default value is TSTART from .TRAN, or TSTART+NBPT/FS if all three other parameters specified.

- **FS**

Sampling frequency. Default value is NBPT/(TSTOP-TSTART).

- **NBPT**

Number of points to be used in the time window. Default value is 1024, or (TSTOP-TSTART)\*FS if FS specified.

**Table 10-13. Default Values for Time Window Parameters**

TSTART	TSTOP	FS	NBPT
User Defined	TSTOP	<b>NBPT / (TSTOP-TSTART)</b>	1024
<b>STARTSMP or TSTART</b>	User Defined	<b>NBPT / (TSTOP-TSTART)</b>	1024
<b>STARTSMP or TSTART</b>	TSTOP	User Defined	(TSTOP-TSTART)*FS
<b>STARTSMP or TSTART</b>	TSTOP	<b>NBPT / (TSTOP-TSTART)</b>	User Defined
User Defined	User Defined	<b>NBPT / (TSTOP-TSTART)</b>	1024
User Defined	TSTOP	User Defined	(TSTOP-TSTART)*FS
User Defined	TSTOP	<b>NBPT / (TSTOP-TSTART)</b>	User Defined

**Table 10-13. Default Values for Time Window Parameters**

<b>TSTART</b>	<b>TSTOP</b>	<b>FS</b>	<b>NBPT</b>
<b>STARTSMP or TSTART</b>	User Defined	User Defined	(TSTOP-TSTART)*FS
<b>STARTSMP or TSTART</b>	User Defined	<b>NBPT / ( TSTOP-TSTART )</b>	User Defined
<b>STARTSMP or TSTART</b>	TSTART+NBPT /FS	User Defined	User Defined
User Defined	User Defined	User Defined	(TSTOP-TSTART)*FS
User Defined	User Defined	<b>NBPT / ( TSTOP-TSTART )</b>	User Defined
User Defined	TSTART+NBPT /FS	User Defined	User Defined
TSTOP-NBPT/FS	User Defined	User Defined	User Defined
User Defined	User Defined	User Defined	User Defined

**Note**



If both **FS** and **NBPT** parameters are specified by the user then **NBPT** will be calculated using the relation **NBPT = (TSTOP - TSTART) \* FS.**

**Note**



**PADDING** parameter can be specified if **TSTART**, **TSTOP**, **FS** and **NBPT** are used and **NBPT > (TSTOP-TSTART)\*FS.**

- **NORMALIZED**

Specifying 1 means that all the points of the result are multiplied by 2/**NBPT**. Default. Specifying 0 means no normalization of the results.

- **INTERPOLATE**

Sets type of interpolation:

- 0 = No interpolation (default)
- 1 = Linear interpolation
- 2 = Cubic Spline interpolation
- 3 = Blocker Sampler interpolation

- **NOROUNDING[=1]**

By default, Eldo truncates and rounds **TSTART** and **TSTOP** values at 1ps. If keyword **NOROUNDING** is specified, this rounding is not performed.

### Notes

1. FFT is much faster when the **NBPT** parameter (either user given, or computed from the three other parameters) can be a product of powers of 2, 3 and/or 5 so that fast algorithms can apply. For example, 1022 can be divided by 2, but is not a good

candidate;  $1024=2^{10}$  or  $960=3\times5\times2^6$  are better candidates. In such cases, the FFT program will make use of several optimizations which do not alter the FFT results.

2. The last time point value actually taken into account by FFT is the value at time: **TSTOP**-1.0/**FS**. This is to deal with the fact that when FFT is done on a circuit which is in steady-state mode, time value F(**TSTOP**) equals the time value F(**TSTART**), and hence that last value must not be used for the FFT since FFT must be computed on a full number of periods.
3. **TSTART** and **TSTOP** time values can be specified as results of an expression that follows the **.EXTRACT** syntax, on [page 10-95](#). When the expression can be measured, FFT sampling will start.

---

**Caution**



When the parameter **TSTOP** is specified as an expression and the parameter **NBPT** is used, then it is impossible for Eldo to determine the sampling frequency before the simulation. Furthermore, Eldo cannot compute exactly the points which will be used to compute the FFT. Therefore, the parameter **INTERPOLATE** cannot be set to 0 (no interpolation). In this case a warning message will be displayed by Eldo and the parameter **INTERPOLATE** will be set to 2.

---

4. A very important point is related to the **INTERPOLATE** parameter: FFT must be done on equally-spaced time value points. However, the time points computed by an analog simulator such as Eldo are not equally spaced. There are two possibilities to obtain the equally-spaced points required by FFT:
  - Force Eldo to compute values at least on the points requested by the FFT (and in between Eldo can do smaller time steps if needed),
  - Interpolate between the time values points computed ‘freely’ by Eldo: this interpolation can be a LINEAR interpolation (**INTERPOLATE** = 1) or a higher order interpolation (**INTERPOLATE** = 2: a cubic SPLINE is used).

FFT results are better when no interpolation occurs. For this reason the parameter **INTERPOLATE** defaults to 0. In such a case, Eldo will actually compute time-value points at least every  $1/\mathbf{FS}$  seconds. Since this behavior is exactly the purpose of the **.OPTION FREQSMP=val** command, this latter command would be ignored in the case of **.OPTFOUR** with **INTERPOLATE** equal to 0, and instead the **FS** value would be used.

If **INTERPOLATE** is set to 1 or 2, Eldo will compute its time steps according to the activity in the design (and if provided, the **FREQSMP** will be used) and the equally-spaced time points required by the FFT will be interpolated from that time-domain waveform generated by Eldo.

---

### Note

 **INTERPOLATION=2** gives better results than **INTERPOLATION=1**. If **INTERPOLATE** is set to a non-zero value, and if the user specifies a **.OPTION FREQSMP=val**, with **val** being equal to **FS**, then FFT results will be the same as that for the case **INTERPOLATE=0**.

---

## PADDING Parameter

Specifies where to add zeros inside the FFT input window: only used if **TSTART**, **TSTOP**, **NBPT** and **FS** are given and **NBPT > (TSTOP-TSTART)\*FS**.

**PADDING** can be set to:

- 1 Zeros are added at the beginning of the FFT input window
- 2 Zeros are added at the end of the FFT input window
- 3 Zeros are added evenly at the beginning and at the end of the FFT input window.

## Display Parameters

- **FMIN**  
Starting frequency used inside the FFT result window.
- **FMAX**  
Last frequency used inside the FFT result window.
- **FNORMAL=freq**  
Adjusts the results around the Y-axis so that the point for the specified frequency is 0.0.
- **DISPLAY\_INPUT**  
Allows visualization of the transient window used as input of the FFT: this waveform contains equally-spaced time value points.
  - 0 = do not display FFT input (default)
  - 1 = display FFT input (if output is displayed)

## Sampling WINDOW Parameter

Specifies the Sampling Window to be used. The following parameters are allowed:

**RECTANGULAR** (default)  
**PARZEN**  
**BARTLETT**  
**WELCH**  
**BLACKMAN**  
**BLACKMAN7**  
**KLEIN**

**HAMMING**  
**HANNING**  
**KAISER**  
**DOLPH\_CHEBYCHEV**

## Associated Parameters

- **ALPHA**

Used when the **WINDOW** is **DOLPH\_CHEBYCHEV** or **HANNING**.

- **BETA**

Used when the **WINDOW** is **KAISER**. Constant which specifies a frequency trade-off between the peak height of the side lobe ripples and the width of energy in the main lobe.



Please refer to the “[Windowing](#)” on page 2-66 of the *Xelga User’s Manual* for further details of the windowing equations.

## Example

```
.optfour tstart=xdown(V(1),2u,1) tstop=xdown(V(1),2u,1000) ..  
.four ...
```

Eldo computes an FFT with 1000 periods of signal **V(1)**. This shows how **TSTART** and **TSTOP** time values can be specified as results of an expression. When the expression can be measured, FFT sampling will start.

```
.optfour tstart=0 tstop={xdown(V(1),10m,13)} nbpt=10000
```

When the parameter **tstop** is specified as an expression and the parameter **nbpt** is used, then it is impossible for Eldo to determine the sampling frequency before the simulation. Eldo cannot compute exactly the points which will be used to compute the FFT. Therefore, the parameter **INTERPOLATE** cannot be set to 0 (no interpolation). In this case the following warning message will be displayed by Eldo and the parameter **INTERPOLATE** will be set to 2.

Warning: **.OPTFOUR** : parameter **INTERPOLATE** is set to 2 because **NBPT** is given and **TSTOP** is an expression

## **.OPTIMIZE**

### Optimization

```
.OPTIMIZE [qualifier=value {, qualifier=value } ]  
+           [PARAM=list_of_parameters | *]  
+           [RESULTS=list_of_targets | *]
```

The global specification of an optimization configuration acting on all the analyses specified in the circuit netlist is done using the **.OPTIMIZE** command.



For the complete description of optimization capabilities, see the separate chapter [Optimizer in Eldo](#).

---

## .OPTION

### Simulator Configuration

`.OPT[ION] OPTION[=VAL] {OPTION[=VAL]}`

The `.OPTION` command allows the user to modify Eldo execution behavior by allowing the setting of parameter values other than the default ones.



For the complete description of options available, see the separate chapter “[Simulator and Control Options](#)” on page 11-1.

---

# .OPTNOISE

## AC Noise Analysis

```
.OPTNOISE [ ALL ON|OFF ] [<CLASS> ON|OFF ]
+ [ R ON|OFF|<max> ] [ OUTSOURCE ON|OFF ] [ NSWEIGHT <FILENAME> ]
+ [ SORT D|V|TD|TV [ SN <n>|SV <value> ] ] [ NBW <FMIN> <FMAX> ]
```

.OPTNOISE allows more flexibility in the output of the AC noise analysis results: noisy elements can be sorted in different ways, and a weight function can be applied before printing out the noise contribution. The .OPTNOISE command must be used in conjunction with the .AC and .NOISE analyses. .OPTNOISE has no effect on .SSTNOISE analysis and results.

## Parameters

- **ALL**  
Specifies that all devices should contribute to the total noise.
- **CLASS**  
Is one of the following keywords: **MOS**, **BJT**, **NPN**, **PNP**, **NMOS**, **PMOS**, **DIODE**, **JFET**, **NJF**, **PJF**. For example: **MOS OFF** means all MOS devices source noise is turned OFF.
- **OUTSOURCE**  
Specifies that printing out of NOISE information in the ASCII output file must be turned ON or OFF.
- **NSWEIGHT <filename>**  
Reads a file for weighted functions: the format of this file is shown on [page 10-200](#).
- **R**  
**ON**: all resistors are assumed to be noisy.  
**OFF**: all resistors are assumed to be noiseless  
**MAX**: all resistors above **MAX** are assumed to be noisy.
- **SORT**  
Choose the way the information is displayed in the ASCII output file:  
**D**: sort by device,  
**V**: sort by value,  
**TD**: sort by technology (CLASS), and in each CLASS by device,  
**TV**: sort by technology (CLASS), and in each CLASS by value,  
**SN <n>**: list the highest n contributions: default n=20  
**SV <value>**: list the device contribution until **value** in % of the total noise is exceeded. Default **value=0.95** (95%).

- **NBW**

stands for Noise BandWidth, which can be different than that of AC band: RMS Average is computed on this bandwidth which defaults to the AC bandwidth. FMIN and FMAX values define the frequency band of this noise bandwidth.

## Multiple output noise

Several **.NOISE** commands can be given for the same run. Eldo will then perform as many NOISE analyses as required. See the OUTPUT/post-processing section below for reporting information.

## Output/Post-processing

1. A noise table that contains general information is first printed out: for each output node there appears the noise RMS value, the Average and total noise power, the maximum and minimum noise power contribution, and the frequency at which these last two values are obtained. In case the weighting function is applied, both weighted and unweighted values are dumped, but for other information below, just weighted or unweighted values will come depending on the content of the **.OPTNOISE** command.
2. For each output noise, total noise source values are displayed, sorted according to the **.OPTNOISE** specification (**SORT SN** or **SORT SV** arguments).
3. For each output and each listed element which appears in 2), the table total noise=f (FREQ) is printed out, each <n> frequency points, where n is the number specified in the **.NOISE** command. Here, frequencies are those of AC: an asterisk (\*) should appear to mark the frequency corresponding to Noise Bandwidth. If there is no sorting of information, then the regular SPICE output will be printed out, with detail of the contribution of each device.
4. Regular frequency output table: this is the content of **.PRINT NOISE**.
5. A file <namecir>\_nsa.cou will also be created, containing the noise response of the devices selected by the **SORT** command option.

## Format of files containing weight

References:

1. IEEE Standard Methods and Equipment for Measuring the Transmission Characteristics of Analog Voice Frequency Circuits. (IEEE Std. 743-1984)
2. SCAMPER Reference Manual, Rel.501, 1990 (noise analysis and option)
3. SCAMPER User's Guide, Rel.501,1990

## Definition

C-Message filter is a frequency-weighting characteristic, used for measurement of noise in voice frequency communications circuits and designed to weight noise frequencies in

proportion to their perceived annoyance effect to a typical listener in telephone service.  
Reference point 1000Hz.

Format of the noise weight table:

```
FreqS|* FreqE|* <flatweight> <weightunit>
+ <freq_interpolation_type>
f1 w1
...
fn wn
```

## Parameters

- **FreqS**  
Starting frequency for noise weight. If “\*” is given then the first frequency in the freq/weight data table will be taken as the starting frequency. Units in Hertz.
- **FreqE**  
Ending frequency for noise weight. If “\*” is given then the last frequency in the freq/weight data table will be taken as the ending frequency. Units in Hertz.
- **flatweight**  
If the flatweight value is given, then it will be taken as a constant (flat) noise weight over the given frequency band (no noise outside the band). The freq/weight data table, if any, will be ignored. If “\*” is given in place of `FreqS` and/or `FreqE`, then the corresponding value will be taken from the `$frequency` line in the scamper netlist.
- **weightunit**

Units of the flatweight or the weight values in the freq/weight data table:

```
DBL  weight given in decibels loss (default)
DB   weight given in decibels
MAG  weight given in numerical value (for noise power)
```

- **freq\_interpolation\_type**  
Type of interpolation for noise weight between frequencies.
  - LOG logarithmic frequency interpolation (default)
  - LIN linear frequency interpolation
- **fi wi**  
Table of frequency and associated weighting values. More than one pair could be given on one line.

## Example weight file

```
-----
' this is a comment
100    4000    DBL    LOG  ' weight is in DBL, LOG interpolation
60      55.7    '      2
```

100	42.5	'	2
200	25.1	'	2
300	16.3	'	2
400	11.2	'	1
500	7.7	'	1
600	5.0	'	1
700	2.8	'	1
800	1.3	'	1
900	0.3	'	1
1000	0.0	'	0      ' reference point
1200	0.4	'	1
1300	0.7	'	1
1500	1.2	'	1
1800	1.3	'	1
2000	1.1	'	1
2500	1.1	'	1
2800	2.0	'	1
3000	3.0	'	2
3300	5.1	'	2
3500	7.1	'	2
4000	14.6	'	3
4500	22.3	'	3
5000	28.7	'	3

### Table noise input source

It is possible to provide Eldo some input noise source values depending on the frequency via the **TABLE** keyword. Eldo accepts the following:

```
Ixx P1 P2 NOISE TABLE [DEC|LOG|LIN] (f1,val1) (f2,val2)...
Vxx P1 P2 NOISE TABLE [DEC|LOG|LIN] (f1,val1) (f2,val2)...
```

Eldo will assume zero current source or zero voltage source in any mode other than NOISE (i.e. for DC, AC, TRAN, etc.).



For further details, please refer to the “[Independent Voltage Source](#)” on page 5-4 and also the “[Independent Current Source](#)” on page 5-11.

# **.OPTPWL**

## Accuracy by Time Window

```
.OPTPWL PARAM=((TIME1,VALUE1) (TIME2,VALUE2)...)
+ PARAM=((TIME1,VALUE1)...))
```

The **.OPTPWL** command allows some precision parameters to be reset during transient simulation for different time windows. <PARAM> can be any one of a number of parameters. Values are given in a piecewise-linear format. **.OPTPWL** can also be changed in Eldo interactive mode.

## Parameters

- PARAM

This can be any of the following parameters: **EPS**, **RELTOL**, **RELTRUNC**, **VNTOL**, **CHGTOL**, **HMAX**, **HMIN**, **ABSTOL**, **FLUXTOL**, **NGTOL**, **OUT\_RESOL**, **ITOL**, **FREQFFT**, **PCS**, **TUNING**.

The **FREQFFT** value can be applied on the time-domain window only if **.OPTFOUR** is not active and if there is no digital simulator connected.

In the **.OPTPWL** command, **FREQFFT** must be used in place of the keyword **FREQSMP** which is used in the **.OPTION** command, i.e. **FREQFFT** in **.OPTPWL** is equivalent to **FREQSMP** in **.OPTION**.

The **PCS** option can be used in conjunction with the **.OPTPWL** command to specify the time after which to turn on PCS. This might be useful when the PCS option is used in the case of non-periodic conditions which may even slowdown the simulation a little.

## Example

```
.OPTPWL RELTOL=(0,1.0e-4) (10n,1.0e-3)
.OPTION RELTOL=1.0e-5
```

In the above example, during DC analysis, the value 1.0e-5 will be used, and during Transient analysis a value of 1.0e-4 will be used until 10n, 1.0e-3 will be used thereafter.

### Note



TIME1 can be 0, in such case VALUE is used for DC, and would overwrite the default value or the value given via **.OPTION**. TIME<sub>x</sub>, VALUE<sub>x</sub> can be parameters set via **PARAM** commands. **.OPTPWL** only works in TRAN.



Please also refer to “**.OPTWIND**” on page 10-204.

## **.OPTWIND**

### Accuracy by Time Window

```
.OPTWIND PARAM=(TIME1,TIME2,VALUE1) . . . (TIME1N,TIME2N,VALUEN)
```

The **.OPTWIND** command allows some precision parameters to be reset during transient simulation for different time windows. **<PARAM>** can be any one of a number of parameters. Values are given per window time. **.OPTWIND** can also be changed in Eldo interactive mode.

### Parameters

- **PARAM**

This can be any of the following parameters: **EPS**, **RELTOL**, **RELTRUNC**, **VNTOL**, **CHGTOL**, **HMAX**, **HMIN**, **ABSTOL**, **FLUXTOL**, **NGTOL**, **OUT\_RESOL**, **ITOL**, **FREQFFT**, **PCS**, **TUNING**.

The **FREQFFT** value can be applied on the time-domain window only if **.OPTFOUR** is not active and if there is no digital simulator connected.

In the **.OPTWIND** command, **FREQFFT** must be used in place of the keyword **FREQSMP** which is used in the **.OPTION** command, i.e. **FREQFFT** in **.OPTWIND** is equivalent to **FREQSMP** in **.OPTION**.

The **PCS** option can be used in conjunction with the **.OPTWIND** command to specify the time after which to turn on PCS. This might be useful when the PCS option is used in the case of non-periodic conditions which may even slowdown the simulation a little.

### Example

```
.OPTWIND RELTOL=(0,10n,1.0e-4) (100n,200n,1.0e-3)
.OPTION RELTOL=1.0e-5
```

In this example, during DC analysis, the value 1.0e-4 will be used, and during Transient analysis a value of 1.0e-4 will be used until 10n, then the default value will be used until 100n (i.e. 1.0e-5), 1.0e-3 will be used between 100n and 200n, and 1.0e-5 again after 200n.

#### **Note**



TIME1 can be 0, in such case VALUE is used for DC, and would overwrite the default value or the value given via **.OPTION**. TIME<sub>x</sub>, VALUE<sub>x</sub> can be parameters set via **.PARAM** commands. For **.OPTWIND** command only, the very last specification can be: (**<TIME>**, , **<VALUE>**) which means that after time **<TIME>**, the parameter value will be equal to **<VALUE>**. **.OPTWIND** only works in TRAN.



Please also refer to “[.OPTPWL](#)” on page 10-203.

# **.PARAM**

## Global Declarations

### Simple Description

```
.PARAM PAR=VAL {PAR=VAL}
```

### Algebraic Description

```
.PARAM PAR=EXPR {PAR=EXPR}
```

### Assigning a Character String

```
.PARAM PAR="NAME"
```

### User Defined Function

```
.PARAM PAR(a,b)=EXPR
```

## Monte Carlo Analysis Parameters

```
.PARAM PAR=VAL|PAR=EXPR
+ LOT|DEV[ /GAUSS | /UNIFORM | /USERDIST ]=VAL | (dtype,-3sig,+3sig
+ [,bi,-dz,+dz [,off,sv [,scale]])
.PARAM PAR=VAL LOTGROUP=my_lot_group
.PARAM PAR=MC_DISTRIBUTION
.PARAM PAR=VAL DEVX=VAL
```

**.PARAM** is used to assign values to parameter variables used in model and device instantiation statements. Parameters and expressions may be used in all of the following cases, for definition and value updating of:

- Device and Model Values.
- Independent Voltage and Current Source Values.
- Linear, Polynomial Coefficients, and Arithmetic Expressions (E & G only) in Dependent Sources.
- Terms used in **.DEFMAC**, **.DEFWAVE** and **.EXTRACT** statements.
- Monte Carlo analysis distribution.

Multiple parameters can be set in a single **.PARAM** command and any combination of parameter types above may be used. There is no limit to the number of parameters that can be set.

**.DC** and **.STEP** statements may be used to define any parameter in the main circuit.

### Note



Parameters and expressions are not allowed in device names and nodes. In commands they are only allowed if explicitly specified in the documentation. Only one definition per parameter is allowed.

If a parameter is specified more than once in a netlist, Eldo will always use the last value given. This includes instances that occur earlier than the last **.PARAM**. For instance, if a netlist includes:

```
.param res=10
r1 4 2 'res'
.param res=13
```

then **r1** will take the value 13.

The **.PARAM** command is order dependent in that all components of any arithmetic expressions used must have been defined earlier in the netlist.

---

#### Note



If a parameter **P** is referred to in a netlist but not defined, Eldo searches for **P** in the **.LIB** files. Only global **.PARAM** definitions are considered. Parameter declarations within a **.SUBCKT** definition will not be considered outside that subcircuit. The **LOT** and **DEV** specifications affect only the parameter before them.

---

Expressions can be used in a netlist with certain restrictions. These expressions must be contained within braces **{ }**. Constants and parameters may be used in expressions, together with the built-in functions and operators.

The following keywords are special in that they may appear in expressions. However, they may not be specified in a **.PARAM** command if an RF analysis is specified in the netlist.

**Table 10-14. Reserved Keywords not available in .PARAM**

AMNOISE	BFACTOR	BOPT	FREQ	GA_mag	GA_dB
GAC	GAM_mag	GAM_dB	GAMMA_OPT	GAMMA_OPT_MAG	GASM_mag
GASM_dB	GAUM_mag	GAUM_dB	GOPT	GP_mag	GP_dB
GPC	INOISE	KFACTOR	LSC	MUFACTOR	NFMIN_mag
NFMIN_dB	ONOISE	PHI_OPT	PHNOISE	POWER	RNEQ
SCALE	SNF_mag	SNF_dB	SSC	TEMP	TGP_mag
TGP_dB	TIME	TNOM <sup>a</sup>	XAXIS		

a. **TNOM** may be specified as a parameter in a **.PARAM** command when **.OPTION DEFPTNOM** is set. The temperature value used by the Eldo model evaluator is always that which is set with **.OPTION TNOM=val**.

If an RF analysis is specified in the netlist, and if any **.PARAM** is named with one of these keywords, it will be rejected. For example, the following statement will generate an error:

```
.PARAM SCALE=VAL
```

## Parameters

- PAR=VAL  
Name and value of the parameter.
- PAR=EXPR  
Name of the parameter and regular arithmetic expression describing it. This expression may use parameter names already defined in the netlist, unless **LOT** or **DEV** are specified. Waves must not be part of the expression as the expression is evaluated prior to the simulation. The parameters can also depend on V or I.



For a list of the available arithmetic functions refer to “[Arithmetic Functions](#)” on page 3-7.

---

- PAR="NAME"  
Assigns a character string to the parameter. May be used to parametrize models and subcircuits. Eldo accepts quoted character strings as parameter values. These string values may be used for model names and filenames. To use a string as a parameter, enclose the string with double quotes, for example:

```
.param TT1="ResMod"
```

To maintain the case of the string enclose the string with single quotes first and then enclose with double quotes, for example:

```
.param TT2=' 'PwlModFile.src' '
```

The value of the string is retrieved simply by specifying the dollar sign (\$) and parentheses () .

See the examples on [page 10-211](#).

- PAR(a,b)=EXPR  
Specifies a user-defined function in order to define a parameter using an expression. The parameter may then be called when required, e.g.

```
.param P(a,b)=expression
      R1 1 2 'P(a,b)'
```

- PAR=VAL | PAR=EXPR  
+ **LOT** | **DEV** [ /**GAUSS** | /**UNIFORM** | /**USERDIST** ]=VAL |  
+ (dtype,-3sig,+3sig  
+ [,bi,-dz,+dz [,off,sv] [,scale]])

This parameter setting can only be used with a Monte Carlo analysis. For more information see “[.MC](#)” on page 10-147

**LOT | DEV=VAL**

Specifies a **LOT** or **DEV** tolerance value of VAL to the parameter for MC analysis. May be used in combination with **GAUSS**, **UNIFORM** or **USERDIST**

which specify Gaussian, uniform or user-defined distributions respectively. The value of VAL may be specified as a percentage using the % sign, or as an absolute value. Parametric expressions are allowed in the value of VAL, but not in the parameter defined after a **.PARAM** statement. **LOT** and **DEV** do not allow other parameters to be specified within parameters. For more information on **LOT** and **DEV**, see “[Tolerance Setting Using DEV, DEVX or LOT](#)” on page 10-150.

Different entities are able to share the same distribution. Anywhere Eldo accepts **LOT/DEV** specifications, you can specify **LOTGROUP=group\_name**.

If no distribution type is specified it will default to **UNIFORM**. For more information on user defined distributions see “[“DISTRIB”](#) on page 10-77.

**dtype**

**nor** for gaussian distribution.  
**uni** for uniform distribution.

**-3sig**

Lower 3 sigma bounds with respect to nominal value, this can be specified as a percentage or an absolute value.

**+3sig**

Upper 3 sigma bounds with respect to nominal value, this can be specified as a percentage or an absolute value.

**bi**

An optional pair of characters specifying that the distribution is bimodal.

**-dz**

Lower limit of the “dead zone” in bimodal distribution, this can be specified as a percentage or an absolute value.

**+dz**

Upper limit of the “dead zone” in bimodal distribution, this can be specified as a percentage or an absolute value.

**off**

An optional offset.

**sv**

A percentage or absolute value that moves the nominal value of a parameter either above or below the “typical” nominal value for that parameter.

**scale**

Specifies whether the calculations are to be held in log or linear scale. The two options are **lin** or **log**.

**Note**

1. The limits for this syntax for Accusim are -3sig, 3sig compared to those of Eldo which are -4sig, 4sig.
2. The minus sign in the values -3sig and -dz is only to specify that they are to the left of the nominal value. Eldo also accepts them as positive values.
3. sv can be > 0 which means a shift to the right, or < 0 which means a shift to the left.

- **LOTGROUP=**my\_lot\_group

Different entities are able to share the same distribution. Anywhere Eldo accepts **LOT/DEV** specifications, you can specify **LOTGROUP=**my\_lot\_group. Please refer to “[.LOTGROUP](#)” on page 10-142 for more information.

- **PAR=MC\_DISTRIBUTION**

When using a Monte Carlo analysis the same random variable will be used each time a parameter affects a model parameter (**LOT** variation). When a declared parameter affects an instance parameter a new random variable is calculated each time it is specified (**DEV** variation).

---

**Note**



It is possible to specify that **DEV** variation will be used for both model and instance parameters, as was default in Eldo versions prior to v6.3\_2, by specifying option **PODEV**. See [page 11-29](#) for more information.

---

A Monte Carlo distribution can be used to specify how the random variables should be distributed between its upper and lower limits. For more information on Monte Carlo analysis see [page 19-1](#).

**MC\_DISTRIBUTION** should be replaced with one of the following keyword statements:

---

**Note**



Parameter values can be specified with the percentage sign %. For example, the following two lines of syntax are equivalent:

```
.param rgauss=gauss(2,20%,1)
.param rgauss=gauss(2,0.2,1)
```

---

**UNIF**(nominal,relative\_variation,[mult])

Defines a uniform distribution. **PAR** can vary between nominal - nominal\*relative\_variation and nominal + nominal\*relative\_variation.

**AUNIF**(nominal,absolute\_variation,[mult])

Defines a uniform distribution. **PAR** can vary between nominal - absolute\_variation and nominal + absolute\_variation.

**GAUSS**(nominal,relative\_variation,sigcoef,[mult])

Defines Gaussian distribution. The standard deviation is equal to relative\_variation\*nominal/sigcoef.

**AGAUSS**(nominal,absolute\_variation,sigcoef,[mult])

Defines Gaussian distribution. The standard deviation is equal to absolute\_variation/sigcoef.

**Note**

The following two distribution statements are equivalent:

```
.PARAM P1=AGAUSS(2,0.3,3)
.PARAM P1=AGAUSS(2,0.6,6)
```

In both cases, the standard deviation will be 0.1.

**LIMIT**(nominal,absolute\_variation)

Outputs **PAR** - absolute\_variation or **PAR** + absolute\_variation depending on whether the random number, varying between -1 and 1, is negative or positive.

**nominal**

The nominal value.

**absolute\_variation**

Absolute value for variation of the nominal value.

**relative\_variation**

Relative value for variation of the nominal value.

**sigcoef**

Normalization coefficient.

**mult**

A positive integer value that acts as a multiplier to set how many times the parameter value is to be calculated. If it is greater than one then the distribution will be bimodal. The result could be either greater or lesser than the **nominal** value. The result with the largest deviation is then used. If **mult** is not specified it will default to 1.

- **DEVX=VAL**

The **DEVX** specification forces Eldo to use a new random value for each instance of a subcircuit. The difference with **DEV** is that even if a parameter is used several times in the same subcircuit, only one value will be used for that particular instance.

**Note**

**DEVX** can only be applied on **.PARAM**, unlike **LOT** and **DEV** which can be applied on both model parameters and **.PARAM** statements.

**Note**

It is impossible to have **DEV** and **DEVX** specified for the same parameter. If **DEV** and **DEVX** are specified for the same parameter, the last specification will be retained.

**Examples**

The following example shows how component values may be defined globally using the **.PARAM** command. Note that **LOT** and **DEV** values of 5% and 10% are also assigned to the parameter lval or MC analysis.

```
r1 1 2 rval
c1 1 2 cval
l1 1 2 lval
.param rval=2k cval=3p lval=2u lot=5% dev=10%
```

The following shows how parameters in a **.MODEL** definition may be assigned symbols which are then declared globally using the **.PARAM** command.

```
.model mod1 nmos level=3 vto=vtodef
*main circuit
m1 1 2 3 4 mod1 w=wdef l=ldef
.param vtodef=1 wdef=20u ldef=3u
```

The following shows arithmetic expressions and previously defined parameters combined in a **.PARAM** command.

```
r1 1 2 p2
.param p1=1k p3=2*p1
.param p2=sqrt(p1)+3*p3
```

The following example shows how parameters may be assigned symbols in a **.SUBCKT** definition. The parameters may then be given values explicitly when the subcircuit is called or globally using the **.PARAM** command.

```
*SUBCKT definition
.subckt inv 1 2
r1 1 3 rval
r2 3 4 rval1
r3 4 2 rval2
.ends inv
*subcircuit call
x1 1 2 inv rval=3 rval2=10
.param rval1=2
```

In the next example, the model name is substituted by the parameter **pmod**.

```
.param pmod="pmos1"
m1 d g s b $(pmod) w=1u l=1u
```

The following two examples show how string parameters are used.

```
.param MOD="Pmos1"
m1 d g s b $(MOD) w=1u l=1u

.param STIMFILE="'Stim.txt'"
v1 1 0 pwl file=$(STIMFILE) R
```

The following example shows **LOT** and **DEV** specifications, defined for the **.param** and **.model** commands:

```
.param p1=10k lot=(UNI,-5%,4%, bi, 3%, 4%)
.model QND NPN BF=100 dev=(NOR,5%,5%,bi,-2%,2%, off, 1%)
```

The following example shows how a user-defined function can be specified. In this case, the value produced for **R1** would be 6 ohms.

```
.param rval(a,b)=a+b
```

```
R1 1 2 'rval(2,4)'
```

In the next example the **DEVX** declaration, placed on parameter **pc**, indicates that during a Monte Carlo analysis, a new value of **pc** has to be randomly generated for X1, and another one has to be generated for X2:

```
.param pc=10p DEVX=10%
.SUBCKT cmod a b
C1 a b pc
C2 a b pc
.ENDS
X1 4 0 cmod 10p
X2 6 8 cmod 10p
```

X1.C1 and X1.C2 will use the same value (same for X2.C1 and X2.C2).

```
v1 1 0 dc 1 pw1 (0 1 10n 9)
r1 1 0 1
v2 2 0 dc 1
r2 2 3 r={p2} tcl=4.2
r3 3 0 1

.param p1=v(1)
.param p2=2*p1

.tran 1n 10n
.extract tran yval(v(3),4n)
.plot tran v(3)
.end
```

This example shows how parameter **p1** can depend on the voltage at **v(1)**.

The following example will return the linearly interpolated value of p2 (y value) at an x value of p1.

```
.param p1=1
.param p2=pwl(p1, 1, 0, 10, 0.5, 20, 1.5 ,30)
```

The value of p2 will be 25, because the linear interpolation between the 2nd and 3rd pairs of values gives 25.

If linear interpolation was not specified, then p2=20. For more information please refer to [page 3-9](#).

The following example shows **LOT** and **DEV** variation usage on MC distribution parameters.

```
.PARAM p1=UNIF(1,0.05)

.MODEL MOD1 NMOS VTO='1*p1'
.MODEL MOD2 PMOS VTO=' -1*p1'

M1 ... W='p1*1u'
M2 ... W='p1*1u'
```

The same random value (**LOT** variation) will be used for the calculation of **VTO** for both **NMOS** and **PMOS** models, since in this case **p1** is affecting model parameters. However independent random values (**DEV** variation) will be used for the MOS instances **M1** and **M2**,

since **p1** is affecting an instance parameter. Use option **PODEV** for backward compatibility for the mechanism in versions of Eldo prior to v6.3\_2 where **DEV** variation was used for both model and instance parameters.

### -compat flag

In **-compat** mode, double quotes are considered as single quotes. (In standard Eldo mode, double quotes are used to specify a parameter string.) Use option **QUOTSTR** to consider double quotes as a parameter string delimiter.

In **-compat** mode, Eldo will check whether there is a **.model** with the same name as the string after the nodes in a model declaration. If not, it will look for a parameter name. This can be confusing, because if there are both a **.model** and a parameter name with the same name, then the simulator will consider the string to be a model name, while a parameter name was desired. This can be overcome by placing the parameter name in single quotes in the instantiation.



Please refer to “[Compatibility Options](#)” on page 12-1 of the *Eldo User’s Manual* for further information on the **-compat** flag.

In this example a parameter, **rmin**, is required as the value of a resistor instantiation. It is placed in single quotes so that it is viewed as a parameter and not a model name.

```
.model rmin fmax=3 nonoise
.param rmin=2k
r1 3 2 'rmin'
```

## **.PART**

### Circuit Partitioning

```
.PART MACH | ELDO | ANALOG | MODSST
+ INST=(<instance_list>) SUBCKT=(<subcircuit_list>)
```

The **.PART** command instructs Eldo to use the specified algorithm in place of the regular transient algorithm, for a certain selection of instances.

The instances to be simulated with the specified algorithm may be listed explicitly, using the **<instance\_list>**. They may also be implicitly designated, using the **<subcircuit\_list>**.

### Parameters

The list below details which simulation algorithm will be used for the selected instances, depending on the keyword specified:

- **MACH**

Use the Mach TA algorithm.

- **ELDO**

Use the Eldo OSR algorithm.

- **ANALOG**

Specified for Eldo or Mach parts to be simulated as analog.

If subcircuit or instance is partitioned to Eldo, simulate using Newton. Only applicable if default Eldo algorithm is changed from Newton to OSR using **.OPTION OSR**.

If subcircuit or instance is partitioned to Mach, simulate with **Mach\_Analog** option.

- **MODSST**

Use the Eldo RF MODSST algorithm. For further details, see [page 2-22](#) of the *Eldo RF User's Manual*.

- **<instance\_list>**

This is enclosed in parenthesis, and contains a list of instance names, separated by commas, possibly using wildcards (\*) and (?) in place of characters.

- **<subcircuit\_list>**

This is enclosed in parenthesis, and contains a list of subcircuit names separated by commas. Subcircuit names may also contain wildcard characters (\*) and (?). If a subcircuit appears in the **<subcircuit\_list>**, all instances of this subcircuit will be handled with the specified algorithm.

### Example

Using these directives will have an effect on how and where simulation occurs, for example:

```
.part MACH SUBCKT= (NOR1, NAND3, AOIX*)
.part MACH INST=(XA1.XM2)
```

**.part** ANALOG INST=(XAMP,XAGC)

## **.PLOT**

### Plotting of Simulation Results

```

.PLOT [ANALYSIS] OVN [(LOW, HIGH)] [(VERSUS)]
+ {OVN [(LOW, HIGH)]} [UNIT=NAME] [(SCATTERED)] [STEP=value]
.PLOT AC|FSST S(i, j) [(SMITH[, zref])] [(POLAR)]
.PLOT FOUR FOURxx(label_name) [(SPECTRAL)]
.PLOT DSP DSPxx(label_name)
.PLOT EXTRACT [MEAS | SWEEP]
.PLOT [CONTOUR] MEAS(meas_name_x) MEAS(meas_name_y) [(SCATTERED)]
+ [(SMITH[, zref])] [(POLAR)]
.PLOT [ANALYSIS] TWO_PORT_PARAM [(SMITH[, zref])] [(POLAR)]

```

The **.PLOT** command takes its name from Berkeley SPICE (2G6). It is used to specify which simulation results have to be kept by the simulator for graphical viewing and post-processing.

### Files created by **.PLOT** Commands

The quantities listed in a **.PLOT** command are written to binary output files, and to the main *.chi* output file (ASCII) as well. The binary files (*.cou/.ext* or *.wdb*) are the input data for the waveform processors compatible with Eldo. The *.cou* format is the binary format for Xelga, whereas the *.wdb* format is the format used by EZwave. The *.ext* files are also binary files with the same format as the *.cou*. The ASCII outputs are still supported for SPICE compatibility reasons, although seldom used. These ASCII outputs are discussed in a specific section below. Saved windows files *.swd* are created with a *.wdb* database and contain information on specific graph windows such as size or position. They also contain the waveforms associated with graph windows. When specifying plots for frequency based analysis, the *.swd* file contains the information for complex waveforms, which can then be viewed by opening the *.swd* file with EZwave.

---

#### Note



Specify option **NOWAVECOMPLEX** (see [page 11-48](#)) to force complex waveform information to be stored within the waveform database (*.wdb*) as well as the *.swd* file. This restores the functionality of Eldo versions prior to v6.3\_2.1.

---

### Types of Waveforms

Many different simulation results can be created by Eldo. The simulator can output simple node voltages, but also currents through devices, currents through device or subcircuit pins, power quantities, S parameters, internal variables from device models, etc. All these results are direct *raw* results from a simulation. The tables shown in the next few pages indicate the exact syntax to use for each category. Example:

```

.PLOT TRAN V(OUT)
.PLOT DC ISUB(XBIAS.VOUT)

```

The **.PLOT** command may also be used to plot *extraction* results. A flexible language exists in Eldo to *extract* characteristics from raw simulation results (for example the maximum value

of a waveform, or the time at which a given threshold is crossed by a waveform, etc.). See the **.EXTRACT** command on page 10-95.

When extraction statements are combined with parameter sweeping statements (see the **.STEP** command on page 10-298), Eldo automatically creates waveforms showing the extraction results versus the swept parameter. For example, a user may extract the width of an output pulse from a transient simulation, and sweep the power supply level. In this case Eldo will automatically create a waveform showing the width of the pulse versus the power supply level. Similarly, if extractions and **.ALTER** statements are combined, Eldo will automatically create waveforms showing the extraction results versus the index of the **.ALTER** runs (see **.ALTER** command on page 10-22). In this case, the X axis will contain 1, 2, 3, etc. The initial display in the viewer can also be prepared using **.PLOT** commands (see the **.PLOT EXTRACT...** description page 10-223). Example:

```
.EXTRACT TRAN label=VMAX MAX(V(out))
.PARAM powersupply=1.2
VDD VDD 0 'powersupply'
.STEP PARAM powersupply list 1.2V 1.3V 1.4V 1.6 2V
.PLOT EXTRACT meas(VMAX) ! this will create a waveform
*                         showing VMAX(powersupply)
```

By default, using the *.wdb* format, the **.EXTRACT** waveforms are saved inside the *EXT* folder in the main *.wdb* file.

If using the *.cou* format, these **.EXTRACT** waveforms are created in a separate file with the *.ext* extension. This file can be loaded by Xelga. The command-line switch **-couext (eldo -couext -i <netlist.cir>)** can be used to merge the extract waveforms into the *.cou* file, when possible. In this case, a single *.cou* file will contain the *raw* waveforms and also the extraction waveforms.

Finally, the **.PLOT** command, combined with the **.DEFWAVE** command (see page 10-70) can also be used to generate so-called *template* waveforms, i.e. piece-wise-linear waveforms defined by a series of arbitrary (x,y) coordinates. Using **.PLOT**, these templates can be displayed together with the simulation results (this is very useful when verifying whether a filter response passes or not specifications such as cutoff frequencies, minimum attenuation etc.). Example:

```
.DEFWAVE FILTERSPEC=PWL(1e-3,0,100k,0,500k,-60,100G,-60)
.PLOT AC VDB(OUT) W(FILTERSPEC)
```

## Complex Modifiers and Initial Formatting

Some results are real quantities, such as the currents and voltages from a transient analysis, while others are complex. In this case, real or imaginary parts, magnitude or phase or group delay, can be required by the user.

It is also possible to specify some initial basic *formatting* of the results. For example, spectral or scattered display modes can be specified, instead of the default *continuous-line* mode. S, Y, Z

parameters can be automatically displayed in a Smith chart, etc. These optional specifications do not alter the raw data, they only tell the waveform viewer how to display the data initially.

## X axis of Waveforms

In general, the created waveforms have their X axis implicitly defined by the corresponding analysis. For example, the X axis of a transient waveform is the time, the X axis of an AC waveform is the frequency etc. The X axis for a transient waveform is defined by the **.TRAN** command, it ranges from  $t=0$  to  $t=t_{max}$  where  $t_{max}$  is specified in the **.tran** **tstep tmax** command (see [.TRAN command on page 10-318](#)). The X axis for an AC waveform is defined by the frequency points in the **.AC** command (see [page 10-12](#)). DC and NOISE waveforms also inherit their X axis from the corresponding analysis command (**.DC** or **.NOISE**).

For all analyses but the transient analysis, the X-axis range and the spacing of points in the X axis are thus predictable. For example, the frequency points in a **.AC** command are either listed explicitly or regularly spaced, in a predictable way. For transient analysis however, this is different. Eldo always uses a variable timestep algorithm, and the spacing of the timepoints where the circuit is solved is dictated by accuracy considerations only (see the [Speed and Accuracy chapter](#)). Thus the total number of timepoints is generally not predictable. By default, all computed timepoints are stored in the binary output files, so that any rapid change or *glitch* can be inspected visually in the waveform processor. However, when working with large circuits and/or long transient simulations and/or storing many waveforms, this may generate huge output files. The loading and post-processing of these huge output files by the waveform viewers may thus be slower. In extreme cases, this may also impact the simulation time, as writing gigabytes to a disk, possibly through a busy network, may take quite a while. The next section discusses several options meant to control the size of the transient output files.

In the case of **.EXTRACT** waveforms, the X axis is defined by the parameter sweeping statement (**.STEP**), or it contains integer indexes (**.ALTER**).

## Limiting the Size of Transient Output Files

At least three options are available in Eldo to limit the size of the output files for transient simulation. The first one is the option **OUT\_STEP=val**, the second one is option **OUT\_RESOL=resolution**, and the third one is option **INTERP=1**. Of course, the many options related to the accuracy settings do generally have an impact upon the number of computed points and thus upon the size of the output files, but this is only qualitative and indirect. There is no way to quantitatively relate, say the *eps* or the *reltol* specification, to the final size of the output files—this all depends on the circuit. In contrast, the options discussed in this section allow controlling the amount of data in a predictable way (the **OUT\_RESOL** option only sets a maximum size, whereas **OUT\_STEP** and **INTERP** allow exact predictions). These options may have a considerable impact on the overall simulation speed, so it is important to define exactly what has to be achieved.

- Option **OUT\_STEP**

The **.OPTION OUT\_STEP=val** command (see also [page 11-49](#)) forces a timepoint at each multiple of `val`, and only these timepoints are stored in the binary output files. These timepoints are forced, i.e. they come in addition to the normal timepoints picked by the simulator. This option has the same effect whether Eldo writes `.cou` or `.wdb` output. When using this option, the timepoints are forced, i.e. they are computed even if they are not required from an accuracy point of view. One of the applications of the **OUT\_STEP** option is for FFT computations. In this case, it is frequent that the user wants to have exact, computed points, at regularly spaced timepoints, to minimize the interpolation artifacts when computing an FFT. However the **OUT\_STEP** option is also a way to control the size of the output files, even if no FFT is scheduled. If not chosen properly, a possible risk of the **OUT\_STEP** option is to slowdown the simulation unnecessarily, by forcing Eldo to compute more points than needed. This is particularly true if they are long periods of time with little or no activity in the simulation. Forcing a short *out\_step* will force many useless timepoints during these periods, whereas Eldo would normally accelerate and compute fewer timepoints. This option is however preferred by some users because the output waveforms ultimately contain only computed points, without interpolation.

- Option **OUT\_RESOL**

Another option used to reduce the size of the output file is the **OUT\_RESOL** option. This option has the same effect whether Eldo writes `.cou` or `.wdb` output. With this option, the user can specify the smallest *resolution* of the output file (see **.OPTION OUT\_RESOL=resolution** on [page 11-49](#)). Computed data is then dumped only if the current time is greater than the previously written timepoint, augmented by the resolution. For example if **OUT\_RESOL** is set to 1ns, and the simulator has written data at time  $t=24.65\text{ns}$ , all timepoints computed until  $t=25.65\text{ns}$  will not be dumped. The first timepoint after  $t=25.65\text{ns}$  will be dumped. This could be  $t=25.76\text{ns}$  for example, or  $t=27\text{ns}$  if the simulator can *accelerate* in this period of time. No timepoints are forced other than those naturally picked by the simulator. Thus the spacing of timepoints in the output file is generally non-constant. The maximum average density of timepoints in the output file is determined by the *resolution* parameter of the option. The minimum density is not guaranteed, neither locally, nor globally. This option is an interesting alternative, combining the certainty that the points in the output are computed points only (no interpolation error), and the ability to handle low activity periods efficiently. It is however, generally not well suited for simulations where an FFT must be computed, because of the non-constant spacing of the timepoints (this would result in interpolation errors when re-sampling for the FFT).

- Option **INTERP**

Both **OUT\_STEP** and **OUT\_RESOL** options dump only computed points to the output file. Sometimes it may be desirable to obtain equally spaced time points, but forcing timepoints (with **OUT\_STEP** for example) would result in an unacceptable CPU penalty. In these cases, interpolating may be useful. An interpolation option is available to force Eldo to sample the output data along the `<tstep>` parameter of the **.TRAN** command (see [page 10-318](#)). If using the option **INTERP=1** (see

[page 11-27](#)), the transient output waveforms are interpolated, and the timepoints written to the file are aligned with multiples of `<tstep>` exactly. For example, if using both the following:

```
.tran 1ns 100ns
.option interp=1
```

the output waveforms will contain exactly 101 points (time 0 is always included), spaced by 1ns. Glitches shorter than 1ns may not be caught in the resulting waveform. This is a radical way to reduce the size of the binary output files (`.cou` or `.wdb`). However, the written data is obtained by interpolation, thus including an unpredictable amount of interpolation error. During low-activity periods, Eldo still picks the largest possible timesteps which will still allow to meet the accuracy requirements, and *fills* the output data with interpolated data (at no or little cost, as opposed to the `OUT_STEP` technique discussed previously).

## ASCII Outputs in the `.chi` File

The quantities listed in a `.PLOT` command are also written to the `.chi` output file. Symbols such as \*, +, etc. are used to identify each quantity, to create ASCII *waveforms* in the `.chi` output file. This used to be the only *graphical* output from SPICE, before waveform viewers such as Xelga and EZwave existed. This was intended for black and white line printers. These outputs, although ignored by most users, are still available by default, because many post-processing utilities have been written that expect this data to be present in the output file, and to respect the original SPICE formatting. In the case of large simulations, this may however create huge `.chi` output files. Thus, it is possible to tell the simulator to skip the creation of these ASCII waveforms and thus reduce the size of the `.chi` output file. To do so, users can specify the `-noascii` command-line flag when invoking Eldo (`eldo -noascii -i <netlist.cir>`), or include the option `NOASCII` in the netlist (see [page 11-44](#)).

### Note



For transient results, the spacing of these points in the ASCII *waveforms* is defined by the `<tstep>` parameter of the `.TRAN` command (data is interpolated). This is for original SPICE compatibility. For all other analyses, the points in the X axis are those specified by the corresponding analysis command.

## Wildcards

You can include wildcards in `.PLOT` commands. See [page 10-262](#) for more information on using wildcards with `.PROBE` commands.

Waveforms specified in previous `.PLOT` commands are excluded from the wildcard plot.

## Related Commands

See the `.PRINT` and `.PROBE` commands.

## Parameters

- ANALYSIS

Can be one of the following:

### **DC**

Specifies that the plots are required for a DC analysis. Provides compatibility with SPICE.

### **AC**

Specifies that the plots are required for an AC analysis. Provides compatibility with SPICE.

### **TRAN**

Specifies that the plots are required for a transient analysis. Provides compatibility with SPICE.

### **NOISE**

Specifies that the plots are required for a noise analysis. Provides compatibility with SPICE.

**SSTAC|SSTXF|SSTNOISE|SSTJITTER|TMODSST|FMODSST|FOURMODSST|TSST|FSST|SSTSTABIL**

Please refer to the [Display Command Syntax](#) chapter of the *Eldo RF User's Manual* for more information regarding these RF options.

### **OPT**

Extraction during optimization analysis. Can only be used with wave type **WOPT**. This wave type is used for optimization analysis only, i.e. it may only be used with **OPT**. For more information on wave type **WOPT** see [page 20-42](#).

- **FOUR**

Displays FFT results. Please see **FOURxx(label\_name)** below for display options.

- **DSP**

Displays DSP results. Please see **DSPxx(label\_name)** below for display options.

- **OVN**

Requests plotting of values of specific nodes on components. The syntax for specifying the list of plot specifications to be monitored is provided on [page 10-224](#).

- **FOURxx(label\_name)**

Should be specified as part of **OVN**. Displays FFT results. **xx** stands for **DB**, **M**, **P**, **R**, **I**, **GD**:

**DB** Magnitude, in dB

**M** Magnitude

**P** Phase

**R** Real part

**I** Imaginary part

**GD** Group Delay

- **DSP<sub>xx</sub>(label\_name)**

Should be specified as part of **OVN**. Displays DSP results. **xx** stands for **DB**, **M**, **P**, **R**, **I**, **GD** (see above).

- **LOW, HIGH**

The optional plot limits **LOW** and **HIGH** may be specified for each of the output variables. All output variables of the same kind (voltage for instance) to the left of a pair of plot limits (**LOW**, **HIGH**) will be plotted using the same lower and upper bounds. If plot limits are not specified, Eldo uses the following default values:

- 0V and 5V for **LOW** and **HIGH** values respectively in voltage plots
- -100dB and 100dB for **LOW** and **HIGH** values respectively in dB plots
- -180 and 180 degrees for **LOW** and **HIGH** values respectively in phase plots

- **UNIT=NAME**

Displays user-defined text representing the Y-axis units of a plot. Useful in conjunction with the **.DEFWAVE** command and with Eldo-FAS.

- **(VERSUS)**

Specifies the waveform viewer plot one, or a number of waves on the Y-axis against a single wave on the X-axis. **(VERSUS)** must be preceded by at least one wave and followed by one wave only.

- **(SCATTERED)**

Only computed points will be represented (no “line” between two successive points). This property is active for all waves of the graph.

- **STEP=value**

Performs a sampling of the waveform(s). A point is dumped every **value**. It can only be specified for databases that support multiple x-axes in the same simulation, e.g. JWDB. Other databases will ignore this parameter.

- **S(i, j)**

Smith chart specification of the S parameter  $S_{ij}$ .

- **(SMITH[ ,zref ] )**

Prompts the waveform viewer to display the waves which are given in the corresponding **.PLOT** command in a Smith chart. Waves must be given without any AC extension, otherwise Eldo will issue an error. The **SMITH** keyword can only be used in conjunction with Smith chart specifications above. **zref** is optional and specifies the impedance.

- **(POLAR )**

Prompts the waveform viewer to display the waves which are given in the corresponding **.PLOT** command in a Polar chart.

- **(SPECTRAL)**

Specifies the waveform viewer to represent the wave in its spectral representation. This keyword must be placed at the end of the command, and be enclosed in parentheses. This works only for **.PLOT** commands specified with the **FOUR** parameter.

- **EXTRACT**

Allows plot combinations in *.ext* file. This is to be used in conjunction with **.ALTER/.STEP/.TEMP** to organize extracted waves in graphs. For example, if you have:

```
.EXTRACT LABEL = a <expression>
.EXTRACT LABEL = b <expression>
```

use **.PLOT EXTRACT MEAS(a) MEAS(b)** if both waves are expected in the same plot on the waveform viewer invocation.

- **CONTOUR**

Eldo will plot the second measurement value with respect to the first. The wave will be displayed in a Smith chart if keyword **SMITH** is specified. **CONTOUR** information is written to the *.ext* file.

---

**Note**



The keyword **VECT** or **CATVEC** must have been specified on the **.EXTRACT** which is referred to in the **.PLOT CONTOUR**, otherwise the corresponding data will be reduced to a single print. The two measurements in the **CONTOUR** must have the same dimension. If not, the **.PLOT CONTOUR** will be ignored. **.PLOT CONTOUR** information is written to the *.ext* file

---

- **TWO\_PORT\_PARAM**

This must be replaced by one of the following keywords:

**Table 10-15. Two-port Parameter Keywords**

<b>BFACTOR</b>	<b>BOPT</b>	<b>GA_mag</b>	<b>GA_dB</b>	<b>GAC</b>
<b>GAM_mag</b>	<b>GAM_dB</b>	<b>GAMMA_OPT</b>	<b>GAMMA_OPT_MAG</b>	<b>GASM_mag</b>
<b>GASM_dB</b>	<b>GAUM_mag</b>	<b>GAUM_dB</b>	<b>GOPT</b>	<b>GP_mag</b>
<b>GP_dB</b>	<b>GPC</b>	<b>KFACTOR</b>	<b>LSC</b>	<b>MUFACTOR</b>
<b>NC</b>	<b>NFMIN_mag</b>	<b>NFMIN_dB</b>	<b>PHI_OPT</b>	<b>RNEQ</b>
<b>SSC</b>	<b>TGP_mag</b>	<b>TGP_dB</b>	<b>YOPT</b>	

---

**Note**



All of these keywords may appear in expressions. However, they may not be specified in a **.PARAM** command if an RF analysis is specified in the netlist. These keywords are all available with both AC and FSST results.

---

## Plot Specifications (OVN)

- **NOISE(element\_name)**

Should be specified as part of OVN. Specifies that the plots are required for a noise analysis of a specific component or subcircuit instance. Multiple occurrences of this parameter can appear in a single line of syntax.

- **INOISE**

Should be specified as part of OVN. Input noise when performing a noise analysis.

- **ONOISE**

Should be specified as part of OVN. Output noise when performing a noise analysis.

### **Note**



**NOISE(element\_name)**, **INOISE** or **ONOISE** can appear singularly or as part of any combination. It is necessary that at least one be specified.

**INOISE** and **ONOISE** may appear in expressions. However, they may not be specified in a **.PARAM** command.



For more information on the use of **INOISE** and **ONOISE**, see the example on [page 10-180](#) and also the description on [page 10-236](#).

- **FOURxx(label\_name)**

Displays FFT results. Should be specified as part of OVN. **xx** can be **DB**, **M**, **P**, **R**, **I**, **GD**:

**DB** Magnitude, in dB  
**M** Magnitude  
**P** Phase  
**R** Real part  
**I** Imaginary part  
**GD** Group Delay

- **DSPxx(label\_name)**

Displays DSP results. Should be specified as part of OVN. **xx** can be **DB**, **M**, **P**, **R**, **I**, **GD** (see meanings above).

- **LSTB\_xx**

Displays loop stability analysis (LSTB) results. Should be specified as part of OVN. **xx** can be **DB**, **M**, **P**, **R**, **I**, **GD** (see meanings above).

- **VDIG(node\_name)**

Should be specified as part of OVN. Enables the plotting of an analog signal as a digital bus. See the appropriate section on [page 10-246](#) for further details.

- **WXxx**(devname.posi)

Should be specified as part of OVN. Returns the value of complex waveforms defined by the **.DEFWAVE** command or implicit declarations such as **W('P(parameter\_value)')** and **W('wave\_dependent\_expression')**. **WXxx** can be: **W**, **WDB**, **WP**, **WR**, **WI**, **WM**, **WGD**, or **WOPT**.

- **VXxx**(devname.posi)

Should be specified as part of OVN. Returns the voltage on the pin. **VXxx** can be: **VX**, **VXDB**, **VXP**, **VXR**, **VXI**, **VXM**, or **VXGD**.

- **IXxx**(devname.posi)

Should be specified as part of OVN. Returns the current on the pin. **IXxx** can be: **IX**, **IXDB**, **IXP**, **IXR**, **IXI**, **IXM**, or **IXGD**. The current is positive when it enters the object by this pin.

devname

Device name.

posi

Position of the pin as given in the **.cir** file. **posi** can be greater than the number of external pins. In such a case, internal pin values are given. Internal pins are sorted in the same order as external pins (5 on a BJT for instance, refers to the 5th pin, which is the 1st internal pin, which is the internal collector). The case for GUDM models (Mextram, HICUM) is different, because the intrinsic structure of such models is not known to Eldo. The internal pins can be sorted in any order, therefore the value of intrinsic pins for GUDM models can be accessed only by the writer of the model.

- **DATA**(dataname, parameter)

Should be specified as part of OVN. Allows to plot a wave defined with a **.DATA** statement. **dataname** is the name assigned to the **.DATA** statement. It is mostly used to define fitting waves during optimization.

**Note**



When the **-compat** flag is set, then the following apply: For R/L/C/E/F/G/H/I/V/D devices, **I2** returns the same value as **I1**.

For M/B/J devices, **I3** is positive when current leaves the object by pin number 3.

Below is a list of devices and the syntax used to plot or print the values of both their extrinsic and intrinsic pins. The information is divided first by device and then into subsections of output type and analysis type.

For devices, if the position number specified is greater than the index available for the device, it will return a value of zero.

The plot specifications must match the corresponding analysis type, i.e. **AC**, **DC**, **TRAN**, or **NOISE**.



Examples of syntax usage can be found on [page 10-246](#).

**Table 10-16. MOSFET Plotting and Printing**

<b>DC or Transient Analysis</b>	
<b>Currents</b>	
ID(Mxx)/I(Mxx.D)/IX(Mxx.1)	Drain current
IG(Mxx)/I(Mxx.G)/IX(Mxx.2)	Gate current
IS(Mxx)/I(Mxx.S)/IX(Mxx.3)	Source current
IB(Mxx)/I(Mxx.B)/IX(Mxx.4)	Bulk current
IBD(Mxx)	Bulk-drain diode current
IBS(Mxx)	Bulk-source diode current
<b>Voltages</b>	
VD(Mxx)/VX(Mxx.1)	Drain voltage, in Volts
VG(Mxx)/VX(Mxx.2)	Gate voltage, in Volts
VS(Mxx)/VX(Mxx.3)	Source voltage, in Volts
VB(Mxx)/VX(Mxx.4)	Bulk voltage, in Volts
VGS(Mxx)	Gate-source voltage, in Volts
VGB(Mxx)	Gate-bulk voltage, in Volts
VDS(Mxx)	Drain-source voltage, in Volts
VBS(Mxx)	Bulk-source voltage, in Volts
VBD(Mxx)	Bulk-drain voltage, in Volts
VT(Mxx)	Threshold voltage value, in Volts
VDSS(Mxx)	Saturation voltage value, in Volts
<b>Conductances</b>	
GDS(Mxx)	$\frac{\partial I_D}{\partial V_{DS}}$
GM(Mxx)	Transconductance: $\frac{\partial I_D}{\partial V_{GS}}$

**Table 10-16. MOSFET Plotting and Printing**

GMBS(Mxx)	$\frac{\partial I_D}{\partial V_{BS}}$
GMIBD(Mxx)	Mosfet diode drain conductances
GMIBS(Mxx)	Mosfet diode source conductances
<b>Capacitances</b>	
CGS(Mxx)	$\frac{\partial QG}{\partial V_S}$ Gate/Source capacitance
CGD(Mxx)	$\frac{\partial QG}{\partial V_D}$ Gate/Drain capacitance
CGG(Mxx)	$\frac{\partial QG}{\partial V_G}$ Gate/Gate capacitance
CGB(Mxx)	$\frac{\partial QG}{\partial V_B}$ Gate/Bulk capacitance
CBS(Mxx)	$\frac{\partial QB}{\partial V_S}$ Bulk/Source capacitance
CBD(Mxx)	$\frac{\partial QB}{\partial V_D}$ Bulk/Drain capacitance
CBG(Mxx)	$\frac{\partial QB}{\partial V_G}$ Bulk/Gate capacitance
CBB(Mxx)	$\frac{\partial QB}{\partial V_B}$ Bulk/Bulk capacitance
CBSJ(Mxx)	$\frac{\partial QB}{\partial V_{Bs}}$ Bulk/Source junction diode capacitance
CBDJ(Mxx)	$\frac{\partial QB}{\partial V_{Bd}}$ Bulk/Drain junction diode capacitance
CDS(Mxx)	$\frac{\partial QD}{\partial V_S}$ Drain/Source capacitance

**Table 10-16. MOSFET Plotting and Printing**

CDD(Mxx)	$\frac{\partial QD}{\partial Vd}$ Drain/Drain capacitance
CDG(Mxx)	$\frac{\partial QD}{\partial Vg}$ Drain/Gate capacitance
CDB(Mxx)	$\frac{\partial QD}{\partial Vb}$ Drain/Bulk capacitance
CSS(Mxx)	$\frac{\partial QS}{\partial Vs}$ Source/Source capacitance
CSD(Mxx)	$\frac{\partial QS}{\partial Vs}$ Source/Drain capacitance
CSG(Mxx)	$\frac{\partial QS}{\partial Vg}$ Source/Gate capacitance
CSB(Mxx)	$\frac{\partial QS}{\partial Vb}$ Source/Bulk capacitance
<b>Charges<sup>a</sup></b>	
QG(Mxx)	Gate charge
QB(Mxx)	Bulk charge
QD(Mxx)	Drain charge
QS(Mxx)	Source charge
QBD(Mxx)	Bulk-drain junction charge
QBS(Mxx)	Bulk-source junction charge
<b>Power</b>	
POW(Mxx)	Power curve calculated as the product of the drain to source voltage and the drain to source current ( $Vds \times Ids$ )
<b>AC Analysis</b>	
<b>Currents<sup>b</sup></b>	
Izz(Mxx.D)	Drain current. See below for meaning of zz
Izz(Mxx.G)	Gate current. See below for meaning of zz
Izz(Mxx.S)	Source current. See below for meaning of zz

**Table 10-16. MOSFET Plotting and Printing**

Izz(Mxx.B)	Bulk current. See below for meaning of zz
<b>Noise Analysis</b>	
IBSNOISE(Mxx)	Base-Source diode noise: shot noise due to IBD
IBDNOISE(Mxx)	Base-Drain diode noise: shot noise due to IBS
RSNOISE(Mxx)	Source access resistor noise
RDNOISE(Mxx)	Drain access resistor noise
RGNOISE(Mxx)	Gate access resistor noise (for MOS model with RG resistance)
THNOISE(Mxx)	Thermal noise
FLKNOISE(Mxx)	Flicker noise
<b>Instance Parameters</b>	
EVAL(Mxx, Leff)	Effective length
EVAL(Mxx, Weff)	Effective width
EVAL(Mxx, ADeff)	Effective drain area
EVAL(Mxx, ASeff)	Effective source area
EVAL(Mxx, PDeff)	Effective drain perimeter
EVAL(Mxx, PSeff)	Effective source perimeter
EVAL(Mxx, Geo)	The GEO parameter (geometry selector)
EVAL(Mxx, RDeff)	Effective drain series resistance
EVAL(Mxx, RSeff)	Effective source series resistance
<b>Special Outputs in the .chi File</b>	
VTH_D	Vgs - Vth

- a. Available for charge controlled models only.
- b. In the above specifications, zz is to be replaced by one of the following:  
DB (Magnitude, in dB); M (Magnitude); P (Phase); R (Real part); I (Imaginary part); GD (Group delay).

**Table 10-17. BJT Plotting and Printing**

<b>DC or Transient Analysis</b>	
<b>Currents</b>	
IB(Qxx)/I(Qxx.B)/I(Qxx.1)	Base current
IC(Qxx)/I(Qxx.C)/I(Qxx.2)	Collector current
IE(Qxx)/I(Qxx.E)/I(Qxx.3)	Emitter current

**Table 10-17. BJT Plotting and Printing**

IS(Qxx)/I(Qxx.S)/I(Qxx.4)	Substrate current
<b>Voltages</b>	
VBEI(Qxx)	Internal node voltage difference base-emitter
VBCI(Qxx)	Internal node voltage difference base-collector
VCEI(Qxx)	Internal node voltage difference collector-emitter
VBE(Qxx)	External node voltage difference base-emitter
VBC(Qxx)	External node voltage difference base-collector
VCE(Qxx)	External node voltage difference collector-emitter
<b>Conductances</b>	
GM(Qxx)	Transconductance: $\frac{\partial I_C}{\partial V_{BE}}$
<b>Capacitances</b>	
CBS(Qxx)	Base/Substrate capacitance (LPNP only)
CBSX(Qxx)	Extrinsic Base/Substrate capacitance (LPNP only)
CBX(Qxx)	Extrinsic Base/Collector capacitance
CCS(Qxx)	Collector/Substrate capacitance
CMU(Qxx)	Base/Collector capacitance
CPI(Qxx)	Base/Emitter capacitance
<b>Resistances</b>	
RO(Qxx)	Resistance between internal nodes C and E
RPI(Qxx)	Resistance between internal nodes B and E
RX(Qxx)	Series resistance between internal and external Base nodes (Base Resistance)
RBB(Qxx)/RX(Qxx)	Base resistance value
<b>Power</b>	
POW(Qxx)	Power curve calculated as the product of the collector to emitter voltage and the collector to emitter current ( $VCE \times ICE$ )
<b>Frequency</b>	
FT(Qxx)	Frequency at which small-signal forward current transfer ratio extrapolates to unity

**Table 10-17. BJT Plotting and Printing**

<b>AC Analysis</b>	
<b>Currents<sup>a</sup></b>	
Izz(Qxx.C)	Collector current. See below for meaning of zz
Izz(Qxx.B)	Base current. See below for meaning of zz
Izz(Qxx.E)	Emitter current. See below for meaning of zz
Izz(Qxx.S)	Substrate current. See below for meaning of zz
<b>Noise Analysis</b>	
ICNOISE(Qxx)	Shot IC noise
IBNOISE(Qxx)	Shot IB noise
THNOISE(Qxx)	Thermal noise
FLKNOISE(Qxx)	Flicker noise
RBNOISE(Qxx)	Thermal noise due to access resistance RB
RENOISE(Qxx)	Thermal noise due to access resistance RE
RCNOISE(Qxx)	Thermal noise due to access resistance RC

a. In the above specifications, zz is to be replaced by one of the following:  
DB (Magnitude, in dB); M (Magnitude); P (Phase); R (Real part); I (Imaginary part); GD (Group delay).

**Table 10-18. JFET Plotting and Printing**

<b>DC or Transient Analysis</b>	
<b>Currents</b>	
ID(Jxx)/I(Jxx.D)/I(Jxx.1)	Drain current
IG(Jxx)/I(Jxx.G)/I(Jxx.2)	Gate current
IS(Jxx)/I(Jxx.S)/I(Jxx.3)	Source current
<b>Voltages</b>	
VD(Jxx)/V(Jxx.D)/V(Jxx.1)	Drain voltage, in Volts
VG(Jxx)/V(Jxx.G)/V(Jxx.2)	Gate voltage, in Volts
VS(Jxx)/V(Jxx.S)/V(Jxx.3)	Source voltage, in Volts
VGS(Jxx)	Gate-source voltage, in Volts
VDS(Jxx)	Drain-source voltage, in Volts
VT(Jxx)	Threshold voltage value, in Volts
VDSS(Jxx)	Saturation voltage value, in Volts

**Table 10-18. JFET Plotting and Printing**

<b>Conductances</b>	
GDS(Jxx)	$\frac{\partial I_D}{\partial V_{DS}}$
GM(Jxx)	Transconductance: $\frac{\partial I_D}{\partial V_{GS}}$
<b>Capacitances</b>	
CGD(Jxx)	$\frac{\partial QG}{\partial Vd}$ Gate/Drain capacitance
CGS(Jxx)	$\frac{\partial QG}{\partial Vs}$ Gate/Source capacitance
<b>AC Analysis</b>	
<b>Currents<sup>a</sup></b>	
Izz(Jxx.D)	Drain current. See below for meaning of zz
Izz(Jxx.G)	Gate current. See below for meaning of zz
Izz(Jxx.S)	Source current. See below for meaning of zz
<b>Noise Analysis</b>	
RGNOISE(Jxx)	Gate access resistor noise (for JFET MODEL with RG resistance)

a. In the above specifications, zz is to be replaced by one of the following:

DB (Magnitude, in dB); M (Magnitude); P (Phase); R (Real part); I (Imaginary part); GD (Group delay).

**Table 10-19. Diode Plotting and Printing**

<b>DC or Transient Analysis</b>	
<b>Currents</b>	
IX(Dxx.1)/I(Dxx.POS)/ ID(Dxx)/I(Dxx)	Positive output
IX(Dxx.2)/I(Dxx.NEG)	Negative output
<b>Conductances</b>	
GD(Dxx)	Diode conductance
<b>Capacitances</b>	
CD(Dxx)	Diode capacitance

**Table 10-19. Diode Plotting and Printing**

<b>Noise Analysis</b>	
FLKNOISE(Dxx)	Flicker noise
RSNOISE(Dxx)	Thermal noise due to access resistance
IDNOISE(Dxx)	Shot noise

**Table 10-20. Common Plotting and Printing**

<b>DC or Transient Analysis</b>	
FLUX(Lxx)	Returns the flux through the inductor Lxx.
ISUB(Xxx.N)	Current flowing into pin N of subcircuit Xxx, where N is the name of the pin in the .SUBCKT statement. The node name can be a node defined in a .global command.
I(dipole_xx[, dipole_yy])	Current difference between two-pin components dipole_xx and dipole_yy of any type (such as resistor, capacitor, source). If dipole_yy and the comma are omitted, the current through dipole_xx is printed. Current is positive when flowing from pin1 to pin2.
I(OPAxx.xyz) <sup>a</sup>	Current flowing into pin xyz, where xyz may be either POS, NEG, CP1 or CN1 for positive output, negative output, positive control, or negative control respectively.
IX(Xxx.N)	Current flowing into the Nth pin of subcircuit Xxx, where N is the order of the pin in the .SUBCKT statement. i.e. IX(XA.1) is the current flowing into the first pin of subcircuit XA. If N is the name of a global node, then Eldo will return the current which enters the X-instance by that global pin.
VX(Xxx.N)	Voltage at the Nth pin of subcircuit Xxx, where N is the order of the pin in the .SUBCKT statement. Works in the same way as IX(Xxx.N) above.
I(dev1[, dev2])	Specifies the current between dev1 and dev2. If dev2 and the preceding comma is omitted, ground is assumed.
V(N1[, N2])	Specifies the voltage difference between nodes N1 and N2. If N2 and the preceding comma is omitted, ground is assumed. In the case of subcircuit nodes, N1 has the form (Xnn.N1) where Xnn is the subcircuit instance.
W(WNAME EXPR)	The value of a waveform WNAME which has been created using the .DEFWAVE command, or the value of the waveform defined by the expression EXPR. .PRINT TRAN W(wave1) or .PRINT TRAN W('V(s)-V(a)')
POW(Xinstance_name) <sup>b</sup>	Returns the power curve dissipated in the Xinstance_name.

**Table 10-20. Common Plotting and Printing**

<b>POWER</b>	Returns the power dissipated in the entire design. This number is the sum for the power dissipated in R, E, H, I, M, B, D, J elements exclusively.
<b>AC Analysis</b>	
<b>VDB(N1[, N2])</b>	Specifies the magnitude of the voltage difference in dB between nodes N1 and N2. If only N1 is specified, ground is assumed for the second node.
<b>VGD(N1[, N2])</b>	Specifies the group delay (derivative of the phase with respect to the frequency) of the voltage difference between nodes N1 and N2. If only N1 is specified, ground is assumed for the second node. VT may be specified instead of VGD, it is equivalent.
<b>VI(N1[, N2])</b>	Specifies the imaginary part of the voltage difference between nodes N1 and N2. If only N1 is specified, ground is assumed for the second node.
<b>VM(N1[, N2])</b>	Specifies the magnitude of the voltage difference between nodes N1 and N2. If only N1 is specified, ground is assumed for the second node.
<b>VP(N1[, N2])</b>	Specifies the phase of the voltage difference between nodes N1 and N2. If only N1 is specified, ground is assumed for the second node.
<b>VR(N1[, N2])</b>	Specifies the real part of the voltage difference between nodes N1 and N2. If only N1 is specified, ground is assumed for the second node.
<b>IDB(dipole_xx[, dipole_yy])</b>	Specifies the magnitude, in dB, of the current difference between dipole_xx and dipole_yy. If only dipole_xx is specified, the current through dipole_xx is printed.
<b>IGD(dipole_xx[, dipole_yy])</b>	Specifies the group delay (derivative of the phase with respect to the frequency) of the current difference between dipole_xx and dipole_yy. If only dipole_xx is specified, the group delay of the current through dipole_xx is printed. IT may be specified instead of IGD, it is equivalent.
<b>II(dipole_xx[, dipole_yy])</b>	Imaginary part of the current difference between dipole_xx and dipole_yy. If only dipole_xx is specified, the imaginary part of the current through dipole_xx is printed.
<b>IM(dipole_xx[, dipole_yy])</b>	Magnitude of the current difference between dipole_xx and dipole_yy. If only dipole_xx is specified, the magnitude of the current through dipole_xx is printed.

**Table 10-20. Common Plotting and Printing**

IP(dipole_xx[, dipole_yy])	Phase of the current difference between dipole_xx and dipole_yy. If only dipole_xx is specified, the phase of the current through dipole_xx is printed.
IR(dipole_xx[, dipole_yy])	Real part of the current difference between dipole_xx and dipole_yy. If only dipole_xx is specified, the real part of the current through dipole_xx is printed.
ISUBDB(Xxx.N)	Magnitude, in dB, of the current flowing into pin N of subcircuit Xxx, where N is the name of the pin in the .SUBCKT statement.
ISUBGD(Xxx.N)	Group delay of the current flowing into pin N of subcircuit Xxx, where N is the name of the pin in the .SUBCKT statement.
ISUBI(Xxx.N)	Imaginary part of the current flowing into pin N of subcircuit Xxx where N is the name of the pin in the .SUBCKT statement.
ISUBM(Xxx.N)	Magnitude of the current flowing into pin N of subcircuit Xxx where N is the name of the pin in the .SUBCKT statement.
ISUBP(Xxx.N)	Phase of the current flowing into pin N of subcircuit Xxx where N is the name of the pin in the .SUBCKT statement.
ISUBR(Xxx.N)	Real part of the current flowing into pin N of subcircuit Xxx where N is the name of the pin in the .SUBCKT statement.
IXDB(Xxx.N)	Magnitude in dB, of the current flowing into the Nth pin of subcircuit Xxx, where N is the order of the pin in the .SUBCKT statement. If N is the name of a global node, then Eldo will return the magnitude in dB of the current which enters the X-instance by that global pin.
IXGD(Xxx.N)	Group delay of the current flowing into the Nth pin of subcircuit Xxx, where N is the order of the pin in the .SUBCKT statement. If N is the name of a global node, then Eldo will return the group delay of the current which enters the X-instance by that global pin.
IXI(Xxx.N)	Imaginary part of the current flowing into the Nth pin of subcircuit Xxx. If N is the name of a global node, then Eldo will return the imaginary part of the current which enters the X-instance by that global pin.
IXM(Xxx.N)	Magnitude of the current flowing into the Nth pin of subcircuit Xxx. If N is the name of a global node, then Eldo will return the magnitude of the current which enters the X-instance by that global pin.

**Table 10-20. Common Plotting and Printing**

<b>IXP(Xxx.N)</b>	Phase of the current flowing into the Nth pin of subcircuit Xxx. If N is the name of a global node, then Eldo will return the phase of the current which enters the X-instance by that global pin.
<b>IXR(Xxx.N)</b>	Real part of the current flowing into the Nth pin of subcircuit Xxx. If N is the name of a global node, then Eldo will return the real part of the current which enters the X-instance by that global pin.
<b>SDB(i, j)</b>	Magnitude in dB, of S parameter Si j.
<b>SR(i, j)</b>	Real part of the S parameter Si j.
<b>SI(i, j)</b>	Imaginary part of the S parameter Si j.
<b>WDB(WNAME)</b>	Magnitude in dB, of the waveform WNAME created using a .DEFWAVE command.
<b>WI(WNAME)</b>	Imaginary part of the waveform WNAME created using a .DEFWAVE command.
<b>WM(WNAME)</b>	Magnitude of the waveform WNAME created using a .DEFWAVE command.
<b>WP(WNAME)</b>	Phase of the waveform WNAME created using a .DEFWAVE command.
<b>WR(WNAME)</b>	Real part of the waveform WNAME created using a .DEFWAVE command.
<b>Noise Analysis</b>	
<b>INOISE<sup>c</sup></b>	Linear input noise when performing a Noise analysis.
<b>ONOISE</b>	Linear output noise when performing a Noise analysis.
<b>DB(INOISE)</b>	Input noise in dB when performing a Noise analysis.
<b>DB(ONOISE)</b>	Output noise in dB when performing a Noise analysis.
<b>NOISE(compx)</b>	Noise contribution of component compx to the total output noise of the circuit.
<b>NOISE NC<sup>d</sup></b>	Noise circle for a given value of a Noise Figure (SNF_val). When this value is not specified, the circle is plotted for a Noise Figure value corresponding to the actual circuit

- a. In the MOS and bipolar current print commands, the current is positive when it enters the device.
- b. In all cases, the power stored in capacitances is ignored, only dissipated power is taken into account. Power is measured only in DC and TRANSIENT analyses.
- c. For more information on the use of INOISE and ONOISE, see the example on page 10-180.
- d. For more information on NOISE NC see “[Two-port Noise Circles](#)” on page 10-243.

## Element Output

`LVnn(Ex)` or `LXnn(Ex)`, where:

- `LV` This formulation is used to obtain user-input parameters.
- `LX` This formulation is used to obtain computed values such as charges, capacitances, and derivatives.
- `nn` Index to select the appropriate output.
- `Ex` Name of the element.

**Note**



In the table below, the `LX` values which correspond to voltage (such as `LX0` for diodes) are computed using the intrinsic nodes, not the extrinsic nodes (the behavior in Eldo releases pre-v5.9).

---

**Table 10-21. Element Output**

<b>Resistors</b>	
LV1	Conductance at analysis temperature
LV2	Resistance at reference temperature
LV3	First temperature coefficient TC1
LV4	Second temperature coefficient TC2
<b>Capacitors</b>	
LV1	Computed capacitance
LX0	Charge stored in capacitor
LX1	Current flowing through capacitor
LX2	Voltage across capacitor
LX3	Capacitance value
<b>Inductors</b>	
LV1	Computed inductance
LX0	Flux in the inductor
LX1	Voltage across inductor
LX2	Current flowing through inductor
LX4	Inductance value

**Table 10-21. Element Output**

<b>Voltage-Controlled Voltage Sources</b>	
LX0	Source voltage (Vdip)
LX1	Current through source (I)
<b>Current-Controlled Current Sources</b>	
LX0	Current through source (I)
<b>Current-Controlled Voltage Sources</b>	
LX0	Source voltage (Vdip)
LX1	Source current (I)
<b>Diodes</b>	
LV1	Diode area factor
LX0	DC/transient voltage across diode (Vdip)
LX1	Current through diode (I)
LX2	Equivalent conductance (GD)
LX3	Charge of diode capacitor (Q)
LX5	Total diode capacitance (CD)
<b>BJTs</b>	
LV1	Area factor
LV5	FT
LV8	LOG10(IC)
LV9	LOG10(IB)
LV10	BETA
LV11	LOG10(BETA) Current
LX0	VBE
LX1	Base-collector voltage
LX2	Collector current
LX3	Base current
LX4	Gpi = Ib/Vbe, constant vbc
LX5	Gmu = Ib/Vbc, constant vbe
LX6	Gm = Ic/Vbe, constant vbc
LX7	GO = Ic/Vce, constant vbe
LX19	cbe capacitance (Cpi)

**Table 10-21. Element Output**

LX20	cbc capacitance (Cmu)
<b>JFETs</b>	
LV1	JFET area factor
LX0	VGS
LX1	Gate-drain voltage (VGD)
LX2	Gate-to-source (CGS)
LX3	Drain current (IDS)
LX4	Gate-to-drain current (IGD)
LX5	Transconductance (GM)
LX6	Drain-source transconductance (GDS)
LX9	Gate-source charge (QG)
LX11	Gate-drain capacitance (GD)
LX18	Drain-body trans-conductance (GMB)
<b>MOSFETs</b>	
LV1	Effective channel Length
LV2	Effective channel width
LV3	Effective area of the drain diode (AD)
LV4	Effective area of the source diode (AS)
LV9	Threshold “on” voltage (VT)
LV10	Saturation voltage (VDSS)
LV11	Effective drain diode periphery (PD)
LV12	Effective source diode periphery (PS)
LV13	Drain resistance (squares) (RDC)
LV14	Source resistance (squares) (RSC)
LV15	Charge sharing coefficient (XQC)
LV16	Effective drain conductance (1/RDeff)
LV17	Effective source conductance (1/RSeff)
LX0	Bulk-drain voltage (VBD)
LX1	Bulk-source voltage (VBS)
LX2	Gate-source voltage (VGS)
LX3	Drain-source voltage (VDS)

**Table 10-21. Element Output**

LX4	DC drain current (IDS)
LX5	DC source-bulk diode current (IBS)
LX6	DC drain-bulk diode current (IBD)
LX7	DC gate transconductance (GM)
LX8	DC drain-source conductance (GDS)
LX9	DC substrate transconductance (GMBS)
LX10	Conductance of the drain diode (GMIBD)
LX11	Conductance of the source diode (GMIBS)
LX12	Bulk charge (QB)
LX14	Gate charge (QG)
LX16	Channel charge (QD)
LX18	dQg/dVgb (CGG)
LX19	dQg/dVdb (CGB)
LX20	dQg/dVsb (CGS)
LX21	dQb/dVgb (CBG)
LX22	dQb/dVdb (CBD)
LX23	dQb/dVsb (CBS)
LX24	Drain-bulk charge (QBD)
LX26	Source-bulk charge (QBS)
LX28	Bulk-source capacitance
LX29	Bulk-drain capacitance
LX32	dQd/dVgb (CDG)
LX33	dQd/dVdb (CDD)
LX34	dQd/dVsb (CDS)

**Printing Internal Pin Values**

To measure the intrinsic value of a pin you use the same syntax as for measuring its extrinsic value. However, you use an index number greater than the number of pins available that corresponds to the pins position. For example, the source voltage pin is the third pin of a MOSFET and there are 4 pins in total. Therefore, to obtain the intrinsic source voltage of this MOSFET you would use:

```
.PLOT V(Mx.7)
```

---

To obtain the intrinsic value of the drain current pin of a JFET you would use:

```
.PLOT I(Jx.4)
```

---

**Note**

If the position number used is greater than the index available for the device it will return a value of zero.

---

## Monitoring of Hierarchical Nodes

More information related to the internal status of devices can be displayed. The following is a list of variables displayed together with the syntax to address them. This output format is used to specify nodes that lie within subcircuits. The node itself cannot be referenced directly from the ‘top-level’ of the circuit and so it must be addressed through the levels of the subcircuit by separating elements with periods (.) in the output control statement. The example below illustrates this output format.

```
.subckt sc1 n10 n12
r10 n10 n11 0.2
x2 n11 n12 sc2
.ends sc1
.subckt sc2 n20 n22
r20 n20 n21 0.1k
r22 n21 n22 0.1k
.ends sc2
x1 a b sc1
.print tran v(x1.x2.n21) v(a) v(b)
```

The above example specifies the output of three nodes. The node `x1.x2.n21` is created as the second pin of resistor `r20`, which is an element of the subcircuit `sc2`, instantiated using instance `x2`. The instance `x2` is itself nested in the subcircuit `sc1`, instantiated using instance `x1`.

## Transmission Lines

- **I(Txx.N1) I(Txx.N2) I(Txx.N3) I(Txx.N4)**

Prints the current out of transmission lines.

## Two-port Parameters

**Table 10-22. Two-port Parameters**

<b>Two-port Stability Factors</b>	
(for more information, see <a href="#">page 3-24</a> of the <i>Eldo RF User's Manual</i> )	
KFACTOR	Computes the stability factor for 2-ports. $\frac{(1.0 -  S_{11} ^2 -  S_{22} ^2 -  S_{11} \times S_{22} - S_{12} \times S_{21} ^2)}{(2 \times  S_{12} \times S_{21} )}$ S11, S22, and so on, are the S parameters.
BFACTOR	Rollet stability factor. $BFACTOR = \frac{1 -  S_{22} ^2}{ S_{11} - \Delta S_{22}*  +  S_{22} \cdot S_{11} }$
MUFACTOR	Rollet stability factor. $MUFACTOR = \frac{1 +  S_{11} ^2 -  S_{22} ^2 +  \Delta ^2}{ S_{11} - \Delta S_{22}* } \text{ where } \Delta = S_{11} \cdot S_{22} - (S_{12} \cdot S_{21})$
<b>Two-port Noise Parameters</b>	
(for more information, see <a href="#">page 3-25</a> of the <i>Eldo RF User's Manual</i> )	
NFMIN_mag	Minimal noise figure (magnitude) of the two-port. $NFMIN = 1 + 2 \cdot \frac{\operatorname{Re}\{C_{12}^A\} + GOPT \cdot C_{11}^A}{4 \cdot k \cdot T}$
NFMIN_dB	Minimal noise figure of the two-port (in decibels).
GOPT	Real part of the optimal source admittance. $GOPT = \frac{1}{C_{11}^A} \sqrt{C_{11}^A \cdot C_{22}^A - (\operatorname{Im}\{C_{12}^A\})^2}$
BOPT	Imaginary part of the optimal source admittance. The sign convention of this parameter can be changed using the <b>NOISE_SGNCONV</b> option on <a href="#">page 11-42</a> . $BOPT = \frac{\operatorname{Im}\{C_{12}^A\}}{C_{11}^A}$
RNEQ	Equivalent noise resistance. $RNEQ = \frac{C_{11}^A}{4 \cdot k \cdot T}$
YOPT	Source admittance that produces the minimal noise figure. $YOPT = GOPT + j \cdot BOPT$

**Table 10-22. Two-port Parameters**

GAMMA_OPT	Complex quantity of the optimal reflection coefficient associated with the minimum noise figure. $\text{GAMMA\_OPT} = \text{GAMMA\_OPT\_MAG} \cdot \exp(j \cdot \text{PHI\_OPT})$
GAMMA_OPT_MAG	Magnitude of the optimal reflection coefficient associated with the minimum noise figure. $\text{GAMMA\_OPT\_MAG} =  \Gamma_{\text{opt}} $
PHI_OPT	Angle of the optimal reflection coefficient associated with the minimum noise figure. The sign convention of this parameter can be changed using the <b>NOISE_SGNCONV</b> option on <a href="#">page 11-42</a> . $\text{PHI\_OPT} = \angle \Gamma_{\text{opt}}$

**Two-port Noise Circles**(for more information, see [page 3-27](#) of the *Eldo RF User's Manual*)

NC	A noise circle is plotted for a given value of a Noise Figure (SNF_val). When this value is not specified, the circle is plotted for a Noise Figure value corresponding to the actual circuit.  $\text{center} = \frac{\Gamma_{\text{opt}}}{1 + N_i}$ $\text{radius} = \sqrt{\frac{N_i^2 + N_i \cdot (1 -  \Gamma_{\text{opt}} ^2)}{1 + N_i}}$ $N_i = \frac{(SNF_{\text{val}} - NFMIN) \cdot  1 + \Gamma_{\text{opt}} ^2}{4 \cdot RNEQ}$
----	--

**Two-port Constant Gain Circles**(for more information, see [page 3-27](#) of the *Eldo RF User's Manual*)

GAC	Available Gain Circle. Determines constant gain contour at the input port. With respect to the definition of <b>GA</b> and <b>GP</b> , <b>GAC</b> represents an optimum match at the output port.  $\text{center} = \frac{GA(S_{11}^* - S_{22}\Delta^*)}{ S_{21} ^2 + GA( S_{11} ^2 -  \Delta ^2)}$ <p>where <math>\Delta = (S_{11}S_{22} - S_{21}S_{12})</math></p> $\text{radius} = \sqrt{\frac{1 - 2KFACTOR S_{21}S_{12} \frac{GA}{ S_{21} ^2} + \left( S_{21}S_{12} \frac{GA}{ S_{21} ^2}\right)^2}{1 + \frac{GA}{ S_{21} ^2}( S_{11} ^2 -  \Delta ^2)}}$
-----	---

**Table 10-22. Two-port Parameters**

GPC	<p>Power Gain Circle. Determines constant gain contour at the output port. With respect to the definition of <b>GA</b> and <b>GP</b>, <b>GPC</b> represents an optimum match at the input port.</p> $\text{center} = \frac{\text{GP}(\mathbf{S}_{22}^* - \mathbf{S}_{11}\Delta^*)}{ \mathbf{S}_{21} ^2 + \text{GP}( \mathbf{S}_{22} ^2 -  \Delta ^2)}$ $\text{radius} = \sqrt{\frac{1 - 2\text{KFACTOR} \mathbf{S}_{21}\mathbf{S}_{12}  \frac{\text{GP}}{ \mathbf{S}_{21} ^2} + \left(  \mathbf{S}_{21}\mathbf{S}_{12}  \frac{\text{GP}}{ \mathbf{S}_{21} ^2} \right)^2}{1 + \frac{\text{GP}}{ \mathbf{S}_{21} ^2} ( \mathbf{S}_{22} ^2 -  \Delta ^2)}}$
<b>Two-port Gain Parameters</b>	
(for more information, see <a href="#">page 3-21</a> of the <i>Eldo RF User's Manual</i> )	
GA_mag	<p>Available power Gain (magnitude). This is the ratio of the power available from the two-port circuit to the power available from the source when the load is conjugately matched to the output port of the circuit (<math>\Gamma_L = \text{conj}(\Gamma_{\text{OUT}})</math>).</p> $\text{GA} = \frac{1 -  \Gamma_s ^2}{ 1 - \mathbf{S}_{11}\Gamma_s ^2}  \mathbf{S}_{21} ^2 \frac{1}{1 -  \Gamma_{\text{OUT}} ^2}$
GA_dB	Available power Gain (in decibels).
GAM_mag	<p>Maximum Available power Gain (magnitude). When the input port of the circuit is conjugately matched to the source impedance and the output port of the circuit is conjugately matched to the load impedance (<math>\Gamma_S = \text{conj}(\Gamma_{\text{IN}})</math> and <math>\Gamma_L = \text{conj}(\Gamma_{\text{OUT}})</math>).</p> <p>For a bilateral case:</p> $\text{GAM} = \left  \frac{\mathbf{S}_{21}}{\mathbf{S}_{12}} \right  (\text{KFACTOR} - \sqrt{\text{KFACTOR}^2 - 1})$ $\text{GAM} = \left  \frac{\mathbf{S}_{21}}{\mathbf{S}_{12}} \right  \quad \text{if KFACTOR} > 1$ $\text{GAM} = \left  \frac{\mathbf{S}_{21}}{\mathbf{S}_{12}} \right  \quad \text{if KFACTOR} \leq 1$ <p>For the <b>KFACTOR</b> see <a href="#">page 10-242</a>.</p> <p>For a unilateral case, see the <b>GAUM</b> definition <a href="#">page 10-245</a>.</p>
GAM_dB	Maximum Available power Gain (in decibels).
GASM_mag	Maximum Available Stable Gain (magnitude). See the <b>GAM</b> definition above.
GASM_dB	Maximum Available Stable Gain (in decibels).

**Table 10-22. Two-port Parameters**

GP_mag	<p>Power Gain (magnitude). This is the ratio of the power delivered to the load to the power input to the two-port circuit when the input port of the circuit is conjugately matched to the source impedance (<math>\Gamma_S = \text{conj}(\Gamma_{IN})</math>).</p> $GP = \frac{1 -  \Gamma_L ^2}{ 1 - S_{22}\Gamma_L ^2}  S_{21} ^2 \frac{1}{1 -  \Gamma_{IN} ^2}$ $\Gamma_L = \frac{Z_L - Z_0}{Z_L + Z_0}$ $\Gamma_{IN} = S_{11} + \frac{S_{12}S_{21}\Gamma_L}{1 - S_{22}\Gamma_L}$ <p><math>Z_S</math> Source impedance  <math>Z_L</math> Load impedance  <math>\Gamma_S</math> Source reflection coefficient  <math>\Gamma_L</math> Load reflection coefficient  <math>Z_0</math> Characteristic impedance (by default <math>50\ \Omega</math>; to modify this value see the ZCHAR option on <a href="#">page 11-60</a>).</p>
GP_dB	Power Gain (in decibels).
GAUM_mag	<p>Maximum Unilateral transducer power Gain (magnitude). This is the transducer gain when the circuit ports are both optimally matched (<math>\Gamma_S = \text{conj}(\Gamma_{IN})</math> and <math>\Gamma_L = \text{conj}(\Gamma_{OUT})</math> and <math>S_{12} = 0</math>).</p> $GAUM = \frac{1}{1 -  S_{11} ^2}  S_{21} ^2 \frac{1}{1 -  S_{22} ^2}$
GAUM_dB	Maximum Unilateral transducer power Gain (in decibels).
TGP_mag	<p>Transducer Power Gain (magnitude). This is the ratio of the power delivered to the load to the power available from the source.</p> $TGP = \frac{1 -  \Gamma_s ^2}{ 1 - S_{11}\Gamma_s ^2}  S_{21} ^2 \frac{1 -  \Gamma_L ^2}{ 1 - \Gamma_{OUT}\Gamma_L ^2}$
TGP_dB	Transducer Power Gain (in decibels).
<h3>Two-port Stability Circles</h3> <p>(for more information, see <a href="#">page 3-28</a> of the <i>Eldo RF User's Manual</i>)</p>	
SSC	<p>Source Stability Circle. It determines the locus of <math>\Gamma_S</math> which produce <math> \Gamma_{OUT}  = 1</math>.</p> $\text{center} = \frac{S_{22}\Delta^* - S_{11}^*}{ \Delta ^2 -  S_{11} ^2}$ $\text{radius} = \sqrt{\frac{ S_{12}S_{21} ^2}{ \Delta ^2 -  S_{11} ^2}}$

**Table 10-22. Two-port Parameters**

LSC	Load Stability Circle. It determines the locus of $\Gamma_L$ which produce $ \Gamma_{IN}  = 1$ . $\text{center} = \frac{S_{11}\Delta^* - S_{22}^*}{ \Delta ^2 -  S_{11} ^2}$ $\text{radius} = \left  \frac{S_{12}S_{21}}{ \Delta ^2 -  S_{22} ^2} \right $
-----	--

**Plotting an analog signal as a digital bus**

To enable the plotting of an analog signal as a digital bus, a setup command is required (**.DEFPLOTDIG**) to be used in conjunction with the **VDIG** parameter of the **.PLOT** command.

```
.DEFPLOTDIG VTH1=2.2 VTH2=2.7
.PLOT TRAN VDIG(n2)
```

When **.DEFPLOTDIG** is used in conjunction with **.PLOT VDIG**, the signal node n2 is plotted as a digital curve and will only have values of “1” or “0”.

- **.DEFPLOTDIG [VTH[1]=VAL1 [VTH2=VAL2]]**

This is used as a precursor to **VDIG** in order to plot an analog curve as digital with reference to any stated threshold voltage(s).

- **VTH[1]=VAL1**

If a voltage threshold is specified, the bus of an analog signal is plotted as a bus (hexadecimal format), else all the different signals of the bus are plotted separately in the waveform viewer as analog waves. (VTH and VTH1 are equivalent to ensure backwards compatibility.)

- **VTH2=VAL2**

This can be used to plot the indeterminate value as shown below:

When only **VTH1** is given:

If value < VTH then logic state 0.  
If value > VTH then logic state 1.

When both **VTH1** and **VTH2** are given:

If value < VTH1 then logic state 0.  
If VTH1 < value < VTH2 then state X.  
If value > VTH2 then logic state 1.

**Examples**

```
.PLOT TRAN V(a) V(b)
```

Specifies that the voltages for **V(a)** and **V(b)** be plotted. EZwave will plot both **V(a)** and **V(b)** on the Y-axis with the time plotted on the X-axis.

```
.PLOT TRAN V(a) (VERSUS) V(b)
```

Specifies that EZwave will plot **V(a)** on the Y-axis against **V(b)**, which will be plotted on the X-axis.

```
.plot tran v(1) v(2) v(3, 4) i(v1, v2)
```

Specifies that the voltage at the nodes 1 and 2, the voltage difference between the nodes 3 and 4, and the current difference of the voltage sources **v1** and **v2** be plotted.

```
.plot ac vdb(4)(-50, 50) idb(v6)(-75, 75)
```

This is the same example as the last except that the dB plot limits have changed. The new limits are -50dB to +50dB for the voltage at node 4, and -75dB to +75dB for the current through voltage source **v6**.

```
.plot ac vm(4) unit=Watts
```

Specifies that the voltage at node 4 be plotted, and that the Y-axis units of the plot be Watts.

```
.PLOT NOISE NOISE(r1)
```

Eldo will plot the noise generated by the resistor **r1** in the circuit.

```
.plot noise noise(x1)
```

Returns the total noise contribution of a subcircuit instance (**x1**). Its value is the sum of the noise of all elements that are part of the specified subcircuit instance.

```
.FOUR LABEL = t1 V(a)
.PLOT FOUR FOURDB(t1)
```

Eldo will plot the value (in dB) at 5 Meg for the FFT done on **V(a)**.

```
.PLOT VX(M1.7)
.PLOT VX(Q1.3)
.PLOT VX(R1.2)
```

Eldo will plot values according to the following: **VX(M1.7)** refers to the intrinsic source voltage, **VX(Q1.3)** refers to the extrinsic emitter voltage, **VX(R1.2)** refers to the ‘second’ pin of resistor **R1**.

```
.PLOT TRAN w1=v(X1.X2.X3.X4.OUT)
```

Here, the wave **v(X1.X2.X3.X4.OUT)** will be plotted inside EZwave, but the legend will be **w1** and not **v(X1.X2.X3.X4.OUT)**.

The following example shows valid syntax for specifying EZwave to display the waves in a Smith chart.

```
.PLOT AC S(1,1) S(2,2) (SMITH)
```

In the example above, both **S(1,1)** and **S(2,2)** will be on the same Smith chart.

The following example shows valid syntax for specifying EZwave to display the waves in a Polar chart.

**.PLOT**

```
.PLOT AC S12 (POLAR)
```

In the example below, the minimal noise figure will be plotted for two port noise.

```
.PLOT NOISE NFMIN_MAG
```

In the example below, the y-axis value of waveform WAVE is returned when the equivalent noise resistance reaches 10k.

```
.PLOT NOISE YVAL(RNEQ, 10k)
```

In the example below the extracted wave FOO which was generated during optimization is plotted.

```
.PLOT OPT WOPT(FOO)
```

Display DSP results:

```
.DSP LABEL=LBL2 MODEL=PSD_PERIODO V(1)
.PLOT DSP DSPDB(LBL2)
```

Eldo will print out the value (in dB) for the DSP done on V(1).

```
.DATA dataname component1 component2
+ x1 c1y1 c2y1
+ x2 c1y2 c2y1
...
+ xn c1yn c2yn
.ENDDATA

.PLOT TRAN DATA(dataname,component2)
```

Eldo will plot a wave defined with a **.DATA** statement.

The following full netlist example shows how to plot the S11, Zin and Yin of a node (IN) on a Smith chart using STOSMITH, ZTOSMITH, and YTOSMITH functions.

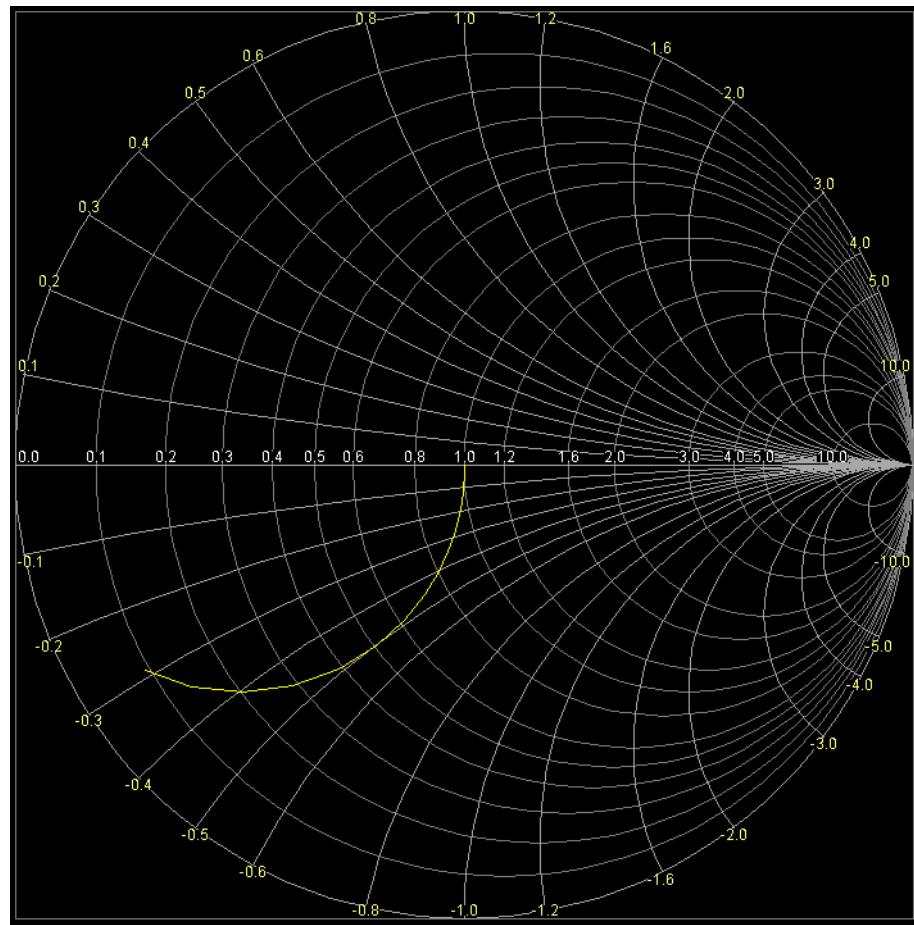
```
VIN IN 0 RPORT=50 IPORT=1 FOUR FUND1 MA (1) 1 0
RIN IN 1 RIN
CIN IN 0 CIN
LIN 1 0 LIN
.PARAM F1=1G Z0=50 RIN=Z0
.PARAM CIN=1p LIN=1f
.SST FUND1=F1 NHARM1=1
.STEP PARAM F1 DEC 10 1k 10G
.DEFWAVE ZIN=(1+S(1,1))/(1-S(1,1))*50
.DEFWAVE YIN=1/W(ZIN)

.EXTRACT FSST LABEL=S_EXT STOSMITH(YVAL(S(1,1),F1))
.PLOT MEAS(S_EXT) (SMITH,50)
.EXTRACT FSST LABEL=Z_EXT ZTOSMITH(YVAL(W(ZIN),F1),50)
.PLOT MEAS(Z_EXT) (SMITH,50)
.EXTRACT FSST LABEL=Y_EXT YTOSMITH(YVAL(W(YIN),F1),50)
.PLOT MEAS(Y_EXT) (SMITH,50)

.END
```

As can be seen in EZwave, the three functions produce the same waveform.

**Figure 10-4. Smith chart plot using STOSMITH/ZTOSMITH/YTOSMITH functions**



## .PLOTBUS

### Plotting of Bus Signals

```
.PLOTBUS BNAME [ VTH[1]=VAL1 [ VTH2=VAL2 ] ]
.PLOTBUS BNAME[MSB:LSB] | BNAME<MSB:LSB> | BNAMEMSB:LSB
```

This command plots all the bits of a bus, previously defined via the **.SETBUS** and using **.SIGBUS** to send a value. **.SETBUS** can be implicitly declared, providing that **BNAME[MSB:LSB]** or **BNAME<MSB:LSB>** or **BNAMEMSB:LSB** syntax is used.

### Parameters

- **BNAME**

Bus name, previously defined via the **.SETBUS** command, unless **BNAME[MSB:LSB]** or **BNAME<MSB:LSB>** or **BNAMEMSB:LSB** syntax is used, in which case **.SETBUS** is implicitly declared.

- **VTH[1]=VAL1**

If a voltage threshold is specified, the bus of an analog signal is plotted as a bus (hexadecimal format), else all the different signals of the bus are plotted separately in the waveform viewer as analog waves. (VTH and VTH1 are equivalent to ensure backwards compatibility.)

- **VTH2=VAL2**

This can be used to plot the indeterminate value as shown below:

When only VTH1 is given:

If value < VTH then logic state 0.  
If value > VTH then logic state 1.

When both VTH1 and VTH2 are given:

If value < VTH1 then logic state 0.  
If VTH1 < value < VTH2 then state X.  
If value > VTH2 then logic state 1.

- **MSB:LSB**

Series of bit names, the most significant bit being defined first. This mechanism can be used to implicitly declare **.SETBUS**.

### Examples

In order to display the output of a bus, you can use the following syntax:

```
.PLOTBUS Y<instance_name> -> <vector_name>
```

The following example shows how a second voltage threshold can be used:

```
.plotbus foo vth=1 vth2=4
.checkbus OUT_BUS_2 vth=1 vth2=4 base=bin
+ 1000ps 01
+ 7000ps x
+ 12000ps 11
+ 15000ps 01
+ 19500ps 01
+ 20000ps 10
+ 22000ps 10
+ 30000ps 01
```



Refer to “[.CHECKBUS](#)” on page 10-30 for more information.

---

**.PLOTBUS** can be used to implicitly declare **.SETBUS**, providing that BNAME [ MSB : LSB ] or BNAME <MSB : LSB> or BNAMEMSB : LSB syntax is used as shown below:

```
.PLOTBUS SELECT[2:0]
.PLOTBUS DOUT<3:0> VTH=1.65
```

is equivalent to:

```
.SETBUS SELECT SELECT[2] SELECT[1] SELECT[0]
.PLOTBUS SELECT
.SETBUS DOUT DOUT<3> DOUT<2> DOUT<1> DOUT<0>
.PLOTBUS DOUT VTH=1.65
```

The following example shows the most significant bit as 1 and the least significant as 5, where the bus name is `foo`.

```
.PLOTBUS foo1:5 vth=3
```

## .PRINTBUS

### Printing of Bus Signals

```
.PRINTBUS BNAME [VTH[1]=VAL1 [VTH2=VAL2]]  
.PRINTBUS BNAME[MSB:LSB] | BNAME<MSB:LSB> | BNAMEMSB:LSB
```

This command prints all the bits of a bus, previously defined via the **.SETBUS** and using **.SIGBUS** to send a value. **.SETBUS** can be implicitly declared, providing that **BNAME[MSB:LSB]** or **BNAME<MSB:LSB>** or **BNAMEMSB:LSB** syntax is used.

### Parameters

- **BNAME**

Bus name, previously defined via the **.SETBUS** command, unless **BNAME[MSB:LSB]** or **BNAME<MSB:LSB>** or **BNAMEMSB:LSB** syntax is used, in which case **.SETBUS** is implicitly declared.

- **VTH[1]=VAL1**

If a voltage threshold is specified, the bus of an analog signal is plotted as a bus (hexadecimal format), else all the different signals of the bus are plotted separately in the waveform viewer as analog waves. (VTH and VTH1 are equivalent to ensure backwards compatibility.)

- **VTH2=VAL2**

This can be used to plot the indeterminate value as shown below:

When only VTH1 is given:

If value < VTH then logic state 0.  
If value > VTH then logic state 1.

When both VTH1 and VTH2 are given:

If value < VTH1 then logic state 0.  
If VTH1 < value < VTH2 then state X.  
If value > VTH2 then logic state 1.

- **MSB:LSB**

Series of bit names, the most significant bit being defined first. This mechanism can be used to implicitly declare **.SETBUS**.

### Examples

In order to display the output of a bus, you can use the following syntax:

```
.PRINTBUS Y<instance_name> -> <vector_name>
```

The following example shows how a second voltage threshold can be used:

```
.printbus foo vth=1 vth2=4
.checkbus OUT_BUS_2 vth=1 vth2=4 base=bin
+ 1000ps 01
+ 7000ps x
+ 12000ps 11
+ 15000ps 01
+ 19500ps 01
+ 20000ps 10
+ 22000ps 10
+ 30000ps 01
```



Refer to “[.CHECKBUS](#)” on page 10-30 for more information.

**.PRINTBUS** can be used to implicitly declare **.SETBUS**, providing that BNAME [ MSB:LSB ] or BNAME<MSB:LSB> or BNAMEMSB:LSB syntax is used as shown below:

```
.PRINTBUS SELECT[2:0]
.PRINTBUS DOUT<3:0> VTH=1.65
```

is equivalent to:

```
.SETBUS SELECT SELECT[2] SELECT[1] SELECT[0]
.PRINTBUS SELECT
.SETBUS DOUT DOUT<3> DOUT<2> DOUT<1> DOUT<0>
.PRINTBUS DOUT VTH=1.65
```

The following example shows the most significant bit as 1 and the least significant as 5, where the bus name is `foo`.

```
.PRINTBUS foo1:5 vth=3
```

## **.PRINT**

### Printing of Results

**.PRINT** [ANALYSIS] [alias\_name=]OVN

The **.PRINT** command defines the contents of a tabular listing of any number of output variables.

### Parameters

- ANALYSIS

Can be one of the following:

#### **DC**

Specifies that the plots are required for a DC analysis. Note, not available for a single analysis. Provides compatibility with SPICE.

#### **AC**

Specifies that the plots are required for an AC analysis. Provides compatibility with SPICE.

#### **TRAN**

Specifies that the plots are required for a transient analysis. Provides compatibility with SPICE.

#### **NOISE**

Specifies that the plots are required for a noise analysis. Provides compatibility with SPICE.

#### **SSTAC|SSTXF|SSTNOISE|SSA|MODSST|SST|TSST|FSST**

Please refer to the *Eldo RF User's Manual* for more information regarding these options.

#### **FOUR**

Displays FFT results. Please see `FOURxx(label_name)` on [page 10-224](#), for display options.

#### **DSP**

Displays DSP results. Please see `DSPxx(label_name)` on [page 10-224](#), for display options.

- alias\_name

Refers to the wave name in the ASCII and binary output files. The alias\_name will be the legend displayed inside the wave viewer for the plotted wave as specified by the plot specifications in OVN.

- OVN

A list of plot specifications can follow. These are listed in the [Plot Specifications \(OVN\)](#) section below.

## Examples

```
.print ac vdb(n2, n4) vp(n2, n4)
```

Requests that the output of the magnitude between node `n2` and node `n4` in dB and the phase between `n2` and `n4` in degrees be printed.

```
.print noise inoise onoise noise(m1) noise(m2)
```

Requests that the input and output noise from a Noise analysis be printed, together with the noise contributions of components `m1` and `m2` to the total output noise.

```
.defwave z=(V(a)-V(b))/i(r1)
.print ac wr(z) wi(z)
```

Requests print out of the real and imaginary parts of the new waveform, `z`, created by the `.DEFWAVE` command.

## .PRINTFILE

### Print Tabular Output File

**.PRINTFILE** [ANALYSIS] OVN FILE=filename [STEP=value]

This command defines the contents of a tabular listing of any number of output variables, and dumps them to the specified file in ascii format. Multiple **.PRINTFILE** commands can share an output file, with variables being added one after the other, or multiple files can be defined.

### Parameters

- ANALYSIS

Can be one of the following:

#### DC

Specifies that the plots are required for a DC analysis. Provides compatibility with SPICE.

#### AC

Specifies that the plots are required for an AC analysis. Provides compatibility with SPICE.

#### TRAN

Specifies that the plots are required for a transient analysis. Provides compatibility with SPICE.

#### NOISE

Specifies that the plots are required for a noise analysis. Provides compatibility with SPICE.

#### SSTAC|SSTXF|SSTNOISE|SSA|MODSST|SST|TSST|FSST

Please refer to the *Eldo RF User's Manual* for more information regarding these RF options.

#### FOUR

Outputs FFT results. Please see [page 10-262](#) for display options.

#### DSP

Outputs DSP results. Please see [page 10-262](#), for display options.

#### SWEEP

Specifies that the plots are required during a multiple-run simulation if defined.

- OVN

The syntax for specifying the list of plot specifications to be monitored is provided on [page 10-224](#).

- FILE=filename

Specifies the file name which the table will be printed.

- **STEP=value**

Performs a sampling of the variable(s). If this parameter is specified in more than one **.PRINTFILE** command using the same output file then the smallest value for **STEP** is used. It can only be specified for databases that support multiple x-axes in the same simulation, e.g. JWDB. Other databases will ignore this parameter.

### Example

```
.printfile tran v(1) i(r1) file="output.txt"
```

This example requests that the voltage on node 1 and the current through r1 be printed to the file *output.txt*.

# **.PROBE**

## Output Shortform

```
.PROBE [ANALYSIS] [ALL | I | IX | ISUB | PORT | PRINT | SG | SPARAM] S | Q | V | VN | VTOP |
+ VX | VXN | W | WTOP] [MASK[=]mask_name] [PRINT] [STEP=val]
.PROBE [ANALYSIS] [MASK[=]mask_name] [alias_name=] OVN [PRINT] [STEP=val]
```

At first glance, the difference in using the **.PROBE** syntax above instead of using the **.PRINT** command may not be clear. When using **.PRINT**, simulation results are written to the *.chi* log file in ASCII format. When using **.PROBE**, the set of signals to be monitored is specified in the same way, but results are written to binary output files (*.cou* or *.wdb*). The *.cou* format is the binary format for Xelga and *.wdb* format is the format used by EZwave. Thus, the simulated results are available for post-processing. Other advantages are binary storage saves significant disk space and is faster to read/write.

Many different simulation results can be created by Eldo. The simulator can output simple node voltages, but also currents through devices, currents through device or subcircuit pins, power quantities, S parameters, internal variables from device models, etc. All these results are direct *raw* results from a simulation. The information in the next few pages together with the tables shown in the **.PLOT** specifications provided on [page 10-224](#) indicate the exact syntax to use for each category. For example:

```
.PROBE TRAN V(OUT)
.PROBE DC ISUB(XBIAS.VOUT)
```

The **.PROBE** command, without specified parameters, forces Eldo to save all node voltages in the binary output file. The **.PROBE** command is a short way of specifying that all nodes should be output. It is not possible to mix analysis types in one command line syntax, therefore the following statement will be rejected:

```
.probe ac I dc V tran I V S
```

It is also not possible to mix in one command line syntax general probe commands (such as **.PROBE v**, **.PROBE I**, **.PROBE vtop**) with specific probe commands (such as **.PROBE v(a)**). Two separate commands must be specified.

When the circuit has more than 1000 nodes, **.PROBE** is ignored unless the **LIMPROBE** parameter, which specifies the maximum number of nodes which may be probed, is increased using the **.OPTION** command.

### Note

The saving of all or large numbers of nodes can generate very large output files.

In order to analyze simulation results for huge circuits, and in case the user is only interested in displaying part of the circuit, a **.PROBE** command with nodes defined via both hierarchy and wildcard characters is available. The wildcard (\*) may be used to select any list of items for probing quantities such as voltage or current.

Subcircuit instances can be specified for keywords **V**, **S**, **W**, **VX**, **IX** and **ISUB**. A specific subcircuit node can be referenced or wildcards (\*) can be used. For example, the below specifies that the node `X1.1` will be probed.

```
.PROBE TRAN V(X1.1)
```

In the following example all nodes in subcircuit `X1` will be probed.

```
.PROBE TRAN V(X1.*)
```

See [page 10-262](#) for more information on using wildcards in subcircuit instances.

Wildcards, masks, etc. can be specified for both voltages and currents.

## Parameters

- ANALYSIS

Can be one of the following:

### **AC**

Specifies that the probes are required for an AC analysis.

### **DC**

Specifies that the probes are required for a DC analysis.

### **TRAN**

Specifies that the probes are required for a transient analysis. This and the above two parameters are optional but can be interesting in the case of multiple types of analysis in the `.cir` file.

### **NOISE**

Specifies that the probes are required for a noise analysis. If specified without any other arguments, the noise of all devices will be written to the output file for viewing. For more information on the use of **INOISE** and **ONOISE** for printing the equivalent input noise and output noise, see the example on [page 10-180](#) and also the description on [page 10-236](#).

### **SSTAC|SSTXF|SSTNOISE|SSA|MODSST|SST|TSST| FSST**

Please refer to the *Eldo RF User's Manual* for more information regarding these RF options.

### **FOUR**

Displays FFT results. Please see `FOURxx(label_name)` on [page 10-262](#) for display options.

### **DSP**

Displays DSP results. Please see `DSPxx(label_name)` on [page 10-262](#), for display options.

- **ALL**  
Specifies that all defwaves, voltages, currents and digital quantities are probed. This is equivalent to specifying **.PROBE W + .PROBE V + .PROBE I + .PROBE S**.
- **W**  
Causes all defwaves to be printed in output files. Subcircuit instances can be specified for this parameter.
- **WTOP**  
Probes defwaves at the top level and dumps them to the output file (not defwaves defined in **.SUBCKT** commands).
- **V**  
Causes all node voltages to be saved—this is the default option. Subcircuit instances can be specified for this parameter.
- **VX**  
Probes voltages on all nodes including all nodes of all subcircuits in the netlist. Subcircuit instances can be specified for this parameter.
- **VN**  
Only probes node names that are not numbers. The purpose being that nodes named with letters come from the designer, while nodes named with numbers come from an automatic netlister and typically designers wish to only see their own explicitly named nodes. The rule is applied only on the last part of hierarchical node names.
- **VXN**  
Equivalent to **.PROBE VX** except that the subcircuit pins are printed instead of indexes, for example **(X1.X2.C)**.
- **VTOP**  
Displays all top level node voltages. The functionality for current is not available (**ITOP** not allowed).
- **I**  
Causes all currents to be saved (node voltages are *not* saved).
- **IX**  
Probes the current flowing in and out of all nodes of all subcircuits. Subcircuit instances can be specified for this parameter.
- **ISUB**  
Equivalent to **.PROBE IX** except that the subcircuit pins are printed instead of indexes, for example **(X1.A)**. Subcircuit instances can be specified for this parameter.

- **S**  
Causes all digital quantities (VHDL, VHDL-AMS, Verilog and Verilog-AMS) to be saved. Subcircuit instances can be specified for this parameter.
- **SG**  
Saves all digital signals.
- **PORT**  
Specifies that all voltages and digital quantities are probed. This is equivalent to specifying **.PROBE V + .PROBE SG**.
- **PRINT**  
Probed defwaves, voltages, currents and digital quantities are printed to both the binary output (*.cou* or *.wdb*) and ASCII (*.chi*) files. Equivalent to specifying **.PROBE ALL PRINT**.
- **SPARAM**  
Forces Eldo to dump all S parameters in the output file. Only analyses that are in the frequency domain can be specified, for example **AC**, **SSTAC** or **FSST**.  
For more information on S parameters see “[Working with S, Y, Z Parameters](#)” on page 15-1.
- **MASK mask\_name**  
Specifying the **MASK** parameter will *not* dump the values on all node names that start with *mask\_name*. The rule is applied only on the last part of hierarchical node names.
- **PRINT**  
The probed output will be printed in the *.chi* file, as well as the binary output file (*.wdb*). The first example below shows that voltages are printed in the *.wdb* and *.chi* files. In the second case, transient probe results are printed in the *.wdb* and *.chi* files.

```
.PROBE V PRINT  
.PROBE TRAN PRINT
```

---

#### Note



To print information to an ASCII file (*.chi*), without printing in the binary output file (*.wdb*), use the **.PRINT** command.

---

- **alias\_name**  
Refers to the wave name in the ASCII and binary output files. The *alias\_name* will be the legend displayed inside the wave viewer for the plotted wave as specified by the plot specifications in **OVN**.
- **OVN**  
A list of plot specifications can follow. The syntax for specifying the list of plot specifications to be monitored is provided on [page 10-224](#). The syntax is separated into a

number of sections depending on the device: MOSFET, BJT, JFET, DIODE, with a common section afterwards.

**Note**

The plot specifications must match the corresponding analysis type, i.e. **AC**, **DC**, **TRAN**, or **NOISE**.

**FOURxx(label\_name)**

Displays FFT results. Should be specified as part of OVN. **xx** stands for **DB**, **R**, **I**, **P**, **M**, **GD**:

- DB** Magnitude, in dB.
- M** Magnitude.
- P** Phase.
- R** Real part.
- I** Imaginary part.
- GD** Group delay.

**DSPxx(label\_name)**

Displays DSP results. Should be specified as part of OVN. **xx** can be **DB**, **R**, **I**, **P**, **M**, **GD** (see above).

- **STEP=val**

Performs a sampling of the waveform(s). A point is dumped every **value**. It can only be specified for databases that support multiple x-axes in the same simulation, e.g. JWDB. Other databases will ignore this parameter.

## Using Wildcards in Subcircuit Instances

The wildcard (\*) may be used to select any list of items for probing quantities such as voltage or current. Subcircuit instances can be specified for keywords **V**, **I**, **S**, **W**, **VX**, **IX** and **ISUB**. For example, the below specifies that all nodes in subcircuit **X1** will be probed:

```
.PROBE TRAN V(X1.*)
```

The example below will store all currents in devices internal to the **Xbias** instance:

```
.PROBE I(Xbias.*)
```

The syntax below can be used to specify wildcards in subcircuit instances:

```
v|i|s|w|vx|ix|isub(<instance_name>.*), or  
v|i|s|w|vx|ix|isub(<instance_name>->*)
```

When specifying parameters **V**, **I**, **VX**, **IX**, **ISUB** with a subcircuit instance the **-R** flag can also be specified. Its usage is shown below.

**.PROBE TRAN V(X1.\*)**

This command will probe all the nodes of subcircuit X1 but will not probe internal nodes of the subcircuit hierarchy.

**.PROBE TRAN -R V(X1.\*)**

When specifying the -R flag all internal and external nodes of subcircuits beginning with the name X1 will be probed. (This can also be performed using the **VX** keyword).

**.PROBE TRAN -Rn V(X1.\*)**

Probe all nodes of subcircuit X1 and enter inside the n level of hierarchy under X1 for plotting nodes. If n is 0 this specifies the top level.

**Note**

 **.PROBE TRAN V(X1.\*)** is equivalent to: **.PROBE TRAN -R0 V(X1.\*)**

**VX**, **IX** and **ISUB** can be used to probe voltages and currents on all subcircuit input nodes within the hierarchy. For example:

**.PROBE TRAN IX(X1.\*)**

Here the current will be probed across all input nodes of subcircuits beginning with the name X1.

Defwaves and digital quantities can also be probed for subcircuit instances using wildcards.

**.PROBE TRAN S(x1.y1->\*)**

In this example all internal states (denoted by the syntax “->”) of macromodel y1 inside subcircuit x1 will be probed.

**.PROBE TRAN W(x1.\*)**

This example all defwaves will be probed for subcircuits beginning with the name x1.

If the name used in a **.PROBE** contains square brackets, “[” and “]”, these are interpreted as special characters when the wildcard character “\*” is used (see [Table 10-23](#)). According to Unix convention, these special characters may be overridden by using the delimiter backslash character “\” which removes the special function of the character that immediately follows it. For example, the following only returns names that begin with X1 . \*, because, according to Unix convention, X[1]. \* means “all names beginning with “X” and followed by the character “1””:

```
.PROBE tran V(X[1].*)
```

Therefore, to display all nodes beginning with V(X[1]. \*) either of the following should be used:

```
.option NOCMPUNIX  
.PROBE tran V(X[1].*)
```

or:

---

```
.PROBE tran V(X\[1\].*)
```

The following table contains a listing of the special characters that can be used with the **.PROBE** command.

**Table 10-23. Special Characters**

Character	Description
*	Matches any sequence of characters.
?	Matches any single character.
[abcd]	Matches any character in the specified set.
[-abcd]	Matches any character in the specified range, e.g. [ A-Z ] matches all alphabetical characters.
[!abcd]	Matches any character not in the specified set, e.g. [ !0-9 ] matches all characters which are not digits.

### Signal Monitoring Specifications used with **.PROBE** Only

- |                          |  |
|--------------------------|--|
| <b>V(Xnn.* )</b>         | Monitors the voltage on all of the nodes of subcircuit <b>Xnn</b> .    |
| <b>S(Xnn.* )</b>         | Monitors the states on all of the nodes of subcircuit <b>Xnn</b> .     |
| <b>S(Xnn.Ynn-&gt;* )</b> | Monitors the states on all of the nodes of macromodel <b>Xnn.Ynn</b> . |

### Examples

```
*.OPTION description
.option limprobe=1550
...
.r1 n30 n400 1k
.c1 n490 n610 5p
.probe
```

Specifies that all node voltages will be saved in the *<circuit\_name>.cou* file. The circuit has more than 1000 nodes, hence the inclusion of the **LIMPROBE** parameter in the **.OPTION** command.

```
* Circuit description
...
.r1 n3 n40 1k
.probe I
```

Specifies that all currents will be saved in the *<circuit\_name>.cou* file.

```
.TRAN 1n 10n
.AC dec 10 1k 10k
.probe
```

Specifies that all node voltages for each analysis specified (i.e. transient and AC) will be saved in the `<circuit_name>.cou` file. The above `.probe` is equivalent to the following two statements:

```
.probe TRAN V
.probe AC V
```

The following example specifies to probe all internal nodes of subcircuit `x1`:

```
.probe TRAN V(x1.*)
```

The following example specifies to probe all internal states of macromodel `x1.y1`:

```
.probe TRAN S(x1.y1->*)
```

The following syntax will display all internal nodes of subcircuit instance `x1` with “2” at the end of the node name:

```
.probe TRAN V(x1.*2)
```

Probe requests such as `I(Q*.C)` as well as `IC(Q*)` are accepted. The same applies for extensions B, E and S for BJT, and S, D, G, B for MOSFET.

The following syntax will display nodes in `x1` at hierarchical levels 1 and 2:

```
.probe TRAN V(x1.[!x]*)
.probe TRAN V(x1.x*.![!x]*)
```

In the following example, the wave `v(X1.X2.X3.X4.OUT)` will be plotted inside EZwave, but the legend will be `w1` and not `v(X1.X2.X3.X4.OUT)`:

```
.probe TRAN w1=v(X1.X2.X3.X4.OUT)
```

The following two examples show valid syntax for specifying EZwave to display the waves in a Smith chart.

```
.probe AC S(1,1) S(2,2) (SMITH)
.probe SSTAC S(1,1) S(2,2) (SMITH)
```

In the first line above, both `S(1,1)` and `S(2,2)` will be on the same Smith chart.

```
.probe VX(M1.7)
.probe VX(Q1.3)
.probe VX(R1.2)
```

Specifies values according to the following: `VX(M1.7)` refers to the intrinsic source voltage, `VX(Q1.3)` refers to the extrinsic emitter voltage, `VX(R1.2)` refers to the ‘second’ pin of resistor `R1`.

In the following example, the current on all pins of all `x` instances will be recorded. Wildcards do not have to be specified.

```
.probe IX
```

In the following example, the mask will be exclusively applied to all nodes of the instances `X1` and `X3`. Instance `X2` will *not* be masked and the nodes of `X2` will be probed.

---

```
.probe tran mask=in V(X1.* ) V(X3.* )
.probe tran V(X2.* )
```

The following example shows option **vxprobe** used with a **.PROBE**. Nodes v(1), v(2), v(x1.b), v(0), v(x1.a) and v(x1.c) will be probed. Without this option only v(1), v(2) and v(x1.b) would be probed.

```
.subckt foo a c
r1 a b 1k
r2 b c 1k
.ends foo

v1 1 0 dc 1
x1 1 2 foo
r1 2 0 1

.tran 1n 10n
.probe tran v
.option vxprobe
```

In the following examples, the current on node 2 of voltage source V1 will be probed, and the current on the first node of resistor R1 will be probed.

```
.probe I(V1.2)
.probe I(R1.1)
```

The example below stores currents in devices on top:

```
.PROBE I(*)
```

The example below will store all currents in devices internal to the X1 instance:

```
.PROBE I(X1.* )
```

# .PROTECT

## Netlist Protection

.PROTECT

This command is used in conjunction with “[.UNPROTECT](#)” on page 10-327 to exclude a section of a netlist from being copied into the output file. Can be specified to control the beginning of the encryption process with the [Eldo Encryption tool](#).

## Example

```
.PROTECT
vin 2 0 ac 0.5
r1 2 3 5k
c3 3 0 0.1p
.ac dec 10 10e+4 10e+8
.plot ac vdb(3)
.UNPROTECT
```

The lines in a netlist between the two commands `.PROTECT` and `.UNPROTECT` are not printed to the output file.

## **.PZ**

### Pole-Zero Analysis

**.PZ OV**

This command is used in conjunction with an AC analysis. When included in the input file, a special file called *<circut\_name>.pz* is generated by Eldo. Upon conclusion of the simulation, the Pole Zero post-processor may be activated to extract Pole-Zero data and perform simplification of the results, visualization, etc.

The poles and zeros are extracted for the transfer function characteristic connecting the input node, specified via the AC analysis command. The output node is specified via the **.PZ** command.

---

#### Note



This command works only if there is a single AC input source, otherwise an error will be issued.

---

### Parameters

- **OV**  
Request output of the specific node or current through a voltage source to the *<circut\_name>.pz* file. The syntax for voltage or current output is as follows:
  - **I(Vxx)**  
Specifies the current through the voltage source **Vxx**.
  - **V(N1[ , N2 ])**  
Specifies the voltage difference between nodes **N1** and **N2**. If **N2** and the preceding comma is omitted, ground is assumed. In the case of subcircuit nodes, **N1** has the form **(xnn.N1)** where **xnn** is the subcircuit instance.



For more details on the Pole-Zero Post-processor, refer to the “[Pole-Zero Post-processor](#)” on page 18-1

---

# .RAMP

## Automatic Ramping

```
.RAMP DC VAL [SIMPLIFY]  
.RAMP TRAN T1 T2
```

This command is used when Eldo encounters difficulties finding a DC operating point with the conventional **.DC** command. There are two options available:

1. DC ramping may be performed if the power supplies in a circuit are increased linearly from time 0, by a fixed voltage step. At each step, a DC operating point is searched up to the maximum power supply voltage.
2. Transient ramping may be performed if the power supplies are increased linearly from time 0 to **T1**, at which point simulation is continued in the form of a transient analysis. When time **T2** has been reached, a DC operating point is searched for by Eldo.

When carrying out Gmin ramping, or DC ramping, there are usually steep discontinuities. Eldo has various algorithms which may pass over these discontinuities, but they are CPU time consuming. When specifying **SIMPLIFY**, “simplified” Gmin ramping or “simplified” DC ramping will be carried out. In this case, Eldo will attempt to pass over only two discontinuities, before switching to another algorithm, whereupon it should find a DC algorithm which works better on the design. If DC convergence is not achieved by any other algorithms, Eldo will restart Gmin ramping and DC ramping, but will try to pass over all discontinuities.

Transient ramping is a more powerful way of finding a DC operating point than by using DC ramping, but has the disadvantage of being more time consuming. It is recommended that DC ramping methods be tried first and only when everything else fails use transient ramping. The **.RAMP TRAN** emulates a sweep on all of the input signals. On completion of the sweep, Eldo performs a DC analysis to ensure that the solution found at the end of the transient analysis is Steady-State.

If the keyword **SIMPLIFY** is specified at the end of the **.RAMP** command, the last DC analysis will not be performed. If a **.TRAN** command is specified, the transient simulation will start from the values obtained at the end of the **.RAMP TRAN**. This is for cases where the user is not interested in the DC, but wishes to use a time constant for increasing transient input voltages without having to change the input signal of the design.



See also **.OPTION PSTRAN** on [page 10-180](#) and **.OPTION DPTRAN** on [page 10-180](#).

**Note**

The DC ramping option, **.RAMP DC** by default, works in conjunction with the **GRAMP** parameter in the **.OPTION** command. If **GRAMP** is set to 0, only **.RAMP DC** is active. It was implemented in this way as it was found that when **GRAMP** and **.RAMP DC** are used together, efficiency is improved. See **.OPTION GRAMP** on page 10-180.

The conductance value that is placed in parallel with all pn junctions and drain and source nodes of MOSFET level 12 and 13 then also changes in accordance with its defined value with the variation of the DC sweep. If DC source ramping *only* is desired, you must set **GRAMP=0** explicitly.

**Parameters**

- **DC**  
Keyword indicating that DC ramping should be performed.
- **TRAN**  
Keyword indicating that transient ramping should be performed.
- **VAL**  
Voltage step at which DC ramping is carried out in volts. The ramping process increases the DC source from 0 up to the nominal value. **VAL** is the largest step that can be used. The default value is 0.1V.
- **SIMPLIFY**  
Eldo will attempt to pass over two discontinuities, before switching to another algorithm, more suited to the design.
- **T1**  
The time at which simulation should be continued. The default value is 1 $\mu$ s.
- **T2**  
The time at which the DC operating point is searched for. The default value is 10s.

# .RESTART

## Restart Simulation

```
.RESTART FNAME [FILE=WNAME]  
.RESTART ["fileBasename"] [NEWEST|LONGEST|TIME=VALUE] [FILE=WNAME]
```

This command restarts a simulation run with information previously saved using either the **.SAVE** or **.TSAVE** command. When a simulation is rerun with **.RESTART**, Eldo looks for a *.iic* or *.sav* file by default. A *.iic* file is the default extension when saving a simulation run with the **.SAVE** or **.TSAVE** command. A *.sav* file is created when a simulation run is interrupted by the user with Ctrl-c. See “[.SAVE](#)” on page 10-274 and “[.TSAVE](#)” on page 10-323.

If there is a mismatch between the actual design and the content of the file when reading back a **.RESTART** file for starting a new TRAN simulation, an error will be returned rather than a warning, and the simulation will stop. That means that the restart command is recommended for a simple transient simulation only.



For more details, refer to the [.SAVE, .USE & .RESTART](#) section in the [Speed and Accuracy](#) chapter.

---

In the case of **.TRAN** and **.RESTART** associated with a **.STEP**, **.MC**, **.WCASE** or **.NOISETRAN**, the same restart file will be used for all runs, see [page 10-273](#) for further information.

## Parameters

- **FNAME**  
Filename containing simulation information which should be used to restart a simulation run.
- **WNAME**  
Name of file containing the previous *.cou* or *.wdb* file. When this is provided, Eldo concatenates the old and the new binary data files.
- **"fileBaseName"**  
Specifies the root name of the *.iic* file that will be used to restart the simulation. Optional. If omitted Eldo will select the *.iic* that has the same root name as the top-netlist i.e. if the top-netlist is called *spice-on-top.cir* Eldo will select the file *spice-on-top\_*. If Eldo can not find a file with the root name *fileBaseName* it will be assumed that the filename (FNAME) has been given.
- **NEWEST**  
The simulation will be restarted with the most recently generated *.iic* file. Optional.

- **LONGEST**

Restarts the simulation using the *.iic* that was generated at the greatest time point in the simulation. Optional.

- **TIME=value**

Restarts the simulation from the specified time. If the file does not exist, Eldo will use the file with the closest time. Optional.

---

**Note**



Eldo always converts filenames specified in **.SAVE/.TSAVE/.RESTART** commands to lower case.

---

## Example

```
.restart test.sav file=test_previous.wdb
```

In this example, Eldo will create *test.wdb* as usual, but will first copy *test\_previous.wdb* into *test.wdb* before adding new points in the *test.wdb* file. If the *file=test\_previous.wdb* string is omitted in the **.RESTART** command, then the first point in the *test.wdb* file will correspond to the time at which simulation actually restarts.

To permit the concatenation of the previous binary output file with the new binary output file, you have to rename it, e.g.

Netlist file: *test.cir*

Output file: *test.wdb* (after the first simulation)

Rename this file: *test\_previous.wdb*

Relaunch Eldo with:

```
.RESTART test.sav file=test_previous.wdb
```

A new wdb file is created, *test.wdb*, which is the concatenation of the previous and the current simulation file.

```
.RESTART "checkpoint" time=10ns
```

The simulation will be restarted using the *.iic* file that was saved at the time *10ns* and has the root name *checkpoint*.

```
.RESTART LONGEST
```

The simulation will be restarted using the *.iic* file that was saved at the latest time point in the simulation. Eldo will select the *.iic* that has the same root name as the top-design unit i.e. if the top-design is called *spice-on-top.cir* Eldo will select the file *spice-on-top\_1.000000E-08.iic*.

## Usage of **.RESTART** and **.AC**

If the user has in his input deck both of the following commands:

- **.RESTART <file\_name>**, where *file\_name* comes from the results of a TRAN simulation

- and if there is a **.AC ... UIC** command,  
then the **.RESTART** will be interpreted as:

```
.USE <file_name> OVERWRITE_INPUT
```



Refer to the **UIC** parameter **.AC** on [page 10-14](#) for more information.

### Usage of **.RESTART** and **.TRAN** with **.NOISETRAN**, **.MC**, **.WCASE**, **.STEP**

In the case of **.TRAN** and **.RESTART** associated with a **.STEP**, **.MC**, **.WCASE** or **.NOISETRAN**, the same restart file will be used for all runs, for example:

```
.MC 5
.TRAN
.SAVE TIME = ...
.END
```

In this example you would assume that six restart files would be generated (five MC and one nominal), and that in subsequent commands:

```
.MC 5
.TRAN
.RESTART
.END
```

each run would use its “restart file ancestor”.

However, this is not how it works. Eldo will use only the latest “restart file” which has been generated for all runs, and unless explicitly specified, Eldo will create a restart file only for the first STEP run or the nominal MC run.

## **.SAVE**

### Save Simulation Run

```
.SAVE [[FILE=]FNAME] DC|END|TIME=VAL1 [REPEAT] [ALT|SEQ]
+ [TEMP=VAL2] [STEP=VAL3] [TYPE=NODESET|IC] [LEVEL=ALL|TOP] [CARLO=index]
```

Writes information at specific times during simulation to a file FNAME. If no filename is given, results are saved in <circuit\_name>.icc. To restart a simulation run with this information saved, use the **.RESTART** command. See “[.RESTART](#)” on page 10-271.



For more details, refer to the [.SAVE, .USE & .RESTART](#) section in the [Speed and Accuracy](#) chapter.

For saving a simulation run at multiple time points, see “[.TSAVE](#)” on page 10-323.

### Parameters

- **DC**

Keyword indicating that the DC part of the simulation should be saved in FNAME. These results may at a later time be used as **.NODESET**, **.IC**, or **.GUESS** values for another simulation via the **.USE** command.

- **END**

Keyword indicating that the simulation results should be saved after transient analysis has been completed in FNAME. These results may then be used as re-starting values of another simulation via the **.RESTART** command.

- **TIME**

The value of this keyword is the discrete instant in time after the start of simulation that the simulation results should be saved in FNAME. These results may then be used as re-starting values of another simulation via the **.RESTART** command. This option may also be used in conjunction with the **REPEAT** parameter (See below).

- **VAL1**

Time, in seconds, at which the simulation results should be saved to FNAME. A number, expression, or parameter can be specified.

#### **Note**



The **TIME** option has the same meaning as the **.OPTION SAVETIME=VAL** command, except that the former specifies simulation time and the latter specifies actual CPU time.

- **FILE**

Optional; used to set the name of the output file.

- **FNAME**

Filename and extension into which simulation information is written. Filename is alpha numeric but must start with an alpha. Note that Eldo always converts filenames specified in **.SAVE** and **.RESTART** commands to lower case.

- **REPEAT**

Keyword, used in conjunction with the **TIME**, **ALT** and **SEQ** options. When selected, Eldo saves the status of the simulation at every time interval **VAL1** to **FNAME** in accordance with the **ALT** and **SEQ** parameters described below.

- **ALT**

Used in conjunction with **REPEAT** to create the files **FNAME.a**, **FNAME.b**, **FNAME.a**, ... in an alternating order, thus overwriting the previous file of the same extension and resulting in only two files **FNAME.a** and **FNAME.b** being created.

- **SEQ**

Used in conjunction with **REPEAT** to create the files **FNAME.1**, **FNAME.2**, **FNAME.3**, ... **FNAME.N**, in a sequential manner, thus creating N files depending on the values of **TIME** and **REPEAT**.

- **TEMP**

Keyword indicating that data should be saved to **FNAME** depending on temperature.

- **VAL2**

Temperature, in degrees, at which the circuit data should be saved to **FNAME**.

- **STEP**

Keyword, used in conjunction with the **.STEP PARAM** command. When selected, Eldo saves the status of the simulation to **FNAME** if the parameter value specified in the **.STEP** command is equal to a specified value.

- **VAL3**

Value at which the status of the simulation should be saved to **FNAME** when equal to the parameter value specified in the **.STEP** command.

- **TYPE**

Used to set the type of information being dumped. Eldo will generate a Spice-format file which can be loaded with the commands **.LOAD** or **.INCLUDE** for subsequent runs.

If **NODESET** is specified, then when Eldo reads back the file from a **.LOAD** command, the node voltage information is considered as being equivalent to a **.NODESET** command.

If **IC** is specified, then the node voltage information is considered as being equivalent to a **.IC** command.

Specifying **TYPE** means the file will contain less information than the native Eldo save file (*iic*). However, it can be re-used on other Spice-like simulators.

- **LEVEL=ALL**

All nodes have to be dumped in the file (default).

- **LEVEL=TOP**

Only top nodes (i.e. those which are not inside a SUBCKT) are dumped in the file.



The last three parameters work in conjunction with the **.LOAD** command on [page 10-139](#).

- **CARLO=index**

Save simulation information for a specific run (index) of a Monte Carlo analysis. This is useful for debugging purposes, as it would be possible to restart a specific run from a series of Monte Carlo simulations. The DC value would be re-injected into a single Monte Carlo run (**IRUN=index** of the **.MC** command).

## Examples

```
.save test.ddt dc
```

Specifies that DC analysis results of the current circuit simulation should be written to the *test.ddt* file.

```
r1 1 2 p2
...
.step param p2 1k 5k 1k
.save status.ccf step=4k
```

Causes status of current simulation to be saved in *status.ccf* file when parameter *p2* reaches 4kΩ.

## .SCALE

### Automatic Scaling of Active Devices

```
.SC[ALE] ELTYPE KEYWORD VALUE [KEYWORD VALUE ...]
+ [ELEMENTS ALL|EXCEPT] [ELNAME1 ELNAME2 ...] [(ELNAME1 ELNAME2)]
.SC[ALE] ELTYPE KEYWORD VALUE [KEYWORD VALUE ...]
+ MODELS MODNAME1 [MODNAME2 ...]

.SC[ALE] MODTYPE KEYWORD VALUE [KEYWORD VALUE ....]
+ [MODELS ALL|EXCEPT] [MODNAME1 MODNAME2] [(MODNAME1 MODNAME2...)]
.SC[ALE] P FACTOR=VALUE [SUBCKT=SUBNAME] [INST=INSTNAME]
+ [PARAMS ALL|EXCEPT] [PARAM1 PARAM2 ...] [(PARAM11 PARAM22)]
```

This command scales device and model parameters of active devices automatically.

The first two syntax shown above are the general forms for devices. The third syntax is the general form for device models. The last syntax is an additional feature for parameter scaling.

### Parameters

- **ELTYPE**  
Type of element.
- **MODTYPE**  
Type of model.
- **KEYWORD**  
Any valid element or model parameter or keyword.
- **ELEMENTS**  
Keyword indicating the end of a keyword/value pair list for elements.
- **PARAMS**  
Keyword indicating parameter scaling.
- **ALL**  
Keyword requesting scaling of all elements or models.
- **EXCEPT**  
Keyword requesting scaling of all elements or models with the exception of the ones listed.

### Limitation

The element parameters or devices are only scaled at the parsing level.

If a **.DC** is used on a device element with a scale factor, the device element takes only the value specified by the DC analysis. There is no scaling effect.

## **.SENS**

### Sensitivity Analysis

**.SENS OVN {OVN}**

This command causes a DC sensitivity analysis to be performed. By linearizing the circuit around a bias point, the sensitivities of each of the output variables in relation to all the device values and the model parameters are calculated and output.

This Eldo command is restricted to DC sensitivity analysis. For AC, TRAN or SST sensitivity analysis, the **.WCASE** command can be used. For more information see “[“.WCASE”](#) on page 10-332.

The DC sensitivity analysis will be performed only for the first DC of a DCSWEEP.

#### **Note**



For MOS, only sensitivity with respect to L and W is implemented. For BJT, Diodes and JFET, sensitivity is implemented with respect to all model parameters.

Device sensitivities are provided for the following types:

- Resistors.
- Independent Voltage and Current Sources.
- Voltage and Current Controlled Switches.
- Diodes and Bipolar Transistors.

### Parameters

- **OVN**  
Request output of the specific node or current through a voltage source to the *<circut\_name>.chi* file. Syntax for voltage or current output is as follows:
- **V(N1[ , N2 ])**  
Specifies the voltage difference between nodes N1 and N2. If only N1 is specified, ground is assumed for the second node.
- **I(Vxx[ , Vyy ])**  
Specifies the current difference between voltage sources Vxx and Vyy. If only Vxx is specified, current through Vxx is printed.

### Example

```
.sens v(2) i(vcc)
```

---

Specifies a sensitivity analysis to be performed on the voltage at node `2` and on the current through the voltage source `vcc`. The results of the analysis are listed in the `<circut_name>.chi` file.



---

An example of this type of analysis can be found in “[Tutorials](#)” on page 24-1.

---

# **.SENSPARAM**

## Sensitivity Analysis

```
.SENSPARAM sub[ckt]=subckt_name param=parameter_list
+ [var[iation]=value] [inst[ance]=instance_list]
+ [sort=inc[reasing] | dec[reasing] | alpha[betical]]
+ [sort_nbmax=value] [sort_abs=value | sort_rel=value]
```

This command activates a sensitivity analysis of extracts versus subcircuit parameters. Designers need sensitivity information for design parameters (parameters they can act upon). This command computes the sensitivity of voltages/currents and extracts to these design parameters.

The first simulation is the nominal simulation, using nominal values for the parameters. The following simulations are sensitivity analysis runs where Eldo applies a small variation to one of the P parameters, keeping all other to their nominal value, and computes the sensitivity of the targets relative to the current parameter. Therefore, there are 1 + P simulations.

## Parameters

- **sub[ckt]**  
A subcircuit name.
- **param**  
Defines the list of parameters of subcircuit `subckt_name` for which the sensitivity must be calculated.
- **var[iation]**  
Defines the variation value for the sensitivity parameters. Default is 0.1%.
- **ins[tance]**  
Defines a list of instances of subcircuit `subckt_name`. The sensitivity analysis will be restricted to parameters of these instances. This keyword supports standard wildcards: \* and ?. For example:  
  
.sensparam subckt=resi instance=x2.\* param=a,b variation=7%  
  
means that all parameters a and b of instances of subcircuit resi inside instance X2 will be subject to sensitivity.  
  
.sensparam subckt=resi instance=x2 param=c,e variation=0.03  
  
means that parameters c and e of instance x2 of subcircuit resi only will be subject to sensitivity.
- **sort**  
Defines the ordering method which will be used to print sensitivity results. Default is alphabetical.

- **sort\_nbmax**  
Only the specified n first contributions will be printed.
- **sort\_abs**  
Only the contributions greater than the given value in absolute will be printed.
- **sort\_rel**  
Only the contributions greater than the given value in relative will be printed.

### Note

If several **.SENSPARAM** commands are specified in the netlist, only the last value for keywords **sort**, **sort\_nbmax**, **sort\_abs** and **sort\_rel** will be retained.

### Example

```
* .sensparam example
.subckt resi a b param: a=1 b=1 c=1 d=1 e=1
R1 a b r={a+2*b+3*c+4*d+5*e}
.ends resi

.subckt div a b param: a=1 b=1 c=1 d=1 e=1
x1 a 1 resi a=a b=b c=c d=d e=e
x2 1 b resi a=a b=b c=c d=d e=e
.ends div

v1 1 0 sin(0 1 0.5g)
x1 1 2 resi a=2 b=1 c=5 d=6 e=10
X2 1 2 resi a=10 b=2 c=6 d=3 e=1
X3 1 2 div
r2 2 0 1

.sensparam subckt=resi instance=x3.* param=a,b variation=2%
.sensparam subckt=resi instance=x2 param=c,e variation=0.03
.sensparam subckt=resi param=d sort_abs=40e-9

.tran 1n 15n

.extract tran yval(i(r2),5n)
.end
```

The three **.SENSPARAM** commands in the above example define which parameters will be used for sensitivity analysis. The first command defines that parameters a, b and d of all instances of subcircuit resi\* inside instance X3 will have a variation of 2% applied to their nominal values. The second command defines that parameters c and e of top instance X2 will have a variation of 3% applied to their nominal values. The third command defines that parameter d of all instances of subcircuit resi will have a default variation of 0.1% applied, that contributions lower than 40e-9 will not be printed, and are sorted in decreasing order.

## .SETBUS

### Create Bus

**.SETBUS** BNAME PN {PN}

This command creates a bus BNAME with a number of bits PN.

### Parameters

- BNAME  
Bus name.
- PN  
Bit name, the most significant bit being defined first.

### Example

**.setbus** bus1 p5 p4 p3 p2 p1 p0

Specifies a bus bus1 with the pins p5, p4, p3, p2, p1 and p0. Pin p5 is the most significant bit.

---

**Note**

 Bus signals are defined using the **.SIGBUS** command.

---

## .SETKEY

### Set Reliability Model Key (Password)

**.SETKEY** [ MODEL=model\_name ] KEY=key\_value

This reliability analysis command defines a key (password), which is associated to a specific model or to all the models of a library.



For the complete description of this command and information on all reliability commands, see the separate chapter [Reliability Simulation](#).

---

## **.SETSOA**

### Set Safe Operating Area

```

.SETSOA [LABEL=<STRING>] E {EXPRESSION=(MIN,MAX[,XAXIS])}
.SETSOA [LABEL=<STRING>] E
+ IF(EXPR) THEN({PARAM=(MIN,MAX[,XAXIS])})
+ ELSE({PARAM=(MIN,MAX[,XAXIS])}) ENDIF
.SETSOA [LABEL=<STRING>] D DNAME [SUBCKT=subckt_list|INST=inst_list]
+ {PARAM=(MIN,MAX[,XAXIS])}
.SETSOA [LABEL=<STRING>] D DNAME [SUBCKT=subckt_list|INST=inst_list]
+ IF(EXPR) THEN({PARAM=(MIN, MAX[, XAXIS])})
+ ELSE({PARAM=(MIN, MAX[, XAXIS])}) ENDIF
.SETSOA [LABEL=<STRING>] M MNAME [SUBCKT=subckt_list|INST=inst_list]
+ {PARAM=(MIN,MAX[,XAXIS])}
.SETSOA [LABEL=<STRING>] M MNAME [SUBCKT=subckt_list|INST=inst_list]
+ IF(EXPR) THEN({PARAM=(MIN, MAX[, XAXIS])})
+ ELSE({PARAM=(MIN, MAX[, XAXIS])}) ENDIF

```

Specifies the safe operating area limits for device (D) and model parameters (M) and Eldo expressions (E).

#### **Caution**



Eldo issues a warning at run time whenever a safe operating limit is violated.

Devices may be selected using either the device name or its model name (if one exists). If limits are specified using both device and model names for a component, then both results are produced. There is no priority selection, and so the particular device must respect all conditions specified to generate no warning (regardless of whether these are the same or different).

Limits set using **.SETSOA** are checked when the **.CHECKSOA** command is used, see [page 10-33](#).

This command applies to all types of analysis, and may also be used in conjunction with **.TEMP**, **.STEP**, **.MC** and **.WCASE** commands. For the last two cases, only a check of the typical case is done.

The IF statement used in conjunction with the keywords THEN, ELSE, ENDIF may be used with expressions to specify conditions for when a SOA should be checked.

It is possible inside SOA expressions to refer to device instance or model parameters via its parameter name, such as: D(\*,<parameter\_name>) or M(\*,<parameter\_name>). The wildcard \* character specifies that Eldo will search for the parameter in the device or model specified in DNAME or MNAME respectively. Model parameters can be specified for devices and device parameters can be specified for models.

At the top level, a list of subcircuits or instances at a lower level of hierarchy can be specified with devices or models.

This command also accepts an optional label as the first argument. The label will appear in the ASCII output file, which can be useful for readability of this file.

## Parameters

- EXPRESSION

An expression whose calculated value is to be checked. This can use the same syntax as described in the **FUNCTION** section listed in the **.EXTRACT** command, e.g.

```
.SETSOA E xup(v(out),4,0,100n,1)
+ -xup(v(out),1,0,100n,1) = (*,3n)
```

- DNAME

Name of a device whose parameter(s) are to be checked. The wildcard character **\*** can be used in a device name to specify that all devices of the same type should be checked (for example **.SETSOA D r\***... will check all resistors in the netlist).

- MNAME

Name of a model whose parameter(s) are to be checked. The following device categories can also be specified: NMOS, PMOS, NPN, PNP, NJF, PJF, D, for example:

```
.SETSOA M NPN ...
```

indicates all NPN devices will be checked.

- PARAM

Name of the parameter to be checked e.g.

**IB**, **IC**, **IE**, **IS**, **VBE**, **VBC**, **VBS**, **VCE**, **VCS**, **VES**, **POW**, **VC**, **VS**, **VB**, **VE** for a bipolar transistor.

**IG**, **IS**, **ID**, **IB**, **VGD**, **VGS**, **VGB**, **VBS**, **VBD**, **VDS**, **POW**, **VS**, **VD**, **VG**, **VB** for a MOS/JFET.

**VPOS**, **VNEG** (for voltage on positive/negative pin), **VDIP**, **I**, for a dipole.

**POW** for power.

To access the **VTH** value, use **VT(device\_name)**,

to access the **VDSAT** value, use **VDSS(device\_name)**.

The wildcard character **\*** can be used for the instance specifier in conjunction with device categories, for example: **VGS(\*)**, **VCE(\*)**

Expressions are also allowed, for example:

```
.SETSOA M NMOS VGS(*) + VDS(*) = (0,*)
```

- MIN

Minimum value of expression/parameter to check. SOA limits can also be defined using **.DATA** statements for expressions.

- MAX

Maximum value of expression/parameter to check. SOA limits can also be defined using **.DATA** statements for expressions.

**Note**

Expressions are also accepted in the MIN/MAX boundaries, together with parameters. Specifying \* for a MIN/MAX boundary means that the appropriate limit should not be checked.

- **LABEL= " <STRING> "**

Specified as the first argument. The label will appear in the ASCII output file, which can be useful for readability of this file.

- **XAXIS**

Specifies the x-axis length that must be achieved before SOA warnings are printed.

- **IF( EXPR )**

Defines an IF statement where EXPR is the expression for the IF condition. The following operators are allowed:

| | for OR  
 && for AND  
 == for EQUAL  
 < or <= for INFERIOR and INFERIOR or EQUAL  
 > or >= for SUPERIOR and SUPERIOR or EQUAL

IF statements can be nested. Only the following parameters are allowed in IF expressions:

V( ), I( ), P( ), E( ), M( ), and EM( ).

- **THEN**

Defines a THEN statement. Used in conjunction with IF. The user can define a parameter(s) to be checked when the IF statement is true.

- **ELSE**

Defines an ELSE statement. Used in conjunction with IF and ELSE. The user can define a parameter(s) to be checked if the IF statement is false.

- **SUBCKT=<subckt\_list>**

Specifies a list of subcircuits to be checked. Valid for the D and M types.

- **INST=<inst\_list>**

Specifies a list of instances to be checked. Valid for the D and M types.

## Examples

```
SOA check
.width out=80
.model n npn
r1 in out 10k
c1 out 0 1p
q1 c b 0 0 n
r1 c vdd 1k
```

```

vin in 0 pwl 0 0 1n 10 20n 10 21n 100
vdd vdd 0 5
vb b 0 pwl 0 -1 40n 1
.setsoa d r1 i=(*, 3m)
.setsoa m n ic=(-1u, 3m)
.setsoa e IC(q1)/IB(q1)=(*, 100)
.checksoa
.option eps=1u
.tran 1n 40n
.plot tran v(out)
.plot tran i(r1) ib(q1) ic(q1)
.end

```

### **Caution**



This produces the following warning on the screen (or in the *.log* file, if you are simulating in background):

\*\*\*WARNING: SOA DETECTION: See output file for details

The following results will be obtained in the *.chi* file:

```

1*****5-Feb-2001 ***** Eldo v5.4 *****10:47:21*****
OSOA CHECK
0**** SOA INFORMATION TEMPERATURE = 27.000 DEG C
0***** ****
* | R1:
* | I(R1)
* | X AXIS WINDOW: [ 20.32351N 31.73637N ] Value superior to 3.000000e-
03
* | IC(Q1)/IB(Q1)
* | X AXIS WINDOW: [ 28.42081N 32.91932N ] Value superior to
1.000000e+02
* | Q1:
* | IC(Q1)
* | X AXIS WINDOW: [ 35.33011N 40.00000N ] Value superior to 3.000000e-
03

```

The following is an example of using the optional 3rd boundary **XAXIS** parameter:

```
.SETSOA D M1 VGS = (1, 2, 3u) VGS = (0, 4, 1u)
```

this means that when **VGS** is outside the limits [1, 2] for a time-period greater than or equal to 3 $\mu$ s, OR when **VGS** is outside the limits [0, 4] for a time-period greater than or equal to 3 $\mu$ s a warning is given.

**Note**

 Safe Operating Area warnings can be searched automatically using the following AWK script:

```
#!/bin/sh
awk '
BEGIN {
    found = 0
}
($2 ~ /SOA/ && $3 ~ /INFORMATION/) {
    found=1
}
(found==1){
    if ($1 == "* | ")
        print $0
}
' $*
```

The optional label specification can be used, as shown in the example below:

```
.setsoa label="My severe error" m ndig
+ Vg(*)-Vb(*)=(-15.0,15.0)
.setsoa label="My warning" m ndig
+ Vg(*)-Vb(*)=(-22.3,22.3)
```

The ASCII output file, with the label appearing, would look similar to the following:

```
*| M#$I10:
*|   VG(*)-VB(*) LABEL="My severe error"
*|     X AXIS WINDOW: [ 110.0000N 1.00000U ]
*|     Value inferior to -1.50000e+01
*|   VG(*)-VB(*) LABEL="My warning"
*|     X AXIS WINDOW: [ 117.3000N 1.00000U ]
*|     Value inferior to -2.23000e+01
```

The following example shows how the boundary specifications can be used inside SOA.

```
.SETSOA E xup(v(out),4,0,100n,1)
+ -xup(v(out),1,0,100n,1) = (*,3n)
```

Here, SOA notification will appear if the extracted value is higher than 3n.

The following example shows how subcircuit parameters can be used inside SOA.

```
.SUBCKT TEST A B
R1 A B 1k
.SETSOA D R1 V=(-CH,CH)
.ENDS TEST

X1 1 0 TEST CH=25
Vd 1 0 30
.DC vd 10 30 0.1
.PLOT DC i(Vd)
.CHECKSOA
```

```
.END
```

The example below shows multiple expressions can be used in a **.SETSOA** command.

```
.SETSOA E IC(q1)/IB(q1)=(*, 100)
+ xup(v(out),4,0,100n,1) = (*,4n)
```

The following examples show IF statements used in **.SETSOA** commands.

```
.SETSOA M NMOS IF(D(*,W) < 10U) THEN VGS(*) = (*,1.2) ENDIF
```

In the example above, the **VGS** parameter on all NMOS transistors with a device parameter **W** of less than 10 $\mu$ m will be checked with respect to a maximum value of 1.2.

```
.SETSOA M NMOS IF((Id(*) >= 0.1u)&&(Vgs(*)>(VT(*)-0.3)))
+ THEN Vds(*) - VDSS(*) = (0,*) ENDIF
```

In the example above, the IF statement sets up the condition that for all NMOS transistors that have a value of parameter **Id** greater than or equal to 0.1 and parameter **Vgs** is greater than the value computed by **VT-0.3**, then the value calculated by **Vds(\*) - VDSS(\*)** will be checked with respect to a minimum value of 0.

```
.SETSOA E IF(P1>0) THEN IF(P2>0) THEN V(1)=(*,0.5) ENDIF ENDIF
```

In the example above, if parameter **P1** is greater than zero, then the second IF statement will be performed. The second IF statement defines that parameter **V(1)** will be checked with respect to a maximum value of 0.5 if parameter **P2** is greater than zero.

```
.DATA vout
+ tt      voutmax      voutmin
+ 0       0.1           0.0
+ 80n    0.1           0.0
+ 120n   5.0           4.5
+ 200n   5.0           4.5
.ENDDATA
.SETSOA label=vout_correct E v(OUT)=[vout(voutmin),vout(voutmax)]
```

The example above shows how SOA limits can be defined using **.DATA** commands.

The following example shows how SOA limits can be checked from the top level using wildcards and across instance/subcircuit lists at a lower level of hierarchy.

```
.SUBCKT NHVT D G S B PARAM: W=1.0 L=0.5 xl=0
.MODEL NHVT NMOS LEVEL=53 XL = p1
M1 D G S B NHVT W=W L=L
.ENDS
X1 D G S B NHVT xl = 0.1u
X2 D G S B NHVT
X3 D G S B NHVT
```

Eldo will create internally two models: one named **X1.NHVT**, used by device **X1.M1**, and another one named **NHVT.NHVT**, shared by **X2.M1** and **X3.M1**. If you wish to perform some SOA checks on the model **NHVT**, specify:

```
.SETSOA M '*.NHVT' VDS=(-1.2,1.2)
```

An equivalent syntax is:

```
.SETSOA M NHVT SUBCKT=NHVT VDS=(-1.2,1.2)
```

It will check for *VDS* on the three instances *X1.M1*, *X2.M1*, and *X3.M1*.

Specifying the following, Eldo will check only for instance *X1.M1*:

```
.SETSOA M NHVT INST=X1 VDS=(-1.2,1.2)
```

# .SIGBUS

## Set Bus Signal

```
.SIGBUS BNAME | BNAMEMSB:LSB | BNAME[MSB:LSB] | BNAME<MSB:LSB>
+ [VHI=VAL1] [VLO=VAL2] [TFALL=VAL3] [TRISE=VAL4]
+ [BASE=OCTAL|DEC|BIN|HEXA] TN VAL {TN VAL} [P] [SIGNED=NONE|1COMP|2COMP]
.SIGBUS BNAME | BNAMEMSB:LSB | BNAME[MSB:LSB] | BNAME<MSB:LSB>
+ [VHI=VAL1] [VLO=VAL2] [TFALL=VAL3] [TRISE=VAL4]
+ [THOLD=VAL5] [TDELAY=VAL6] [BASE=OCTAL|DEC|BIN|HEXA]
+ PATTERN $(PAT) {$(PAT)} | VAL {VAL} [Z] [SIGNED=NONE|1COMP|2COMP]
```

This command sets signals on a bus `BNAME` that has been previously defined via the `.SETBUS` command. `.SETBUS` can be implicitly declared, providing that `BNAMEMSB:LSB`, `BNAME[MSB:LSB]` or `BNAME<MSB:LSB>` syntax is used.

In its first form shown above, `.SIGBUS` defines each transition value and the time at which it occurs as a list of `TN VAL` value pairs. In its second form, all consecutive signal values are listed in order after the `PATTERN` keyword, and a hold time specifies the duration of each signal value specified.

Bus values for both the standard and pattern definitions accept exactly the same syntax.

- parameters are denoted `$(P)` or `'P'`
- values can be specified with or without double quotes and use the notation `HX[...]`, `DX[...]`, `OX[...]`, and `BX[...]` to specify in which base this number is given (by default it is the base of the bus). All these notations can be mixed in the same bus definition. For example:

```
.sigbus a[0:7] VHI=3 VLO=0 THOLD=10ns BASE=BIN
+ PATTERN $(PAT_RAMP) 101 "001" 'PAT2' BX010 "DX123"
```

Parameters can only be strings, except for buses declared with `BASE=DEC`. This is mandatory to allow sweeps. If this rule is not satisfied, the following message is issued:

```
ERROR 1545: COMMAND .SIGBUS: bus A, parameters of type real can only be
used with buses using base=DEC
```

If the parameter holds a decimal value, the message issued is:

```
ERROR 1544: COMMAND .SIGBUS: decimal non-integer value found in bus A
```

## Parameters

- `BNAME`

Bus name, previously defined via the `.SETBUS` command, or can be implicitly declared with `MSB:LSB` syntax specified. A limitation of the simplified `BNAMEMSB:LSB` notation is that bus names cannot contain numbers, i.e. `BUS18:0` will be equivalent to `BUS[18:0]` but not `BUS1[8:0]`.

- **MSB : LSB**  
Series of bit names, the most significant bit being defined first. This mechanism can be used to implicitly declare **.SETBUS**.
- **TN**  
Time in seconds at which the bus signal is equal to **VAL** Volts.
- **VAL**  
Bus signal voltage level at time **TN**.
- **VHI=VAL1**  
Upper bus signal voltage level.
- **VLO=VAL2**  
Lower bus signal voltage level.
- **TFALL=VAL3**  
The time for a falling signal to reach **VLO** Volts. Default is 2ns.
- **TRISE=VAL4**  
The time for a rising signal to reach **VHI** Volts. Default is 2ns.
- **THOLD=VAL5**  
The hold time for each signal value, measured from the 50% point of the transition.
- **TDELAY=VAL6**  
The delay time before a signal pattern starts.
- **BASE**  
Keyword indicating that the bus signal number system is to be defined. The default number system is decimal.

**OCTAL**

Keyword indicating that bus signals are defined in octal.

**DEC**

Keyword indicating that bus signals are defined in decimal.

**BIN**

Keyword indicating that bus signals are defined in binary.

**HEXA**

Keyword indicating that bus signals are defined in hexadecimal.

- **P**  
Keyword indicating the bus signal is periodic.
- **PATTERN**  
Keyword indicating the bus signal is pulsating (digital-like). Integer values can be specified, a bus value specifying `base=bin pattern 101` has the same effect as specifying

base=dec pattern 5. The **PATTERN** on a **.SIGBUS** command can also be specified via a parameter using \$(PAT) where PAT is the parameter as specified in the **.PARAM** statement. For example:

```
.PARAM PAT=3
.SIGBUS toto VHI=1.8 VLO=0 TFALL=100ps
+ TRISE=100ps THOLD=5ns TDELAY=0 BASE=DEC
+ PATTERN $(PAT)
```

- \$(PAT)

PAT is a previously declared parameter in the special case of the **PATTERN** specification above.

- Z

Releases the applied signal of a bus, i.e. when the Z state is active, the signal will be disconnected from the node, and the node will be computed by Eldo as if it were a non-input signal.

- **SIGNED=NONE | 1COMP | 2COMP**

Defines how negative values inside bus patterns are managed.

**NONE**—negative values are forbidden (default)

**1COMP**—the one's complement of the value is done

**2COMP**—the two's complement of the value is done

## Example

**.SIGBUS** can be used to implicitly declare **.SETBUS**, providing that BNAMEMSB:LSB, BNAME[MSB:LSB] or BNAME<MSB:LSB> syntax is used as shown below:

```
.SIGBUS BUS2:0
.SIGBUS SELECT[2:0]
.SIGBUS DOUT<3:0>
```

is equivalent to:

```
.SIGBUS BUS BUS2 BUS1 BUS0
.SIGBUS BUS
.SETBUS SELECT SELECT[2] SELECT[1] SELECT[0]
.SIGBUS SELECT
.SETBUS DOUT DOUT<3> DOUT<2> DOUT<1> DOUT<0>
.SIGBUS DOUT
```

A Z state can be specified for a single signal of a bus, shown in the example below:

```
.SIGBUS B<1:0> VHI=2 VLO=0 TFALL=1n TRISE=1n THOLD=200n
+ TDELAY=0 BASE=BIN
+ PATTERN 00 01 0Z 00 01
```

## **.SINUS**

### Sinusoidal Voltage Source

```
.SINUS NODE VO VA FR [ TD [ THETA ] ]
```

#### Parameters

- **NODE**  
Node name between which the source is connected to and ground.
- **VO**  
Offset voltage in volts.
- **VA**  
Sine wave amplitude in volts.
- **FR**  
Frequency in Hertz.
- **TD**  
Delay time in seconds. Optional.
- **THETA**  
Damping factor in  $s^{-1}$ . Optional.

---

**Note**

The generated waveform is described by:

$$V = VO + VA \times \sin(2\pi \times FR(t - TD)) \times \exp(-(t - TD) \times THETA)$$

---



For more information, refer to the [Sources](#) chapter.

---

#### Example

```
.sinus n2 4.0 1.0 1meg 0.0
```

Specifies a sine wave applied between node `n2` and ground with a 4V offset voltage, 1V amplitude, 1MHz frequency and zero delay.

## .SNF

### Spot Noise Figure

```
.SNF INPUT=(LIST_OF_DEVICES) OUTPUT=(LIST_OF_DEVICES)
+ [INPUT_TEMP=VAL] [NOISETEMP=VAL]
```

A new number, SNF (Spot Noise Figure), is calculated. When specifying the **INPUT\_TEMP=VAL**, the noise at the input will be computed at **INPUT\_TEMP**. When **NOISETEMP** is specified in the **.SNF** command, this is equivalent to specifying the parameter on all sources in the netlist. This option will take priority over the **INPUT\_TEMP** parameter whether it is used on a source or in the **.SNF** command. Then, the total noise contribution will be calculated by:

$$\text{SNF} = \frac{(\text{Input noise at INPUT\_TEMP} + \text{Other noise sources at TREF} - \text{Output noise at TREF})}{\text{Input noise at INPUT\_TEMP}}$$

#### **Note**

 When **NOISETEMP** is specified, it will replace **INPUT\_TEMP** in the equation above.

If neither **INPUT\_TEMP** nor **NOISETEMP** are not specified, then the method of calculation will be:

$$\text{SNF} = \frac{(\text{Noise in circuit} - \text{noise due to output})}{\text{Noise due to source}}$$

### Parameters

- **LIST\_OF\_DEVICES**

This can contain wildcard '\*' characters anywhere in the name. Each item must be separated with a white space or a comma. Brackets are optional. However, if more than one name is provided, then commas and brackets *must* be used.

### Example

```
.SNF INPUT=(XM1*.M*, XM[1-3]* ) OUTPUT=M19
```

The example above shows how it is possible to use wildcard characters.

```
.SNF INPUT=(XM1, XM2*) OUTPUT=M19
```

This number is frequency dependent, hence a curve is generated that can be plotted via **.PLOT NOISE SNF** in exactly the same way one can plot:

```
.PLOT NOISE INOISE
```

#### **Note**

 The **.SNF** command provides results only if a **.NOISE** or **.SSTNOISE** (Eldo RF) command is present.

## **.SOLVE**

### Sizing Facility

```
.SOLVE PARAM param_name MIN MAX expr=expr [TOL=VAL]
+ [RELTOL=VAL] [GRID=VAL]
.SOLVE obj_name [W|L] MIN MAX expr=expr [TOL=VAL]
+ [RELTOL=VAL] [GRID=VAL]
.SOLVE CNAME [W|L] MIN MAX OPSIZE [TOL=VAL]
```

Used in conjunction with a DC analysis only. Eldo sizes the specified component to match given constraints. This command can be used to solve the value of a parameter. The target may be **V(NODE)=<VAL>**, or **I(Vxx)=<VAL>**, or an expression (e.g. **V(NODE1)+V(NODE2)**).

### Parameters

- **CNAME**  
Name of the component to size. Legal names are **R**, **MOS**, **Vxx**, **Ixx**.
- **MIN**, **MAX**  
Bounds to search for a solution, expressed in correct units.
- **OPSIZE**  
Request component sizing so that the voltage on a specific node or current through a voltage source matches given constraints. The syntax is as follows:

**V(NODE)=VAL**

Specifies that the component should be sized so that the voltage on the specified node be equal to **VAL**, with tolerance **TOL**.

**I(Vxx)=VAL**

Specifies that the component should be sized so that the current through the specified voltage source be equal to **VAL**, with tolerance **TOL**.

- **W**, **L**  
Keywords identifying width or length for MOS transistors. Optional.
- **TOL=VAL**  
Required accuracy, specified in absolute units. Default is the value of the **EPS** parameter. Optional.
- **GRID=VAL**  
Means the result must be a multiple of **VAL**. Optional.
- **RELTOL=VAL**  
Adds the relative tolerances. Optional.

There are two points to bear in mind when using the **.solve** command:

1. **.solve** works in the case where there is a 0 in the interval, and if the function is monotonous in the interval, but cannot be used to find a MIN or a MAX.
2. the default absolute tolerance for **.solve** is **EPS**; specify another value in the **.solve** command to reset this default.

## Examples

```
.solve m1 w 10u 50u v(2)=0.245v tol=1u
```

Specifies that the width of transistor **m1** be sized to make the voltage on node 2 equal to 0.245V with a tolerance of 1 $\mu$ V. Maximum and minimum values of the width are 50 $\mu$ m and 10 $\mu$ m respectively.

```
.solve m3 l 1.5u 3.5u i(v2)=0.2m tol=0.05m
```

Specifies that the length of transistor **m3** be sized to make the current through voltage source **v2** equal to 0.2mA, within a tolerance of 0.05mA. Maximum and minimum values of length are 3.5 $\mu$ m and 1.5 $\mu$ m respectively.

## **.STEP**

### Parameter Sweep

```

.STEP TEMP | DIPOLE INCR_SPEC
.STEP MOS W | L INCR_SPEC
.STEP MNAME PARAM_NAME INCR_SPEC
.STEP PARAM PARAM_NAME INCR_SPEC {PARAM_NAME INCR_SPEC}
.STEP PARAM PARAM_NAME INCR_SPEC, {[VALSTART] VALSTOP VALUE}
.STEP ITEM INCR_SPEC {ITEM2 BOUND}
.STEP (ITEM1,ITEM2... ITEMn)
+ LIST | = (VALi1, VALi2... VALin)... (VALj1, VALj2... VALjn)

```

Used to perform several simulations while sweeping one circuit parameter or several circuit parameters simultaneously (multiple-sweep). Nested sweeps can also be specified. Further nesting levels can be applied by additional **.STEP** commands. There is no limit to the depth (levels) of nested sweeps possible.

When specifying multiple sweeps, secondary `incr_spec`'s cannot define the number of points. Eldo will determine the number of points from the first `incr_spec` defined and then set the same number for subsequent increment steps.

This makes it possible to set multiple items and have them change concurrently, then subsequent **.STEP** commands would provide the next level of nesting.

The series of parameters on the right of the equals sign indicate a series of 'n' values enclosed in brackets, which must be specified. The number of terms of this series corresponds to the number of simulations, and 'n' items would change at each simulation.

To perform a parameter vector sweep, with all parameters taking their values from lists, use the last syntax shown above and ensure that `ITEM` is correctly specified for parameter items, for example specifying `(P(IG1), P(LR1))` to sweep parameters IG1 and LR1.

The command has the ability to sweep several parameters at the same time using the **.STEP PARAM** syntax. Multiple increment step specifications can be made. This is in order to be able to have "windows" with more points than in other regions.

**.MPRUN** can be used to take advantage of multi-processor machines for the **.STEP** command. Please see "[.MPRUN](#)" on page 10-166 for further information.

By default, the second run in **.STEP** uses the result of the previous STEP as an initial guess. For example `.step temp 0 70` in a tran analysis. Setting option **NOMEMSTP** means Eldo will not use the results of the previous STEP run as an initial guess for the next one.

### Parameters

- **TEMP**

Keyword indicating that temperature is to be swept.

- **DIPOLE**  
Name of the Dipole (R, C, L) component whose value is to be swept.
- **MOS**  
Name of the MOS component whose **W** or **L** parameter is to be swept.
- **W, L**  
Keywords identifying either MOS width or length respectively.
- **MNAME**  
Model name, of which a parameter **PARAM\_NAME** is to be swept.
- **PARAM\_NAME**  
Name of the model parameter to be swept.
- **PARAM**  
Keyword to identify that a globally declared parameter is to be swept. Multiple increment step specifications can be made. Non-primitive parameters can be specified.
- **LIST**  
Keyword specifying that a list of individual values are to be swept.

**Note**

For multiple increment step specifications: Be careful of the character “,” which is used for separating the different windows. If not specified, **VALSTART** is assumed to be the **VALSTOP** value of the preceding window.

**Note**

**.TRAN ... SWEEP** will not work if **PARAM** is specified and a warning message will be issued.

- **ITEM**

Can be one of the following:

**P(global\_var)**  
**E(device,parameter)**  
**M(model\_name,parameter)**  
**EM(device\_name,model\_parameter\_name)**  
**LIB(libname)**

**P**

Parameter item.

**E**

Element item.

**M**

Model item.

**EM**

Model item attached to a specific Element. Affecting the model parameter of this new private model created for `device_name`, leaves the original model unchanged. Once **.STEP** is complete, this private model is discarded, and the original model is attached back to the device.

**LIB**

Library item. A list of strings is expected, of type `LIST` as specified in `INCR_SPEC` below. Used to make several Eldo runs, each of them using a different variant of the libraries.

- **INCR\_SPEC**

Specification of the increment step may be either of the following:

```
VALSTART VALSTOP [DEC|OCT|LIN|INCR] VALUE
LIST {VAL1 VAL2 ... VALN}
```

**VALSTART**

Initial value of sweep.

**VALSTOP**

Final value of sweep.

**LIN**

Optional keyword, explicitly selecting a linear sweep.

**INCR**

Incrementing value of sweep. This is the default.

**DEC**

Keyword to select a logarithmic sweep.

**OCT**

Keyword to select an octave sweep.

**LIST**

Keyword specifying that a list of individual values are to be swept. Accepts character strings for dealing with the `ITEM` of type **LIB**(`libname`).

**VALUE**

Specifies the number of points per decade and octave for DEC and OCT respectively, the total number of points for LIN, and the value of increment for INCR.

**VAL1 .. VALN**

A list of values to be applied to the specified parameter at each point in the sweep.

- **BOUND**

Same as `INCR_SPEC`, except that only boundaries are provided. The increment value is chosen according to the number of runs imposed by the first swept parameter.

**Note**

 To use this command with a device that is situated in a subcircuit, use the nested output format.



For more details, refer to “[.PRINT](#)” on page 10-254.

Parameters **TEMP** or **PARAM** specified in the **.STEP** command may also appear on the x-axis of a graphical output when performing a DC analysis.

Eldo will select the model to be assigned to MOS devices according to the geometric size of each device, even if these geometric sizes are modified at run-time via **.STEP** commands. In previous versions, the selection of the model was done just once at the very beginning of the simulation, and was not changed at run time.

## Examples

```
.step temp 25 35 2
```

Specifies that simulator runs should be carried out between the temperatures 25 and 35 degrees in increments of 2 degrees Celsius.

### Note

The **INCR** keyword is optional as this is the default.

```
.step mos_1 w 25u 40u 5u
```

Specifies that simulator runs should be carried out with the width of the transistor **mos\_1** being swept from 25  $\mu\text{m}$  to 40  $\mu\text{m}$  in increments of 5  $\mu\text{m}$ .

```
.step qmod rb 80 110 10
```

Specifies that simulator runs should be carried out with the base resistance parameter of devices with transistor model **qmod** being swept from  $80\Omega$  to  $110\Omega$  in increments of  $10\Omega$ .

```
c1 1 2 20p  
...  
.step c1 1p 10p 1p
```

Specifies that simulator runs should be carried out with the capacitor **c1** being swept from 1 pF to 10 pF in steps of 1 pF.

```
.step x1.r1 0.1k 0.5k 0.1k
```

Specifies that simulator runs should be carried out with the value of the resistor **r1**, instantiated using the subcircuit instance name **x1**. The resistor value should be swept from  $0.1\text{k}\Omega$  to  $0.5\text{k}\Omega$  in increments of  $0.1\text{k}\Omega$ .

```
.param r1 1k  
...  
.step param r1 1k 2k 1k, 2k 4k 500
```

Specifies that the value of resistor **r1** will be swept from  $1\text{k}\Omega$  to  $2\text{k}\Omega$  with a step of  $1\text{k}\Omega$ , and then from  $2\text{k}\Omega$  to  $4\text{k}\Omega$  with a step of  $500\Omega$ .

---

```
.param p2=100k
.param p1=sqrt(p2)
...
.step param p1 1k 5k 1k
```

This example shows how a non-primitive parameter `p1` can be specified.

The following example shows how the command can be used to define a multiple run by a list of parameters:

```
.step param (p1 p2 ...pn) list (val11 val21 ...valn1)
+ ...(valj1...valjn)
```

This will perform `j` runs:

```
run1: p1=val11, p2=val21 ... pn=valn1
run2: p1=val21 ...
...
runj: p1=valj1, ... pn=valjn
```

The following example shows how the command can be used to sweep Models attached to specific Elements:

```
M1 ... NMOS W=...
M2 ... NMOS W=...
.MODEL NMOS NMOS VT0 = 1

.STEP EM(M1,VT0) 0 5 1
```

For this example, Eldo will:

1. Emulate a command `.MODDUP M1`  
A new model `NMOS.M1` is created, which is private to `M1`.  
All parameters of `NMOS.M1` are initialized to the NMOS values.
2. Five Simulations will be performed, for `NMOS.M1.VT0` varying from 0 to 5.  
`NMOS.VT0` remains at 1.0.
3. `NMOS.M1` is discarded: model attached to `M1` is NMOS again.

---

### Note



As in the `.MODDUP` case, parameter dependencies are propagated.

```
.PARAM P1 = 550
M1 ... NMOS W=...
M2 ... NMOS W=...
.MODEL NMOS NMOS VT0=1 UO=P1
```

```
.STEP (EM(M1,VT0),P(P1)) = (1,550) (2,600)
```

In the example above, for both `NMOS.M1` and `NMOS`, the `UO` value will be identical.

---

The following examples show how the command can be used to sweep Parameters, Elements, Models, or Models attached to specific Elements:

```
.PARAM P1=1u
.MODEL NMOS NMOS LEVEL=1 VTO=2
M1 p1 p2 p3 p4 NMOS w=p1 l=3u
M2 p1 p2 p3 p4 NMOS w=p1 l=3u
...
.end
```

After the above specification, the following four **.STEP** commands are valid:

```
.STEP P(P1) 1u 10u 1u
```

Parameter P1 varies from 1 $\mu$  to 10 $\mu$  in steps of 1 $\mu$ .

```
.STEP E(M1,l) 3u 4u 1u
```

Length of Element M1 varies from 3 $\mu$ m to 4 $\mu$ m in steps of 1 $\mu$ m.

```
.STEP M(NMOS,VT0) 1 3 1
```

Parameter VT0 of model NMOS varies from 1 to 3 in steps of 1.

```
.STEP EM(M2,VT0) 1 4 1
```

A private model will be assigned to M2. VT0 of model attached to M2 will vary from 1 to 4 in steps of 1; VTO of model attached to M1 will remain unchanged.

The following defines a two-level nested sweep. The first loop is defined by the first **.step**. The resistor, R22, will be swept from 20k $\Omega$  to 100k $\Omega$  in a linear 1k $\Omega$  increment. This defines 81 points for the first loop.

```
.STEP E(R22,R) 20K 100K LIN 1K M(MOS1,COX) 0.4M 0.3M
.STEP P(GLOBAL_VAR) 1 1000 DEC 10 E(C_CUPL,C) 100p 500p LIN
```

The second item, the MOS model named MOS1 would have the parameter COX swept from 0.4M down to 0.3M in increments that produce 81 points (i.e.  $abs(0.3M-0.4M)/81$ ). The order is important and would need to be preserved. The number of points is defined by the first increment specification so that the entire **.step** command will have a consistent number of points.

The resistor and COX parameter would increment/decrement together:

```
(20K, 0.4M)
(21K, 0.38977M)
.
.
.
(100K, 0.3M)
```

The second nesting level is defined by the second **.step** command. The global parameter named GLOBAL\_VAR will be swept from 1 to 1000 in decade logarithmic steps, producing 31 increments. The capacitor, C CUPL, would be stepped from 100pF to 500pF linearly with 31 steps together with the GLOBAL\_VAR parameter.

```
.STEP PARAM (E(R1,R) TEMP LIB(corna.lib)) LIST
+ (20k 27 typ)
+ (10k -40 fast)
```

Specifying the above means that two simulations will be performed:

- The first with R1 value of 20k, TEMP is 27 and uses the LIB variant typ for file corna.lib.
- The second with R1 value set to 10k, a TEMP value of -40 and the LIB variant fast of the file corna.lib.

The following example shows a parameter vector sweep, with all parameters taking their values from lists. Eldo will sweep parameters IG1 and LR1 over four simulations. The first simulation with IG1 at 10u and LR1 at 1000u. The last simulation with IG1 at 100u and LR1 at 100u.

```
.param IG1=10u LR1=1000u
.step (p(IG1),p(LR1)) LIST (10u, 1000u) (20u, 500u) (50u, 200u)
+ (100u, 100u)
i1 1 0 ig1
r1 1 0 '10e6*lr1'
.plot dc v(1)
.dc
```

The following example shows how the **.STEP** command is used to make several Eldo runs, each of them using a different variant of the libraries, using the keyword LIB(libname):

```
.LIB mylib TYP
...
.STEP LIB(mylib) TYP MIN MAX
.END
```

Three simulations will be performed: one using the variant TYP of mylib, another one using the variant MIN, and a last one using the variant MAX.

---

**Note**

 The LIB specification can be used with any other specification.

---

**Note**

 TYP, MIN, and MAX are not keywords, but are the character strings that can appear as the 2nd arguments of the **.LIB** command (**.LIB** FNAME [LIBTYPE]). Usually, variant names are TYP, MIN, and MAX, but they can be any string.

---

```
.STEP P(P1) 1 2 0.5 LIB(mylib) TYP MIN MAX
```

Here, Eldo will perform three simulations:

- one with both P1=1.0 and mylib using variant TYP
- one with both P1=1.5 and mylib using variant MIN
- one with both P1=2.0 and mylib using variant MAX

## Limitations on Library Variants

1. Subcircuits are not replaced.

When **.LIB** is specified in the input file, it can be used to select a subcircuit to be

included in the design. Taking a subcircuit from another library than the library specified in the input file could result in a change of topology (and changing topology of the current design is strictly impossible), therefore Eldo would issue an error whenever the user attempted to substitute one subcircuit for another one.

2. Models which were used by Eldo-XL cannot be substituted.
3. Switching between GUDM and non-GUDM models is prohibited.  
Similarly, switching between GUDM models is prohibited.
4. Only MOS, BJT, DIODE, JFET, R, L, and C `.model` commands can be substituted.  
Attempting to substitute other kinds of models will lead to an error.

## **.SUBCKT**

### Subcircuit Definition

```

.SUBCKT NAME NN {NN} [(SWITCH|ANALOG|OSR|DIGITAL)]
+ [(NONOISE)] [(INLINE)] [PARAM: PAR=VAL {PAR=VAL}]
...
<CIRCUIT_COMPONENTS>
...
.ENDS [NAME]
.SUBCKT LIB FNAME SNAME [LIBTYPE]

```

A subcircuit consists of Eldo, FAS and FIDEL elements. The subcircuit is defined in the circuit description file by groups of elements beginning with a **.SUBCKT** command and terminating with a **.ENDS** command. All components and nodes used in the subcircuit definition are known only locally, unless they have been globally defined via the **.GLOBAL** or **.PARAM** commands. There is no limit to the size or complexity of the subcircuit.

---

**Note**

 Care must be taken when using the **.PARAM** command to ensure that no unwanted parameter assignments are made.

---

Subcircuit definitions may be nested. The inner subcircuit definitions are local to the subcircuit definition in which they are defined, i.e. they are not valid outside of it.

Subcircuits may be parameterized. Symbols may be used as values within the subcircuit body which may be assigned values when the subcircuit is instantiated. A subcircuit may be stored in a library file, in which case the second syntax above and the **.LIB** or **.ADDLIB** commands should be used. The second syntax allows no continuation lines.

The subcircuit may optionally be simulated using the differentiated accuracy system. The iteration technique used to process the subcircuit is chosen by the **(ANALOG)** optional keyword. Another optional keyword **(SWITCH)** is available which can be specified instead of or as well as **(ANALOG)**. Usually, a circuit with **(ANALOG)** specified indicates 'high accuracy' and therefore it is more likely not to specify **(SWITCH)**.

---

**Note**

 **.MACRO** is equivalent to **.SUBCKT**.

---

### Parameters

- **NAME**

Name of the subcircuit. May also be specified with the **.ENDS** command if desired.

- **NN**

Names of the subcircuits nodes. Nodes are referenced in the same order that they are called in the subcircuit call statement. Ground (or 0V) may not be referenced in this list.

- **LIB**  
Keyword indicating a subcircuit library file is to be used.
- **FNAME**  
Name of the library file that contains the subcircuit description.
- **SNAME**  
Name of the subcircuit stored in library file **FNAME**. See the [Library variant management](#) in the **.LIB** command description for further details.
- **(ANALOG)**  
Keyword used with the differentiated accuracy system indicating that the subcircuit should be solved using Newton block iteration techniques. These techniques are used in conjunction with the **EPS** parameter in the **.OPTION** command. **(ANALOG)** basically means “high accuracy.”
- **(SWITCH)**  
Keyword used with the differentiated accuracy system. See also [.OPTION NOSWITCH](#) page 10-180.



Before using the differentiated accuracy system see the relevant section in the [Speed and Accuracy](#) chapter.

---

- **(OSR | DIGITAL)**  
Used to stop propagation of the **ANALOG** flag across the hierarchy. In addition the flag will request Eldo to use OSR in the selected blocks, if possible (i.e. MOS subcircuit with **OSR** flag could then be solved by OSR, but BJT subcircuit will still be solved by Newton even if flag **OSR** is set).
- **(NONOISE)**  
Specifies that all subcircuit objects are noiseless. However, the appearance of a flag **NOISE** at a lower level of subcircuit hierarchy overrides the effect of a **(NONOISE)** flag at a higher level. This parameter must be specified before any **PARAM:** specification, if any.
- **(INLINE)**  
When specified, Eldo will not print the full hierarchical name of the device which has the same name as the **.SUBCKT**. The effect of this option is for print out in the **.OP** table only. This parameter must be specified before any **PARAM:** specification, if any.
- **PARAM:**  
Keyword indicating parameter allocation within the subcircuit definition.
- **PAR=VAL**  
Specifies that the parameter **PAR** is assigned the value **VAL** inside the subcircuit, unless another value is assigned to the parameter when the subcircuit is instantiated.

**i** In case of M factor usage, see also **.option m53** on page 11-28. With some descriptions, Eldo will generate the following error message:  
ERROR 712:SUBCKT "XXX": cannot define a parameter named 'M'

- NAME  
Name of the subcircuit. May or may not be specified.
- LIBTYPE  
Name of a library variant to be used.

**i** For details of subcircuit instance syntax refer to [page 4-1](#).

## Examples

```
.subckt inv n1 n2 n3 n4 param:p1=10u r1=8u
m1 n3 n2 n4 0 pmos w=p1 l=r1
m2 n3 n2 n1 0 nmos w=p1 l=r1
.ends inv
...
x1 vss n8 n9 vdd inv p1=5u r1=5u
```

Specifies the subcircuit `inv`. Nodes declared in the subcircuit are local. The parameters `p1` and `r1` are assigned values both at instantiation and in the **.SUBCKT** command.

```
.subckt w1 n1 n2 n3 n4 (analog)
+ param:r1=10u r2=5u
m1 n1 n4 n5 0 nmos w=r1 l=r2
m2 n5 n5 n2 0 nmos w=r1 l=r2
m3 n4 n5 n3 0 nmos w=r1 l=r2
.ends w1
...
x1 vss 0 0 vdd w1
```

Specifies the subcircuit `w1`. This subcircuit is defined to be solved only using Newton block iteration techniques due to the presence of the **(ANALOG)** optional keyword.

```
*SUBCKT definition
.subckt inv n1 n2 n3 n4
m1 n3 n2 n4 n4 pmos w=w1 l=l1
m2 n3 n2 n1 n1 nmos w=w2 l=l2
.param l1=5u
.ends inv
...
*main circuit
x1 vss n8 n9 vdd inv w1=60u w2=20u
.param l2=5u
```

Specifies the subcircuit `inv` using variables `w1`, `l1`, `w2` and `l2`. Note the use of the **.PARAM** command in this example. The value of the parameter `l1` has been assigned a value in the subcircuit `inv` and so is local to this subcircuit, whereas the parameter `l2` has been

assigned a value on the main circuit level and so any subsequent instances of the parameter 12 in other subcircuits would also be assigned this value.

```
.subckt outer n1 n2 n3 n4
q1 n1 n5 n6 pbip
...
    .subckt inner n1 n2 n3
    r1 n1 n3 1k
    r2 n3 n8 4k
    ...
    .ends inner
x1 n10 n12 vdd inner
...
.ends outer
x1 n21 n25 n30 vdd outer
```

Specifies and instantiates a subcircuit `inner` nested inside another subcircuit `outer`.

## Order of Precedence of Subcircuit Parameter Assignments

Parameters used within a subcircuit may be defined in several ways.

- In the subcircuit instantiation:

```
x1 N1 N2 N3 MY_SUBCIRCUIT P3=10
```

- Internally, within the subcircuit:

```
.SUBCKT MY_SUBCIRCUIT
.PARAM p3=20
.ENDS
```

- In the subcircuit declaration:

```
.SUBCKT MY_SUBCIRCUIT P1 P2 P3 PARAM: p3=2
```

- Or globally in the top level netlist:

```
.PARAM p3=5
x1 N1 N2 N3 MY_SUBCIRCUIT
```

The precedence of these parameter assignment methods is, in descending order, subcircuit instantiation, internal subcircuit, subcircuit declaration which acts as a default only, and finally global declaration.

Use option `PARHIER` to control the priorities for parameters.

**Note**

 .PARAM assignments are order dependent within the netlist. Thus, a parameter value assigned using the .PARAM command will only apply to subcircuits instantiated after this .PARAM command. Therefore, for the below:

```
.SUBCKT INV
.PARAM p1=v0
.ENDS
x1 ... inv
.param p1=v1
```

This is accepted, and p1 retains the value of v0 within x1 (when option PARMIER is set to local).

## Accessing Nodes Inside Subcircuit Instances from Outside

The following example illustrates how inner nodes may be accessed from outside of a subcircuit instance in which they defined.

```
.subckt inner n1 n2 n3
r1 n1 n3 1k
r2 n3 n8 4k
q27 n1 n7 n8 nbip
...
.ends inner
.subckt outer n1 n2 n3 n4
q1 n1 n5 n6 pbip
...
x1 n10 n12 vdd inner
...
.ends outer
x13 n21 n25 n30 vdd outer
cmill1 x13.n1 x13.n5 1.2p
cmill2 x13.x1.n1 x13.x1.n5 0.9p
```

Where cmill1 is the Miller Capacitance of q1 located inside the subcircuit x13 and cmill2 is the Miller Capacitance of q27 located inside x1 which in turn is located inside x13. Note that in this example, the capacitances are declared from outside of the subcircuit definitions.

## (INLINE) keyword

The following example illustrates how the (INLINE) keyword can be specified on the .SUBCKT instance, so that Eldo will not print the full hierarchical name of the device which has the same name as the .SUBCKT.

## Example

```
x1 ... MFOO
x2 ... MFOO
.SUBCKT MFOO ... (INLINE)
```

```
M1 ...
MFOO
M2
...
.ENDS
```

Then, in the OP table, it will not be X1.MFOO and X2.MFOO which appear, but simply X1 and X2.

X1.M1, X2.M1, X1.M2 and X2.M2 names will be printed out unaffected.

### -compat flag

When Eldo is run with the -compat flag, the usage of **.GLOBAL** node names in the **.SUBCKT** definition node list are affected. Eldo will check the node names in the subcircuit list definition prior to the global node list. For example:

```
.GLOBAL QWE
VG QWE 0 DC 3
* A .SUBCKT with a global node name in argument list
.SUBCKT B QWE
R1 QWE 0 1K
.ENDS
V2 PIN2 0 DC 2
X2 PIN2 B
```

Without the -compat flag specified, Eldo will connect X2 and R1 between global node QWE and 0. In other words, only the voltage source V2 will be connected to PIN2.

With the -compat flag specified, X2 and R2 will be connected between PIN2 and 0.

## .SUBDUP

### Subcircuit Duplicate Parameters

**.SUBDUP** SUBCKT\_NAME

This command is used to inform Eldo of the duplicate parameters and models which are local to the subcircuit. Also, it allows access to parameters in the hierarchical form for SimPilot. This command is useful when Eldo is running in conjunction with SimPilot.

# **.TABLE**

## Value Tables

```
.TABLE NAME AC|DC|TRAN (X1 Y1) {(XN YN)}
```

This command defines tables of run time values to be used in **AC**, **DC** or **TRAN**sient analyses. Values from the table may then be accessed from arithmetic expressions using the function call **TABLE(NAME)**.

## Parameters

- **NAME**  
Name of the table being defined.
- **AC**  
Specifies an AC analysis for which the data table is to be used. An analysis type MUST be specified.
- **DC**  
Specifies a DC analysis for which the data table is to be used. An analysis type MUST be specified.
- **TRAN**  
Specifies a transient analysis for which the data table is to be used. An analysis type MUST be specified.
- **Xxx Yxx**  
Data value pairs, whose units depend on the analysis type.

## Example

```
.table ac_table_ex ac (1 1) (10k 10)
.defwave wave1=vdb(s)-table(ac_table_ex)
```

Here, `ac_table_ex` contains the AC analysis values 1 at 1Hz and 10 at 10kHz.

## **.TEMP**

### Set Circuit Temperature

```
.TEMP TS {TS}
```

The **.TEMP** command may be used to execute several successive simulations at various temperatures. The **.TEMP** command works for all analysis types. All input data for Eldo is assumed to have been measured at 27°C. Eldo also assumes a nominal temperature of 27°C unless a **TNOM** statement is present in the **.OPTION** command.

---

**Note**

Any model temperature defined by the **TMOD** parameter has priority over the **.TEMP** command, and furthermore, the **T** parameter has priority over both **TMOD** and **.TEMP**, i.e. **T** has the highest priority.

---

**.MPRUN** can be used to take advantage of multi-processor machines for the **.TEMP** command.



Please see “[.MPRUN](#)” on page 10-166 for further information.

---

### Parameters

- **TS**  
The temperature(s) for circuit simulation.

### Example

```
.temp 0 27 60
```

Specifies circuit analyses at 0, 27 and 60 °C.

## **.TF**

### Transfer Function

**.TF OV IN**

The **.TF** command causes the small signal Transfer Function to be calculated by linearizing around a bias point. The gain from **IN** to **OV** is output along with the input and output impedances.

### Parameters

- **IN**  
Input voltage source name. Must be an independent source
- **OV**  
Requests the output voltage of a specific node or current through a voltage source. The syntax is as follows:

**V(N1[ , N2 ])**

Specifies the voltage difference between nodes **N1** and **N2**. If **N2** and the preceding comma are omitted, ground is assumed.

**I(Vxx[ , Vyy ])**

Specifies the current difference between the voltage sources **Vxx** and **Vyy**. If **Vyy** and the comma are omitted, the current through **Vxx** is output.

### Example

```
* voltage source definition
vin 1 0 5
...
.tf v(3) vin
```

Specifies that the small signal Transfer Function be calculated for the voltage at node **3** with respect to the independent voltage source **vin**.

## .TITLE

### Set Title of Binary Output File

**.TITLE** name

The first line of the *.cir* file provides the title of the binary output file (*.wdb* or *.cou*). However, under a graphical environment the user is not supposed to manually modify the netlist, so this command can be used to define the title of the binary output file.

**.TITLE** also affects the different banners that appear in the ASCII output file.

## .TOPCELL

### Select the TOP Cell Subcircuit

**.TOPCELL [=] <SUBCKT\_NAME>**

Usually, an automatic extraction tool is used to generate a hierarchical Spice netlist. However, in order to simulate the design, it may be necessary to add an X instance of the TOPCELL subcircuit, and re-specify all interface nodes. In these cases it is easier to use **.TOPCELL SUBCKT\_NAME**, when the associated **.SUBCKT SUBCKT\_NAME** and the **.ENDS** command will be ignored by the Eldo parser.

---

#### Note



The command **.TOPCELL** has to be placed before the **.SUBCKT SUBCKT\_NAME** command.

---

## **.TRAN**

### Transient Analysis

#### Point-Driven Analysis

```
.TRAN TPRINT TSTOP [TSTART [HMAX]] [SWEET DATA=dataname] [UIC] [MONTE=val]
.TRAN TPRINT TSTOP [TSTART [HMAX]] [SWEET parameter_name
+ TYPE nb start stop] [UIC] [MONTE=val]
.TRAN TPRINT TSTOP [TSTART [HMAX]] [SWEET parameter_name start stop incr]
+ [UIC] [MONTE=val]
```

#### Parameterized Analysis

```
.TRAN INCRn Tn [{INCRn Tn}] [TSTART=val] [SWEET DATA=dataname]
+ [UIC] [MONTE=val]
.TRAN INCRn Tn [{INCRn Tn}] [TSTART=val] [SWEET parameter_name
+ TYPE nb start stop] [UIC] [MONTE=val]
.TRAN INCRn Tn [{INCRn Tn}] [TSTART=val] [SWEET parameter_name
+ start stop incr] [UIC] [MONTE=val]
```

#### Data-Driven Analysis

```
.TRAN DATA=dataname [SWEET DATA=dataname] [UIC] [MONTE=val]
.TRAN DATA=dataname [SWEET parameter_name TYPE nb start stop]
+ [UIC] [MONTE=val]
.TRAN DATA=dataname [SWEET parameter_name start stop incr]
+ [UIC] [MONTE=val]
```

This command activates a transient analysis. Transient output variables (i.e. those variables contained within **.PRINT** and **.PLOT** commands in the input description file) are calculated as a function of time over a user specified time interval. The initial conditions are automatically determined by a DC analysis (unless the **UIC** parameter is specified) with all sources that are not time dependent being set to their DC values.

The Integral Equation Method (IEM) is used in **.TRAN** to solve FNS functions. The algorithm is just used for FNS, and not for the rest of the circuit. **.OPTION NOFNSIEM** reverts to the previous implementation based on state variables.

Multi-threading can be activated for a single DC or TRAN simulation, Eldo will share computer resources on a multi-processor machine. Alternatives are:

- command line flag **-mthread** (see [page 2-9](#)) at Eldo invocation, or option **MTHREAD** in the netlist. Eldo will make use of all the possible CPUs on the machine.
- command line flag **-usethread #** (see [page 2-10](#)) at Eldo invocation, or option **USETHREAD=val**. This forces Eldo to use at maximum the specified (#) number of CPU. The number specified can exceed the number of CPUs available, but this is not recommended, even though Unix will allow it.

Statistics, generated at the end of simulation, show how many CPUs have been used for the current simulation. This number will also be printed out at the beginning of the TRAN simulation.

## Parameters

- **TPRINT**

The time interval used for the printing or plotting in the ASCII output *.chi* file of the results of transient analysis (in seconds). Also used to compute a default **HMAX** value in case the circuit does not contain any signals (no **PWL/SIN** etc.), which is often the case in oscillator circuits. **TPRINT** can be specified as a parameter or as an expression

- **TSTOP**

The transient analysis duration in seconds. This can be specified as a parameter or as an expression.

- **TSTART**

No outputs are stored from 0 to **TSTART** seconds. This can be specified as a parameter or as an expression.

- **HMAX**

Sets the maximal internal timestep. When **HMAX** is specified both in the **.OPTION** command and in the **.TRAN** command, the **HMAX** in **.OPTION** is considered by Eldo. See **.OPTION HMAX=VAL** on page 11-15.

- **DATA=dataname**

Used in conjunction with the **.DATA** command. The **dataname** parameter should be specified using the **.DATA** command. Please refer to the **.DATA** command on page 10-51 for more information.

- **UIC**

Keyword which indicates that the user does not want Eldo to solve for the quiescent operating point before beginning the transient analysis. Eldo automatically initializes all the node voltages itself as well as any user defined initial node voltages included in a **.IC** command. The **UIC** option is recommended for the simulation of astable or very large digital circuits.

- **MONTE=val**

Monte Carlo analysis. Equivalent to **.MC val**. The syntax allows a different **MONTE** value for each run. However, the actual implementation in Eldo does not account for that: it is the last **MONTE** value to be specified in the netlist which will be taken into account.

## -compat flag

When Eldo is run with the **-compat** flag, the arguments of the **.TRAN** command are a list of **INCRn Tn**, and not the standard **TPRINT TSTOP [TSTART [HMAX]]**. Syntax:

```
.TRAN INCR1 T1 [{INCRn Tn}] [TSTART=val] [UIC]
```

- **INCR1, ...n**

Used in the **.PRINT/.PLOT** command for printout purposes only. Between 0 and **T1**, a value will be printed for each **INCR**, and so on.

- **Tn**

Simulation end time.

## Sweep Parameters

This section contains **SWEEP** related parameters that are previously unspecified in the Parameters section.

- **SWEEP**

Specifies that a sweep should be performed on a parameter or device name.

### **Note**



**.TRAN ... SWEEP** will not work if **.STEP PARAM** is specified and a warning message will be issued stating that these two ways for specifying a sweep are not compatible.

- **parameter\_name**

Name of the parameter or device name to be swept.

- **TYPE**

Can be one of the following:

#### **DEC**

Keyword to select logarithmic variation.

#### **OCT**

Keyword to select octave variation.

#### **LIN**

Keyword to select linear variation.

#### **POI**

Keyword to select a list of frequency points. **POI** is the same as **LIST** except that **POI** expects the number of points **nb** to be specified as it's first argument.

#### **INCR**

Increment of the parameter or device name to sweep. When **INCR** is specified as the **TYPE** parameter, the value which directly follows (**nb**) is the incrementing value.

- **nb**

Number of points required, e.g.

```
.TRAN 1n 10n SWEEP P1 POI 3 1k 10k 100k
```

- **start**

Start value of the parameter or device.

- **stop**

Stop value of the parameter or device.

- **incr**

Increment of the parameter or device name to sweep.

---

**Note**


---

 When **INCR** is specified as the **TYPE** parameter, the value which directly follows (nb) is the incrementing value. If **INCR** is not specified, the incrementing value (**incr**) must be placed after the **start** and **stop** values.

---

For a circuit not containing any signals (no **PWL/SIN** etc.), which is often the case in oscillator circuits, the value of **TPRINT** is also used to compute a default **HMAX** value. In such circuits, unless a **.OPTION HMAX** command was supplied, or a very low **EPS** value was given, Eldo would sometimes not detect the transitions that trigger oscillations, or the transitions that would cause them to die out, and designers who are familiar with Spice-like simulators would not understand the reason for Eldo not behaving in a Spice-like way. The reason for such a difference is only because Spice-like simulators use **TPRINT** as a **HMAX** value, while in Eldo, this was not the case.

The scenario inside Eldo is now the following: if the circuit does not contain any stimuli, then the **HMAX** value is either **TPRINT**, or **TSTOP/DENOM**, whichever value is larger. **DENOM** is based on **EPS**; **DENOM** is 50 by default, and 100 for **EPS < 100U**.

### Examples

```
.tran .2u 40u sweep c1 INCR 5 100 5k
.tran .2u 40u sweep c1 100 5k 5
```

The two lines above are equivalent. Either can be used to specify a transient analysis from 0.2us to 40us. The sweep specification will force Eldo to carry out a transient analysis on each value of **c1** starting at 100Ω and stopping at 5kΩ with an incrementing value of 5Ω.

```
.tran 1ns 100ns
.plot tran v(2)
```

Specifies a transient analysis from 0 to 100ns with a print step of 1ns. The results of the transient analysis are plotted for the voltage at node 2 of the circuit within the limits specified in the **.TRAN** command.

```
.tran 2ns 100ns 50ns uic
.plot tran v(4)
```

Specifies a transient analysis from 0 to 100ns and a print-out from 50 to 100ns with a print step of 2ns. The **UIC** option is also specified indicating that a DC analysis will not be performed and that initial voltage will be set up automatically. The results of the transient analysis are plotted for the voltage at node 4 of the circuit within the limits specified in the **.TRAN** command.




---

Examples of this type of analysis can be found in the [Tutorials](#) chapter.

---

Parameters are allowed in transient analysis commands as shown in the following example:

```
.param p1 = 1e9
.tran dec 10 1 p1
```



NOISE analysis may also be run from within a **.TRAN** command, see “[AC in the middle of a .TRAN](#)” on page 10-17 for more details.

---

# **.TSAVE**

## Save Simulation Run at Multiple Time Points

```
.TSAVE [REPLACE|NOREPLACE] TIME=VALUE [FILE="fileBasename"]
```

The **.TSAVE** command will save the state of the simulation at a specified time point. The state of the simulation is saved to a *.iic* file. The file can be used to restart the simulation from the specified time point using the **.RESTART** command. See “[“.RESTART”](#) on page 10-271. The state of the simulation can be saved at more than one time point by specifying the **.TSAVE** command for each time point.

For saving a simulation run without specific multiple time points, see “[“.SAVE”](#) on page 10-274.

### Parameters

- **REPLACE**

All previously saved checkpoint files in the output directory will be removed and replaced with the checkpoint file specified. Default. Optional.

- **NOREPLACE**

Only the checkpoint file with the same name will be modified, all the remaining checkpoint files will remain unchanged. The checkpoint file will be saved in the output directory. Optional.

- **TIME=VALUE**

Specifies the time at which the simulation will be saved. Mandatory.

- **FILE="fileBasename"**

Specifies the first part of the checkpoint file name. The file name will take the form *fileBasename\_timepoint.iic*, where *timepoint* is the time that the simulation was saved. The file will contain the information from the simulation run for the specified time point. If omitted Eldo will save the file with the name of the top-netlist i.e. if the top-netlist is called *spice-on-top.cir* Eldo will use the name *spice-on-top\_....*. Optional.

### Examples

```
.TSAVE TIME=200ns FILE="spice_ontop"
```

The state of the simulation will be saved at 200ns. All previously saved checkpoint files in the output directory will be removed and replaced with the checkpoint file *spice\_ontop\_2.000000E-7.iic*.

```
.TSAVE NOREPLACE TIME=200ns FILE="spice_ontop"
```

The state of the simulation will be saved at 200ns. All previously saved checkpoint files in the output directory will remain unchanged if the filename is unique. The checkpoint file will be saved with the name *spice\_ontop\_2.000000E-7.iic*.

```
.PARAM sim=250ns
.PARAM chkpnt_file='timepoint'
```

**.TSAVE**

```
.TSAVE noreplace time=sim file=$(chkpnt_file)
```

The state of the simulation will be saved at 250ns as defined on the parameter `sim`. The checkpoint file name will be *timepoint* as defined by the string parameter `chkpnt_file`.

```
.TSAVE NOREPLACE TIME=10ns FILE="spice_ontop"  
.TSAVE NOREPLACE TIME=100ns FILE="spice_ontop"  
.TSAVE NOREPLACE TIME=1000ns FILE="spice_ontop"
```

The state of the simulation will be saved at 10ns, 100ns and 1000ns to three independent files. If `noreplace` was not specified on the last `.TSAVE` command the checkpoint files saved at 10ns and 100ns would be removed.

## .TVINCLUDE

### Test Vector Files

**.TVINCLUDE [FILE=]FILENAME [COMP=ON|OFF] [ERRNODE[=YES|NO]]**

Mach TA test vectors can be included in a netlist. This file allows the user to define a bus, specify inputs and check output values. It is possible to compare Mach TA simulation results using test vectors. A test vector is an external file containing a record of circuit stimulus and response.

### Parameters

- **FILE=**  
Specifies the filename. Optional.
- **FILENAME**  
Test Vector filename.
- **COMP=ON | OFF**  
Define whether the output vector is compared to simulation results. Default is ON. Optional.
- **ERRNODE[=YES | NO]**  
If set to YES (default value), an error is printed for signals using undeclared nodes. If set to NO, the following warning is displayed:

```
Warning 445: COMMAND .TVINCLUDE: node %s not found (%s).  
Test vector specifications ignored for this node.
```

A test vector file consists of the following parts:

- **Header**  
The header specifies the units of time used in the test vectors and the direction and order of the inputs and outputs.
- **Comments**  
Comments can appear anywhere in the file.
- **Test Vectors**  
Test vectors consist of a time stamp followed by the input and output signal data at the time indicated by the time stamp.



Please refer to the *Mach-TA User's And Reference Manual* (Chapter 7: Test Vector Files) for further information.

---

Boundaries are set by using the following keywords:

```
.OPTION LOWVOLTAGE=VAL HIGHVOLTAGE=VAL LOWVTH=VTH1  
+ HIGHVTH=VTH2
```

- **LOWVOLTAGE** and **HIGHVOLTAGE** are used for input signals
- **LOWVTH** and **HIGHVTH** are used for output checking
- Default values are the same as in Mach: **LOWVOLTAGE=0** **HIGHVOLTAGE=5**  
**LOWVTH=2.4** **HIGHVTH=2.6**

---

**Note**



For more details regarding the setting of boundaries with these options, please refer to [page 11-27](#).

---

## **.UNPROTECT**

### Netlist Protection

**.UNPROTECT**

This command is used in conjunction with “[.PROTECT](#)” on page 10-267 to exclude a section of a netlist from being copied into the output file. Can be specified to control the end of the encryption process with the [Eldo Encryption tool](#).

### Example

```
.PROTECT
vin 2 0 ac 0.5
r1 2 3 5k
c3 3 0 0.1p
.ac dec 10 10e+4 10e+8
.plot ac vdb(3)
.UNPROTECT
```

The lines in a netlist between the two commands **.PROTECT** and **.UNPROTECT** are not printed to the output file.

## **.USE**

### Use Previously Simulated Results

```
.USE FILE_NAME [NODESET|IC|GUESS|OVERWRITE_INPUT]
```

This command takes a set of voltages previously saved using the **.SAVE <fname> DC** command, and inserts these as **.NODESET**, **.GUESS** or **.IC** values at the point of the circuit where this command is present. It is also possible to have multiple occurrences of the **.USE** command in an input netlist. See “[.SAVE](#)” on page 10-274.



For more details, refer to the [.SAVE, .USE & .RESTART](#) section in the [Speed and Accuracy](#) chapter.

This command cannot be used with previously saved simulation results from a **.TSAVE** command.

### Parameters

- **FILE\_NAME**  
Filename into which the DC values were saved via the **.SAVE <fname> DC** command.
- **NODESET**  
Indicates that the read in voltages should be used as **.NODESET** values.
- **IC**  
Indicates that the read in voltages should be used as **.IC** values.
- **GUESS**  
Indicates that the read in voltages should be used as **.GUESS** values.
- **OVERWRITE\_INPUT**  
Has the same effect as **GUESS**, however Eldo is allowed to change the DC values of input if they don't match the **.ic** file. This parameter allows the user to perform AC analysis close to a previous transient saved point (saving with the **.SAVE end** or **.SAVE time** command).

#### Note



If **.USE OVERWRITE\_INPUT** is used in conjunction with **.AC UIC**, then the file referred to in the **.USE** statement will be used for each AC analysis, and not only for the first one as was the case in versions prior to v6.3. For example, if there is a **.STEP** command, then the *use file* will be used for each AC analysis triggered by the **.STEP**, and not only by the first one.

### Examples

```
.use test1.exa nodeset
```

Specifies that DC values found in the file *<test1>.exa* should be read and used as **.NODESET** values.

```
.use circuit1.iic ic
```

Specifies that DC values found in the file *<circuit1>.iic* should be read and used as **.IC** values.

## .USEKEY

### Use Reliability Model Key (Password)

**.USEKEY** [ MODEL=model\_name ] KEY=key\_value

This reliability analysis command provides the encryption key (password) to be used to allow the UDRM API to retrieve encrypted model parameter's value.



For the complete description of this command and information on all reliability commands, see the separate chapter [Reliability Simulation](#).

---

## **.USE\_TCL**

### Use Tcl File

**.USE\_TCL FILENAME**

This command loads the Tcl file `FILENAME` into Eldo's Tcl interpreter.

This allows you to dynamically manage User Defined Functions (UDF) written in Tcl language. The functions of the post-processor library and other commands are available through the Tcl interface.

After loading the Tcl file, Eldo can use all the functions written in this file as if they were macros. This means that these functions can accept any argument, are defined for the whole netlist, and can return both numbers and/or waves.



For performing a single call to a Tcl function, see the command “[.CALL\\_TCL](#)” on page 10-28. For further information on both commands, see the [Post-Processing Library](#) chapter of this manual.

---

### Example

*resi.tcl* contains:

```
proc PARAM_EVAL { a } {  
    return [expr $a * 3]  
}
```

*tutorial.cir* contains:

```
.use_tcl resi.tcl  
v1 1 0 pwl(0 0 10n 10)  
r1 1 0 r=PARAM_EVAL(2)  
.tran 1n 10n  
.plot tran i(r1)  
.end
```

This example demonstrates how to use a very simple Tcl function in a spice-like netlist.

**.USE\_TCL** loads the file into an internal Tcl interpreter, making all functions known to Eldo. Thus resistor `r1` gets its value from the Tcl function `PARAM_EVAL` just as if `PARAM_EVAL` is a macro (defined with `.DEFMAC`).

## **.WCASE**

### Worst Case Analysis

```
.WCASE DC|AC|TRAN [OUTPUT=MIN|MAX|BOTH] [VARY=LOT|DEV|BOTH]
+ [TOL=VAL] [ALL]
```

Worst Case Analysis computes worst case values for waveform data extracted using the **.EXTRACT** command according to a set of parameters that have **LOT** and **DEV** variations.

- 
- i** For DC sensitivity analysis, the **.SENS** command can be used. For more information see “**.SENS**” on page 10-278.
- 

Worst Case Analysis uses the same **LOT** and/or **DEV** parameters as MC analysis to set these parameters to be varied. A Worst Case Analysis, **.WCASE**, may not be used in a netlist together with a **.MC**, Monte Carlo Analysis command.

Different entities are able to share the same distribution. Anywhere Eldo accepts **LOT/DEV** specifications, you can specify **LOTGROUP=group\_name**. Please refer to “**.LOTGROUP**” on page 10-142.

Considering a design which contains **P** parameters that have a statistical variation, and **T** targets (**.EXTRACT** commands), Eldo will perform several simulations:

1. The first simulation is the nominal simulation, using nominal values for the **P** parameters.
2. Next **P** simulations are sensitivity analysis runs, where Eldo applies a small variation to one of the **P** parameters, keeping all other parameters to their nominal value, and computes the sensitivity of the **T** targets relative to the current parameter.

For each of the **T** targets, Eldo outputs a message like:

```
With respect to MODEL R LOT
Variation of MAX(V(1)) is negative: -1.000000e-02 (Unit/%) or
1.000000e+00 (%/%)
```

Negative variation means that when parameter is increased, target is decreasing.

3. Then with all the sensitivity data, Eldo computes the combination of min or max parameter values that will generate the best and worst cases (**MIN** and **MAX**) for each of the **T** target.

There can be at most **2\*T** worst case simulations. During these simulations, Eldo does not output anything.

4. Finally, Eldo computes the worst case information for all the targets, and outputs this information.

Thus, there can be at most  $1 + P + 2*T$  simulations. Worst Case analysis assumes the influence of each parameter to be linear, and not correlated with other parameters.

## Parameters

- **DC**  
Specifies a DC analysis. An analysis type MUST be specified.
- **AC**  
Specifies an AC analysis. An analysis type MUST be specified.
- **TRAN**  
Specifies a transient analysis. An analysis type MUST be specified.

---

**i** .WCASE can also be used in Steady State analysis (.SST). For more information see “Steady-State Worst Case Analysis” on page 2-18 of the *Eldo RF User’s Manual*.

---

- **OUTPUT**  
Specifies the type of Worst Case Analysis:
  - MIN**  
Only the minimum case is extracted.
  - MAX**  
Only the maximum case is extracted.
  - BOTH**  
Both minimum and maximum cases are extracted. Default=BOTH.
- **VARY**  
Specifies the kind of variation; **DEV**, **LOT** or **BOTH**. Default=BOTH.
- **TOL**  
Specifies that a variation or tolerance is to be applied to the sensitivity measurement.
- **VAL**  
The percentage tolerance value for the sensitivity: <value>. Default=2%. If in the .chi file Eldo states that sensitivity is zero, this may be due to the **TOL** value being set too small.
- **ALL**  
Causes all intermediate waves will be plotted/printed. This can result in large output ASCII/Binary files, depending on the number of waves and the number of Worst Case runs.

## Related options

[CARLO\\_GAUSS](#) (page 11-24), [SIGTAIL](#) (page 11-30), [DISPLAY\\_CARLO](#) (page 11-45)

**Example**

```
Worst Case analysis example
* circuit: RC filter
r1 in out rmod 10k
c1 out 0 cmod 1p
vin in 0 ac 1
.model rmod res dev=10%
.model cmod cap dev=10%
* extract cutoff frequency at -3 dB
.extract xycond (xaxis, vdb(out)<=max(vdb(out))-3)
.wcase ac
.ac dec 10 1 1G
.plot ac vdb(out) vp(out)
.end
```

The following results will be obtained in the *.chi* file:

Sensitivity:

```
With respect to MODEL RMOD DEV: Object :R1
Variation of XYCOND(XAXIS, ...) is negative: -1.598533e+05 (Unit/%)
With respect to MODEL CMOD DEV: Object :C1
Variation of XYCOND(XAXIS, ...) is negative: -1.598533e+05 (Unit/%)
```

Worst case values:

```
*XYCOND(XAXIS, ...) NOM: 1.587800E+0 MIN: 1.312462E+0 MAX: 1.959982E+07
```

**Note**

If incorrectly used, run times for this command may become excessive, especially where results are completely out of range. **LOT/DEV** parameters must be applied correctly to devices/models to get reasonable results.

# **.WIDTH**

## Set Printer Paper Width

**.WIDTH OUT=80 | 132**

Sets the paper width in number of characters of the output print device.

When multiple occurrences of the **.WIDTH** command are specified, a warning is issued. The last **.WIDTH** command will override all previous commands.

### Parameters

- **OUT=80**  
Specifies that the number of columns or characters on the output device is 80.
- **OUT=132**  
Specifies that the number of columns or characters on the output device is 132. The default value is 132.

### Example

**.width out=80**

Sets the output width on the output device to 80 columns.

**.WIDTH OUT=80**  
**.WIDTH OUT=132**

Sets the output width on the output device to 132 columns, as the previous command is ignored.



# Chapter 11

## Simulator and Control Options

---

### Introduction

The **.OPTION** command allows the user to modify Eldo execution behavior by allowing the setting of parameter values other than the default ones.

Multiple **.OPTION** commands can be used in a netlist. A single **.OPTION** statement can contain multiple options in any combination and any order.

If an option is stated more than once, the last specified value is used. Unless otherwise described, if an option is not stated, it defaults to zero.

## **.OPTION**

```
.OPT[ION] OPTION[=VAL] {OPTION[=VAL]}
```

The parameter descriptions have been divided into sections according to what area of the program they affect.

**Note**

Parameters from the SPICE language that are not included in the following list are ignored by Eldo.

### Parameters

**OPTION[=VAL]**

The following tables show a list of the possible options which can be used with the **.OPTION** command. Click on an option to jump to the description of that option. Each option is described in greater detail on the following pages.

**Table 11-1. Simulator Compatibility Options**

COMPAT	COMPMOD	COMPNET	MOTOROLA
PRECISE	SDA	SPI3ASC	SPI3BIN
SPICEDC	SPIOUT	WSF	WSFASCII

**Table 11-2. Netlist Parser Control Options**

ADMS_FAST_PARSE	ADMSBS	ALTINC	BSLASHCONT
CHECKDUP	COMPEXUP	CONTINUE_INCLUDE	CNTTHREAD
MTHREAD	NOBSLASHCONT	NOCMPUNIX	NOELDOLOGIC
NOKEYWPARAMSST	NOMATSING	NOSSTKEYWORD	NOZSINXX
PARHIER	PSTRICK	QUOTREL	QUOTSTR
STOPONFIRSTERROR	SUBFLAGPAR	USETTHREAD	

**Table 11-3. Simulation Speed, Accuracy & Efficiency Options**

ABSTOL	ABSVAR	ADJSTEPTRAN	CAPANW
CHGTOL	DVDT	EPS	FASTRLC
FLUXTOL	FREQSMP	FROM_TO	FT
HMAX	HMIN	HRISEFALL	INCLIB
ITL1	ITL3	ITL4	ITL6
ITL7	ITL8	ITOL	LIBINC
LIMNWRMOS	LVLTIM	MAXNODES	MAXTRAN

**Table 11-3. Simulation Speed, Accuracy & Efficiency Options**

MAXV	NETSIZE	NGTOL	NMAXSIZE
NOCONVASSIST	NOLAT	NONWRMOS	NOQTRUNC
NOSWITCH	PCS	PCSSIZE	PCSPERIOD
PIVREL	PIVTOL	PSOSC	QTRUNC
RATPRINT	RELTOL	RELTRUNC	RELVAR
SAMPLE	SPLITC	STARTSMP	STEP
TIMESMP	TRTOL	TUNING	UNBOUND
VMAX	VMIN	VNTOL	WDB_IDELTA
WDB_VDELTA	WDB_NOSYNCHRO	XA	

**Table 11-4. Miscellaneous Simulation Control Options**

AMMETER	AUTOSTOP	AUTOSTOPMODULO	CARLO_GAUSS
CPTIME	DEFPTNOM	DSCGLOB	DSPF_LEVEL
FALL_TIME	FLOATGATE0	FLOATGATECHECK	FLOATGATERR
HIGHVOLTAGE	HIGHVTH	ICDC	ICDEV
INTERP	LICN	LOWVOLTAGE	LOWVTH
M53	NOMEMSTP	PARAMOPT_NONINITIAL	PODEV
RANDMC	RGND	RGNDI	RISE_TIME
SIGTAIL	TNOM	TPIEEE	ULOGIC
ZOOMTIME			

**Table 11-5. Model Control Options**

ACM	ASPEC	BSIM3VER	DEFAD
DEFAS	DEFL	DEFNRD	DEFNRS
DEFPD	DEFPS	DEFW	ELDOMOS
FNLEV	GMIN	GMIN_BJT_SPICE	GMINDC
GRAMP	GENK	KLIM	KWSCALE
IBIS_SEARCH_PATH	MAXADS	MAXL	MAXPDS
MAXW	MINADS	MINL	MINPDS
MINRESISTANCE	MINRVAL	MINW	MNUMER
MOD4PINS	MODWL	MODWLDOT	NGATEDEF
NOAUTOCTYPE	NOKWSCALE	NWRMOS	PGATEDEF
RAILRESISTANCE	REDUCE	RESNW	RMMINRVAL

**Table 11-5. Model Control Options**

RMOS	RSMALL	RZ	SCALE
SCALEBSIM	SCALM	SOIBACK	SPMODLEV
TMAX	TMIN	USEDEFAP	VBICLEV
WARNING_DEVPARAM	WARNMAXV	WL	YMFAC
ZDETECT			

**Table 11-6. RC Reduction Options**

RC_REDUCE	RC_REDUCE_FMAX	RC_REDUCE_METHOD	RC_REDUCE_PORT
-----------	----------------	------------------	----------------

**Table 11-7. Noise Analysis Options**

FLICKER_NOISE	IKF2	JTHNOISE	THERMAL_NOISE
NOISE_SGNCONV	NONOISE		

**Table 11-8. Simulation Display Control Options**

ACSIMPROG	DCSIMPROG	ENGNOT	INGOLD
MSGBIAS	MSGNODE	NOTRCLIB	NOWARN
NUMDGT	PRINTLG	VERBOSE	WBULK

**Table 11-9. Simulation Output Control Options**

ACOUT	ALTER_SUFFIX	ASCII	BLK_SIZE
CAPTAB	DISPLAY_CARLO	DUMP_MCINFO	EXTCGS
EXTFILE	EXTMKSA	HISTLIM	INPUT
JWDB_EVENT	JWDB_PERCENT	KEEP_HMPFILE	LCAPOP
LIMPROBE	LIST	MAX_CHECKBUS	MAX_DSPF_PLOT
NEWACCT	NOASCII	NOBOUND_PHASE	NODCINFOTAB
NODE	NOEXTRACTCOMPLEX	NOMOD	NOOP
NOPAGE	NOSIZECHK	NOSTATP	NOTRC
NOWAVECOMPLEX	NOXTABNOISE	OPTYP	OUT_RESOL
OUT_SMP	OUT_STEP	POST	PRINT_ACOP
SIMUDIV	STAT	TEMPCOUK	TIMEDIV
VBCSAT	VXPROBE		

**Table 11-10. Optimizer Output Control Options**

OPSEUDO_ABSTRACT	OPSEUDO_DETAIL	OPSEUDO_DISPLAY_GOALFITTING	OPSEUDO_FORCE_GOALFITTING
OPSEUDO_NETLIST	OPSEUDO_NOGOALFITTING	OPSEUDO_OUTER	OPSEUDO_OUTPUT
RESET_MULTIPLE_RUN			

**Table 11-11. File Generation Options**

AEX	ALIGNEXT	ASCII=val	COU
CSDF	INFODEV	INFOMOD	ISDB
JWDB	NOAEX	NOCKRSTSAVE	NOCOU
NOIICXNAME	NOJWDB	NOPROBEOP	OUT_ABSTOL
OUT_REDUCE	OUT_RELTOL	PROBE	PROBEOP
PROBEOP2	PSF	PSFASCII	SAVETIME

**Table 11-12. Mathematical Algorithm Options**

ANALOG	BE	BLOCKS=IEM	BLOCKS=NEWTON
CSHUNT	DCPART	DIGITAL	DPTRAN
GEAR	GNODE	GSHUNT	IEM
MAXORD	METHOD=GEAR	NEWTON	NODCPART
NODEFNEWTON	NORMOS	OSR	PSTRAN
SMOOTH	TRAP		

**Table 11-13. Mixed-Mode Options**

D2DMVL9BIT	DEFA2D	DEFD2A	DEFCONVMSG
DYND2ALOG	DYND2ALOG2	MIXEDSTEP	

**Table 11-14. Other Options**

CTEPREC	DCLOG	EPSO	MAXNODEORD
NODUPINSTERR	NOELDOSWITCH	NOFNSIEM	NOINIT
SEARCH	VAMAXEXP	ZCHAR	

## Cadence Compatibility Options

- **WSF**  
When **SDA=2**, Eldo creates a Cadence **WSF** binary output (*.wsf*) file for EDGE and OPUS. This option requires license authorization.
- **WSFASCII**  
When **SDA=2**, Eldo creates a Cadence **WSF** ASCII output (*.wsf*) file for EDGE and OPUS. This option requires license authorization.
- **SDA**  
When set to 2, this parameter enables Cadence **WSF** compatible ASCII or binary output files for EDGE and OPUS. This option requires license authorization. Default is 0.0.

## Precise Compatibility Options

- **PRECISE**  
Forces Precise models to be used. This option also implies option **NEWTON**, unless option **NODEFNEWTON** is set. This means that option **PRECISE** ensures compatibility with the Precise simulator, and not only with the Precise models.

## SPICE Compatibility Options

- **SPI3ASC**  
Forces Eldo to create a SPICE3 compatible ASCII output (*.spi3*) file.
- **SPI3BIN**  
Forces Eldo to create a SPICE3 compatible binary output (*.spi3*) file.
- **SPICEDC**  
Forces Eldo to use the Berkeley SPICE mechanism when calculating the DC operating point rather than using the normal method. By using this method, VBE junctions of BJT's are set to 0.7V, while voltages across other PN junctions are set to 0V. This option can increase the efficiency on circuits containing BJT elements.
- **SPIOOUT**  
DC operating point is sorted alphanumerically.

## Simulator Compatibility Options

- **COMPAT**  
This is equivalent to the `-compat` flag used when invoking Eldo. It is important to note that this option must be set at the top of the design, otherwise results can be unpredictable.

Option **COMPAT** is equivalent to setting both **COMPMOD** and **COMPNET** options shown below:

- **COMPMOD**

Triggers only the automatic conversion of models. This is equivalent to the `-compmod` flag used when invoking Eldo.

- **COMPNET**

Causes the netlist to be interpreted as compatible format, but the models themselves are treated as Eldo Spice models. This means it is assumed that models are already Eldo models. This is equivalent to the `-compnet` flag used when invoking Eldo.



Please refer to the “[Compatibility Options](#)” on page 12-1 for further information.

---

- **MOTOROLA**

Invokes the Motorola mode of Eldo.

## Netlist Parser Control Options

- **ADMS\_FAST\_PARSE**

This option is used to accelerate the elaboration phase of the design with ADVance MS.



Please refer to [Black-box mode for Eldo and Mach](#) in the *ADVance MS User’s Manual* for further information.

---

- **ADMSSBS**

This option enables extended identifiers to be used in VHDL-AMS descriptions. This option is used with ADVance MS.

- **ALTINC**

Forces Eldo to replace the first **.INCLUDE** statement found in an input netlist by the first **.INCLUDE** statement found in the **.ALTER** section of the netlist. This allows the replacement of one file by another one when using the Eldo re-run facility.

- **BSLASHCONT**

Allows two backslashes to be used for the continuation of the line. This can also be achieved by invoking Eldo with the `-compat` flag. Example:

```
.option BSLASHCONT
R1 1 2 \\
3k
R2 1 2 1k
```

**R1** will be set to **3K**. When invoking Eldo with the `-compat` flag, it is possible to disable option **BSLASHCONT** by using option **NOBSLASHCONT**.

- **CHECKDUPL**

Forces Eldo to perform the check of duplicate instance names, even when the netlist is larger than the internal limit (1000 lines). A warning is displayed telling the user that they can use option **CHECKDUPL** if it isn't already selected.

- **COMPEXUP**

Forces Eldo to keep the name of the extraction results (from **.EXTRACT** and **.MEAS** statements) in uppercase when in simulator compatibility (-compat) mode. Prior to Eldo v6.6 the names were always in uppercase, beginning Eldo v6.6 they are in lowercase.

- **CONTINUE\_INCLUDE**

Specifies that continuation lines with + as the first character apply to the **.INCLUDE** command in the file. For example, if the main netlist has the two lines:

```
.include file.inc
+ b=2
```

With this option specified, if file *file.inc* contains as its last line:

```
.param a=1
```

Eldo would interpret this as:

```
.param a=1
+ b=2
```

- **CNTTHREAD**

Checks how many CPUs Eldo can access on a multi-processor machine. Eldo will share computer resources for multi-threading a single DC or TRAN simulation. This option is equivalent to the Eldo **-cntthread** command line flag.

- **MTHREAD**

Activates multi-threading for a single DC or TRAN simulation. Eldo will share computer resources on a multi-processor machine. Eldo will make use of all the possible CPUs on the machine. This option is equivalent to the Eldo **-mthread** command line flag.

- **NOBSLASHCONT**

Disables option **BSLASHCONT** when invoking Eldo with the **-compat** flag.

- **NOCMPUNIX**

The characters "[" and "]" are interpreted as special characters when the wildcard character "\*" is used. According to Unix convention, these special characters may be overridden by using the backslash character "\\" which removes the special function of the character that immediately follows it.

- **NOELDOLOGIC**

Specifies that subcircuit X instances can be named beginning with any Eldo logic or macro primitive name. With this option, Eldo will not attempt to parse such instances as logic gates. This option is automatically set when invoking Eldo with the **-compat** flag. This

option affects the following Eldo built-in logic primitives: AND, OR, NAND, NOR, XOR, CMP, CMPD, INV; and the following Eldo built-in macromodels: ADC, DAC, DEL, FNS, FNZ, OPA.

- **NOMATSING**

This option can be used to complete a simulation if the simulation stops with a warning regarding a singularity. Otherwise, it will be stopped if a singularity is found in DC.

However, the design has to be improved if there is a singularity. Even if the simulation continues, this does not mean that the solution in DC is correct.



Please refer to the appendix “[Improved Diagnostics for Certain Erroneous Models](#)” in the *ADVance MS User’s Manual* for further information.

---

- **NOSSTKEYWORD | NOKEYWPARAMSST**

Tells the parser to bypass the RF keyword check in expressions or in [.PARAM](#).



For a full list of Eldo RF keywords that cannot be used in [.PARAM](#) see [page 10-206](#).

---

- **NOZSINXX**

Remove the effect of the  $\sin x/x$  term used in the AC response of Z transforms. The response was modified in Eldo v6.3, previously the term was not taken into account.

- **PARTHIER= 'local' | 'global' | 'hier' | 'hierlocal'**

This parameter controls the priorities for parameters. Quotes are optional. Default for Eldo is **parhier=hier** (was **local** in pre-v6.6 versions of Eldo). When **-compat** is set, default is **global**.

When PARHIER is GLOBAL or LOCAL, then the parameter priorities adopted in X statements are performed as if the operations were executed inside the SUBCKT. The following rules apply:

When PARHIER is GLOBAL, the priority is the following:

1. .PARAM statement at a higher level
2. Value at the X call (instance)
3. SUBCKT definition

When PARHIER is LOCAL, the priority is:

4. SUBCKT call (instance)
5. SUBCKT definition
6. .PARAM statement

When PARHIER is set to HIER or HIERLOCAL, then the mechanism for evaluating parameters mimics that of the function call in programming languages such as C. The parameters found on the right-hand side of the expressions are sought first in the current environment, while the name on the left-hand side refers to the parameter name inside the subcircuit being called. This is a very different mechanism than for GLOBAL or LOCAL. For example:

```
ID VDD 0 1m
RC1 S1 VDD 1k
X1 S1 0 sub1

.op

.subckt sub1 c b
.param w = 1k
XM1 c b sub2 w2 = w
.ends sub1

.subckt sub2 c b
.param w = 2k
r1 c b 'w2'
.ends sub2
.param w = 7k
.end
```

If PARHIER = GLOBAL w2 is 7k. Because it is as if we had:

```
XM1 c b sub2 xm1.w2 = xm1.w
```

and xm1.w takes the definition at the highest level.

If PARHIER = LOCAL w2 is 2k. Because it is as if we had:

```
XM1 c b sub2 xm1.w2 = xm1.w
```

and xm1.w takes the definition at the lowest level, i.e inside the subcircuit SUB2.

If PARHIER = HIER w2 is 1k. Because it is as if we had:

```
XM1 c b sub2 xm1.w2 = w
```

and local value for w is 1k.

If PARHIER = HIERLOCAL w2 is 1k: same as PARHIER=HIER.

However, in case of PARHIER = HIER or PARHIER = HIERLOCAL, if a parameter is specified at the X instance, then this value will have precedence over the local value if any. For example:

```
ID VDD 0 1m
RC1 S1 VDD 1k
X1 S1 0 sub1

.op

.subckt sub1 c b
.param w = 1k
```

```

XM1 c b sub2 w=3k w2 = w
.ends sub1

.subckt sub2 c b
.param w = 2k
r1 c b 'w2'
.ends sub2
.param w = 7k
.end

```

Here, Eldo will assume that the w on the w2 =w statement is 3k because w is explicitly specified on the X instance. If w=3k had not been there, then the local value 1k would have been taken.

Note that in case a parameter which appears on the right-hand-side of an expression on a X instance has both a value in the current environment, and in the SUBCKT being called, and is not specified on that X instance, then Eldo will issue a warning that the two values exist, just to warn the user about possible problems in the netlist.

The difference between HIER and HIERLOCAL is for the case a parameter which appears on the right-hand-side of an expression on a X instance, is neither specified on the same X call, neither specified on the current environment: if PARHIER is set to HIER, Eldo will issue an error that the parameter is not found, if PARHIER is set to HIERLOCAL, then Eldo will check for this parameter inside the SUBCKT being called, and will use that value if found. For example:

```

ID VDD 0 1m
RC1 S1 VDD 1k
X1 S1 0 sub1

.op

.subckt sub1 c b
XM1 c b sub2 w2 = w
.ends sub1

.subckt sub2 c b
.param w = 4k
r1 c b 'w2'
.ends sub2

```

If PARHIER is set to HIER, Eldo will issue an error while evaluating w2 = w, because w is not found in the current environment. If PARHIER is set to HIERLOCAL, Eldo will check inside the sub2 if w exists, and will find it.

- **PSTRICT**

Eldo accepts parameter names that contain special characters '! " < > ? & (in addition to letters, numbers and characters '\_ '\$ '[' ']' '@' '\''~' ':' '#' and '%'). If option **PSTRICT** is set, Eldo will issue an error when using such special characters.

- **QUOTREL**

This option instructs Eldo to consider double quotes as single quotes. This provides improved compatibility with other simulators. However, character strings are not

**.OPTION**

recognized as such in **.PARAM** statements. This option is set by default in compat mode (**-compat** flag or option **COMPAT**).

- **QUOTSTR**

This option instructs Eldo to consider double quotes as a parameter string delimiter. This is the default in Eldo, but disabled in compat mode (**-compat** flag or option **COMPAT**). Therefore, only use this option in compat mode to identify parameter strings with double quotes.

**Note**

If both **QUOTREL** and **QUOTSTR** options are specified, the last to be specified is used.

- **STOPONFIRSTERROR=1 | 2**

When set to 1, Eldo will stop parsing the netlist on the first error. This is to prevent the case where Eldo cannot recover correctly from a first error, and would display many subsequent meaningless errors. If the netlist file contains **.ALTER** statements for multiple simulation runs, Eldo will stop on the first **.ALTER** that has an error, but continue with the remaining **.ALTER** statements.

When set to 2, the first error stops the simulation even if the netlist file contains **.ALTER** statements for multiple simulation runs.

- **SUBFLAGPAR**

When this option is set, then **SWITCH**, **ANALOG** and **ELDO** are no longer considered as keywords, and the subcircuit definition/instantiation of a SUBCKT named **SWITCH**, **ANALOG** or **ELDO** will be allowed. This option must be specified at the top of the netlist, just after the title line.

By default, **SWITCH**, **ANALOG** and **ELDO** are considered to be keywords inside Eldo, i.e. it is not possible, by default, to create and instantiate a SUBCKT named **SWITCH**, **ANALOG** or **ELDO**. Therefore, the following design would fail:

```
Title
.subckt switch a b
r1 a b 1
.ends
x1 a b switch
v1 a 0 1
rb b 0 1
.dc
.end
```

However, if option **SUBFLAGPAR** is set, then **SWITCH**, **ANALOG** and **ELDO** are no longer considered as keywords, and the subcircuit definition/instantiation of a SUBCKT named **SWITCH**, **ANALOG** or **ELDO** will be allowed. In such a case, and if the user also wants to set the subcircuit as **SWITCH** (i.e. the Eldo-XL flag that would force Eldo to convert MOS into their switch equivalent), then that flag must be enclosed in brackets, e.g.

```
.OPTION SUBFLAGPAR
x1 a b switch (switch)
```

The first occurrence of the parameter `switch` in the above example stands for the subcircuit name, while the second occurrence, which is enclosed in brackets, stands for the XL flag `SWITCH`.

- **`USETHREAD=VAL`**

Activates multi-threading for a single DC or TRAN simulation. Eldo will share computer resources on a multi-processor machine. Eldo will make use of the number of CPUs as specified with this option. The number specified can exceed the number of CPUs available, but this is not recommended. This option is equivalent to the Eldo `-usethread #` command line flag.

## Simulation Speed, Accuracy & Efficiency Options

---

**i** Before using the following options it is recommended that the relevant section be consulted in the [Speed and Accuracy](#) chapter.

---

- **`ABSTOL=VAL`**

Absolute current accuracy. The default value is `VNTOL × ITOL`. Note `ABSI` is synonymous to `ABSTOL`.

- **`ABSVAR=VAL`**

Used for timestep control. Sets the maximum voltage change from convergent iteration to iteration (timestep to timestep) for the condition that `LVLTIM=1` and `DVDT=0`. If the simulator produces a convergent solution that is greater than `ABSVAR`, the timestep is reduced for the next solution. Default is 0.2V.

- **`ADJSTEPTRAN`**

When this option is set the `TPRINT` parameter of the `.TRAN` command is used to calculate minimum and maximum time step values. They are calculated as follows:

$$\begin{aligned} \text{MIN} &= \text{TPRINT}/10 \\ \text{MAX} &= \text{MIN}(\text{TIME\_DURATION}/50, \text{TPRINT} * \text{RMAX}) \end{aligned}$$

Where `TIME_DURATION` is the time duration specified in the `.TRAN` command and `RMAX` is the value given by the option `RATPRINT`. When `RATPRINT` is not specified and `ADJSTEPTRAN` is active then `RATPRINT` defaults to 2.

- **`CAPANW=VAL`**

Value below which coupling capacitors could be treated by OSR. Defaults to 50 femto Farads, unless option `DIGITAL` is used, in which case it is 1 nF.

- **`CHGTOL=VAL`**

Is the absolute tolerance on charge and is used by charge control devices, (e.g. BIP, DIODE, JFET, BSIM1...). Default value is set by `EPS`. This option is only used by the “Gear” algorithm or if option `QTRUNC` is set.

**.OPTION**

- **DVDT=VAL**

Used for timestep control on Newton blocks. When set to 0, the **DVDT** algorithm is activated for the condition that **LVLTIM**=1. Default is -1.

- **EPS=VAL**

Sets the internal simulator accuracy. Default value is 5mV. Values smaller than  $1.0 \times 10^{-10}$  V need to be defined with the **UNBOUND** parameter (see [page 11-23](#)). Please refer to “[Global Tuning of the Accuracy—EPS](#)” on page 16-10.

- **FASTRLC**

For R, L and C elements which have values that vary at run time, a new device evaluation is performed at each iteration to ensure accurate results. When **FASTRLC** is set, the value used at a given time is not recomputed at each iteration. Instead, the value of previous time step is used. This accelerates the simulation because expressions are not re-evaluated at each time step, at the cost of a lower accuracy. This option only has an effect in TRANSIENT simulation.

- **FLUXTOL=VAL**

Is the absolute tolerance on flux (for inductors). Default value is set by **EPS**. This option is only used by the “Gear” algorithm or if option **QTRUNC** is set.

- **FREQSMP=VAL | {VAL1 ,VAL2 , . . . VALn }**

Forces Eldo to compute a time point at every multiple time interval of  $1/\text{FREQSMP}$ , the sampling frequency. Useful when performing a Fourier analysis. Multiple values can be specified as a list in which case Eldo computes timepoints corresponding to all sampled points. For example: a design has two clocks at 2MHz and 320 kHz. For uniform sampling, time intervals of  $0.5\mu\text{s}$  and  $3.125\mu\text{s}$  are needed to calculate the necessary exact points (for FFT post processing). Note that if options which impose the sampling frequency are set (for example **interpolate=0** on **.optfour**), the freqsmp array is reset to a single value

**Note**

 It is possible to assign expression parameters to **FREQSMP**, e.g.

```
.PARAM p1=1k
.OPTION FREQSMP=p1
```

- **FROM\_TO=0 | 1**

When enabled (=1), this option will affect the functions **min**, **max** and **avg** of the **.MEAS** command. The simulation will stop as soon as all measurements other than **min**, **max** and **avg** which don't have a **from= to=** parameter are complete. Though the results of these measurements will be not be complete, the simulation will run faster. The default value is 0.

- **FT=VAL**

Used for timestep control. When a solution is rejected because of non-convergence (in conjunction with **ITL3** and **ITL4**) the timestep is reduced (multiplied) by **FT**. Default value is 0.125.

- **HMIN=VAL**

Sets the minimal internal timestep. Default value is 1ps, which is a value well suited for typical MOS circuits. The default for Switched Capacitor circuits is 2 ns. The default for bipolar circuits is 10ps.

- **HMAX=VAL**

Sets the maximum internal timestep. Default **HMAX** value is 1/10 of the wave period when using **SIN** and **SFFM** functions.



See also **.TRAN TPRINT TSTOP [TSTART [HMAX]] [UIC]** on page 10-318.

---

- **HRISEFALL=VAL**

Forces Eldo to take timesteps not greater than  $dt/val$  during transitions as defined by PULSE or PWL signals. **VAL** is an integer. **dt** is the transition time between two time-points corresponding to different values. Default value is 0.

- **INCLIB**

Same as the **LIBINC** option. Specifies that the full contents of every library are read by Eldo in one pass upon completion of reading the input file. This was the default mechanism in the v5.8 version of Eldo. All the libraries (**.LIB**) are included without filtering the objects (model, card, or subcircuit) that are not used in the specific netlist.

- **ITL1=VAL**

Sets a limit on the maximum number of DC iterations. Default value is 100.

- **ITL3=VAL**

Used for timestep control. If convergence is reached in less than **ITL3** iterations, the next timestep is doubled. Default value is 3.0.

- **ITL4=VAL**

Used for timestep control. If convergence is not reached within **ITL4** iterations, the present timestep is rejected and the next one reduced by **FT**. Default is 13.0.

- **ITL6**

Equivalent to **ITL3** when DC convergence assist is in use. Default value is 5; however this default is 6 in Pseudo-Tran algorithm.

- **ITL7**

Equivalent to **ITL4** when DC convergence assist is in use. Default value is 30; however this default is 20 in Pseudo-Tran algorithm.

- **ITL8**

This controls the maximum number of iterations allowed for each trial of ramping algorithm (**GRAMP**, **PSTRAN**, **.RAMP DC**). Default is 10000.

- **ITOL=VAL**

Controls the current accuracy of the simulator when solving circuits using Newton iterations. Circuit convergence is reached when:

$$|I(i) - I(i-1)| < \text{RELTOL} \times |\max(|I(i)|, |I(i-1)|)| + \text{VNTOL} \times \text{ITOL}$$

where **I(i)** is the current value at voltage iteration *i* and **I(i-1)** is the previous iteration. Default value is  $1.0 \times 10^{-6}$  A/V.

- **LIBINC**

Same as the **INCLIB** option. Specifies that the full contents of every library are read by Eldo in one pass upon completion of reading the input file. This was the default mechanism in the v5.8 version of Eldo. All the libraries (**.LIB**) are included without filtering the objects (model, card, or subcircuit) that are not used in the specific netlist.

**Note**



This mode is also activated in the case of option **COMPAT**. Option **LIBINC** can also be activated by using the command line flag **-libinc** at invocation of Eldo.

- **LIMNWRMOS=VAL**

This option is used to set the value below which MOS resistors are not handled at all. They are not created as objects and not handled in the approximative manner described in the **NONWRMOS** option.

This option is used to collapse intrinsic MOS transistor nodes. With this option, the effect of the parasitic resistors upon the channel current is still taken into account, although not as accurately as when the nodes are explicitly created. Particularly, some of the overlap capacitances in the MOS model become connected to the external drain/source, whereas they really are connected to the internal nodes. This may change results slightly.

- **LVLTIM=VAL**

Sets the simulator Time Step algorithm. There are three options available:

**LVLTIM=4**

This algorithm is used in conjunction with Newton and the **LVLCNV** parameter. The new time step prediction is based on the voltage and time values of the last 3 steps computed (i.e. 3 pairs of time/voltages). The Newton convergence criteria can be set to 0, 2, or 3 using the **LVLCNV** parameter. Other related parameters are: **RELTOL**, **VNTOL** and **ABSTOL**. This algorithm speeds up the simulation. Accuracy is also lost. Ratios of 2x are commonly obtained. This algorithm is not able to be selected with circuits containing BJT elements.

**LVLTIM=2**

Selects the Local Truncation Error algorithm. Eldo estimates the Local Truncation

Error (controlled by the **EPS** parameter) and adjusts its timestep accordingly. The smaller the **EPS** parameter is, the smaller the timestep will be. The timestep may also be controlled using the **TRTOL** parameter, a multiplier of the Local Truncation Error. When the Local Truncation Error is exceeded, the current step is rejected. This algorithm is the most accurate and is selected as default.

**LVLTIM=1**

This algorithm is used in conjunction with the **DVDT** parameter. If **DVDT**=0, the **DVDT** algorithm is activated, whereby the timestep is adjusted according to the **ABSVAR** and **RELVAR** parameters. The timestep is decreased if the current value, compared with the last value, is greater than either **ABSVAR** or a factor of **RELVAR**. When no **DVDT** value is given, the Local Truncation Error algorithm is used in the same way as for **LVLTIM=2**, except that no time step rejection occurs. This algorithm is normally faster than the Local Truncation Error algorithm in terms of CPU time, but is less accurate.

**LVLTIM=0**

Selects the Iteration Count algorithm, which may only be used if the circuit is solved in a single Newton block (**.OPTION NEWTON**). If the number of iterations needed to obtain convergence at the current timestep is less than the defined minimum (set in the **ITL3** parameter) the next timestep is increased. If, however, the number of iterations is greater than the defined maximum (set in the **ITL4** parameter) the current timestep is rejected and the next timestep is divided by the **FT** parameter.

- **MAXNODES=VAL**

To optimize memory allocation when simulating large circuits, the maximum number of circuit nodes may be specified. We recommend use of a value larger than the number of nodes in the circuit, to prevent memory re-allocation procedures.

- **MAXTRAN=VAL**

Specifies the maximum number of circuit components in order to optimize memory allocation when simulating large circuits. We recommend a value larger than the number of components in the circuit, to prevent memory re-allocation procedures.

- **MAXV=VAL**

Sets maximum voltage values for which Eldo searches for the DC operating point of a circuit. The options **VMIN** and **VMAX** also set bounds on DC operating point voltages, but **MAXV** overrides **VMAX**. Default is  $1.0 \times 10^{13}$  V. Thus **MAXV** must be specified if there are operating points greater than  $1.0 \times 10^{13}$  V in the circuit.

- **NETSIZE=VAL**

Provides a rough guess about the size of the Newton matrix.

- **NGTOL=VAL**

Is the absolute tolerance on voltage. Default value is set by **EPS**. This option is only used by the “Gear” algorithm or if option **QTRUNC** is set.

- **NMAXSIZE=VAL**

Sets the maximum number of nodes that can be contained in a single Newton block. Default value is  $6.0 \times 10^4$ . However, if the **NEWTON** option is explicitly specified, it has priority, and **NMAXSIZE** defaults to  $5.0 \times 10^6$ . This is in order for very large MOS circuits, containing over 60,000 nodes, to be simulated by default with OSR.

The **NMAXSIZE** option works in the following way:

Eldo attempts to partition:

If `nb_nodes < NMAXSIZE`, Eldo can attempt only one block, or it will attempt several blocks. This depends upon **EPS** and the option **BLOCK=>** specification.

Else, Eldo will attempt several blocks.

However, in the process of creating blocks, **NMAXSIZE** is not checked. The matrix grows depending on the connectivity, and the matrix size may exceed **NMAXSIZE**. The growth of the matrix cannot be stopped at that step, since the simulation would probably not work if we decide to split the matrix at those points where Eldo sees potential important feedback.

Once the blocks are completed, Eldo will attempt to concatenate blocks in case it sees feedback between blocks. However, it will not concatenate blocks if the sum of their size exceeds **NMAXSIZE**.

- **NOCONVASSIST**

Disables the automatic convergence aid mechanisms. This is specified to avoid other algorithms when non-convergence occurs. In this case Eldo will try the first algorithm and if there is a non-convergence problem the simulation is stopped and Eldo returns the list of nodes.

- **NOLAT**

Suppresses latency optimizations.

- **NONWRMOS**

This option is used to collapse intrinsic MOS transistor nodes. With this option, the effect of the parasitic resistors upon the channel current is still taken into account, although not as accurately as when the nodes are explicitly created. Particularly, some of the overlap capacitances in the MOS model become connected to the external drain/source, whereas they really are connected to the internal nodes. This may change results slightly.

See also [page 16-17](#) for further information.

- **NOQTRUNC**

Disables the charge-based *Local Truncation Error* algorithm. See option **QTRUNC**.

- **NOSWITCH**

Forces Eldo to ignore the **SWITCH** keyword specified as part of a subcircuit instance in the netlist.

- **PCS=0 | 1 | 2 | 3**

(Periodic Circuit Speedup) Used to increase the simulation speed for circuits with periodic or nearly periodic nature. PLLs in near-lock state belong to this category. The amount of speedup depends on the design nature. The speedup will be more significant with relatively large circuits. With circuits showing no periodicity at all, the option will not usually provide any speedup and may even slow down the simulation. Periodic Circuit Speedup is invoked by setting **PCS=1|2** for BSIM3v3 models only (**PCS=1|2** represent two possible speed optimization methods, **PCS=1** is more recommended) or **PCS=3** for BSIM4 and BSIM3v3 models (with same speed optimization as **PCS=1**). Default is 0. This option only supports the BSIM3v3 and BSIM4 models.

---

**Note**



BSIM4, unlike BSIM3, has many parasitic configurations: gate resistors, body resistors network, bias dependent access resistors, etc. These parasitic configurations are controlled by a set of instance and model parameters (Rdsmod, Rbodymod and Rgatemod). As the complexity of the parasitic configuration around the core model increases, the gain expected by the PCS decreases.

---

---

**Caution**



Use of the PCS option in the case of non-periodic conditions may even slowdown the simulation a little, which is why this option is not set by default. In this case, try using the PCS option in conjunction with the [.OPTPWL](#) or [.OPTWIND](#) commands to specify the time after which to turn on PCS.

---

- **PCSSIZE=VAL**

This option is used to specify the memory size (in MB) to be used by the PCS algorithm. The default value is 128. In many cases increasing this value will enhance the speedup achieved by the PCS option. However this value should not exceed the available physical memory to prevent memory allocation problems and excessive slowdown due to disk access.

- **PCSPERIOD=VAL**

This option can be used to specify a guess for the period of the circuit. When provided, this guess can be used to enhance the speedup achieved by the PCS option.

- **PIVREL**

Used in the LU-Factorization algorithm in order to find a compromise between the value of a pivot (The larger is a pivot the better it is from a numerical point of view) and the fill in of the matrix (the less fill-in we get the more efficient will be future matrix manipulations). Default value is 1.0e-3. User is invited to change this value whenever Eldo informs that the matrix is singular.

- **PIVTOL**

Is the absolute minimum value that can be accepted in the matrix to be a pivot for the LU-Factorization algorithm. Default is 1.0e-16. User is invited to change this value whenever Eldo informs that the matrix is singular.

- **PSOSC=VAL**

Allows control of the maximum number of oscillations that can occur when using a DC pseudo-transient algorithm during DC. This is to avoid a never-ending loop in cases where the amplitude of the oscillations does not decrease. Default value is 10. If the user knows that the oscillations amplitude decreases after n periods, **psosc=n** can be specified to increase the Eldo limit.

- **QTRUNC**

Forces Eldo to use a *Local Truncation Error* algorithm as in a SPICE like simulator. By default, the Eldo timestep is calculated from voltages using a predictor-corrector algorithm. This algorithm is suitable for IC circuits, however, for PCB-like circuits, it is preferable to use a *Local Truncation Error* timestep algorithm calculated from charges (and flux) as in SPICE like simulators. **QTRUNC** is automatically switched on if the circuit contains large current sources of above 1A, magnetic models or large power supplies. The option **NOQTRUNC** may be used to disable this algorithm. The *Local Truncation Error* algorithm, when working on charges or fluxes, uses the same control variables as the **GEAR** option, (i.e. **RELTRUNC**, **CGHTOL**, **FLUXTOL** and **NGTOL**).

- **RATPRINT=VAL**

If specified, then **DELMAX** is computed as:

$$\min\left(\frac{TSTOP}{50}, \text{RATPRINT} \times \text{TPRINT}\right)$$

If not specified, then the default algorithm is used and **DELMAX** is not computed from **.TRAN** specifications.

- **RELTOL=VAL**

Controls both the timestep size and the accuracy of Newton and/or IEM iterations. For voltages, convergence of iterations is reached when:

$$|V(i) - V(i-1)| < \text{RELTOL} \times \max(|V(i)|, |V(i-1)|) + \text{VNTOL}$$

where **V(i)** is the voltage value at current iteration *i* and **V(i-1)** is the previous iteration. For currents, circuit convergence is reached when:

$$|I(i) - I(i-1)| < \text{RELTOL} \times \max(|I(i)|, |I(i-1)|) + \text{VNTOL} \cdot \text{ITOL}$$

where **I(i)** is the current value at voltage iteration *i* and **I(i-1)** is the previous iteration. Default value is  $1.0 \times 10^3$ .

**RELTOL** also controls the time step size via the Local Truncation Error (LTE). This feature was added for compatibility with SPICE.

- **RELTRUNC=VAL**

Is the relative tolerance on the **CHGTOL**, **NGTOL** and **FLUXTOL** parameters above. Default value is set by **EPS**. This option is only used by the “Gear” algorithm or if option **QTRUNC** is set.

- **RELVAR**

Used for timestep control. Sets the relative voltage change for the condition that **LVLTIM**=1 and **DVDT**=0. If the nodal voltage at the present time point exceeds the nodal voltage at the last time point by **RELVAR**, the next timestep is reduced and a new solution is calculated. Default is 0.15.

- **SAMPLE=tval**

May be used to speed-up simulation of sampled circuits containing **OPA** macromodels and/or switch macromodels. The simulation timestep is set to the value **tval** specified. Eldo then computes additional intermediate timesteps (i.e. at **tval**/2, 3**tval**/2, 5**tval**/2, ...), in order to compute values when the signals should have reached their steady state values. The **SAMPLE** option also causes simplified **OPA** and switch macromodels to be used. The slew-rate of the simplified **OPA** macromodel is assumed to be infinite, and the simplified switch resistor value is either **RON** or **ROFF**, with no linear transition in-between.

- **SPLITC[ =val ]**

Simulation of floating capacitors (for example, those derived from extraction) can be optimized by changing the value of this option. This will force Eldo to split a capacitor object into two capacitors, each connected between ground and one node of the original capacitor. This leads to faster, but less accurate simulations. This splitting occurs for capacitors with values greater than **val**. Default is 50fF (1nF if **DIGITAL** option is specified).

This option is also used with Eldo Mach in a slightly different way. A floating capacitor affects the partitioning between Eldo and Mach solvers, if it connects two blocks that are assigned to the two different engines. Using this option allows the defined partitioning if capacitor value is less than **val**. In Eldo Mach, **val** defaults to 1fF.

- **STARTSMP=VAL**

Used in conjunction with **FREQSMP**. The **FREQSMP** command will be active after **STARTSMP**.

- **STEP=VAL**

Imposes a fixed timestep to be used by Eldo as defined by the **VAL** value. By default, Eldo uses a varying timestep.

---

**Note**



**.OPTION STEP** could produce incorrect results near breakpoints when used with the IEM method. This can occur if the **STEP** value is set greater than the rise and fall times of the circuit.

---

- **TIMESMP=VAL | {VAL1, VAL2, . . . VALn}**

Forces Eldo to compute a time point at every multiple time interval of **TIMESMP**, the sampling time interval. Equivalent to  $1.0/\text{FREQSMP}$ . Useful when performing a Fourier analysis. Multiple values can be specified as a list in which case Eldo computes timepoints corresponding to all sampled points. For example: a design has two clocks at 2MHz and 320 kHz. For uniform sampling, time sampling of  $0.5\mu\text{s}$  and  $3.125\mu\text{s}$  are needed to calculate the necessary exact points (for FFT post processing). Note that if options which impose the sampling frequency are set (for example **interpolate=0** on **.optfour**), the **timesmp** array is reset to a single value.

- **TRTOL=VAL**

Used for timestep control. Serves as a multiplier of the internal timestep generated by the Local Truncation Error timestep algorithm (**LVLTIM=2**). It is a factor that estimates the amount of error introduced in truncating a series used in the algorithm. This error is a reflection of what the minimum value of the timestep should be to reduce simulation time and maintain accuracy. The larger **TRTOL** is, the larger the timestep will be. Default value is 7.0.

- **TUNING=[FAST | STANDARD | ACCURATE | VHIGH | BACKANNOTATE]**

Selects the default mode of operation of Eldo as regards to precision and speed. This **TUNING** option acts as a macro-controller of Eldo in the sense that it enables the selection of algorithm and parameter settings via **EPS**. Any given parameter can be explicitly imposed, which will supersede the value related to the tuning setting. Note: the order of the option is important, the tuning type must be set first and then the user can adjust any parameter with specific values.

**FAST** keeps Newton as the default algorithm, unless options **OSR** or **IEM** were explicitly imposed. **FAST** gives milder values to some tolerance parameters (**ITOL**, **VNTOL** and **ABSTOL**, but not **EPS**) in order to have faster but still reliable simulation, even if slightly less accurate. Well suited to efficiently simulate large analog or mixed circuits

**STANDARD** (this is the default) causes **EPS** to be set to  $5.0\text{e-}3$  and activates the **NEWTON** option unless options **OSR** or **IEM** were explicitly imposed.

**ACCURATE** causes **EPS** to be set to  $1.0\text{e-}6$  and activates the **NEWTON** option unless options **OSR** or **IEM** were explicitly imposed.

**VHIGH** causes **EPS** to be set to  $1.0\text{e-}8$  and activates the **NEWTON** option unless options **OSR** or **IEM** were explicitly imposed.

**BACKANNOTATE** enables an improved Eldo solver to be used for handling back-annotated netlists with many parasitic elements. It can provide significant capacity and speed improvements (up to  $10\times$ ) for DC, AC, TRAN and all RF analyses for circuits which contain parasitic elements. The option is most efficient when the parasitics are defined in DSPF format in a separate file.

For greater flexibility, the improved solver can be activated or deactivated for a particular analysis by using one or several of the following options:

**TUNING=DC\_BACKANNOTATE** (activates the solver for DC analysis)  
**TUNING=AC\_BACKANNOTATE** (activates the solver for AC analysis)  
**TUNING=TRAN\_BACKANNOTATE** (activates the solver for Tran analysis)  
**TUNING=SST\_BACKANNOTATE** (activates the solver for all RF analyses)  
**TUNING=NODC\_BACKANNOTATE** (deactivates the solver for DC analysis)  
**TUNING=NOAC\_BACKANNOTATE** (deactivates the solver for AC analysis)  
**TUNING=NOTRAN\_BACKANNOTATE** (deactivates the solver for Tran analysis)  
**TUNING=NOSST\_BACKANNOTATE** (deactivates the solver for all analyses)

**Note**

 **TUNING** can also be specified using the time window **.OPTPWL** or **.OPTWIND** commands to apply or disable it during or outside some time intervals.

---

- **UNBOUND**

Enables an **EPS** value smaller than  $1.0 \times 10^{-10}$  to be specified. This option may be used for special applications that use low currents. Care must be taken, however, as problems with convergence may occur due to accumulation of round-off errors.

- **VMIN=x1,**  
**VMAX=x2**

Sets the minimum and maximum voltage values for which Eldo searches for the DC operating point of a circuit.

- **VNTOL=VAL**

Controls the voltage accuracy of the simulator when solving circuits using Newton Raphson techniques. Circuit convergence is reached when:

$$(|V(i) - V(i-1)| < \text{RELTOL} \times |\max(|V(i)|, |V(i-1)|)| + \text{VNTOL})$$

where **V(i)** is the voltage value at current iteration *i* and **V(i-1)** is the previous iteration. Default value is 1  $\mu$ V. Note **ABSV** is synonymous to **VNTOL**.

- **WDB\_IDELTA=VAL**

This option is used to define the delta I for use with all current plots, using this option will reduce the size of the wave database. Only for use with the JWDB (.wdb) output format. Default value is 0.0.

- **WDB\_VDELTA=VAL**

This option is used to define the delta V for use with all voltage plots, using this option will reduce the size of the wave database. Only for use with the JWDB (.wdb) output format. Default value is 0.0.

- **WDB\_NOSYNCHRO**

This option is used to send unsynchronized waves to the database using default VDELTA and IDELTA values. Using this option will reduce the size of the wave database. Only for use with the JWDB (.wdb) output format.

- **XA=VAL**

Diffusion length for MOS S/D calculation. Default value is 6 $\mu$ m.

$$W_{eff} = W - DW - 2 \times k_l$$

$$A_{Deff} = W_{eff} \times x_a$$

$$A_{Seff} = W_{eff} \times x_a$$

$$P_{Deff} = W_{eff} + 2 \times x_a$$

$$P_{Seff} = W_{eff} + 2 \times x_a$$

## Miscellaneous Simulation Control Options

- **AMMETER**

Prevents Eldo from eliminating zero voltage sources which are frequently used as current probes.

- **AUTOSTOP=0 | 1 | 2**

This option only works in Transient simulation, and only when the ‘Transient-extraction language’ is used, i.e. when functions **TPD**, **TRISE**, **TFALL**, **TCROSS** are used. This means that if the following line is present in the netlist, then the **AUTOSTOP** specification will be ignored, since **YVAL** is a keyword of the ‘General-purpose extract language’.

```
.EXTRACT YVAL(V(S), 5n)
```

**AUTOSTOP=0**

Deactivates autostop if necessary. Default.

**AUTOSTOP=1**

Causes Eldo to stop the simulation when all extracted waveform information (**.EXTRACT**) has been measured.

**AUTOSTOP=2**

Used in multi-step simulations. Causes Eldo to stop when all sweep measurements are completed.

- **AUTOSTOPMODULO=VAL**

This option is for use with the **AUTOSTOP** option. It only has an effect in TRANSIENT analysis. The evaluation of MEAS/EXTRACT will be performed only at every **AUTOSTOPMODULO** steps. This can be used to overcome the simulation slow down sometimes caused by setting **AUTOSTOP**, especially when the time spent to evaluate the MEAS/EXTRACT is large compared to the time taken to solve for the circuit. Default is 0, meaning that evaluation is performed at each step.

- **CARLO\_GAUSS**

Option for Gaussian distribution in Monte Carlo analysis. This option has two effects:

- All **LOT/DEV** statements having no distribution type specification, will be assumed to have a Gaussian distribution, i.e. default is:

**LOT/GAUSS or DEV/GAUSS**

Example:

```
VT0 = 1 LOT/UNIFORM = 20%
EOX = 10n LOT = 12%
.OPTION CARLO_GAUSS
```

In this example the distribution on **EOX** will be Gaussian, while the distribution on **VT0** will remain uniform.

- The envelope returned in the binary output files (or in the **.PRINT** commands) is computed from the standard deviation and do not correspond anymore to (**MIN**, **MAX**) of the waveform.
- **CPTIME=VAL**  
Stops the simulation if the CPU time exceeds **VAL** seconds. Message is displayed in the **.chi** file as well as being sent to the screen.
- **DEFPTNOM=VAL**  
Allows a parameter to be defined with the name **TNOM**. In such a case, this value will be used inside parameter expressions, instead of the value of **TNOM** set using option **TNOM=val**. The temperature value used by the Eldo model evaluator is always that set with option **TNOM=val**.
- **DSCGLOB=X | GLOBAL**

**X**

This is the default. When specified, nodes which appear in subcircuit definitions have priority over the nodes defined with **.GLOBAL** statements. See “[.GLOBAL](#)” on page 10-121. This option must be specified at the very beginning of the netlist.

**GLOBAL**

When specified, nodes defined with **.GLOBAL** statements have priority over the nodes which appear in subcircuit definitions. This is the default in **-compat** mode. See “[.GLOBAL](#)” on page 10-121. This option must be specified at the very beginning of the netlist.

#### **Note**



When Eldo finds a subcircuit definition which uses global nodes, Eldo issues a warning to let the user know how the situation is handled.

- **DSPF\_LEVEL=C | RC | RCC**

Defines the level of parasitics to use from a specified DSPF file. The option will use part of the information stored in the RC extracted DSPF file specified by **C**, **RC** or **RCC**.

**DSPF\_LEVEL=C**

Where **C** is the intrinsic and coupling capacitance. The pre-layout nets will obtain a grounded capacitance, as specified in the DSPF file in the line **\* | NET**.

**DSPF\_LEVEL=RC**

Where **RC** is the distributed RC elements. The ideal pre-layout nodes will be

replaced with RC elements. Also the coupling capacitance between wires/layers is ignored, that is any capacitor that is not grounded will ignored.

**DSPF\_LEVEL=RCC**

Where **RCC** is both intrinsic and coupling capacitance and distributed RC elements.

It is assumed that the DSPF file has been extracted with all RCC information.

**Note**

This option can also be specified in the **.DSPF\_INCLUDE** command, see [page 10-79](#). If specified in the command it will overwrite this option.

- **FLOATGATECHECK**

Enables Eldo to issue a warning when a floating gate is detected and resume the simulation. Eldo will not consider reverse-biased diodes as active elements when checking for floating gates. Reverse-bias conditions are detected as follows:

- If the node is connected to the 1<sup>st</sup> pin of a diode, then the other pin must be connected to the positive node of a power supply to be considered as reverse-bias.
- If the node is connected to the 2<sup>nd</sup> pin of a diode, then the other pin must be connected to either ground or the negative pin of a power supply to be considered as reverse-bias.

This option can be used in conjunction with the option **FLOATGATERR** and **FLOATGATE0**.

- **FLOATGATERR**

Enables Eldo to stop the current process and generate an error when floating gates are detected. This option must be used in conjunction with the option **FLOATGATECHECK**. It can not be specified by itself.

- **FLOATGATE0**

This option will force detected floating gates to 0. It can be useful to change the topology of a circuit and achieve better convergence. It can be used in conjunction with the option **FLOATGATECHECK**.

- **FALL\_TIME=VAL**

Transition time in seconds for signal changing from high to low state. Used by the interactive LOW command. Default value is 1ns.

- **RISE\_TIME=VAL**

Transition time in seconds for signal changing from low to high state. Used by the interactive HIGH command. Default value is 1ns.

**Note**

The following two options (**HIGHVOLTAGE** and **LOWVOLTAGE**) are only used by the **.TVINCLUDE** command and by interactive commands.

- **HIGHVOLTAGE=VAL**  
Sets the upper bus signal voltage level. Default is 5V.
- **LOWVOLTAGE=VAL**  
Sets the lower bus signal voltage level. Default is 0V.

**Note**



The following two options (**LOWVTH** and **HIGHVTH**) are only used by the **.TVINCLUDE** command.

---

- **LOWVTH=VTH1**  
See below. Default value is 2.4V
- **HIGHVTH=VTH2**  
Default value is 2.6V.

The VTH parameters are required by Eldo to compute the **HEX** (or **DEC**, **OCT**, or **BIN**) value on the bus from the analog value inside Eldo. Then, it compares this value with the value expected and displays the error if they are not the same.

When only vth1 is given:

If value < vth1 then logic state 0.

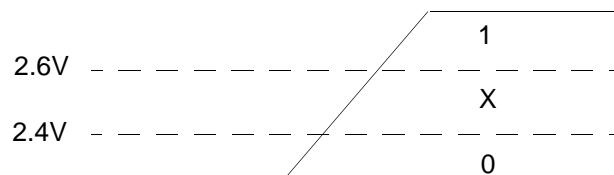
If value > vth1 then logic state 1.

HighVth=vth2 is used to plot the indeterminate value as shown below:

If value < vth1 then logic state 0.

If vth1 < value < vth2 then state X.

If value > vth2 then logic state 1.



- **INTERP**

Causes Eldo to generate data in the binary waveform output file in the same way it does in the **.chi** file. Therefore, rather than producing data as it is calculated it will only produce points at each timestep specified in the analysis. Example:

```
.TRAN 1n 10n
.OPTION INTERP
```

This will generate data in the binary output file every 1ns. If **INTERP** is not specified, then Eldo will dump points that it has actually computed.

**.OPTION**

- **ICDC** and **ICDEV**

By default, Eldo behaves like Spice 2g6 regarding **.IC** commands and **IC** parameters specified on devices: **.IC** commands are taken into account only for DC done before Transient analysis, or when **.TRAN ... UIC** is specified. However, **IC** parameter specifications on devices are taken into account only in the case of **.TRAN ... UIC**.

- **ICDC**

In this case, **.IC** commands (and **IC** parameter specifications on devices when option **ICDEV** specified) will be taken into account for any DC analysis.

- **ICDEV**

In this case, **IC** parameters specified on devices and **.IC** commands will be handled the same way.

- **LICN**

By default, the first IC specification has precedence over subsequent IC specifications. Setting **LICN**, the last IC specification will have precedence.

The same remark applies to **.NODESET**, or **.GUESS**. Setting **LICN**, the last **.NODESET** (or **.GUESS**) specification will have precedence over the previous **.NODESET** (or **.GUESS**) specifications.



See “**.IC**” on page 10-125, “**.GUESS**” on page 10-122, and “**.NODESET**” on page 10-179

**Note**

Whether or not **LICN** is specified, it remains that **.NODESET** always has precedence over **.GUESS**, and whenever **.IC** commands are active (i.e. during the DC which is done prior to TRANSIENT simulation, or whenever option **ICDC** is active), then **.IC** has precedence over **.NODESET**.

- **M5.3**

Specifies the ruling surrounding the M factor should be as it was for versions of Eldo before and including v5.3. Example:

```
.SUBCKT mysubckt in out m=5
r1 in 1 1k m='2*m'
r2 1 out 1k
.ends
X1 in out mysubckt m=3
```

The above example used to work for previous versions of Eldo (Eldo would simulate as if there were six **X1.r1** devices in parallel, but only one **X1.r2**). For Eldo version v5.4 and higher, the rule regarding the **m** factor changed. The **m** value can appear on the X instance or on the device. The **m** value cannot appear anymore on the Right-Hand-Side of the expression. In the example above, Eldo will issue an error. To avoid this error, it is

therefore mandatory to replace the **m** parameter in any expression, by another parameter name. In the example below, no confusion is possible:

```
.SUBCKT mysubckt in out mr=5
r1 in 1 1k m='2*mr'
r2 1 out 1k
.ends
X1 in out mysubckt mr=3
```

Here, the user will still have six devices of `X1.r1`, and only one of `X1.r2`. Of course, if the user put the **m** factor on the subcircuit instantiation:

```
X1 in out mysubckt m=3 mr=3
```

Here, 18 devices of `X1.r1`, and three devices of `X1.r2` will be emulated.

For backward compatibility, you can invoke Eldo with the `-m53` flag, or use option **m53** (to be placed at the beginning of the `.cir` file, just after the title).

- **NOMEMSTP**

Setting this option means Eldo will not use the results of the previous STEP run as an initial guess for the next one. By default, without this option, the second run in **.STEP** uses the result of the previous STEP as an initial guess.

- **PARAMOPT\_NOINITIAL**

This option can be used for the Eldo optimizer. Allows the initial value parameter of the **.PARAMOPT** syntax to be omitted. The value already available in the netlist through the normal **.PARAM** specification will instead be used.

- **PODEV**

This option is used to specify **DEV** variation for a declared Monte Carlo analysis distribution parameter when the parameter is used as part of either a model or instance parameter. By default **LOT** variation is used for model parameters and **DEV** is used for instance parameters. In Eldo versions prior to v6.3\_2 **DEV** variation was the default for both model and instance parameters.

- **RANDMC**

Reset the MC randomly. The sequence of random numbers differs from one run to the next.

- **RGND=val**

When this option is set a resistance is connected between a node and ground for nodes that have no DC path to ground (warning message 118), for example a node to which only capacitors are connected. This allows the simulation to proceed instead of producing errors. It differs from option RGNDI which is used only for dangling nodes connected to a current source. This option connects a resistance of `val` and allows the simulation to proceed.

- **RGNDI=val**

When this option is set a resistance is connected between the dangling nodes of independent current sources and ground. Normally if Eldo detects a dangling node on a current source an

**.OPTION**

error is issued and the simulation stops. This option connects a resistance of `val` and allows the simulation to proceed.

- **SIGTAIL=VAL**

This option can be used in Monte Carlo analysis. The value represents the relative maximum extension of the Gaussian tail when Gaussian distribution is used. The default is 4. Example:

```
.param p1=1 lot/gauss=0.1
.option sigtail=6
```

Here, `p1` values will be in the range [-0.6, 0.6]. If **SIGTAIL** is not specified, `p1` values will be in the range [-0.4, 0.4].

- **TNOM=VAL**

Sets the model temperature, i.e. the temperature defined in the model. Default is 27°C.

- **TPIEEE**

For each port for which **NOISETEMP** is not specified, then the value 16.85 will be used rather than the temperature of the circuit. See [page 5-9](#) and [page 5-16](#) in the V&I Sources.

- **ULOGIC**

Causes a current source/RC combination to be used on the output of Eldo logical primitives **AND**, **NAND** etc. By default, the output of Eldo logic gates acts as a pure voltage source, making simulation very fast, but not allowing easy arbitration of signals when logical outputs are connected together. By using **ULOGIC**, depending on the values of driving resistance used, such conflicts may be resolved. Note that in this mode of operation, simulation speed is slightly slower than the default voltage source method.



See the **.MODEL ... LOGIC** command on [page 7-3](#) for details of output resistor and propagation delay model parameters.

- **ZOOMTIME=VAL**

The hold times of **PWL**, **PULSE** and **PATTERN** sources are extended by multiplying them by a factor `VAL`. Source rise and fall times remain unchanged.

## Model Control Options

- **ACM**

Usually **ACM** is just a parameter which is an alias to **ARLEV**; i.e. it overwrites the values of **ALEV** & **RLEV**. However, if you specify **ACM** in the netlist, then **ACM** has a much more effective role. This option is automatically set if COMPAT mode is used.



Please refer to “[.OPTION ACM](#)” on page 12-38 of the *Eldo Device Equations Manual*.

- **ASPEC**

Used with ASPEC MOSFET models (Eldo level 16). This controls use of **SCALE** and **SCALM** values with the models. When this option is specified, the following conditions are set:

```
.OPTION SCALE = 1U
.OPTION SCALM = 1U
.OPTION WL
```

- **BSIM3VER=VAL**

There are several versions of the BSIM3v3 model: BSIM3v3 (selected by **VAL=3 . 0**), BSIM3v3.1 (selected by **VAL=3 . 1**) or BSIM3v3.2 (selected by **VAL=3 . 2**). Default is 3.2. (Note, in Eldo 4.5\_x.x, the default was 3.0.)

- **DEFAD=VAL**

Sets the default value for MOS drain diffusion area. Default is zero.

- **DEFAS=VAL**

Sets the default value for MOS source diffusion area. Default is zero.

- **DEFL=VAL**

Sets the default value for MOS channel length. Default is 100 $\mu\text{m}$ .

- **DEFW=VAL**

Sets the default value for MOS channel width. Default is 100 $\mu\text{m}$ .

- **DEFNRD=VAL**

Sets the default value of MOS parameter **NRD**. Default is zero.

- **DEFNRS=VAL**

Sets the default value of MOS parameter **NRS**. Default is zero.

- **DEFPD=VAL**

Sets the default value for MOS drain perimeter. Default is zero.

- **DEFPS=VAL**

Sets the default value for MOS source perimeter. Default is zero.

- **ELDOMOS**

Sets option **ACM** together with the items listed below:

- sets the default value of **TNOM=25** instead of the normal 27
- if **LD** is not specified, **LD** =  $0.75 \cdot XJ$
- $Ids = Ids + g_{min} \cdot V_{ds}$
- for **RLEV=5**: (if **defnrd** & **defnrs** not defined)
  - sets the default value of **NRD=0**
  - sets the default value of **NRS=0**

**.OPTION**

- if (accusim2), sets a lower limit for ***Isat*** which is  $1 \times 10^{-18}$  for BSIM2 and  $1 \times 10^{-15}$  for any other model
- calls the impact ionization model if !accusim2 & !BSIM2
- required for the usage of LEVEL 17 (SPICE-A LEVEL 8)
- allows the separate use of both ***COX*** and ***TOX*** at the same time for Levels 1, 2 & 3
- for Level 2:  
uses ***CAPLEV=4*** for the calculation of the intrinsic charges & capacitances
- for Levels 1, 2, 3, 16 & 17:  
 $\text{Leff} = \text{Ldrwan} \cdot \text{LMLT} + \text{XL} - 2(\text{LD} + \text{DEL})$   
 $\text{Weff} = \text{Wdrwan} \cdot \text{WMLT} + \text{XW} - 2\text{WD}$
- for Levels 1, 2, 3, 16 & 17:  
uses SPICE-A mobility temperature equations  
uses SPICE-A surface potential (***Phi***) temperature equations according to ***TLEV***  
uses SPICE-A VBI · VTO temperature equations according to ***TLEV***
- for the AMS model Level 15:  
uses another capacitance model rather than SPICE 2G6 if ***CPOP=0***
- switches automatically:  
 $\text{Level1} \Rightarrow \text{Level 11}$   
 $\text{Level2} \Rightarrow \text{Level 12}$   
 $\text{Level3} \Rightarrow \text{Level 13}$
- if !BSIM2 & !MOSP9: adds ***2WD*** to ***Weff*** used for the calculation of the overlap capacitance to cancel the effect of the already subtracted ***2WD***.

**• FNLEV**

Selects between the two flicker noise models of the BSIM3v3 model with **NOIMOD=1&4**.

**• GMIN=VAL**

Sets the conductance value that is placed in parallel with all pn junctions and drain and source nodes of MOSFET models. It enhances the convergence properties that are degraded by having too low a value of OFF conductance for pn junctions and MOSFET devices. Large values of ***GMIN*** may cause unreasonable circuit response. Default is  $1.0 \times 10^{-12}$ .

**• GMIN\_BJT\_SPICE**

A defect was fixed in the BJT level 1 and 5 models for the Gmin handling. This defect caused unrealistic high leakage current values. This defect was also found in the original Spice code. The fix affects the backward compatibility of results in some test cases. Use option ***GMIN\_BJT\_SPICE*** to reproduce results for backward compatibility.

**• GMINDC=VAL**

Similar to option ***GMIN***, however the value specified via ***GMINDC*** is used for DC analysis only. Defaults to ***GMIN***.

- **GRAMP=VAL**

Ramps the conductance value that is placed in parallel with all pn junctions and all drain and source nodes of MOSFET level 12 and 13 from **GMIN**.10<sup>GRAMP</sup> down to **GMIN**. This helps DC convergence in some circuits. This option may also be used in conjunction with the **.RAMP** command. See “**.RAMP**” on page 10-269. Default value is 0.

- **GENK[ =VAL ]**

Forces Eldo to generate 2nd order mutual inductors. It is activated if **val** is any value greater than zero or if no value is stated. If option **COMPAT** is set it is automatically activated and can be deactivated using **GENK=0**. Used together with the **KLIM** option.

- **KLIM=VAL**

Used together with the **GENK** option to force Eldo to generate 2nd order mutual inductors. Default is 0.02.

In the case of the following example:

```
K1 L1 L2 val1
K2 L1 L3 val2
```

by specifying the **GENK** option, a new mutual **K3** between **L2** and **L3** will be generated, if there is no such mutual already present in the netlist. The value of the new mutual created would be **val1×val2**, and the mutual coefficient is only created if **val1×val2** is higher than the value of **KLIM**. However, the new mutual **K3** will not be generated if:

- **GENK** is equal to zero,
- the mutual coefficient is lower than the value of **KLIM**, or
- a mutual is already present in the netlist.

**Note**

A list of the mutuals, which are automatically created, is dumped in the ASCII output file.

- **IBIS\_SEARCH\_PATH="FILEPATH"**

Specifies the path to the directory to search for IBIS files.

- **MAXADS=VAL**

Sets the maximum value for the MOS source diffusion area or drain diffusion area. No default is specified.

- **MAXL=VAL**

Sets the maximum value for the MOS channel length. No default is specified.

- **MAXPDS=VAL**

Sets the maximum value for the MOS source diffusion perimeter or drain diffusion perimeter. No default is specified.

- **MAXW=VAL**  
Sets the maximum value for the MOS channel width. No default is specified.
- **MINADS=VAL**  
Sets the minimum value for the MOS source diffusion area or drain diffusion area. No default is specified.
- **MINL=VAL**  
Sets the minimum value for the MOS channel length. No default is specified.
- **MINPDS=VAL**  
Sets the minimum value for the MOS source diffusion perimeter or drain diffusion perimeter. No default is specified.
- **MINW=VAL**  
Sets the minimum value for the MOS channel width. No default is specified.
- **MINRACC=VAL**  
If the resistor value is below **MINRACC**, Eldo will not create access resistor of devices MOS, BJT, Diode or JFET. Default is undefined, i.e. access resistors are always created.
- **MINRESISTANCE=VAL**  
Using this option is equivalent to using the options **RMMINRVAL** and **MINRVAL**. Resistors with a value less than the specified **VAL** will be removed *before* partitioning occurs and a zero voltage source will be inserted between the two pins, and then one of the pins removed. The resistor cannot be reactivated when this option is used. Resistors that are connected to a pin which appears in a **.SUBCKT** line, will not be removed.

**i** To allow a resistor to override this option use the **KEEPRMIN** parameter. For more information see [page 4-5](#). For resistors connected directly to a voltage source, see option **RAILRESISTANCE**.

- **MINRVAL=VAL**  
Removes resistors with absolute values below **val**. It emulates a zero voltage source between the two pins of the resistor. By default, **MINRVAL** is not specified. This means that no action will be taken regarding resistor devices.

**i** To allow a resistor to override this option use the **KEEPRMIN** parameter. For more information see [page 4-5](#).

**Note**



The connection is not allowed in a case where suppression of the resistor would create a Voltage loop. For the same reason, (i.e. prevention of a Voltage loop) resistors connected to Y elements are never removed.

To prevent cases where the resistor value would become larger than **MINRVAL** in a new simulation, the resistor is reactivated. This could happen in instances where the resistor's value is controlled by a parameter, varying according to a **.STEP** command for instance.

• **MNUMER**

Applies to MOS devices. Sets the derivative computation method to the finite difference method (DERIV=0). By default, DERIV=1 for analytical derivatives.

• **MOD4PINS**

Sets the number of pins that must be specified in the MOS instantiation to 4. (The only model that can have more than 4 pins is the BSIM3SOI model.) Setting this option has the effect that for the following line:

```
M1 D G S B MOD W L AD AS PD PS NRD NRS M=<value>
```

MOD is considered as the model name. Otherwise, Eldo would look for a model named NRS, with M1 assumed to have 12 pins.

• **MODWL**

By default, Eldo behaves as if this option is set. It enables the use of MOS model versions which can be selected via **.MODEL** command **W** and **L** parameters (binning parameters). Whenever Eldo finds a MOS device for which the model name has no **.MODEL** command, it searches through all defined models for a model of the same root name and whose **W/L** range matches the specified device size, e.g.

```
M1 1 2 3 4 MODROOT w=2u l=3u
.MODEL MODROOT.1 NMOS VT0=1 WMIN=3u WMAX=5u
+ LMIN=1u LMAX=5u
.MODEL MODROOT.2 NMOS VT0=2 WMIN=1u WMAX=5u
+ LMIN=1u LMAX=5u
```

In this case, Eldo will assign the model MODROOT.2 to the MOSFET M1.

**Note**



The separator in the **.MODEL** command should be the dot character “.”. Eldo selects the model to be assigned to MOS devices according to the geometric size of each device, even if these geometric sizes are modified at run-time via **.STEP** commands. In previous versions, the selection of the model was done just once at the very beginning of the simulation, and was not changed at run time.



For further details please see option **MODWL** on [page 4-131](#).

- **MODWLDOT**

This option is used for binned models. By default, extension of binned model is either:  
`<root>.<extension>` or `<root>_<extension>`

This can be confusing when `<root>` also contains the character `'.'`. If **MODWLDOT** is set, only `'.'` will be allowed as a separator, i.e. character `'_'` can appear in the `<root>`. This option is automatically activated with the `-compat` flag.



See also option **MODWL** on [page 11-35](#).

- **NGATEDDEF [ =node\_name ]**

Specifies NMOS floating gates to be connected to the specified node name. Default is node 0.

- **NOAUTOCYPE**

Disables automatic checking by Eldo of the dependencies of the capacitor value. Without this option, if Eldo finds that the capacitor value does not depend on the bias across the terminal of the capacitor, then Eldo will behave on that device as if CTYPE=1 had been set. With this option set, Eldo will behave as if CTYPE=0 (default) has been set, unless it is otherwise explicitly specified.

- **NWRMOS**

Forces all access resistors of MOS that are connected to nodes which are to be solved by NEWTON, to be created as objects, i.e. as if the access resistors had been explicitly instantiated in the netlist. This is done for sake of accuracy. Default is **NWRMOS**.

- **PGATEDDEF [ =node\_name ]**

Specifies PMOS floating gates to be connected to the specified node name. Default is node 0.

- **RAILRESISTANCE=VAL**

This option is similar to the **MINRESISTANCE** option. Resistors connected directly to a voltage source and below the specified value `VAL` will be removed. A zero ohm resistor will be inserted between the two pins, and then one of the pins removed. The resistor cannot be reactivated when this option is used.

- **REDUCE**

When this is set, multiple identical subcircuits that follow each other are reduced into a single instance using the `M` parameter, for example:

```
X1 1 2 FOO A = 1 B = 1
X2 1 2 FOO A = 1 B = 1
X3 1 2 FOO A = 1.0 B = 1
.OPTION REDUCE
```

Here, X instances `X1` and `X2` will be replaced by:

```
X1 1 2 FOO A = 1 B = 1 M = 2
```

but `X3` will remain as it is because the character string for A does not match.



This also applies to BJTs, MOSFETs and diodes. For more information see [page 4-111](#), [page 4-127](#) and [page 4-104](#) respectively.

---

- **RESNW=val**

Resistors with values higher than `RESNW` are allowed to be solved by OSR. Default is  $1.0e18 \Omega$  (i.e. virtually infinity).

- **RMMINRVAL**

With this set option `MINRVAL=val` will emulate a `.CONNECT` through the 2 pins of the resistor. The advantage of this is that it reduces the number of nodes to be solved, but the disadvantage is that the resistor cannot be reactivated.



To allow a resistor to override this option use the `KEEPRMIN` parameter. For more information see [page 4-5](#).

---

- **RMOS**

When this is set, all MOS access resistors are unconditionally created as objects. Default is `NORMOS`.

**Note**



Whenever MOS access resistors are not created as objects, Eldo uses an iterative process to handle their effects. This process is less CPU time consuming than the resolution of the additional nodes which are created to connect the access resistors to the MOS devices.

---

- **RSMALL=VAL**

Resistors with a value smaller than `val` will be set to `val`. Default is  $1.0e^{-6} \Omega$ .

- **RZ=VAL**

This will set the *global* value (VAL) for `RZ` which will enable Eldo to detect a ‘Z’ state on *all* A2D nodes when the option `ZDETECT` is used. A ‘Z’ state is detected when the equivalent impedance of the A2D node exceeds the specified `RZ` value.

- **SCALE=VAL**

Multiplier for MOS width, length, perimeter of drain and source. This command also scales the `.MODEL` parameters `WMIN`, `WMAX`, `LMIN`, `LMAX` in the same way. The parameters `AD` and `AS` are multiplied by  $x^2$ .



For more information on these parameters, refer to the [Device Models](#) chapter. For more information on scale factors see [page 3-5](#). The keyword `SCALE` can be used in expressions. For more information on such keywords see [page 3-3](#).

---

- **(NO) KWSCALE**

**SCALE** can be considered as a keyword by Eldo (**KWSCALE** set), or not (**NOKWSCALE** is set).

When **SCALE** is considered as a keyword, **SCALE** can appear in expressions, and its value will be that assigned via option **SCALE=val**.

When **SCALE** is not considered as a keyword, **SCALE** can also appear in expressions, but its value must be defined via a **.PARAM** statement, as is the case for any other parameter.

When **-compat** is set at invocation of the simulator, **NOKWSCALE** is assumed to be set, and can be switched with option **KWSCALE**.

When **-compat** is not set at invocation of the simulator, **KWSCALE** is assumed to be set, and can be switched with option **NOKWSCALE**.

- **SCALEBSIM=VAL**

Scales all sensitivity parameters of the BSIM1 and BSIM2 MOSFET models. For example, **VFB** is a basic parameter which is corrected by the length and width sensitivity parameters, **LVFB** and **WVFB**. **SCALEBSIM** will scale these parameters by the factor **VAL**. Default value is 1, e.g.

$$vfb = VFB + (LVFB \cdot SCALEBSIM/L) + (WVFB \cdot SCALEBSIM/W)$$

- **SCALM=VAL**

Scaling factor for the model parameters **LDIF**, **DL** and **DW** (MOS), **DW** and **DLR** (RC wire). e.g. **DL** is equivalent to the **LVAR** parameter for the MM9 models (see the “[Philips MOS 9 Model \(Eldo Level 59 or MOSP9\)](#)” on page 4-157). **SCALM** can be individually defined for each model card using the model parameter **SCALM**. This overrides the global **SCALM** value defined using the **.OPTION** command.



For more information on these parameters, refer to the [Device Models](#) chapter.

- **SOIBACK=<grounded\_voltage\_source>**

This option applies to the SOI model only (BSIM3SOI model). Instances of such models can accept 4 or 5 pins in the netlist. When the **SOIBACK** option is specified and the MOS device is defined as a 4 pin device, then the 4th pin is the internal body of the device. This option defines a voltage source allowing Eldo to translate a 4-pin SOI instance into a 5-pin SOI instance. The backgate will be inserted and connected automatically to the voltage source defined by the **SOIBACK** option.

- **SPMODLEV**

Beginning Eldo v6.5 the SP model was enhanced with analytical derivatives. Analytical derivatives replace the numerical derivatives and improves the speed of the model. By default Eldo uses the new implementation. To select the old model implementation, set the Eldo option **SPMODLEV** for backward compatibility.

- **TMAX | TMIN=VAL**

Some models contain self-heating effects, such as VBIC or Hicum models. For such models, when self-heating is active, the temperature of the device becomes an unknown to the system. In order to prevent possible overflow in model evaluation, device temperature is limited by Eldo, and by default cannot exceed 1000K, or go below 0K. These values can be overwritten using options **TMIN=VAL** or **TMAX=VAL**.

- **USEDEFAP**

If the model parameter **ACM** is set to 2 or 3, then **AD**, **AS**, **PD** and **PS**, when unspecified in the instance command, are computed from **W**, **L** and **HDIF**, regardless of what values are given to **DEFAD**, **DEFAS**, **DEFPD** and **DEFPS**. If you wish to use **DEFAD**, **DEFAS**, **DEFPD** and **DEFPS**, option **USEDEFAP** has to be set.

- **VBICLEV**

Beginning Eldo v6.5 the VBIC model was modified to improve the speed of the model by a factor of up to 4x. This was achieved with code restructuring and the implementation of a new algorithm. By default Eldo uses the new implementation. To select the old model implementation, set the Eldo option **VBICLEV** for backward compatibility.

- **WARNING\_DEVPARAM**

Forces Eldo to print a warning instead of an error when an unknown parameter is specified on a device instance. For example:

```
ERROR 254: OBJECT "M1": Unknown parameter MULU0
```

will be replaced by:

```
Warning 209: OBJECT "M1": Parameter ignored MULU0
```

- **WARNMAXV=VAL**

Returns a warning if a Voltage on a node is higher than **VAL**. This applies to DC analysis only.

- **WL**

Reverses the order of MOS length and width specification.

- **YMFACt**

Allows the use of the device multiplier **M** in Y instantiations. This must be specified at the beginning of the netlist.

- **ZDETECT**

When this option is set, it will enable Eldo to detect 'Z' states on A2D nodes where an **RZ** value has been specified either in the **.A2D/.model** card or with the option **RZ=VAL**. A 'Z' state is detected when the equivalent impedance of the A2D node exceeds the specified **RZ** value command.

## RC Reduction Options

- **RC\_REDUCE**

Simplifies complex RC nets, reducing the number of resistors, capacitors, and nodes with the Branch Merge Reduction<sup>1</sup> and the TICER method<sup>2</sup>. Eldo will remove some RCLK networks and replace them by equivalent RCLK nets. Therefore some nodes can be removed. The user can explicitly prevent the **RC\_REDUCE** option from eliminating a node during RCLK reduction by including a corresponding **RC\_REDUCE\_PORT** option in the netlist. Using the **RC\_REDUCE** option will slow the Eldo parser a little depending on the number of RC nets. However, the simulation speed should increase significantly. (This option was named **ELDO\_RC\_REDUCE** in releases prior to Eldo v6.5\_2.)

- **RC\_REDUCE\_METHOD=RC | RCLK**

**RC** indicates that the TICER method of RC reduction is to be used.

**RCLK** indicates that the Branch Merge method of RCLK reduction is to be used.

By default, when this option is not set, Eldo performs a RCLK Branch Merge Reduction followed by a TICER RC reduction.

- **RC\_REDUCE\_PORT=" {node\_name} "**

Prevents the **RC\_REDUCE** option from eliminating a node during RC reduction. Note that the double quotes are mandatory. (This option was named **ELDO\_RC\_PORT** in releases prior to Eldo v6.5\_2.) By default, nodes related to output statements and nodes connected to any non-RC device are not eliminated and need not be protected with this option.

- **RC\_REDUCE\_FMAX=VAL**

Sets the cut-off frequency for RC/RCLK reduction. Eldo tries to preserve the response of the circuit for frequencies ranging from DC up to the specified cut-off frequency when RC/RCLK reduction is used. By default, the cut-off frequency is the highest frequency that can be numerically represented according to the simulation commands and options present in the netlist.

---

**Note**



For transient simulations the default cut-off frequency is the inverse of the minimum internal step (see HMIN option); this is a strongly conservative choice and the user is highly recommended to specify a cut-off frequency more representative of the highest (internal) signal frequency expected for the circuit.

1. B. Sheehan, *Branch Merge Reduction of RLCM networks*, proceedings of the IEEE International Conference on Computer-Aided Design, 9-13 Nov. 2003, pp. 658-664.

2. B. Sheehan, *TICER: Realizable Reduction of Extracted RC Circuits*, proceedings of the IEEE International Conference on Computer-Aided Design, 7-11 Nov. 1999, pp. 200-203.

**Note**



While using Eldo RF for steady-state simulation of autonomous circuits (.SSTOSCIL analyses) the default cut-off frequency cannot be deduced from the netlist. Consequently a default of 50 GHz that is conservative for most oscillator circuits is used. For such simulations, the user is recommended to specify a cut-off frequency that is a close upper bound of the expected highest signal frequency in the circuit (e.g. for a single-tone .SSTOSCIL analysis, an upper bound of the expected oscillation frequency times the number of harmonics).

---

## Noise Analysis Options

- **FLICKER\_NOISE=VAL**

Used in Noise Analysis as a frequency dependent noise model selector. Default value is zero. Values 0, 1, 2, 3 are used. Same functionality as using the **FLKLEV** model parameter.

- **THERMAL\_NOISE=VAL**

Used in Noise Analysis as a temperature dependent noise model selector. Four values are used: 0, 1, 2, 3. Same functionality as using the **THMLEV** model parameter.



See the appropriate sections in the [Device Models](#) chapter for details of the device noise models used for each case for the above two noise analysis options.

---

- **IKF2**

Specifies that the **ONOISE** value returned is in  $V^2/Hz$ , instead of  $V/\sqrt{Hz}$ .

- **JTHNOISE=VAL**

Selects the equations for thermal noise in JFETs. The equations are as follows:

**JTHNOISE=0**

Default equation is used:  $id = \frac{8kT}{3} \cdot gm$

**JTHNOISE=1**

If  $vds > vdsat$   $Sid = \frac{8kT}{3} \cdot gm$

Else  $Sid = 4kT \cdot (gm + gds)$

**JTHNOISE=2**

$Sid = \frac{8kT}{3} \cdot (gm + gds) \cdot \left( \frac{3}{2} - \frac{Vdseff}{(2 \cdot VDSAT)} \right)$

where  $Vdseff = \min(Vds, VDSAT)$

- **NONOISE**

By default, it is assumed that all devices contribute to the total output noise, unless this option is specified. Setting **NONOISE** assumes all devices to be noiseless.

- **NOISE\_SGNCONV**

Allows the user to change the sign convention used for the computation of the noise parameters **BOPT** and **PHI\_OPT**.

## Simulation Display Control Options

- **ACSIMPROG**

This option displays in the terminal window the simulation progress (percentage) during an AC analysis. This can be useful to monitor long simulations.

- **DCSIMPROG**

This option displays in the terminal window the simulation progress (percentage) during a DC analysis. This can be useful to monitor long simulations.

- **MSGBIAS=[VAL]**

Usually there will only be three messages mentioning that PMOS are connected to 0. Use the **MSGBIAS** option without specifying a value if you want all such messages, or specify a limit with **VAL**.

- **WBULK**

Eldo would print out a warning about positive bias on NMOS bulk (or negative bias on PMOS bulk) only once, unless **WBULK** is set. When set, all warnings will be printed out.

- **MSGNODE=VAL**

Limits the number of node connection faults reported. Eldo reports four types of node connection faults as follows:

```
Warning 107: node "xxx": Less than two connections.
Warning 108: node "xxx": This node is a floating gate.
Warning 113: node "xxx": Not connected to any element.
               This node is removed from the netlist.
Warning 252: OBJECT "xxx": Self-connected object not created.
```

If **MSGNODE=0** then all connection fault warnings are displayed. By default, **MSGNODE** is set to 3, which means Eldo displays each type of connection fault for the first three nodes on which the fault is detected. If the number of nodes at which a fault is detected exceeds the number specified by this option, then the following warning message is issued:

```
Warning 29: Set .option MSGNODE=0 to receive all such warnings.
```

- **NOWARN**

Suppresses the display of warnings. By default, all warnings are displayed. This option must be placed at the top of the design, just after the title line, otherwise the warnings will continue to be displayed.

- **NOWARN=xxx**

Only warning `xxx` will *not* be printed by Eldo. This option must be placed at the top of the design, just after the title line, otherwise the warnings will continue to be displayed.

- **INGOLD=VAL**

This option is used for the printout of double precision numbers. `VAL` can be 0, 1 or 2.

**INGOLD=0**

Use engineering format: exponents are given from a single character with the same convention as that for the input file. However, the `1.0e6` is expressed as X rather than as MEG.

**INGOLD=1**

Numbers between 0.1 and 999 are written without E format. Other numbers use the SPICE 2G6 format.

**INGOLD=2**

SPICE 2G6 format. This is the default.

- **NUMDGT=INTEGER\_VAL**

Numerical values written to the ASCII output file are forced to have `NUMDGT` digits.

Applies only to data generated by the `.OP` command, to data corresponding to DC values or to data corresponding to `.PRINT` statements and `.EXTRACT` or `.MEAS` values.

- **ENGNOT**

Numerical values are printed out with scaling factors (e.g. U, MEG etc.). Applies only to data generated by the `.OP` command, to data corresponding to DC values or to data corresponding to `.PRINT` statements and `.EXTRACT` or `.MEAS` values.

- **PRINTLG=VAL**

This option is used for printout purposes only: if number of items which appear in a `.PRINT` card exceeds `PRINTLG`, values are printed out in different tables. This is to prevent lines being too long in the ASCII output file, making reading of the file uncomfortable for the user. Default is 8.

- **NOTRCLIB**

This option removes the printout of `.MODEL` and `.SUBCKT` included from library files via `.LIB` or `.ADDLIB` commands.

- **VERBOSE**

This option forces Eldo to display more detailed reporting with some information messages in the standard output terminal. Eldo will print hints about syntax which is valid but ignored if the appropriate analysis is not found in the netlist. For example:

Warning 10001: No optimization command has been found in the netlist.

As a consequence:

- 1) .option OPSELD0\_NETLIST is ignored
- 2) .PARAMOPT are interpreted like .PARAM using the initial value, or ignored if no initial value has been specified.

---

```

3) GOAL=MINIMIZE is ignored on measurement FOO
4) GOAL is ignored on measurement FOO2
5) UBOUND is ignored on measurement FOO3
6) GOAL=MAXIMIZE is ignored on measurement FOO4

```

Warning 10002: COMMAND .MC has not been found in the netlist.

As a consequence:

1) .option DISPLAY\_CARLO is ignored

## Simulation Output Control Options

- **ACOUT=VAL**

This option controls how expressions **VX(a,b)** and **IX(a,b)** are computed in AC analysis: X in this case stands for **DB**, **M**, **P** or **GD**. Only two values are allowed, 0 or 1. In -compat mode, **ACOUT** defaults to 1, else it defaults to 0.

0

**VX(a,b)** or **IX(a,b)** are computed from the complex value **v(a)-v(b)** or from **i(a)-i(b)**.

1

**VX(a,b)** or **IX(a,b)** are computed from the complex value **VX(a,0)-VX(b,0)** or from **IX(a,0)-IX(b,0)**.

- **ALTER\_SUFFIX**

Change the naming convention for swept waves used in **.ALTER** statements to be switched between: xxx and xxx\_alter:XX.

- **NOASCII**

This option forces Eldo to suppress the printing of the output curves in the ASCII output file. This file is created anyway, but its size may be much smaller since these results are not printed in it. This option is equivalent to the **-noascii** flag upon invocation of Eldo.

- **ASCII**

Specifying **ASCII** forces Eldo to store the results including the printing of the output curves in the ASCII output file. This is the default behavior when not running Eldo in ST mode. When running Eldo in the ST (**-stver**) mode, the default behavior is *not* to store the results in the ASCII output file (i.e. **NOASCII** is the default).

- **BLK\_SIZE=VAL**

Sets the total size of the buffer when using the Xelga 4.7 cou file format. With this cou file format, data is buffered. The format is k values for X-axis, followed by k values for wave 1, then k values for wave 2 and so on. The value of 'k' (i.e. the size of the buffer) is computed by Eldo according to the number of waves. Specified in Octets. Default is **BLK\_SIZE=1e6** (i.e. 1Meg).

- **CAPTAB**

Prints out in the ASCII output file the capacitance values which are dropped in the AC matrix. The term  $C_{ij}$  corresponds to the derivative of the charge on node 'i' with respect to the voltage on node 'j'. **CAPTAB** must be used in conjunction with a **.AC**, **.OP** or **.TRAN** command. Information will not be output when there is only a **.DC** command. The output in the ASCII file has the following format:

```
Node <i>
<j1> C = <val1>
<j2> C = <val2>
...
CFIX = <valfix> CVAR = <valvar> CTOT = <valtot>
```

where:

`val1` is the derivative of the charge on node 'i' with respect to the voltage on node 'j1', etc.

`valfix` is the total capacitance connected to node 'i' which is not dependent on other node voltages.

`valvar` is the total capacitance connected to node 'i' which is dependent on other node voltages.

`valtot` is the sum of `valfix` and `valvar`.

- **DISPLAY\_CARLO**

Display all the updated values for all the Monte Carlo and Worst Case runs.

- **DUMP\_MCINFO**

Dump a summary of the **.EXTRACT** distributions during Monte Carlo analysis to the *.aex* file and the file declared by the **.EXTRACT** parameter **FILE=FNAME**, if they are specified. The information is also still printed to the *.chi* file.

- **EXTCGS**

The default unit for **.EXTRACT** current values ( $I(\text{object\_name})$ ) is Amperes (since Eldo v6.5\_2). Set this option to specify mA as the default unit for backward compatibility.

- **EXTFILE**

Forces the creation of a *.ext* file. Extraction results are by default saved inside the EXT folder in the main *.wdb* file (since Eldo v6.6\_1). The extraction results file (*.ext*) is not automatically created.

- **EXTMKS**

When set, the **.EXTRACT** values which are relevant to the transient-extract language are expressed in *mksA* (meter-kilogram-second-Ampere) units instead of *cgs* (centimeter-gram-second) units, i.e. if **EXTMKS** is set, a result such as:

`TPDUU(V(1),V(1),VTH=0.5) = 1.0001E+06 nS`

would be returned as:

---

```
TPDUU(V(1),V(1),VTH=0.5) = 1.0001E-03
```

- **HISTLIM**

In case History has to be saved on FAS, **HISTLIM** would save data only if there is a significant change (based on **RELTOL**) between the last saved value and the current value. This is in order to conserve memory when running very long transient simulations.

- **INPUT**

Forces a list of the user-specified inputs present in the netlist to be written to the binary output (.wdt or .cou) file for checking purposes.

- **JWDB\_EVENT**

Specifies the time interval in seconds for updating marching waveforms displayed in the EZwave viewer. Default value is 10 s.

- **JWDB\_PERCENT**

Specifies the simulation percentage interval for updating marching waveforms displayed in the EZwave viewer. Default value is 5%.

- **KEEP\_HMPFILE**

By default, when **.NOISETRAN ...MRUN** is specified in a netlist, the **.hmp** file will not be generated under the JWDB output format. This default can be overridden with this option.

- **LCAPOP**

Produces a table of the capacitances attached to each net in the circuit in the **.chi** file. Both floating and grounded capacitances are listed.

- **LIMPROBE=VAL**

Sets the maximum number of nodes that may be monitored via the **.PROBE** command. Default value is 1000.

- **LIST**

Causes a listing of the elements contained in a circuit netlist, together with the names of their pins, to be printed to the ASCII output (**.chi**) file. Grounded capacitors are not considered as elements.

- **MAX\_CHECKBUS=ALL|value**

This option allows the user to control the number of checkbus errors printed to the **.chi** file. Default value is 20. Specifying 0 means no errors are printed. ALL specifies all errors are printed (this can also be set by specifying -1). Also if the **.CHECKBUS** command has specified the **LOCK** parameter then all check points are printed whether they have passed or failed.

- **MAX\_DSPF\_PLOT=VAL**

Sets the maximum number of DSPF interface nodes plotted. Default value is 100. This can be useful when a plot (or probe) in DC or TRAN is asked on a node which has been replaced

by a DSPF parasitics network (**.DSPF\_INCLUDE** command), then the plot of this node will be replaced by a plot of all the DSPF network interface nodes. Inside EZwave, the generated plots will be grouped inside an analog bus.

- **NEWACCT**

Specifies that simulation display statistics are displayed in a better format. By default, this option is disabled. When this option is made active, the subsequent output will be generated with one item and one number allowed per line. This simplifies the writing of post-processors. Example:

Number of Input signals	3
Number of resistors	0
Number of floating capacitors	0
Number of inductors	0
Number of voltage sources	3
Number of current sources	0
Number of dependent sources	0
Number of diodes	0
Number of BJT	0
Number of JFET	0
Number of MOS	19
Number of SWITCHES	0
Number of Transmission lines	0
Total number of elements	22
Number of equations	13
Number of non-zero terms	73
Percent Zeros	5.680e+01
Number Newton iterations	178
Average number Newton iterations	2.000e+00
Number of accepted time steps	57
Number of rejected time steps	4
due to LTE	4
due to newton to Newton	0
Evaluation of active devices	11864

- **NOBOUND\_PHASE**

Option to avoid the fold-down for the phase. When this option is set, the phase is not displayed in modulo 360 degrees. This option only applies to outputs of type XP() (e.g. vp(), ip() or wp()). It does not apply to outputs of another type, or to the internal representation of waveforms of type phase. Therefore, to avoid fold-down of phase for a computed waveform, it should be written in the form:

```
.plot ac wp(computed_waveform)
```

- **NODCINFOTAB**

Disables the printout of the DC node information in the ASCII output file. This can be useful to save disk space if the circuit is very large and the DC solution is not of interest.

- **NODE**

Causes printing of a node table to the log (.chi) file. It contains a list of all elements in a circuit netlist, together with the total grounded capacitance value for each node.

- **NOEXTRACTCOMPLEX**

The **.EXTRACT** command by default provides complex results information instead of only real values with the `yval` function. This can be disabled by setting this option. Complex results information is used by default instead of real values with the **.EXTRACT** `yval(v(out),10k)` would deliver the result:

```
* EX1 = 10.011 , -1.8
```

The general form is: `real_part, imaginary_part`.

- **NOMOD**

Suppresses the print-out of the model parameters to the ASCII output (*.chi*) file.

- **NOOP**

Suppresses the printing of operating point (OP) table information in the ASCII output (*.chi*) file when a **.AC** analysis is specified.

- **NOPAGE**

This is synonymous to the option **LIST**.

- **NOSIZECHK**

Eldo checks the size of the *.chi* file, and does not write to it if its size exceeds the total disk space available (Unix Only). The **NOSIZECHK** option disables this check.

- **NOTRC**

Suppresses the rewriting of the circuit description file in the ASCII output (*.chi*) file, when inserted immediately after the title line of a circuit description file.

- **NOWAVECOMPLEX**

Disables the generation of complex waves. By default, complex waveform information is only written to the saved windows file (*.swd*) with JWDB format output. This option forces Eldo to write complex waveform information to the waveform database (*.wdw*) as well as the saved windows file. This can be used to restore the functionality of Eldo versions prior to v6.3\_2.1. This can also be specified using the `-jwdb_nocomplex` flag when invoking Eldo.

- **NOXTABNOISE**

X instances that do not contain any other X instances are considered as Eldo primitives, and are taken into account in the noise table that is displayed in the ASCII output files. This is particularly useful in the case of designs where devices are modeled by subcircuit rather than by Eldo primitive. Such X instances will then appear in the ASCII table as if Eldo primitives had been used. This can be disabled with the **NOXTABNOISE** option.

- **OPTYP=VAL**

Used to change the way Operating Point information related to MOSFETs is displayed in the ASCII output file. Default: `optyp=1`. See “**.OP**” on page 10-187.

- **OUT\_RESOL=VAL**

Save data in binary waveform output file only if the time increment in the output file is at least the value of **OUT\_RESOL**. This option was originally named differently (**COURRESOL** for pre-v6.5\_2 Eldo versions), although the option name referred to *cou* format, it was only historical, and has the same effect whether Eldo writes *.cou* or *.wdb* output.

- **OUT\_SMP=VAL**

This can be used as an alternative to the **FREQSMP/TIMESMP** with **OUT\_STEP** options. With **OUT\_SMP** Eldo expects a frequency value to be specified whereas a time interval is expected with **OUT\_STEP**. This option was originally named differently (**COUSMP** for pre-v6.5\_2 Eldo versions), although the option name referred to *cou* format, it was only historical, and has the same effect whether Eldo writes *.cou* or *.wdb* output.

- **OUT\_STEP**

Used in conjunction with option **FREQSMP=VAL**. Forces Eldo to save into the binary waveform output file only the timesteps in 1/val intervals as defined by **FREQSMP**. All intermediate timesteps, although calculated, will not be saved. The results in this output file (using **.OPTION FREQSMP=val OUT\_STEP** in the netlist) are close to those using option **FREQSMP=val** in the netlist, but not to the output file without option **FREQSMP** in the netlist. This option was originally named differently (**COUSTEP** for pre-v6.5\_2 Eldo versions), although the option name referred to *cou* format, it was only historical, and has the same effect whether Eldo writes *.cou* or *.wdb* output.

---

**Note**



It is possible to assign expression parameters to **OUT\_STEP**. For example:

```
.PARAM p1 = 1k
.OPTION OUT_STEP = p1
```

---

- **POST**

This is equivalent to **.PROBE V**, but in addition, current values of grounded voltage sources are also displayed.

- **PRINT\_ACOP=0 | NO | 1 | YES**

This option will limit the amount of operating point information produced in the output file, when a **.AC** analysis is specified.

- **SIMUDIV=VAL**

Specifies how many times status information will be printed out during simulation. For example, **SIMUDIV=10** causes a print out after each 10<sup>th</sup> of the simulation, e.g. each 10<sup>th</sup> of **tstop**. Status information recorded includes the elapsed CPU time, estimated total CPU time, percentage of simulation done, plus any data selected using the **STAT** option. Can also be used independently of **STAT**. Cannot be used in conjunction with **TIMEDIV**. Default is 20. Set **SIMUDIV** to 0 to remove the print out.

- **STAT=VAL**

Used to set debug levels for simulation. Four values are allowed: 0, 1, 2, 3.

0

Default. Specifies no statistics are to be recorded.

1

Reports **NEWTON/OSR** partitioning in the *.chi* file and to the terminal. Writes a trace of parameter and object updates in *.chi* log file. Simulation time data is written to the *.chi* file when **SIMUDIV** or **TIMEDIV** are used.

2

As level 1 plus: a node list is written to the *.chi* file.

3

As level 2 plus: parameter and object update tracing is also listed to the terminal (`stdout`). The number of rejected/accepted timesteps is reported when **SIMUDIV** or **TIMEDIV** are used.

- **NOSTATP**

Disables the print out of parameter values when **STAT** option is set to a non-zero value (**STAT** is then used only for partitioning information).

- **TIMEDIV=VAL**

Works in conjunction with **STAT**. This option is equivalent to **SIMUDIV** except **VAL** now specifies a CPU time interval. Status information will be printed out after every **VAL** minutes of elapsed CPU time. This may significantly slow down simulation. Cannot be used in conjunction with **SIMUDIV**.

- **TEMPCOUK**

By default, all the temperatures are plotted in Celsius in the *.cou* file. Setting this option causes the temperatures to be plotted in Kelvin.

- **VBCSAT=VAL**

Default is 0. Sets the ‘region of work’ that is displayed on the **OP** table for BJT as follows:

```
if(VBC > VBCSAT)
    if(VBE > 0) "SATURATION"
    else "INVERSE"
else
    if (VBE > 0 ) "ON"
    else "OFF"
```

- **VXPROBE**

Adding this option with **.PROBE V** forces Eldo to dump ground and subcircuit node voltages to the output file.

## Optimizer Output Control Options

- **OPSELDO\_ABSTRACT**

Generates a summary table of simulations containing parameter and extract values for each run.

- **OPSELDO\_DETAIL=[ NONE | ALL ]**

If this option is set to `NONE`, only the last run will be stored in generated files (`.wdb`, `.cou`, `.aex`) and no other simulation information will be displayed. When set to `ALL`, simulation information for all runs will be stored. Default is `NONE`. (This option was named `OPSELDO_NODETAIL` in pre-v6.6 Eldo versions and was not enabled.)

- **OPSELDO\_DISPLAY\_GOALFITTING**

Displays the intermediate goals when splitting DC measurements during optimization. Each `.DATA` point is optimized as an independent goal.

- **OPSELDO\_FORCE\_GOALFITTING**

Forces Eldo to split DC fitting extracts in independent goals even if data points are correctly ordered (as opposed to `OPSELDO_NOGOALFITTING`). It can also split AC fitting extracts.

Disables the splitting of DC measurements during optimization. By default, each `.DATA` point is optimized as an independent goal.

- **OPSELDO\_NETLIST**

Generates a netlist modified from the original input file, which contains the optimized parameter values but also every parameter set under `#ifdef` statements.

- **OPSELDO\_NOGOALFITTING**

Disables the splitting of DC measurements during optimization. By default, each `.DATA` point is optimized as an independent goal only if they are not correctly ordered.

- **OPSELDO\_OUTER**

Allows a reverse behavior of optimization and sweep simulations (`.TEMP`, `.DATA` or `.STEP`). A full optimization will be performed for each set of sweep parameters.

- **OPSELDO\_OUTPUT**

This option controls results of optimization (Opsim or bisection method).

**OPSELDO\_OUTPUT=0**

Print results in a simplified format: parameter name, goal, optimized value

**OPSELDO\_OUTPUT=1**

Print results using simplified format (as for `OPSELDO_OUTPUT=0`) and generate a `.ops0` file using the same format.

**OPSELDO\_OUTPUT=2**

Print results in a detailed format: parameter name, minimum value, maximum value, weight, goal, optimized value

---

**OPSELDO\_OUTPUT=3**

Print results using detailed format (as for **OPSELDO\_OUTPUT=2**) and generate a *.ops0* file using the same format.

- **RESET\_MULTIPLE\_RUN**

This option is used internally by Eldo to disable multiple-run analyses (**.STEP**, **.TEMP**). It is only used in the files generated by Eldo to replay some optimization runs. Parameters are assigned values and multiple-run analyses are disabled.

**Caution**



The behavior cannot be predicted if this option is specified by the user instead of by Eldo.

---

## File Generation Options

- **AEX**

Forces Eldo to dump the results of the **.EXTRACT** command into a file *filename.aex*. The values are also still written into the *.chi* file. This is the default.

- **NOAEX**

Suppresses the **AEX** option and hence cancels the creation of *filename.aex*. The values are only written into the *.chi* file. See option **AEX**.

- **ALIGNEXT**

Specifies that Eldo will write tabulated **.EXTRACT** results in the *.aex* file. By default the results are not tabulated (aligned). Remember, the *.aex* file is only created if the option **AEX** is activated.

- **ASCII=VAL**

If **VAL** is set to 0, the ASCII output file generation is terminated. If **VAL** is set to 1, the ASCII output file is created. The default value is 1. This option does not have the same meaning as the option **ASCII** described on [page 11-44](#).

- **COU**

Used to generate output in binary Cou format. This can also be specified using the invoke command **-gwl cou**.

- **CSDF**

Used to generate output in CSDF format. CSDF is the Common Simulation Data Format. This can also be specified using the invoke command **-gwl csdf**.

- **INFOMOD=file\_mod**

Eldo will generate a file **file\_mod** that contains all of the **.MODEL** commands with their parameter values evaluated.

- **INFODEV=***file\_dev*

Eldo will generate a file *file\_dev* that contains all of the R/L/C/M/B/D/J device names with their corresponding parameter values evaluated.

- **ISDB**

Used to generate output in ISDB format for SimWave. This can also be specified using the invoke command `-gwl isdb`.

- **JWDB**

Used to generate output in JWDB format (extension *.wdb*). This can also be specified using the invoke command `-gwl jwdb`.

- **NOJWDB**

Specifying this option will disable the generation of the output in JWDB format. This can also be specified using the `-nojwdb` flag when invoking Eldo.

- **NOCOU**

Suppresses the generation of the binary monitored results (*.cou*) file.

- **NOIICXNAME**

Used to restore the full name display in the *.iic* file. (This was the default behavior in releases prior to v6.5.) By default, the saved simulation run file (*.iic*) format compresses the hierarchical names, in order to reduce the size of the file. With this option, the file produced can become very large when simulating large circuits with lots of hierarchy because the node names are stored flat.

- **NOCKRSTSAVE**

By default, if the same file name appears in a **.SAVE** command and in a **.RESTART** command, and if conditions for reading and writing the file are not the same, then Eldo issues an error. This is in order to prevent Eldo unexpectedly overwriting restart files which take a long time to create. **NOCKRSTSAVE** disables that check.

- **NOPROBEOP**

When running Eldo in the ST mode, this will remove creation of the *probeop* file.

- **OUT\_REDUCE**

Used to reduce the accuracy of output waveforms by performing a slope based comparison of output points. This is useful to reduce the number of points in case you have both constant (or slow varying) waves and varying waves in the same file. The comparison works by searching for groups of three points which are aligned. The middle point is ignored if the following condition is met:

$$|current\_slope - previous\_slope| < OUT\_ABSTOL + OUT\_RELTOL \times |current\_slope|$$

Where **OUT\_ABSTOL** and **OUT\_RELTOl** are options which can also invoke the waveform reduction comparison.

As soon as this option is set, output waveforms are stored asynchronously i.e. each wave will have its own X values. Thus the size of the .wdb waveform database file may decrease or increase depending on the variations of waveforms. This option is useful if output waveforms contain constant signals, pulses, or slow varying signals. For example, power supplies which are constant require only two points at tstart and tstop.

- **OUT\_ABSTOL**=value

Used to reduce the accuracy of output waveforms and set the absolute tolerance value. Default is 1.0E-5. See [OUT\\_REDUCE](#) for details.

- **OUT\_RELTOL**=value

Used to reduce the accuracy of output waveforms and set the relative tolerance value. Default is 1.0E-6. See [OUT\\_REDUCE](#) for details.

### **Caution**



Simulation time could increase when specifying options **OUT\_REDUCE**, **OUT\_ABSTOL** or **OUT\_RELTOL**. They are not performance options.

- **PROBEOP**

This command will be active only if **.OP** is explicitly required. By default, **.AC** will not create the `probeop` file.

- **PROBEOP2**

Eldo will generate a `<circut>.op<x>`, where `x` is the index of the run, in which DCOP information is dumped. This file is an ASCII file which is read by the DA-IC environment. The content of this file is similar to that created when option **PROBEOP** is used, but some additional information is dumped: temperature, node information, and current out of X leaf-instances (leaf-cells are instances which do not call any other X instances).

- **PROBE**

When `-compat` is set, then the Eldo **.PROBE** command is emulated, i.e. all node voltages are dumped in the cou file. In order to have only those items specified in **.PLOT/.PROBE** to be dumped in the cou file, add option **PROBE**.

- **PSF**

Used to generate output in Cadence format. This can also be specified using the invoke command `-gwl psf`.

- **PSFASCII**

Used to generate PSF format data in ASCII. This can also be specified using the invoke command `-gwl psfascii`.

- **SAVETIME=VAL**

Creates a `.sav` file which saves the context of a simulation every `VAL` hour of CPU time used in a transient analysis. If the system crashes during a run, the simulation may be restarted from the last saving time.

## Mathematical Algorithm Options



For more details on these options, please refer to the [Speed and Accuracy](#) and [Integral Equation Method \(IEM\)](#) chapters.

---

- **TRAP**

Resets the integration method to “Trapezoidal” from “Backward Euler.” Useful in interactive mode only. Related parameters: **SMOOTH**.

- **SMOOTH**

When the “Trapezoidal” integration method is used, oscillation may occur on some output curves. This **SMOOTH** option forces Eldo to display at the mid-point of the computed time intervals, this operation acts as a filter and oscillations due to **TRAP** are removed.

- **BE**

Forces “Backward Euler” integration to be used. “Trapezoidal” integration is used by default.

- **GEAR**

Forces “Gear” integration to be used. Trapezoidal integration is used by default. Related parameters: **MAXORD**, **CHGTOL**, **NGTOL**, **FLUXTOL**, **RELTRUNC**. Take care with the gear algorithms as you may potentially miss some oscillations in circuits.

- **GNODE**

Assists DC convergence for **.RAMP DC** and option **GRAMP**. **GNODE** is the conductance which is applied between each node and ground. The value input by the user is the conductance value to be used as the first **RAMP** point. Eldo will reduce the value down to 0 for the final **RAMP** point, when conducting a **RAMP** analysis.

- **METHOD=GEAR**

See the option **GEAR** above (used only for compatibility with SPICE).

- **MAXORD=VAL**

Used in conjunction with option **GEAR**. Specifies the maximum order of the Gear integration method and must be in the range 1 to 6. The value 1 is equivalent to the **BE** option. Default value is 2.

- **NEWTON**

Forces Eldo to use only one Newton block on the design. This is the default algorithm.

- **NODEFNEWTON**

Forces Eldo to use OSR by default, instead of Newton. See option **NEWTON**.

- **IEM**

Forces Eldo to use only one IEM block on the design.

- **BLOCKS=NEWTON**

Forces Eldo to use OSR on the whole circuit and Newton-Raphson algorithm on tightly coupled sub-blocks. If the whole circuit was tightly coupled, then Eldo will automatically switch to **NEWTON**, after issuing a message.

- **BLOCKS=IEM**

Forces Eldo to use OSR on the whole circuit and IEM algorithm on tightly coupled sub-blocks. If the whole circuit was tightly coupled, then Eldo will automatically switch to **IEM**, after issuing a message.

- **ANALOG**

Forces Eldo to create only one Newton block for the whole circuit and adjusts **EPS** to a maximum value of  $1.0 \times 10^{-4}$  in order to achieve higher accuracy results.

- **DIGITAL**

When the OSR algorithm is selected, this forces nodes which are not in **ANALOG** blocks and which are connected to MOSFET, grounded capacitors, current sources or floating capacitors of less than **CAPANW** to be solved by OSR, all other nodes being treated by Newton-Raphson. This option is useful for large MOS circuits with extracted parasitic coupling capacitances.

- **OSR**

Forces Eldo to use the OSR (One Step Relaxation) iteration technique whenever possible, i.e. for MOS loosely coupled circuits. Related parameters: **DIGITAL**, **CAPANW**.

- **NORMOS**

This option allows the removal of access resistances in analog blocks if low precision simulation is required. For circuit blocks simulated using OSR at a relatively low precision level, MOS access resistances are not treated as separate elements. Their effect is taken into account during MOS model evaluation in an approximate manner. It is sometimes possible for analog blocks simulated using either Newton Raphson or IEM to have these devices (i.e. MOS transistor with an approximate access resistance effect). This happens for example when the circuit is dynamically re partitioned during simulation for convergence problems. Analog blocks (solved by Newton or IEM) should maintain a minimum level of precision hence access resistors should be created as new elements.

Hence, specifying **NORMOS** selects the mechanism that allows dynamic creation/removal of these devices depending on the algorithm used: removal for OSR and creation for both Newton and IEM.

- **PSTRAN**

Forces Eldo to use the PSTRAN (PSseudo TRANsient) algorithm prior to any other convergence aid. This algorithm is one of the DC convergence algorithms that are automatically used by Eldo when the standard DC algorithm fails to converge.

- **DPTRAN**

Forces Eldo to use the DPTRAN algorithm prior to any other convergence aid. This algorithm is one of the DC convergence algorithms that are automatically used by Eldo when the standard DC algorithm fails to converge.

- **NODCPART**

Do not insist on DC partitioning in DC analysis. In case of convergence difficulties, immediately go to the convergence aid algorithm.

- **DCPART=VAL**

Here, **VAL** is the number of time partitions that will occur, the default is 5. **DCPART=0** is equivalent to option **NODCPART**.

- **CSHUNT**

Add a capacitance between each node and ground. Default is 0. This can be used to eliminate oscillation problems caused by numerical noise or high frequency oscillation.

- **GSHUNT**

Add a conductance between each node and ground. Default is 0. This can be used to eliminate oscillation problems caused by numerical noise or high frequency oscillation.

## Mixed-Mode Options

- **MIXEDSTEP=FREE | LOCKED**

Used for controlling time-step synchronization between Eldo and a digital solver (ADVANCE MS or Verilog-XL). This option can take two values: **FREE** or **LOCKED**. Default is **FREE**.

When **MIXEDSTEP** is **FREE**, simulation is usually faster, but may fail on some rare cases with a message inviting the user to re-run the simulation with **MIXEDSTEP=LOCKED**.

When **MIXEDSTEP** is **LOCKED**, the analog kernel will stop on each digital event even if these digital events have no immediate impact on the analog side. This mode of simulation is very robust, but is usually slower except in cases where most of the digital events have an immediate impact on the analog side. When partitioning in ADMS Mach GUI mode without the option **MACH** set in the command file, the user will have to manually set the option **MIXEDSTEP=LOCKED** to send a block of the circuit to Mach.

- **D2DMVL9BIT**

Enables direct connection between HDL-A ports of type BIT with mixed-mode nodes connected to Eldo via convertors of type MVL9 (or `std_logic`).



Further information can be found [page 10-4](#) for **.A2D** or [page 10-44](#) for **.D2A**.

---

**.OPTION**

- **DEF<sub>A</sub>2D**=model\_name

Eldo will automatically insert an implicit A2D convertor when needed. Without this option specified, it is normally necessary to specify an explicit A2D between ANALOG nodes and HDL-A ports.



Further information can be found [page 10-10](#).

- **DEF<sub>D</sub>2A**=model\_name

Eldo will automatically insert an implicit D2A convertor when needed. Without this option specified, it is normally necessary to specify an explicit D2A between ANALOG nodes and HDL-A ports.



Further information can be found [page 10-49](#).

- **DEFCONVMSG**

Forces Eldo to print out in ASCII output files the convertor information related to the **.A2D** and **.D2A** which are added when options **DEF<sub>A</sub>2D** and **DEF<sub>D</sub>2A** are specified.

- **DYND2ALOG**

Enables the threshold voltages VTHI and VTLO to be calculated dynamically for digital macromodels using actual values of the high and low bias. The bias values, VHI and VLO are computed using two extra pins defined by the user in the digital model instance. The value specified on the first pin provides the high voltage digital output value (VHI) and the value specified on the second pin provides the low voltage digital output value (VLO). The two extra pins are placed after the list of regular pins and before the model name in the model instance syntax, see “[Digital Macromodels](#)” on page 7-1 for more information. This option must be placed at the top of the design, just after the title line.

The *active* VTHI and VTLO threshold values are also computed dynamically using the following equations:

$$\begin{aligned} \text{VTHI\_ACTIVE} &= \text{VLO} + \text{VTHI} * \text{DV} \\ \text{VTLO\_ACTIVE} &= \text{VLO} + \text{VTLO} * \text{DV} \end{aligned}$$

VTHI and VTLO are the values specified in the **.MODEL** command defining the digital macromodel or in the macromodel instance, VLO is the low output voltage value given on the second extra pin defined by the user, and DV is the voltage difference given by VHI - VLO.

- **DYND2ALOG2**

Specifying this option activates the already existing option **DYND2ALOG** for dynamically calculating threshold values of digital macromodels, and also changes the way the thresholds are detected.

For the detection of the rising edge, and in **DYND2ALOG2** mode, Eldo checks for  $V > VTH$  and  $V_{LAST} \leq VTH\_LAST$  where V and VTH are the voltage and the threshold

value at the current time point, and VLAST and VTH\_LAST the same entities but at the previous time step.

If **DYND2ALOG** is set, Eldo will check for: for  $V > VTH$  and  $VLAST \leq VTH$ , which is also the formula used in case none of the options **DYND2ALOG2** and **DYND2ALOG** are set.

This change of the expression has a major impact in the threshold detection for the case the pin which is used to carry the voltage used as the threshold value is also an input pin: the output of the small circuit will change depending on the option set:

## Other Options

- **CTEPREC**

With this option, fundamental physical constants are as follows (SI units):

Free space permittivity (epso) =  $8.854187817 \times 10^{-12}$ ;  
Boltzmann constant (boltz) =  $1.38065812 \times 10^{-23}$ ;  
Electron charge (charge) =  $1.6021773349 \times 10^{-19}$ ;  
Twice pi (twopi) =  $4.0 \times \text{asin}(1.0)$ ;

By default, Spice2G6 values are used:

epso =  $8.854214871 \times 10^{-12}$ ;  
boltz =  $1.3806226 \times 10^{-23}$ ;  
charge =  $1.6021918 \times 10^{-19}$ ;  
twopi = 6.283185308;

- **DCLOG=VAL**

When Eldo is used with a circuit containing digital gates, a DC analysis is run at first, then the output of the digital gates is updated, and then another DC analysis is run, i.e. there are relaxations between DIGITAL and Eldo. There are **DCLOG** relaxations. Default is 1.

- **EPSO=VAL**

Allows overwriting of the **EPSO** value.

- **MAXNODEORD**

If number of nodes in the netlist is less than **MAXNODEORD**, then in the **OP** node table, node names are listed by alphabetic order. If the number of nodes exceeds **MAXNODEORD**, node names are listed in the order that Eldo has chosen. Default is 300.

- **NODUPINSTERR**

When specified in conjunction with the **stver** flag/option, will change **ERROR 838** into **WARNING 204**. This option is ignored if not specified with the **stver** flag/option.

- **NOELDOSWITCH**

When specified with the **-compat** flag/option, it informs Eldo that devices beginning with an **S** are not switches (Eldo default) but S-parameter block instantiations. This option is ignored if not specified with the **compat** flag/option.

- **NOINIT**

Prior to an AC or Transient run with UIC active (i.e. no DC analysis to be performed), Eldo performs a logical initialization, unless **NOINIT** is defined.

The concept behind this ‘logical initialization’ is as follows: consider a PMOS element, with 0 V imposed on the gate (via an independent voltage source for instance), and the Source connected to power supply VDD. Then, if Drain has not been initialized, Eldo would initialize its value to VDD, and the value will propagate as far as it can. This algorithm would correctly initialize CMOS designs.

- **NOFNSIEM**

The Integral Equation Method (IEM) is used in Transient Analysis to solve FNS functions. Specifying **NOFNSIEM** reverts to the implementation based on state variables.

- **SEARCH=path1[{:path2}]**

When Eldo does not find a subcircuit called <name> or a model definition called <name>, it will look for the file path1/<name>.inc. The name is converted into lowercase before searching, and, if more than one **SEARCH** option is given, Eldo will search all the directories in the path specified by the user in the order that they appear inside the input file.

If more than one path is specified it will search all of the paths. Each path name must be separated by a colon. There is no limit to the number of paths that can be specified.

---

**Note**

This has the same effect as using `-searchpath` in the Eldo instantiation. See [“Running Eldo from the Command Line”](#) on page 2-1.

---

- **VAMAXEXP=val**

In Verilog-A, the argument of any exponential is limited to **VAMAXEXP**. Default is 80.0. This is to limit exp(value) when value is too large.

- **ZCHAR=VAL**

Characteristic impedance of the circuit. Default is 50 Ohms.

# Chapter 12

## Compatibility Options

---

### Introduction

Compatibility can be obtained by using the `-compat` flag at runtime or by using `.OPTION COMPAT` in the netlist.

This chapter shows the different effects that this command has. It has been separated into the following sections:

- [Devices](#)
- [Commands](#)
- [Options](#)
- [Netlist](#)
- [Arithmetic Functions and Operators](#)

Additional flexibility is offered. Instead of using `-compat`, the flags `-compmod` and `-compnet` can be set with the following effects:

- `-compmod` Triggers only the automatic conversion of models (can alternatively be set with `.OPTION COMPMOD`); see [Devices](#).
- `-compnet` Causes the netlist to be interpreted as compatible format, but the models themselves are treated as Eldo Spice models. This means it is assumed that models are already Eldo models (can alternatively be set with `.OPTION COMPNET`).

---

#### Note



Flag `-compat` (or `.OPTION COMPAT`) is equivalent to setting both `-compmod` and `-compnet` flags (or `.OPTION COMPMOD` and `.OPTION COMPNET`).

---

## Devices

### Eldo Levels

The following Eldo Levels are set:

**Table 12-1. MOS Levels with -compat**

LEVEL	Model Name (Eldo Level)
2	Eldo2 (Eldo LEVEL 12)
3	Eldo3 (Eldo LEVEL 13)
6	Modified Lattin-Jenkins Grove Model (Eldo LEVEL 16)
8	Enhanced Berkeley LEVEL 2 (Eldo LEVEL 17)
13	Berkeley BSIM1 (Eldo LEVEL 8)
39	Berkeley BSIM2 (Eldo LEVEL 11)
49	Berkeley BSIM3 v3.0 & BSIM3 v3.1 (Eldo LEVEL 53)
50	Philips MOS Model 9 (Eldo LEVEL 59)
54	Berkeley BSIM4.0.0 (Eldo LEVEL 60)
57	Berkeley BSIM3SOIv2 PD (Eldo LEVEL 56, SOIMOD=1)
59	Berkeley BSIM3SOIv2 FD (Eldo LEVEL 56, SOIMOD=3)



For more information see [page 4-129](#).

---

**Table 12-2. BJT Models with -compat**

LEVEL	Model Name (Eldo Level)
2	Improved Berkeley Model (Eldo LEVEL 5)
3	STMicroelectronics LEVEL 1 (Eldo LEVEL 2)
4	VBIC v1.2 (Eldo LEVEL 8)
	(VERSION=1.15) VBIC v1.1.5 (Eldo LEVEL 8)
6	Philips Mextram 503.2 Model (Eldo LEVEL 4)
8	HICUM Model (Eldo LEVEL 9)



For more information see [page 4-113](#).

**Table 12-3. Diode Models with -compat**

LEVEL	Model Name (Eldo Level)
2	Fowler-Nordheim (Eldo LEVEL 3)
3	Berkeley Level 1 (Eldo LEVEL 1), by default SCALEV is set to 3
4	JUNCAP Diode Model (Eldo LEVEL 8), by default DIOLEV is set to 9



For more information see [page 4-106](#).

**Table 12-4. Resistor Level with -compat**

LEVEL	Model Name (Eldo Level)
1	RC Wire (Eldo LEVEL 3)



For more information see [page 4-29](#).

## PNP and NPN devices

The `-compat` flag affects the default type for PNP and NPN devices. `SUBS` defines the type of BJT. By default, NPN and PNP are vertical:

If `SUBS` is -1, BJT is lateral.

If `SUBS` is 1, BJT is vertical.

If `-compat` is set, NPN are vertical, PNP are lateral.

For more information see [page 4-111](#).

## Group Delay

`VT` is equivalent to `VGD`, and `IT` is equivalent to `IGD` (Group Delay on voltage or current). For more information see [page 10-225](#).

## Node GROUND/GND

Node `GROUND` or `GND` are assumed to be the global node 0.

## Value “x”

“x” is equivalent to “meg”, i.e. when the `-compat` flag is set:

```
R1 1 0 1x
```

is equivalent to:

```
R1 1 0 1meg
```

Otherwise, `1x` corresponds to 1.0.

## Commands

- **.ALTER**

This command is cumulative. For example:

```
r1 1 0 1
r2 2 0 1
.alter 1
r1 1 0 2
.alter 2
r2 2 0 2
```

The second “alter” simulation will be done with both `r1` set to 2 (inheritance from later number 1), and `r2` set to 2. For more information see “[.ALTER](#)” on page 10-22.

- **.DC**

When only a DC analysis is specified in conjunction with the `-compat` flag, by default the initial transient value will be used. For more information see “[.DC](#)” on page 10-54.

- **.LIB**

`.LIB` behaves as `.INCLUDE`. For more information see “[.LIB](#)” on page 10-133.

- **.PLOT/.PRINT**

The sign convention of `Ix` is that the current is positive when it enters the device by pin `x`. However, when `-compat` is set, then the following apply:

For R/L/C/E/F/G/H/I/V/D devices, `I2` returns the same value as `I1`.

For M/B/J devices, `I3` is positive when current leaves the object by pin number 3. For more information see “[.PLOT](#)” on page 10-216 and “[.PRINT](#)” on page 10-254.

- **.PROBE**

The Eldo `.PROBE` card is emulated, i.e. all node voltages are dumped in the cou file. In order to have only those items specified in `.PLOT/.PROBE` to be dumped in the cou file, add `.OPTION PROBE`. For more information see “[.PROBE](#)” on page 10-258.

- **.SUBCKT**

Usage of **GLOBAL** node names in the `.SUBCKT` definition node list. Eldo will check the node names in the subcircuit list definition prior to the global node list. For more information see “[.SUBCKT](#)” on page 10-306.

- **.TEMP**

The model parameter **TREF** can be specified. This is equivalent to the parameter **TNOM** when the **.TEMP** command is used to apply temperature effects. When **TREF** is specified, **TNOM** will be ignored.

- **.TRAN**

The argument of the **.TRAN** card is a list of **INCRn Tn**, and not the standard **TPRINT TSTOP [TSTART [HMAX]]**. Syntax:

```
.TRAN INCR1 T1 [{INCRn Tn}] [TSTART=val] [UIC]
```

For more information see “[.TRAN](#)” on page 10-318.

## Options

- **NOINIT**

Set for cases where **UIC** is set on the **.TRAN** card. For more information see [page 10-319](#).

- **ACM**

The **-compat** flag causes the **ACM** option to be set. For more information see [page 11-30](#).

- **ICDC** and **ICDEV**

These options are assumed to be set. This means that IC specifications given on devices are taken into account for the DC which is performed prior to TRAN. When **-compat** is not set, and neither are the **ICDC** and **ICDEV** options, then IC specifications given on devices are ignored, unless the **UIC** keyword is specified on the **.TRAN** card. For more information see [page 11-28](#).

- **(NO)KWScale**

**NOKWScale** is assumed to be set. Therefore, **SCALE** is not considered as a keyword and can be used as a parameter. For more information see [page 11-38](#).

- **ACOUT=VAL**

**ACOUT** defaults to 1.

1

**Vx(a,b)** or **Ix(a,b)** are computed from the complex value **Vx(a,0)-Vx(b,0)** or from **Ix(a,0)-Ix(b,0)**. For more information see [page 11-44](#).

- **PROBE**

**.PROBE** command is emulated, i.e. all node voltages are dumped in the cou file. In order to have only those items specified in **.PLOT/.PROBE** to be dumped in the cou file, add **.OPTION PROBE**. For more information see [page 11-54](#).

- **LIBINC**

This option is invoked. It specifies that all the libraries (**.LIB**) should be included without filtering the objects (model, card or subcircuit) that are not used in the specific netlist. For more information see [page 11-16](#).

- **BSLASHCONT**

This option is invoked. It allows two backslashes to be used for the continuation of the line.  
Example:

```
.OPTION BSLASHCONT
R1 1 2 \\
3k
R2 1 2 1k
```

**R1** will be set to **3K**. It is possible to disable this option by using **.OPTION NOBSLASHCONT**.

- **PARTHIER='local' | 'global'**

Default is set to **global**. Global “parent” values have precedence. Example:

```
.SUBCKT R1 1 2
R1 1 2 a
.ends
X1 in out R1 a=2
.param a=3
```

When **parhier** is **local**, **X1.R1** will be 2, while when **parhier** is **global**, **X1.R1** is 3. For more information see [page 11-9](#).

- **GENK<=val>**

This option is invoked. It forces Eldo to generate 2nd order mutual inductors. It is used together with the **KLIM** option. For more information see [page 11-33](#).

- **TNOM**

**TNOM** defaults to 25°C instead of 27°C. For more information see [page 3-15](#).

- **DEFPTNOM**

This option is invoked. Allows a parameter to be defined with the name **TNOM**. In such a case, this value will be used inside parameter expressions, instead of the value of **TNOM** set using **.OPTION TNOM=val**.

- **MODWLDOT**

This option is invoked. It is used for binned models. For more information see [page 11-36](#).

- **NOELDOSWITCH**

When specified with the **-compat** flag/option, it informs Eldo that devices beginning with an **S** are not switches (Eldo default) but S-parameter block instantiations.

- **QUOTREL**

The `-compat` flag causes the **QUOTREL** option to be set. For more information see [page 11-11](#).

- **NOELDOLOGIC**

The `-compat` flag causes the **NOELDOLOGIC** option to be set. For more information see [page 11-8](#).

- **DSCGLOB**

The `-compat` flag causes the default for the **DSCGLOB** option to be **GLOB** instead of **X**. Nodes defined with **.GLOBAL** statements have priority over the nodes which appear in subcircuit definitions. For more information see [page 11-25](#).

For example:

```
.GLOBAL QWE
VG QWE 0 DC 3
*.SUBCKT with a global node name in argument list
.SUBCKT B QWE
R1 QWE 0 1K
.ENDS
V2 PIN2 0 DC 2
X2 PIN2 B
```

With the `-compat` flag and option **DSCGLOB** specified, Eldo will connect **X2** and **R1** between global node **QWE** and **0**. In other words, only the voltage source **V2** will be connected to **PIN2**. For more information see [“.GLOBAL” on page 10-121](#).

## Netlist

### Comment character

`$` is used as the comment character in the netlist line instead of `!`.

The comment character is usually considered as such only if the preceding character is a blank space or if the comment is at the beginning of a line. In `-compat` mode with the comment character `$`, then the above rule still applies; additionally, the `$` character is a comment character if:

1. the preceding character is not a white space, *and*
2. the first character of the string is ‘,’, “, a number, or a period (.)

Examples:

- Circumstances where `$` is considered as a comment:
  - `$` is considered a comment because the string begins with the digit 0:

`M1... 1=0.1u$`

- \$ is considered a comment because the preceding character is a blank space:

```
M1... l=0.1u $
```

- \$ is considered a comment because the first character is a number:

```
M1 1$ d g s b nmos w=lu l=3
```

- Circumstances where \$ is *not* considered as a comment:

```
M1 a1$ d g s b...
M2 '1$' d g s b ...
```

## Parameters

Multiple affectation of parameters which can be overwritten sequentially is allowed.

In model instantiations, Eldo will check whether there is a **.model** with the same name as the string after the nodes in a model declaration. If not, it will look for a parameter name.

## Quotes

Double quotes are considered as single quotes. (In standard Eldo mode, double quotes are used to specify a parameter string.) Use option **QUOTSTR** to consider double quotes as a parameter string delimiter.

## Order of analyses

Analyses will be performed in the order which is specified in the netlist. The order in which analyses will be performed can then differ from the original Eldo order as described above, and there can be multiple analyses of the same type to be run sequentially. In addition, **.PRINT/.PLOT/.EXTRACT/.MEAS** commands will be seen only by the preceding command. Example:

```
.tran 1n 10n
.print tran v(1)
.tran 1n 50n
.print tran v(2)
.end
```

With the **-compat** flag set, the first **.tran** command will force Eldo to do a transient simulation between 0 and 10n. Only **v(1)** will be displayed. The second **.tran** command will force Eldo to do a transient simulation between 0 and 50n. Only **v(2)** will be displayed for that simulation.

## Arithmetic Functions and Operators

When the **-compat** flag is active, the following arithmetic function/operator rules apply:

```
log(x) = sign(x) * log(abs(x))
log10(x) = sign(x) * log10(abs(x))
db(x) = sign(x) * 20.0*log10*abs(x)
```

`sqrt(x)` is `-sqrt(abs(x))` if `x` is negative.

`x**n` is computed as `x**n` if `x` is positive, `-(abs(x)**n)` if `x` is negative, and 0 if `x` is 0.

The power operator (`^`) has highest precedence (same as standard Eldo); prior to v6.3\_2 it had lower precedence in -compat mode than the multiplication and division operators.

---

**Note**

---



In Eldo standard mode:

`sqrt(x)` returns an error if `x` is negative.

`x**n` is computed as `exp(n*log(x))` if `x` is strictly positive, 0 otherwise.

---



For more information see [page 3-7](#).

---

## EVAL keyword

When the `-compat` flag is active, the function keyword `EVAL` is not required in conditional expressions. For example:

```
r1 1 2 '(p1 > p2 ? p2: p1)'
```

Will be the equivalent of:

```
r1 1 2 'eval(p1 > p2 ? p2: p1)'
```



# Chapter 13

## TIs spice Compatibility

## Introduction

Eldo provides partial compatibility with TIs spice version 3.40. TIs spice is developed by Texas Instruments Incorporated. This chapter describes some specific TIs spice syntax that is compatible inside Eldo.

Eldo TIs spice compatibility is invoked as follows:

```
eldo -tispice ... cir_file_name
```

or by adding in the netlist the following option:

```
.OPTION TISPICE
```

This chapter shows the different effects that this compatibility mode has.

## Devices

### Model mapping

The following Eldo Levels are set:

**Table 13-1. MOS Levels in TIs spice Compatibility Mode**

LEVEL	Model Name (Eldo Level)
4	Berkeley BSIM1 (Eldo LEVEL 8)
8 or 49	Berkeley BSIM3 v3.0 & BSIM3 v3.1 (Eldo LEVEL 53)
9	Berkeley BSIM3SOIv3.1.1 PD (Eldo LEVEL 56, SOIMOD=0)
14 or 54	Berkeley BSIM4.0.0 (Eldo LEVEL 60)

**Table 13-2. BJT Models in TIs spice Compatibility Mode**

LEVEL	Model Name (Eldo Level)
2	VBIC v1.2 (Eldo LEVEL 8)
	(VERSION=1.15) VBIC v1.1.5 (Eldo LEVEL 8)
504	Philips Mextram 504 Model (Eldo LEVEL 22)

## Resistor and capacitor model syntax

.MODEL R or C: SCALE equivalent to Eldo R or C

.MODEL R and no level => level = 5

.MODEL C and no level => level = 5

### Resistor model level 5

This is the default resistor model when in Eldo TIsipce compatibility mode. L and W are instance parameters. Model parameters:

RSH	Sheet resistivity. Unit is Ohm/square. This value is mandatory.
LR	The reduction in length from side etching. NARROW overrides this parameter. Default is 0.
NARROW	The narrowing of the resistor due to side etching. The units for this are meters, and the default value is 0.
WR	The reduction in width from side etching. NARROW overrides this parameter. Default is 0.
SCALE	The scaling factor (on top of all other calculations) used to multiply the value of the resistance before analysis. The default value is 1.

The R value is computed as:

$$R = RSH \times (L - LR)/(W - WR)$$

If neither L or W are specified, then it is assumed that the R value is specified on the instance card: then only model parameter SCALE will be considered.

### Capacitor model level 5

This is the default capacitor model when in Eldo TIsipce compatibility mode. Model parameters:

cj	The junction bottom capacitance for semiconductor capacitors. Units for this are farads/square meter. This parameter must be specified for a semiconductor capacitor.
cjsw	The sidewall junction capacitance in farads/square meter for a semiconductor capacitor. This is also a required model parameter for a semiconductor capacitor.
defw	The default width for a semiconductor capacitor. A W=width statement on capacitors overrides this value. The units for this are meters, and the default value is 1E-6.

narrow      The narrowing of the capacitor due to side etching in the case of semiconductor capacitors. The units for this parameter are meters, and the default value is 0.

scale      The scaling factor (on top of all other calculations) used to multiply the value of the capacitance before analysis. The default value is 1.

Usage of these model parameters depends on how the device has been instantiated:

- if W or L specified:

$$C = CJ \times (L - NARROW) \times (W - NARROW) + 2 \times CJSW \times (L + W - 2 \times NARROW)$$

- if AREA or PERI specified:

$$C = CJ \times (AREA - NARROW \times PERIMETER / 2 + NARROW^2) + CJSW \times (PERIMETER - 4 \times NARROW)$$

- None specified: C is assumed to be specified on the instance card.

## Resistor, self inductor and capacitor

The model name, if any, is placed after the nominal value for a non-geometric model, and before any parameter for a geometric model:

```
R1 a b value MODEL TC1 = 1
R1 a b MODEL L = ...
```

In the case of a non-geometric model, the 5th token can be a special parameter name rather than a model name. If the 1st letter of the 5th argument is a P, then Eldo will look for a parameter rather than a model. The value of that parameter is a multiplier of the device. For example:

```
R1 1 2 1k pr
.param pr = 2
```

The actual value of R1 will be 2k.

## MOS length and width

Length and width of MOS are supposed to be given in Microns. Same for PD and PS. AD and AS are also expected in Microns<sup>2</sup>. The default in Eldo is meters.

# Commands

## .PARAM command

The '=' is not mandatory on the **.PARAM** but if the '=' is omitted, there can be only one parameter defined per **.PARAM** statement.

## .LIB command

If dns=/path on a **.LIB** statement is specified, the dns= is ignored. This was old TIspice syntax.

## .TRAN command extension

The full TIspice **.TRAN** syntax is accepted.

TMIN	sets hmin
TSHIFT	sets time at which all independence sources begins their evaluation
TPUNCH	creates a save file for each tpunch timepoint
WRINIT RDINIT RDFORCE WRFINAL	handles init filenames

The following options are accepted but ignored: SAVE, CHECKSTOPDELAY, TRNOISE, INITERROR.

## .INCLUDE/.LIB commands

These commands accept the syntax SECTION = <>, which gives the corner name of the library.

When **NOPRINT** is specified the library netlist is not dumped in the output (.chi) file.

## .RERUN command

Same as the Eldo **.ALTER** command except that a **.END** statement is required between each **.RERUN**.

## .PRINT/.PLOT command extensions

General form:

```
.print analyse_type signal_list
.plot analyse_type signal list
```

signal\_list is the collection of output variables. TIspice output variables can be specified in one of five ways:

- using a full name format
- using a partial name format
- using regular expression format
- specifying depth of hierarchy
- using a mathematical-expression format

Currently only partial name and full name formats are supported in Eldo. An example of partial name format is:

```
.print ac all(v i) except(i(x1.x1))
```

This prints all node voltages and all currents except any current in the subcircuit x1.x1.

Output variables supported by Eldo are:

- Voltage at nodes, for example:

```
v(4) v(5,3) v(2 4,5 out x1.1)
```

- Voltage on devices, for example:

```
E(Q1,BE,CE M1,DS)
E(C1,PN) E(C2)
```

Note: reversing voltage is not supported yet in Eldo. i.e. E(Q1,EB) and E(C1,NP) will both be ignored.

Note: default keyword handling is fully supported by Eldo. E(C1) is similar to E(C1,PN). But the voltage may not exist in Eldo, so E(M1) will be ignored in Eldo as it is similar to E(M1,DG).

The supported keywords are:

**Table 13-3. Supported Keywords for Voltage on a Device**

Element type	Supported keywords
JFET	DS, GS
MOSFET	DS, GS, GB, BD, BS
BJT	CE, BE, BC

- Current in a device terminal, for example:

```
I(Q1,C,B,E D J1,D,G,S M1,D,G,S,B XA.Q1,B,C)
```

Note: default keyword handling is fully supported by Eldo.

- Current density in a device

Not supported for resistors, diodes, JFETS, MOS and BJTS, otherwise its is equivalent to current.

- Power, for example:

```
P(Q1 Q2)
```

Note: Time averaging is not currently supported. P(Q1,TA) will be ignored.

- Noise variables, for example:

```
INOISE ONOISE(DB) ONOISE
```

When in TIspace compatibility mode the output variable names are assumed to be in the TIspace format. To keep the Eldo format with TIspace compatibility mode you must use the option **TIELDOOF**.

So in TIspace compatibility mode and with option **TIELDOOF** specified, the following statement will give an error:

```
.print dc E(D1,PN R1)
```

## .PUNCH command

The TIspace **.PUNCH** command is synonymous with the Eldo **.PROBE** command.

## .FOUR command extension

This calculates the fourier coefficients for the sinusoidal harmonic components of any output variable in the circuit. This command creates a table containing:

- DC component
- FOUR\_NCOEFF first Fourier coefficients
- Total harmonic distortion THD

## .FORCE command

Forces one or more nodes to specified voltage(s) with respect to ground for the initial transient solution. This is similar to the Eldo **.IC** command, values will be used only for the DC performed prior to TRAN analysis. Both commands are used to give values replacing the DC solution. With the **.FORCE** syntax, initial values of inductors current can be given while with Eldo **.IC** only node voltages can be initialized including nodes inside subcircuits. Syntax:

```
.FORCE [NODE] node_name1 value1... node_name2 value2...
+ [IND|OBJECT] object_name1 value1...
```

node\_name Node name.

VALUE Value at node.

IND|OBJ Inductor or object. For objects, only voltage source components can be specified to set the current on.

object\_name Inductor or voltage source component.

## .INITIAL command

Specifies estimated solutions of DC analyses to improve DC convergence. This is similar to the Eldo **.NODESET** command, but analysis type can be specified, and **.INITIAL** can also be applied to impose **.NODESET** conditions on devices. Syntax:

```
.INITIAL ANALYSIS [NODE] node_name1 value1... node_name2 value2...
+ [IND|OBJECT] object_name1 value1...
```

**ANALYSIS** Type of analysis can be OP, TR or DC[SWEEP].

**OP**

Initial conditions for operating point analysis.

**TR**

Initial conditions for transient analysis.

**DC**

Initial conditions for DC analysis.

**node\_name** Node name.

**VALUE** Value at node.

**IND|OBJ** Inductor or object. For objects, only voltage source components can be specified to set the current on.

**object\_name** Inductor or voltage source component.

Examples:

```
.INITIAL OP A 5V
.INITIAL TRANSIENT A 4V
```

When an OP analysis is being performed, 5V will be used as a NODESET value. When a TRAN analysis is being performed, 4V will be used as a NODESET value

```
.INITIAL DC IND L1 5m
```

During a DCSWEEP analysis, a value of 5m will be used as NODESET value for the current flowing through object L1.

## .ECHO command

Echo on/off is used to switch on/off the netlist lines to write to the output (*.chi*) file. The netlist lines between the echo off/on statements are disabled.

```
.ECHO ON|OFF [<string>]
```

The optional string on the echo command is always dumped.

Example:

```
*test ti
.param n=1
.param k=0
.param k1=n**k
.echo off "END OF DUMP"
v1 1 0 #n**k#
R1 1 0 1k
.echo on "RESTART THE DUMP"
.op
.print op v(1)
.end
```

The part of the netlist between echo off and echo on will not be written in the output file because echo has been disabled (**.echo off**). The last part will be written because echo has been re-enabled (**.echo on**).

## Netlist

### Model selection using parameter string

```
X1 ... MOD = foo
M1... %MOD% ...    -> model foo will be used
```

### Value “x”

x indicates unit 1e6 i.e. 2x represents 2.0e6. MEG is required in Eldo.

### Expressions

Mathematical expressions can be enclosed in pound signs (#) or single-quotes (').

### Parameter statements

Two ways of writing TIspace parameter statement allowed:

```
P pname [=] value [TC=<TC1> [, <TC2>]]
or:
```

```
.PARAM pname value [TC=<TC1> [, <TC2>]]
```

PR, PL, and PC are pre-defined parameter names that act as scaling factors for all resistors, inductors, and capacitors respectively in the circuit. These default to 1 and can be changed using this statement. These can be defined local to subcircuits and also passed as subcircuit parameters. Use extreme caution in using these parameter names, as they change the values of all R, L, and C elements in the circuit.

# Functions and Sources

## Delay

DELAY is equivalent to TD or SHIFT in PWL signals.

## Sources

Current and voltage sources SIN, SFFM, AM, EXP allowed.

# Options

## Dangling nets

ALLOW\_DANGLING\_NETS causes Eldo to find all nets in the circuit with only one connection and connect each one to ground through a resistance of 1/GMIN.

NOALLOW\_DANGLING\_NETS causes Eldo to give a topology error and stop if any nets are found with only one connection.

# Eldo Extensions for Tlspice Compatibility

## .DC command extensions

- Sweep of gain for linear controlled sources

The **.DC** command has been extended to support the sweep of gain for linear controlled sources VCVS, CCVS, CCCS, VCCS. Only the linear case is supported and only the gain of these sources may be swept. Example:

```
.dc E1 3 5 0.5
```

The gain of the VCVS E1 will be swept from 3 to 5 in increments of 0.5.

See [page 10-54](#) of this manual.

- Third level dc sweep

The DC sweep can be nested up to three levels deep. Example:

```
.dc v1 2 3 0.5 v2 5 9 0.5 v3 -1 2 0.1
```

## .DEFAULT command

This command resets the default values for elements, device initial conditions and model parameters.

The general form for resistors, capacitors, and inductors is:

```
.DE[FAULT] TYPE VALUE
```

The general form for other types is:

```
.DE[FAULT] TYPE {KEYWORD [VALUE]}
```

Eldo implementation of **.DEFAULT** does not support the Lossy Transmission Line (LDTL) model and IC for active devices are ignored.

See [page 10-65](#) of this manual.

## **.MEAS command extension**

For the trigger and target specification formats of the **.MEAS** command, SIG\_H and SIG\_L are also available. These represent the High and Low signal values respectively. Default high and low values are trig\_val for the trigger signal and targ\_val for the target signal. These high and low values are used to validate a transition before incrementing the cross, rise or fall counter.

These specifications are allowed in both standard Eldo mode and TIs spice mode.

See [page 10-155](#) of this manual.

## **.MSELECT command**

Automatic model selection. This command allows the user to select model automatically for MOS devices. The selection is based on:

- the size and temperature of the specific device (W, L, TEMP)
- the size and temperature constraints of each model in the list provided (WMIN, WMAX, LMIN, LMAX, TEMPMIN, TEMPMAX)

Syntax:

```
.MSEL[LECT] dummy [models] mod1 [mod2 [mod3 [...]]]
```

It is not allowed to have a model statement with the same name as an mselect dummy model name.

## **.SCALE command**

This command scales device and model parameters of active devices automatically.

The general form for devices (elements) is:

```
.SC[ALE] ELTYPE KEYWORD VALUE [KEYWORD VALUE ...]
+ [ELEMENTS ALL|EXCEPT] [ELNAME1 ELNAME2 ...] [(ELNAME1 ELNAME2)]
```

```
.SC[ALE] ELTYPE KEYWORD VALUE [KEYWORD VALUE ...]
+ MODELS MODNAME1 [MODNAME2 ...]
```

The general form for devices models (models) is:

```
.SC[ALE] MODTYPE KEYWORD VALUE [KEYWORD VALUE ....]
+ [MODELS ALL|EXCEPT] [MODNAME1 MODNAME2] [(MODNAME1 MODNAME2...)] ]
```

An additional feature is parameter scaling:

```
.SC[ALE] P FACTOR=VALUE [SUBCKT=SUBNAME] [INST=INSTNAME]
+ [PARAMS ALL|EXCEPT] [PARAM1 PARAM2 ...] [(PARAM11 PARAM22)] ]
```

The element parameters or devices are only scaled at the parsing level.

If a **.DC** is used on a device element with a scale factor, the device element takes only the value specified by the DC analysis. There is no scaling effect.

The naming convention of devices, parameters, and models follows the Eldo naming convention if the **TISPICE** option is not set. The Tlispice naming convention is followed if the option is set. Tlispice uses partial names instead of wildcards, i.e. MOD1 means all names beginning with MOD1: MOD12, MOD1TT, MOD1, MOD1\_A11, for example. In Eldo mode wildcards must be specified: MOD1\*.

## EBIT and PBIT source functions

The Tlispice EBIT and PBIT source functions have been implemented. These represent exponential pulse with bit pattern and trapezoidal pulse with bit pattern sources respectively.

The full Tlispice specifications are used in Tlispice mode. However, standard Eldo also has access to this command, with the following changes:

- the Tlispice optional parameter TD is specified as a mandatory first argument in Eldo
- the Tlispice optional parameter NONPERIODIC is replaced in Eldo by the R keyword which can be found also on the original PATTERN Eldo statement
- the \$ sign which must be set before the pattern string in Tlispice is not required in Eldo
- Eldo does not allow specifying a file containing the bit pattern

For example, in Tlispice, the statement:

```
v1 1 0 ebit 1 5 1n 0.5n 0n 1n 5n $101011101000011
```

is written in Eldo standard mode as:

```
v1 1 0 ebit 1 5 0n 1n 0.5n 0n 1n 5n 101011101000011 R
```

See [page 5-37](#) of this manual.



# Chapter 14

## Transient Noise Analysis

---

### Introduction to Transient Noise Analysis

In most analog design applications, knowledge of noise levels generated by the circuit is of great importance. In all traditional SPICE-like simulators, noise computation is only available in AC analysis. In this case the circuit is assumed to have a fixed bias (DC operating point), and noise simulation can only be applied to circuits working under small signal conditions. For this reason, many applications cannot be simulated and their noise performance is unknown until physical measurement is possible on the manufactured design. The only other method available to obtain this important information is to perform tedious hand calculations (when possible).

Eldo provides a solution to this noise analysis problem:

#### NOISE SIMULATION DURING TRANSIENT ANALYSIS

Transient noise simulation can be applied to all types of circuit without restriction. To perform transient noise analysis, physical noise of electrical devices is emulated by time dependent current sources. The frequency characteristics of these sources are referred to the noise models of the noisy components. The method used is simple, fast and does not disturb the simulated behavior of the circuit because the noise signals introduced are continuous and fully deterministic.

Simulation results from transient noise analysis include:

- Transient response of the circuit, as generated by a pure transient analysis.
- Transient response with the added noise generated by the circuit (plotted as an oscilloscope or experimental measurements).
- The RMS value of the noise versus time which gives the accuracy of the ideal transient response.

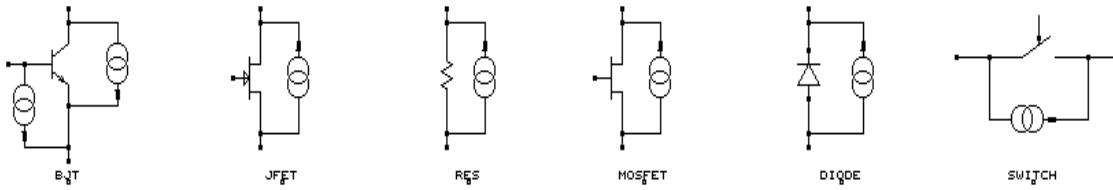
To compute the RMS value of the generated noise, Eldo performs a normal transient analysis of the circuit, plus  $N$  transient simulations which include noise sources within the noisy devices. The RMS value of noise is computed at each time-step as follows:

$$\text{RMS}(t) = \sqrt{\frac{1}{N} \sum_{i=1}^N (V_i(t) - V_o(t))^2}$$

Where  $V_o(t)$  represents the noiseless transient response of the circuit, and  $V_i(t)$  is the  $i^{\text{th}}$  transient response of the circuit including the noise sources;  $N$  is the number of noise simulations performed.

To perform transient noise analysis, Eldo adds a noise contribution to the same components as in AC noise analysis (R, M, B, J, D, see below). Moreover, an additional noise model, connected with the Switch macromodel (S), has been introduced. It is therefore possible to simulate noise in switched capacitor circuits (see “[Example 2—Switched Capacitor Filter](#)” on page 14-9).

**Figure 14-1. Circuit Components with their Added Noise Sources**



Please refer to the noise models in the respective sections of the [Device Models](#) chapter.

## .NOISETRAN Command

**.NOISETRAN FMIN=VAL FMAX=VAL NBRUN=VAL [OPTIONS]**

This command is used to control the transient noise analysis of a circuit and must be used in conjunction with a transient analysis (**.TRAN**).

### Parameters

- **FMIN, FMAX**

Define the range of the noise frequency band (same function as **FSTART** and **FSTOP** in AC noise analysis).

---

#### Note



**FMIN** and **FMAX** define the frequency band of the noise sources. This frequency range may sometimes not correspond to the noise frequency band at the output of the circuit. For instance, the band (**FMIN, FMAX**) does not correspond to the output noise frequency band in the case of filters, oscillators and mixers that exhibit frequency conversion.

---

- **NBRUN**

Defines the number of noise simulations to perform (those which include the noise sources). This parameter defines the accuracy of the noise analysis.

---

#### Caution



Care must be taken when setting the **NBRUN** parameter as it strongly influences the CPU time used by the simulation.

---

- **OPTIONS**

Other options are available for more advanced and more efficient use of the command.



---

Please refer to “[.NOISETRAN](#)” on page 10-182 for more details.

---

During the transient noise analysis, Eldo generates noise sources in the time domain for each noisy component. These noise sources were generated as a sum of **NBF** sinusoids distributed between **FMIN** and **FMAX**. It means that the generated noise sources have a spectrum of discrete points and the energy of the noise sources is localized in a limited number of frequency points. When a frequency band is wide the corresponding generated noise source may have a poor frequency resolution.

Consequently, a new algorithm has been developed to generate the noise sources, leading to a continuous spectrum in the frequency band zero to **FMAX**. The new algorithm is approximately twice as fast and with more accurate results. However, it cannot generate a noise source with a frequency band beginning at a value of **FMIN** other than zero. This algorithm is automatically activated by putting **FMIN** to zero. When **FMIN** is different from zero, the method using

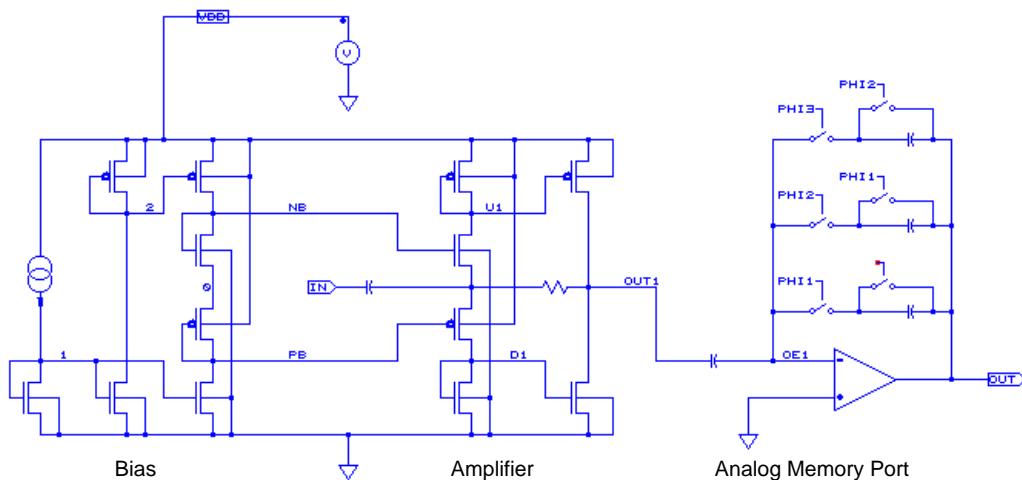
sinusoids is then used. Both methods can work together, one for some components, the other method for different noisy devices.

**Note**

The new method is used to generate flicker noise and white noise sources (thermal and shot noise). Because it is not possible to have a flicker noise source with a frequency band starting from zero, the generated flicker noise source will have a white spectrum from zero to  $F_1$  and a flicker noise spectrum from  $F_1$  to  $F_{MAX}$ . The frequency  $F_1$  is calculated from the transient simulation duration:  $F_1=1/TSTOP$ .

## Example 1—High-rate Particle Detector

**Figure 14-2. High-rate Particle Detector Circuit**



This circuit was designed at CERN and is used in high-rate particle detection. It comprises a front-end pre-amplifier and an analog memory port. Performance analysis of the complete circuit is given in the following sections. Transient simulations have been performed and noise characteristics investigated. The netlist for the circuit, describing the analysis performed, can be found after the simulation results.

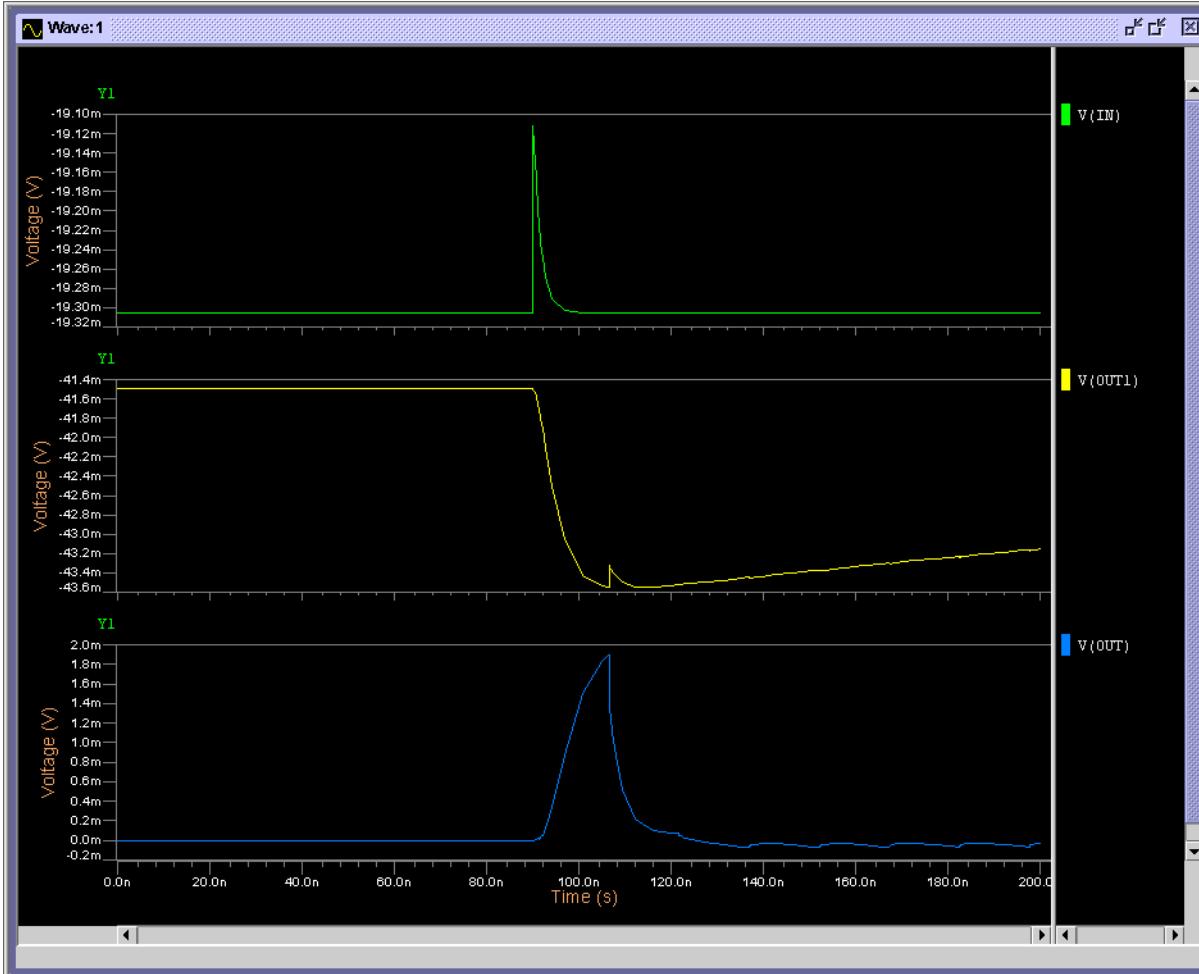
## Description of the Particle Detector Behavior

The pre-amplifier has a rise time of less than 20ns and a large fall time constant compared to the rise time. Therefore, for this application it can be considered as an integrator. The function of the analog memory port is signal storage and noise shaping. It takes successive samples of the signal into the different feedback capacitors. The complete system of analog memories acts as a charge sampler and is equivalent, from a signal processing point of view, to a discrete differentiator.

Shown below is the signal at the input, the voltage at the amplifier output and the signal at the output of the complete system. The simulation was performed for an input charge of 2.5 fC across a 5 pF input capacitor and with a 15 ns clock period.

The clock of the analog memory port is synchronized with the input signal in order to store the maximum amount of charge in the first feedback capacitor. We can note that most of the charge is deposited in one storage capacitor within the 15 ns clock period.

**Figure 14-3. Simulation Results—Input & Output Signals**



## Analysis of the Noise Performance

The resolution of this detection system is a function of the noise generated by the circuit. It is therefore the major performance criterion of the device.

The noise performance of the complete circuit strongly depends on the performance of the pre-amplifier. The analog memory port does not generate noise, it only shapes the noise generated by the amplifier.

## Transient Noise Analysis

### Example 1—High-rate Particle Detector

A number of transient noise simulations were performed on the complete circuit.

**Figure 14-4. Noise at Amplifier O/P**

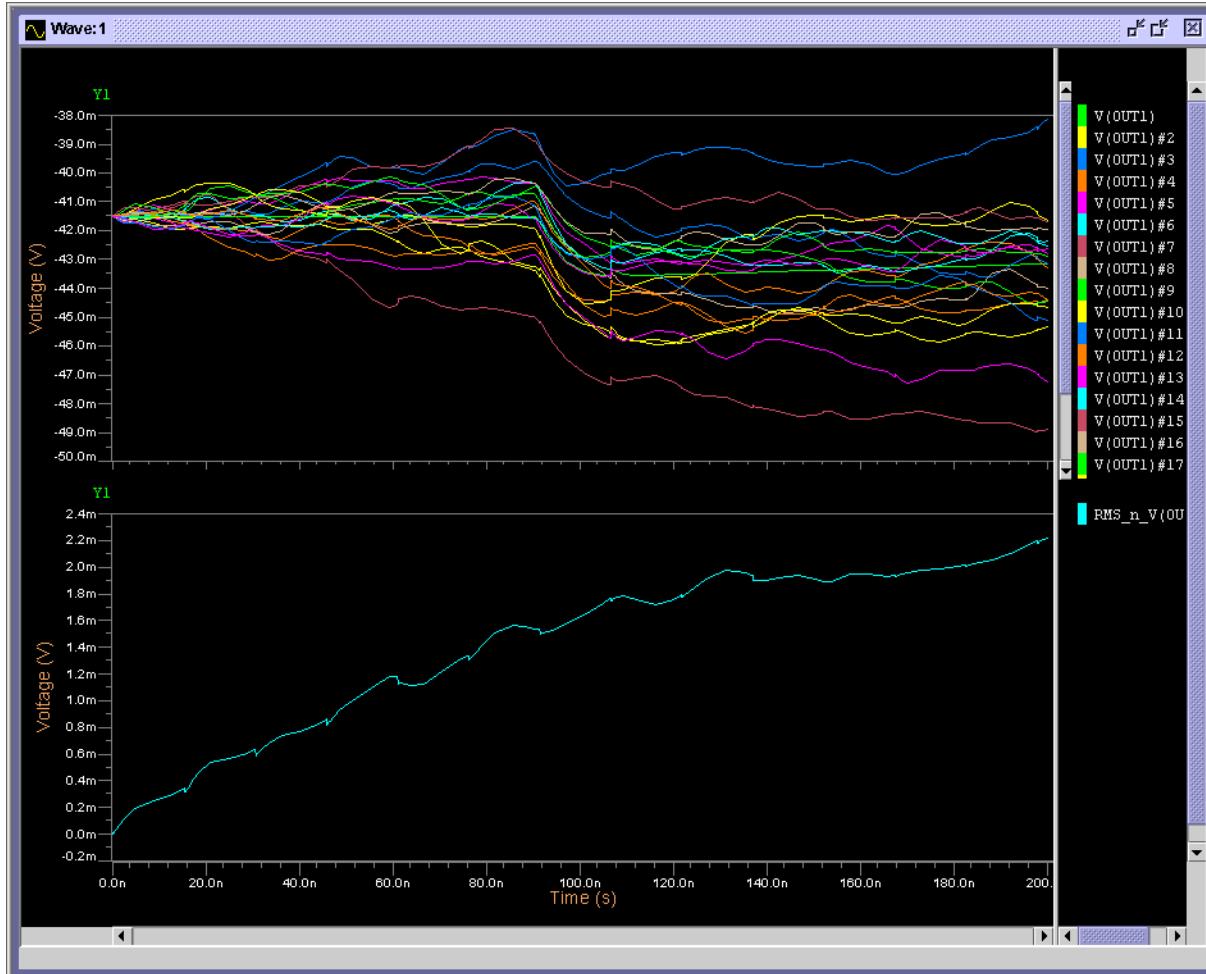


Figure 14-4 shows the results of several transient simulation runs including noise sources, with different initial conditions and the RMS value of the noise at the amplifier output calculated from the different runs described above.

#### Note

The RMS value of the noise increases with time and the Signal to Noise Ratio is limited by the low frequency noise components.

**Figure 14-5. Noise at Analog Memory O/P**

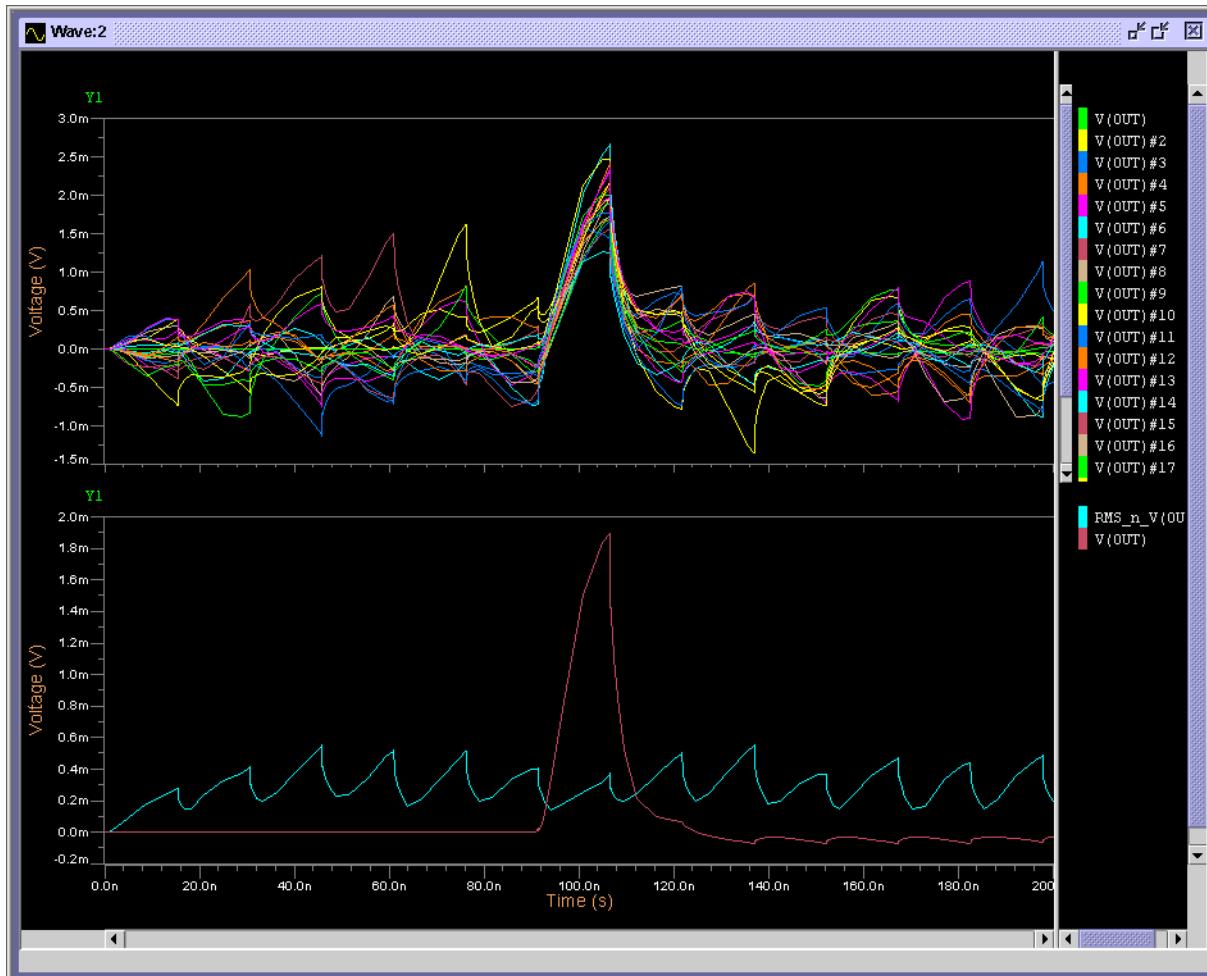


Figure 14-5 shows the curves related to the different runs with the noise sources, the signal at the output of the complete circuit, and the RMS value of the corresponding noise.

**Note**

The effect of the analog memories on the noise behavior acts as a Correlated Double Sampling, meaning that output noise is reset at each period of the clock.

For this type of circuit, the noise performance is expressed in terms of Equivalent Noise Charge referred on the input (**ENC**):

$$\text{ENC} = \frac{\text{Noise}}{\text{Signal}} \text{Qin}$$

Where **Noise** is the RMS noise value at the output (extracted from the figure above), **Signal** is the signal at the output and **Qin** is the charge injected across the input capacitor (in  $e^-$ ). In this case, the **ENC** is about **2000e<sup>-</sup>** across Cin (**Noise** = 0.9 mV, **Signal**=7 mV, **Qin**=2.5 fC). This

circuit has been manufactured and experimental measurements match the simulation results very closely.

## Netlist for Example 1

```
***** MOS MODELS *****
.model nmost nmos level=3 vto=0.75 gamma=0.4 phi=0.457
+ nsub=2.0e15 theta=0.05 ld=0.2e-06 xj=0.3e-6 tox=32.5e-9
+ delta=0.5 uo=700 rsh=37 eta=0.03 cj=20e-5 cjsw=3.5e-10
+ cgso=2.1e-10 cgdo=2.1e-10 mj=0.6 mjsw=0.4 cgbo=8.3e-10
+ pb=0.6 vmax=100e3 kappa=0.48 kf=16.8e-28 nfs=1e10
.model pmost pmos level=3 vto=-0.75 gamma=0.4 phi=0.6
+ nsub=1.0e16 theta=0.14 ld=0.3e-06 xj=0.4e-6 tox=32.5e-9
+ delta=1.5 uo=220 rsh=85 eta=0.08 cj=33e-5 cjsw=3.5e-10
+ cgso=3.1e-10 cgdo=3.1e-10 mj=0.42 mjsw=0.3 cgbo=8.3e-10
+ pb=0.6 vmax=206e3 kappa=15 kf=7.04e-29 nfs=1e10
***** SWITCH MACROMODEL *****
.MODEL swi NSW vh=0.1 vth=2 ron=1 cl=0 c2=0 crec=0
***** CIRCUIT DESCRIPTION *****
**** THE BIAS
Ipol vdd 1 60u
mn1 1 1 vss vss nmost w=35.5u l=5u
mnt1 2 1 vss vss nmost w=35.5u l=5u
mpt1 2 2 vdd vdd pmost w=70u l=5u
mnb2 pb 1 vss vss nmost w=35.5u l=5u
mpb2 nb 2 vdd vdd pmost w=70u l=5u
mnb1 nb nb 0 vss nmost w=540u l=1.5u
mpb1 pb pb 0 vdd pmost w=1080u l=1.5u
**** THE PRE-AMPLIFIER
mni1 u1 nb in vss nmost w=540u l=1.5u AD=+3.375E-09
+ AS=+3.6E-09 PD=+1.635E-03 PS=+1.744E-03
mpi1 d1 pb in vdd pmost w=1080u l=1.5u AD=+3.375E-09
+ AS=+3.6E-09 PD=+1.635E-03 PS=+1.744E-03
mni2 d1 d1 vss vss nmost w=35.5u l=5u AD=+9.0E-11
+ AS=+1.79E-10 PD=+5.0E-05 PS=+9.3E-05
mpi2 u1 u1 vdd vdd pmost w=70u l=5u AD=+9.0E-11
+ AS=+1.79E-10 PD=+5.0E-05 PS=+9.3E-05
mno1 out1 d1 vss vss nmost w=35.5u l=5u AD=+9.0E-11
+ AS=+1.79E-10 PD=+5.0E-05 PS=+9.3E-05
mpo1 out1 u1 vdd vdd pmost w=70u l=5u AD=+9.0E-11
+ AS=+1.79E-10 PD=+5.0E-05 PS=+9.3E-05
Cin inin in 5pf
Rfb in out1 100meg
Cout out1 0 .2pf
Vdd vdd 0 3V
Vss vss 0 -3V
**** ANALOG MEMORY PORT
Coel out1 oel 0.4pF
S1 phil oel s1 swi
Sr1 phi3 s1 out swi
Cfb1 s1 out 0.4pF ic=0
Vphil phil 0 pulse(-5 5 .1n .1n .1n 15n 45.6n)
S2 phi2 oel s2 swi
Sr2 phil s2 out swi
Cfb2 s2 out 0.4pF ic=0
Vphi2 phi2 0 pulse(-5 5 15.3n .1n .1n 15n 45.6n)
```

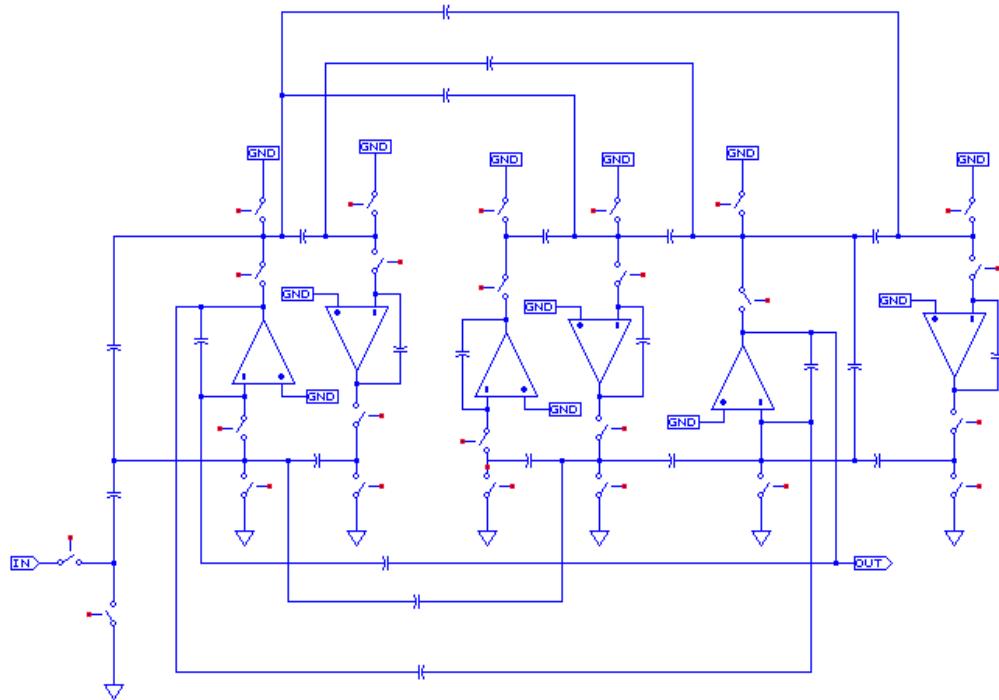
```

S3 phi3 oe1 s3 swi
Sr3 phi2 s3 out swi
Cfb3 s3 out 0.4pF ic=0
Vphi3 phi3 0 pulse(-5 5 30.5n .1n .1n 15n 45.6n)
**OTA definition**
Gamp 0 out 0 oe1 .002
Ramp out 0 0.5Meg
Camp out 0 1pF
Cpip oe1 0 1pF
Ritg out oe1 lg
.ic v(oe1)=0 v(s1)=0 v(s2)=0 v(s3)=0 v(out)=0
**** TRANSIENT NOISE SIMULATION
Vin inin 0 pw1(0 0 90n 0 90.1n 0.3mv 600n 0.3mv)
.tran 200n 200n
.noisetrans fmin=100k fmax=100meg nbrun=20
.plot tran v(in)
.plot tran v(out1)
.plot tran v(out)
**** AC ANALYSIS OF THE AMPLIFIER
Vin in 0 dc 0 ac 1
.ac dec 10 100k 100meg
.noise V(out1) Vin 10
.plot ac vdb(out1)
.plot noise inoise onoise
.option thermal_noise=2
.end

```

## Example 2—Switched Capacitor Filter

**Figure 14-6. Switched Capacitor Filter Circuit Schematic**

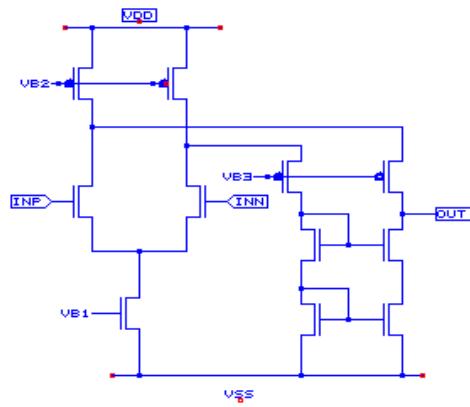


This second example, shown above, is a 6th order switched capacitor bandpass filter used in telecommunications applications. Complete simulation results of the circuit and its noise performance are given in this section. To increase simulation speed, modeling is implemented at a macromodel level rather than transistor level. The Eldo macromodels **SWITCH** and **OPA** are used. We will first describe the macromodels used and simulation results follow thereafter. Finally the netlist describing the simulation conditions is listed after the simulation results.

## Characterization of the Amplifier

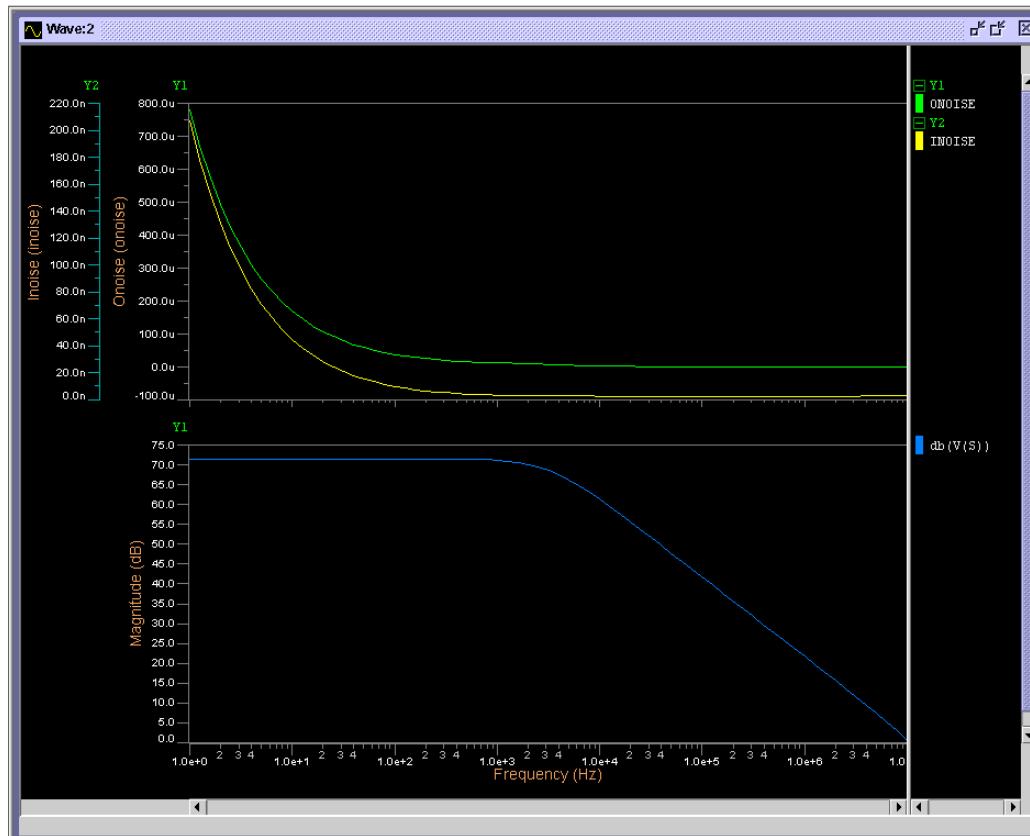
The structure of the amplifier is a classical folded cascade. AC and transient simulations have been performed at a transistor level in order to determine the macromodel parameters of the circuit shown below:

**Figure 14-7. Amplifier Schematic**

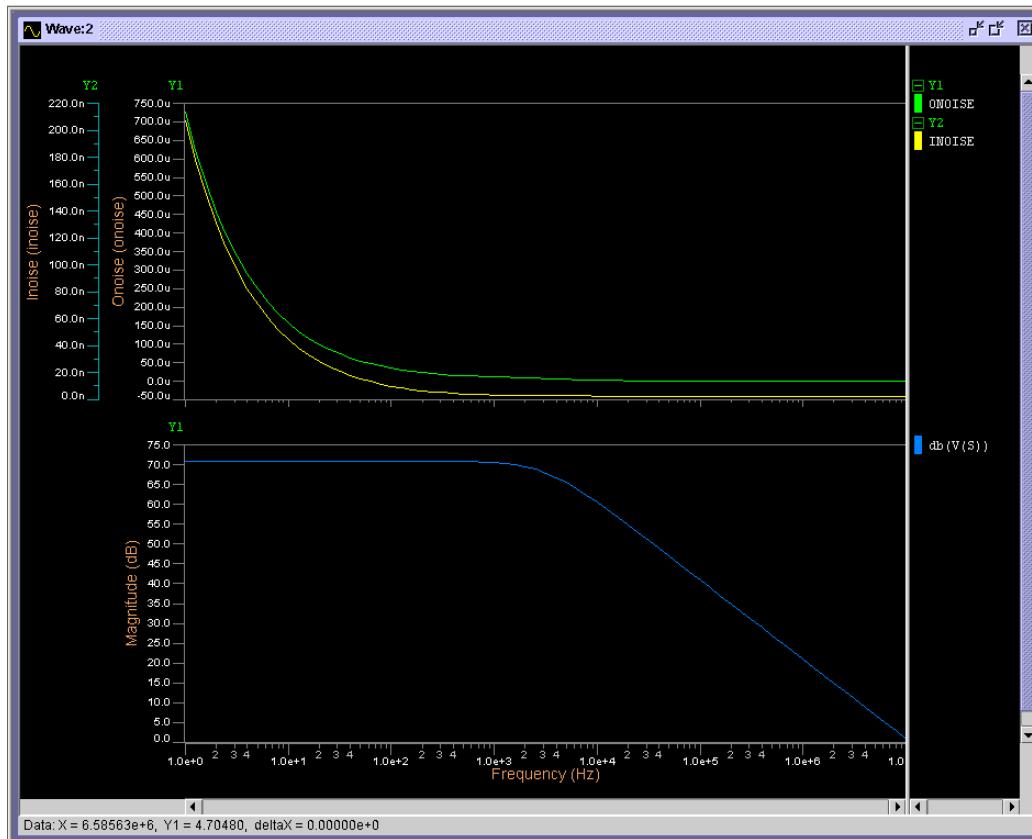


A subcircuit composed of an **OPA1** macromodel, and a voltage source representing the equivalent noise referred at the input, is used for the simulation of the complete circuit.

Figure 14-8. AC & Noise Simulation Results of Amplifier



**Figure 14-9. AC & Noise Simulation of Amplifier Macromodel**



The dominant pole of the amplifier is defined by the output impedance and load capacitance of the amplifier. Comparison of simulation results between the amplifier and its macromodel are shown in [Figure 14-8](#) and [Figure 14-9](#).

## Netlist Used for the Amplifier Simulations

### AOP Analysis at Transistor Level

```

AOP ANALYSIS
*** AOP Analysis at transistor level
.GLOBAL VDD VSS
.MODEL MN NMOS NIV=6 EOX=200E-10 MU0=520 DPHIF=0.8 DW=1.0E-6
+ DL=0.1E-6 VT0=0.75 KB=0.62 REC=0.1E-6 TG=0.08 VL=1.0E5
+ GL=0.5E-6 KL=0.3E-6 KW=0.2E-6 DINF=0.3 GW=0.4E-6
+ LDIF=3.3E-6 CDIFS0=1.4E-4 CDIFP0=8E-10 VE=20E4 LMIN=1.2E-6
+ WMIN=3.4E-6 RSH=525 AF=1.33 Kf=2.7e-26
.MODEL MP PMOS NIV=6 EOX=200E-10 MU0=190 DPHIF=0.8 DW=1.0E-6
+ DL=0.3E-6 VT0=-0.8 KB=0.36 REC=0.1E-6 TG=0.13 VL=2E5
+ GL=0.34E-6 KL=0.2E-6 KW=0.4E-6 DINF=0.12 GW=0.22E-6
+ LDIF=3.3E-6 CDIFS0=3.2E-4 CDIFP0=8E-10 VE=10E4 LMIN=1.4E-6
+ WMIN=3.4E-6 RSH=1225 AF=0.89 KF=2e-29

```

```

.SUBCKT BIAS VB VB1 VB2 VB3
M1 VSS VB 4 VSS MN L=4.00U W=66.00U
M3 VSS VB VB VSS MN L=4.00U W=66.00U
M5 VB1 VB1 VSS VSS MN L=4.00U W=37.00U
M6 VB1 VB3 VB3 VSS MN L=8.00U W=144.60U
M9 VB3 VB3 8 8 MP L=1.40U W=97.20U
M11 4 4 VB2 VB2 MP L=4.00U W=241.00U
M15 VDD VB2 8 VDD MP L=3.00U W=96.40U
M17 VDD VB2 VB2 VDD MP L=3.00U W=96.40U
C19 VSS VB3 6.32179E00PF
C20 VSS VB2 6.32179E00PF
C21 VB1 VSS 6.32179E00PF
.ENDS
.SUBCKT OPAMP AOUT INP INN VB1 VB2 VB3
M1 VSS 13 11 VSS MN L=10.00U W=194.40U
M5 VSS VB1 12 VSS MN L=4.00U W=118.00U
M7 3 INP 12 VSS MN L=1.20U W=669.00U
M17 4 INN 12 VSS MN L=1.20U W=669.00U
M27 VSS 13 13 VSS MN L=10.00U W=194.40U
M31 13 14 14 VSS MN L=1.50U W=25.60U
M35 AOUT 14 11 VSS MN L=1.50U W=25.60U
M39 3 VB2 VDD VDD MP L=2.00U W=178.00U
M43 4 VB2 VDD VDD MP L=2.00U W=178.00U
M47 3 VB3 14 VDD MP L=1.40U W=194.80U
M51 4 VB3 AOUT VDD MP L=1.40U W=194.80U
.ENDS
***
vvdd vdd 0 2.5
vvss vss 0 -2.5
Xpol vb vb1 vb2 vb3 bias
Xaop s inp inn vb1 vb2 vb3 opamp
Rcha vdd vb 50k
Ccha s 0 50p
vinn inn 0 ac 1
vinp inp 0 0
.ac dec 10 1 10meg
.noise v(s) vinn 10
.plot noise onoise inoise
.plot ac vdb(s)
.option flicker_noise=1
.end

```

## AOP Analysis at Macromodel Level

```

*****
*** AOP analysis at macromodel level
.MODEL AMP OPA LEVEL=1 VOFF=0 IMAX=200UA
+      CIN=1.0E-12 RS=1Meg
+      GAIN=3500 FNDP=150Meg
+      VSAT=2.5 CMRR=-100db
.subckt opamp o1 o2 i1 i2
opax e1 i2 o1 o2 AMP
Vinoi e1 i1 noise 7.86e-18 4.29e-14 1.33
.ends

```

```
Xaop s 0 0 inn opamp
Ccha s 0 50p
vinn inn 0 ac 1
.ac dec 10 1 10meg
.noise v(s) vinn 10
.plot noise onoise inoise
.plot ac vdb(s)
.end
```

## Switch Noise Model

Instead of MOS transistors, a switch macromodel is used to perform the complete circuit simulations. To compute transient noise simulations, noise sources have been added to the switches. We have considered that a switch only generates thermal noise. A current source has been included between the source and the drain and its Power Spectral Density is a function of  $R_{on}$ , as follows:

$$S_1 = \frac{4kT}{R_{on}}$$

## Simulation Results of the Complete Circuit

Transient simulation results are shown below. In [Figure 14-10](#) the first curve represents the impulse response of the circuit. The second curve represents the RMS value of noise generated at the output of the filter in the frequency band (1Hz, 50kHz). We can see that this result is about 5mV. This has been obtained by performing about 20 runs including noise sources. This curve would be smoother if more runs had been performed. [Figure 14-11](#) shows the Fourier transform of the circuit output; it therefore represents the frequency response of the filter.

Figure 14-10. Simulation Results of the Filter

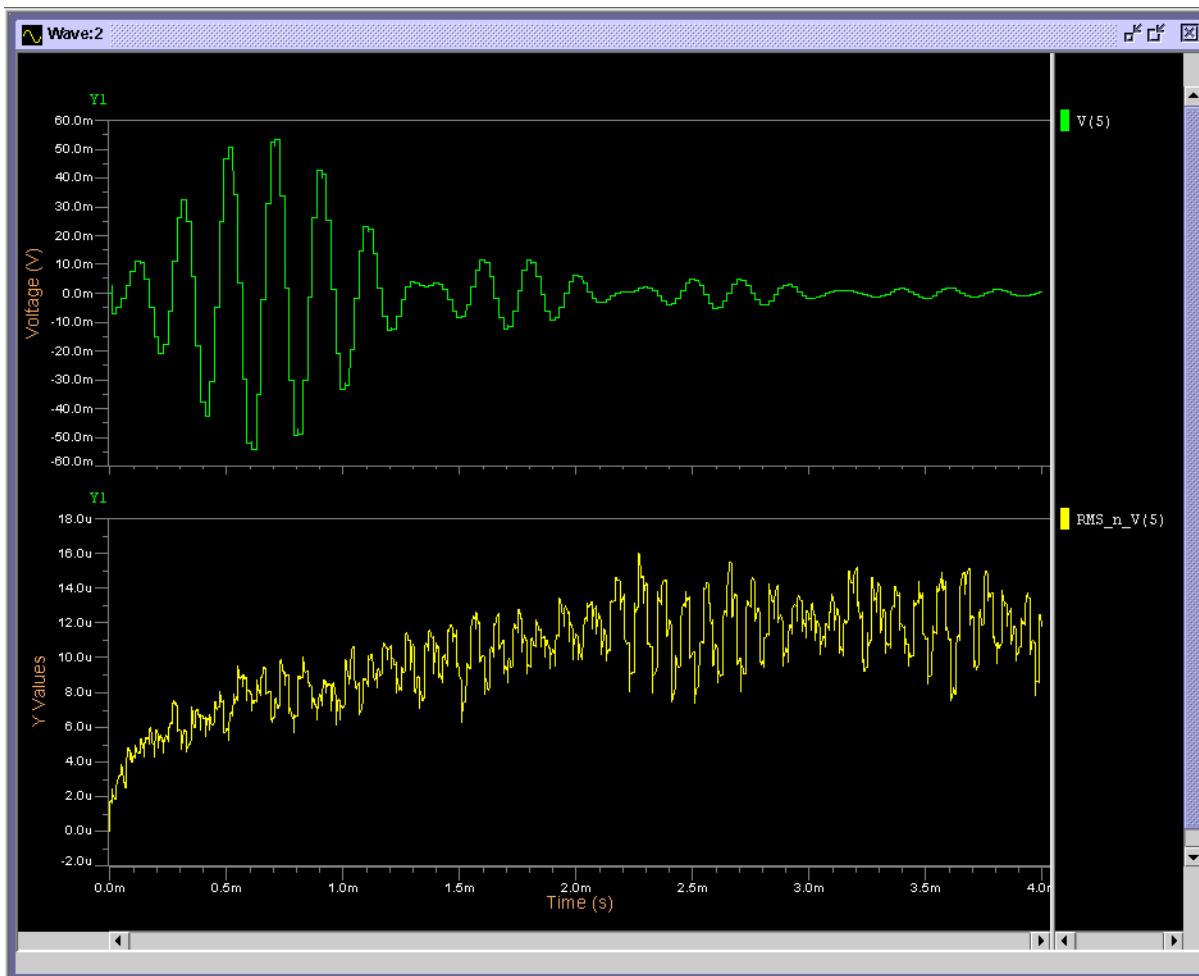
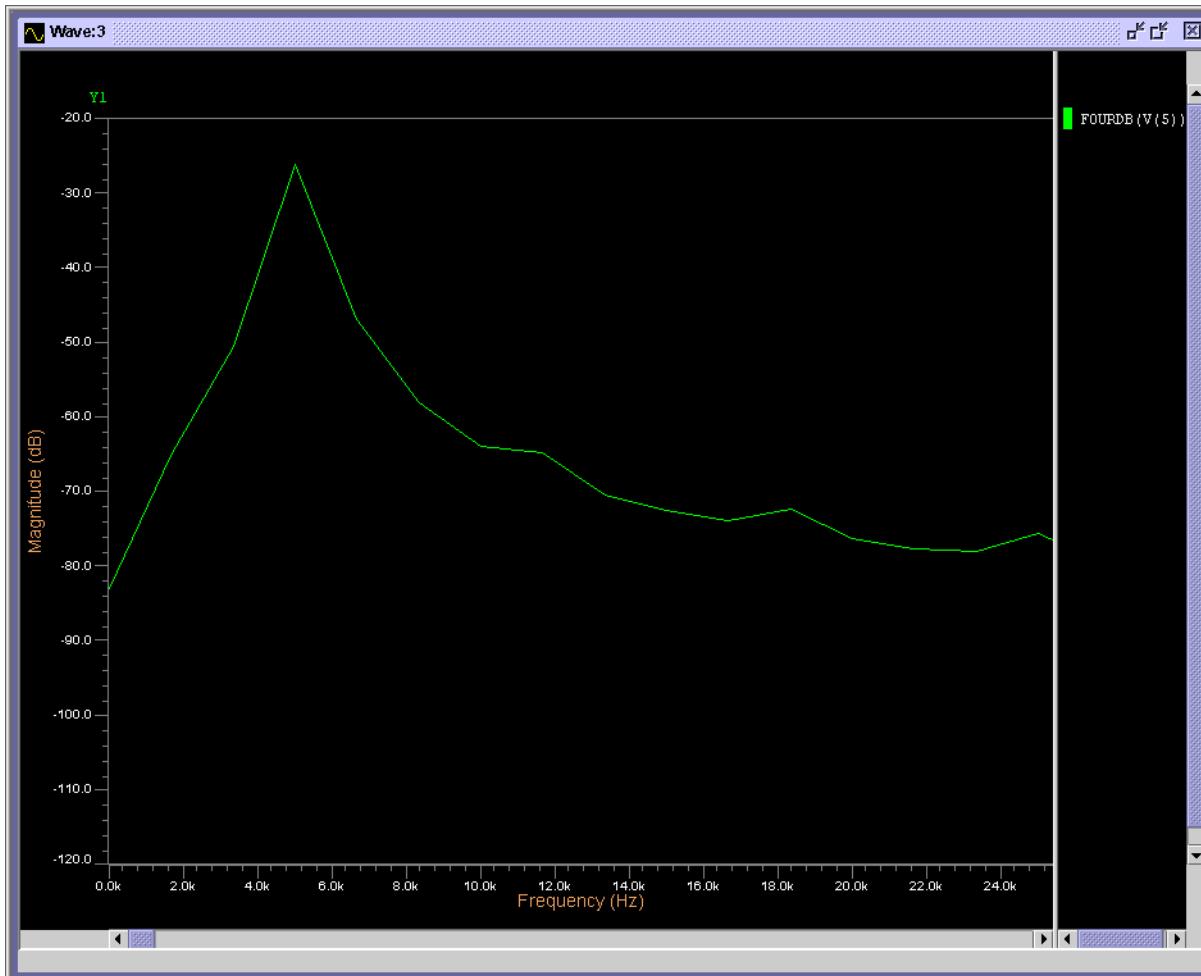


Figure 14-11. Frequency Response of the Filter



## Netlist for Example 2

```
* 6TH ORDER BAND-PASS FILTER *
.GLOBAL VDD VSS
VVDD VDD 0 2.5
VVSS VSS 0 -2.5
.MODEL AMP OPA LEVEL=1 VOFF=0 IMAX=200UA CIN=1.0E-12 RS=1MEG
+ GAIN=3000 FNDP=150MEG VSAT=2.5 CMRR=-100DB
.MODEL SWI NSW VH=0.5 VTH=0.4732 GOFF=0.01U RON=1.5K
.SUBCKT OPAMP O1 O2 I1 I2
VINOI E1 I1 NOISE 7.86E-18 4.28E-14 1.33
OPAX E1 I2 O1 O2 AMP
.ENDS
.SUBCKT IT PHI INP OUT
S0 PHI INP OUT SWI
.ENDS IT
```

```
*** INTEGRATOR 1
XS1 C 9 0 IT
XS2 CB 9 8 IT
XS3 CB 10 0 IT
XS4 C 10 7 IT
XS5 C 11 0 IT
XS6 CB 11 1 IT
XS7 C 12 0 IT
XS8 CB 12 4 IT
XS9 C 13 0 IT
XS10 CB 13 2 IT
*** INTEGRATOR 2
XS11 CB 15 0 IT
XS12 C 15 14 IT
XS13 CB 16 5 IT
XS14 C 16 0 IT
*** INTEGRATOR 3
XS15 C 18 0 IT
XS16 CB 18 17 IT
*** INTEGRATOR 4
XS17 CB 20 0 IT
XS18 C 20 19 IT
XS19 CB 21 3 IT
XS20 C 21 0 IT
*** INTEGRATOR 5
XS21 C 23 0 IT
XS22 CB 23 22 IT
XS23 C 24 0 IT
XS24 CB 24 6 IT
*** INTEGRATOR 6
XS25 CB 26 0 IT
XS26 C 26 25 IT
*** OUTPUT NODE 5
X1 1 0 0 8 OPAMP
X2 2 0 0 14 OPAMP
X3 3 0 0 17 OPAMP
X4 4 0 0 19 OPAMP
X5 5 0 0 22 OPAMP
X6 6 0 0 25 OPAMP
X7 OUT 0 5 OUT OPAMP
```

Transient Noise Analysis  
**Example 2—Switched Capacitor Filter**

---

```
C20675 10 9 9.23530E-01PF
C20678 9 11 4.52374E-01PF
C20680 12 9 8.43906E-01PF
C20683 13 9 6.42120E00PF
C20698 8 1 9.10828E00PF
C20719 8 5 2.66016E-01PF
C20720 12 18 7.09376E-01PF
C20722 17 3 1.26865E00PF
C20725 12 23 1.64195E00PF
C20729 23 16 1.23994E00PF
C20732 24 23 6.27077E00PF
C20747 22 5 8.70857E00P
C20767 22 1 7.09376E-01PF
C21108 11 15 9.29551E00PF
C21124 16 15 2.66016E-01PF
C21125 2 14 1.70001E01PF
C21153 11 20 9.73922E-01PF
C21155 16 20 7.09376E-01PF
C21157 20 21 6.92543E00PF
C21169 4 19 1.02160E01PF
C21186 16 26 3.50407E00PF
C21192 11 26 2.66016E-01PF
C21193 6 25 6.22576E00PF
*** INPUT SIGNAL AND CLOCK ***
VIN 7 0 pwl (0 0 200n 1 11u 1 11.2u 0)
VCLK00 C 0 PWL (0 -5 200n +5 10000n +5 10200n -5 20000n -5 R)
VCLKB00 CB 0
+ PWL (0 +5 200n -5 10000n -5 10200n +5 20000n +5 R)
.TRAN 1M 4M
.NOISETRAN FMIN=1 FMAX=50K NBRUN=20
.OPTION FREQSMP=150K BE
.PLOT TRAN V(5)
.END
```

# Chapter 15

## Working with S, Y, Z Parameters

---

### Introduction

A set of commands in Eldo allow the user to extract the large signal S parameters (Scattering parameters), the Y parameters (Admittance) or the Z parameters (Impedance) in the frequency domain for a specified circuit. The circuit can have any number of ports.

Eldo enables the simulation of circuits including any number of N-port blocks described by a frequency tabulation of their S (Scattering), Y (Admittance) or Z (Impedance) parameters. It does so by reading the S, Y, Z parameter data from a Touchstone® format file.

### Simulation Setup for S, Y, Z Parameter Extraction

Special sources must be added at each port of the circuit to be analyzed. The number of the port and the reference impedance for S parameters must be specified in the Eldo control file as follows:

#### Source Syntax

```
VYY NP NN IPORT=VAL [RPORT=VAL] [CPORT=VAL] [LPORT=VAL] [MODE=KEYWORD]
VYY NP NN IPORT=VAL ZPORT_FILE=string [CPORT=VAL] [LPORT=VAL]
+ [MODE=KEYWORD]
IYY NP NN IPORT=VAL [RPORT=VAL] [CPORT=VAL] [LPORT=VAL] [MODE=KEYWORD]
IYY NP NN IPORT=VAL ZPORT_FILE=string [CPORT=VAL] [LPORT=VAL]
+ [MODE=KEYWORD]
```

#### Parameters

YY	Name of the port.
NP	Name of the positive node.
NN	Name of the negative node.
IPORT	This is a strictly positive number that is unique and is used as the port number: this number is used for naming the outputs (for instance, .PLOT AC S(1,2)). An error message will be issued if two port instances have the same value for IPORT, or if an IPORT is missing (e.g. maximum IPORT number found in the netlist is 4, and there is no instance with IPORT 3).
RPORT	Value of the Reference Impedance in Ohms. Default value is 50Ω.

**CPORT** Capacitor placed in series with **RPORT**. Defaults to 0, in which case it behaves like a zero voltage source (i.e. **CPORT** would have no effect).

**LPORT** Inductor placed in series with **RPORT**. Defaults to 0.

**ZPORT\_FILE**

Specifies the Touchstone file name that contains the port source with a complex impedance from which the S parameters will be extracted from.

**MODE=SINGLE | COMMON | DIFFERENTIAL**

Mixed-mode S parameter selection.

**SINGLE** specifies the port as single ended, it is dedicated to S parameter extraction. Default.

**COMMON** and **DIFFERENTIAL** specify that the port is not single ended. Such ports are split into two linked sources that are either common (same amplitude and same phase) or differential (same amplitude but opposite phases). During S parameter extraction a “not single ended” port is equally common and differential depending on which display is required. During simulation (DC, AC or TRAN) this port is either common or differential depending on the specified mode keyword.

---

**Note**



Port numbers in **VYY** instances should range from 1 to the total number of ports without discontinuity. The simulation parameters **FMIN**, **FMAX**, and Number of frequency points for the analysis are specified with a **.AC** command.

---

## S, Y, Z Parameter Extraction

Once the simulation has been setup, see [page 15-1](#), S, Y, Z parameters can be extracted during simulation by use of the **.PRINT** and **.PLOT** commands as described below:

### For S Parameter Extraction

```
.PLOT  AC  SR(i, j)
.PRINT AC  SI(i, j)
.PRINT AC  SM(i, j)
.PLOT  AC  SDB(i, j)
.PRINT AC  SP(i, j)
.PRINT AC  SGD(i, j)
```

### For Mixed Mode S Parameter Extraction

Mixed mode S parameters can be extracted using the following syntax:

```
S[mn]TYPE(i, j)
```

**TYPE** can be one of the following:

- R** Real part
- M** Magnitude
- I** Imaginary part
- DB** Magnitude (dB)
- P** Phase
- GD** Group Delay

**mn** specifies the mode of ports **i** and **j** respectively, can be one of the following:

- cc** common-common
- dd** differential-differential
- dc** differential-common
- cd** common-differential
- sc** single-common
- sd** single-differential
- cs** common-single
- ds** different-single
- ss** single-single. Default.

The default mixed mode for S parameter extraction is single-single. If no mixed extension is specified on the output the default will change depending on how ports 1 and 2 are setup. The default rule is shown in [Table 15-1](#).

**Table 15-1. Default Rule**

Port 1	Port 2	Default	Available quantities
Single	Single	<b>ss</b>	<b>S[ss](1,1)</b> <b>S[ss](1,2)</b> <b>S[ss](2,1)</b> <b>S[ss](2,2)</b>
Single	Balanced	<b>SD</b>	<b>Sss(1,1)</b> <b>Ssd(1,2) Ssc(1,2)</b> <b>Sds(2,1) Scs(2,1)</b> <b>Sdd(2,2) Sdc(2,2) Scd(2,2) Scc(2,2)</b>
Balanced	Balanced	<b>DD</b>	<b>Sdd(1,1) Sdc(1,1) Scd(1,1) Scc(1,1)</b> <b>Sdd(1,2) Sdc(1,2) Scd(1,2) Scc(1,2)</b> <b>Sdd(2,1) Sdc(2,1) Scd(2,1) Scc(2,1)</b> <b>Sdd(2,2) Sdc(2,2) Scd(2,2) Scc(2,2)</b>
Balanced	Single	<b>DS</b>	<b>Sdd(1,1) Sdc(1,1) Scd(1,1) Scc(1,1)</b> <b>Sds(1,2) Scs(1,2)</b> <b>Ssd(2,1) Ssc(2,1)</b> <b>Sss(2,2)</b>

The default can be set using the **.LIN** command, with syntax:

```
.LIN mixedmode2port=dd|dc|ds|cd|cc|cs|sd|sc|ss
```

## Example

```
V1 in1 in2 iport=1 MODE=single
V2 in4 in5 iport=2 MODE=differential
.PLOT AC Sdb(1,1) Sscm(1,2) Ssdp(1,2) Sddr(2,2) Scdi(2,2)
```

## For Y Parameter Extraction

```
.PLOT AC YR(i, j)
.PRINT AC YI(i, j)
.PRINT AC YM(i, j)
.PLOT AC YDB(i, j)
.PRINT AC YP(i, j)
.PRINT AC YGD(i, j)
```

## For Z Parameter Extraction

```
.PLOT AC ZR(i, j)
.PRINT AC ZI(i, j)
.PLOT AC ZM(i, j)
.PLOT AC ZDB(i, j)
.PRINT AC ZP(i, j)
.PRINT AC ZGD(i, j)
```

Where  $S_{xx}(i, j)$ ,  $(Y_{xx}(i, j), Z_{xx}(i, j))$  give the influence of port  $j$  on port  $i$ .

## Matrix Parameter Extraction

Once the simulation has been setup, see [page 15-1](#), G, H, T, A parameters can be extracted during an AC or FSST simulation by use of the **.PRINT** and **.PLOT** commands as described below:

## For G Parameter Extraction

```
.PLOT AC GR(i, j)
.PRINT AC GI(i, j)
.PRINT AC GM(i, j)
.PLOT AC GDB(i, j)
.PRINT AC GP(i, j)
.PRINT AC GGD(i, j)
```

## For H Parameter Extraction

```
.PLOT AC HR(i, j)
.PPRINT AC HI(i, j)
.PPRINT AC HM(i, j)
.PLOT AC HDB(i, j)
.PPRINT AC HP(i, j)
.PPRINT AC HGD(i, j)
```

## For T Parameter Extraction

```
.PLOT AC TR(i, j)
.PPRINT AC TI(i, j)
.PPRINT AC TM(i, j)
.PLOT AC TDB(i, j)
.PPRINT AC TP(i, j)
.PPRINT AC TGD(i, j)
```

## For A Parameter Extraction

```
.PLOT AC AR(i, j)
.PPRINT AC AI(i, j)
.PPRINT AC AM(i, j)
.PLOT AC ADB(i, j)
.PPRINT AC AP(i, j)
.PPRINT AC AGD(i, j)
```

Where  $S_{xx}(i, j)$ ,  $(Y_{xx}(i, j))$ ,  $Z_{xx}(i, j)$ ) give the influence of Port  $j$  on Port  $i$ .

## Output File Specification

```
.ffile S|Y|Z|G|H|T|A [SINGLELINE] FILENAME [HZ|KHZ|MHZ|GHZ] [RI|MA|DB]
```

### Parameters

- S** Specifies S (Scattering) frequency parameters tabulation.
  - Y** Specifies Y (Admittance) frequency parameters tabulation.
  - Z** Specifies Z (Impedance) frequency parameters tabulation.
  - G** Specifies G (Hybrid-G) matrix parameters tabulation.
  - H** Specifies H (Hybrid-H) matrix parameters tabulation.
  - T** Specifies T (transfer scattering) matrix parameters tabulation.
  - A** Specifies A (chain or ABCD) matrix parameters tabulation.
- FILENAME** Name of the file where the S, Y, Z, G, H, T and A parameters will be stored.

#### SINGLELINE

This enables the user to obtain the S-parameter file in single line format as shown below.

```
Freq S11 S21 S12 S22
```

**Hz** Specifies the units to be Hz. This is the default.

**KHz** Specifies the units to be kHz.

**MHz** Specifies the units to be MHz.

**GHz** Specifies the units to be GHz.

**RI** Specifies Real Imaginary storage format. This is the default.

**MA** Specifies Magnitude Angle storage format.

**DB** **MA** with magnitude in dB.

Two-port noise parameters NFMIN\_MAG, GAMMA\_OPT\_MAG, PHI\_OPT and RNEQ are automatically written to the specified output file when a **.NOISE** command is specified in the netlist and the circuit to be analyzed is a two-port circuit.

#### Examples

```
r1 1 2 100k
c1 2 0 10pf
v1 1 0 iport=1 rport=100
v2 2 0 iport=2 rport=20
.ac dec 10 1 100meg
.plot ac sdb(2,1)
.Ffile S sb1.par khz ri
```

In this example, the S parameters of an RC circuit are extracted between 1 Hz and 100MHz with 10 points per decade. The reference impedance is 100 for port1 and 20 for port2. The magnitude of S21 is plotted in dB, and the extracted S parameters are stored in the file *sb1.par* with the frequency in kHz. The data is stored in the form of the Real and Imaginary parts.

```
v1 1 0 iport=1 rport=50

R1 1 n1 1k
C1 n1 0 100p
R2 n1 2 1k
.Rc1 n1 0 100k

v2 2 0 iport=2 rport=50

.ac lin 21 1meg 21meg
.noise v(n1) V1 3

.plot noise rneq gopt bopt nfmin_mag
.ffcile Z Z.par kHz ma
```

In this example the Z parameters are being extracted between 1MHz and 21MHz with 21 analysis points. The extracted Z parameters are stored in the file *Z.par* with the frequency in kHz. The data is stored in the form of the Magnitude Angle. As the circuit is a two-port circuit and there is a **.NOISE** command specified in the netlist then the two-port noise parameters are also stored in the output file. The output file is shown below:

```

! Data from ffile_test
# KHZ Z MA R 5.000000E+01
!
1.0000000000000001E+03 1.8928908821086993E+03 -5.7202544106307627E+01
1.5913478969070338E+03 -8.9088186330215706E+01

...
7.5788046363417195E+01 -8.9956576643609125E+01 7.5788046363417209E+01
-8.9956576643609125E+01 1.0029250742358629E+03 -4.3338006361865542E+00

! Noise Data: Nfmin(dB) GammaOpt PhiOpt Rneq/R0
1.0000000000000001E+03 5.2661113010485536E+00 9.6890047604419338E-01
1.7851510536508837E+02 4.8497683522933400E+01

...
2.100000000000000E+04 2.8441528902446137E+01 9.0554147314407418E-01
1.7956971588917614E+02 3.5225984336136335E+03

```

In the following example the S parameters of the block TWO\_PORT\_SUBCKT are being extracted between 10kHz and 10MHz with 10 analysis points per decade. The Touchstone file *Z1.par* defines the complex impedance of the source at port 1. The extracted S parameters are stored in the file *subckt.par* with the frequency in kHz. The data is stored in the form of the Magnitude Angle.

```

Vin 1 0 iport=1 zport_file="Z1.par"
Xsub1 1 0 2 0 TWO_PORT_SUBCKT
Vout 3 0 iport=2 rport=50
.ac dec 10 10000 10meg
.plot sdb(1,2) sdb(1,1) sdb(2,2) sdb(2,1)
.ffcile S subckt.par KHZ MA

```

## Transient Simulation of Circuits Characterized in the Frequency Domain

### Introduction

Traditionally, high frequency circuits are characterized and simulated in the frequency domain. This is because of the difficulty of handling extremely short rise times of the order of picoseconds and the simplicity of frequency measurement.

Today with the technological advent in high frequency circuits, there is a vital need to simulate circuits in the time domain using electrical simulators. This is needed to simulate, for example,

a linear (lossy and maybe coupled) interconnection having a non-linear termination. Such problems may be solved in the frequency domain using harmonic balance. If we are interested in using pulse stimuli, the circuit must be analyzed in the time domain. Another example is the microwave simulation of passive elements, either discrete or integrated. A user defined passive element may be simulated by extracting its scattering (S) parameters using a standard Electro-Magnetic (EM) solver and then using this file as an input to an S-Model. This procedure enables the user to simulate this passive element in Eldo either with or without other linear or non-linear elements.

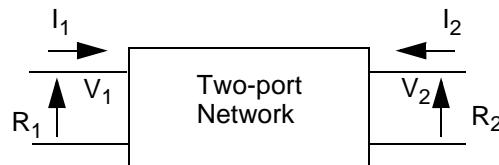
The main object of the S-Model GenLib library is to allow the transient analysis of pre-characterized (in the frequency domain) linear high frequency circuits with any other non-linear components. The circuit is usually characterized by its scattering parameters. S-Model also allows the simulation of circuits characterized using their admittance (Y) or impedance (Z) parameters. The pre-characterized circuit may have any number of ports. The following paragraph gives a brief technical presentation of matrix representation of linear circuits.

## Technical Background

A two port network is completely presented by its Z, Y, h or S matrix. As an example, consider the impedance Z matrix:

$$\begin{bmatrix} V_1 \\ V_2 \end{bmatrix} = \begin{bmatrix} Z_{11} & Z_{12} \\ Z_{21} & Z_{22} \end{bmatrix} \begin{bmatrix} I_1 \\ I_2 \end{bmatrix}$$

It is clear that this matrix relates the currents and voltages at the terminals of a given block:



An equivalent presentation is:

$$\begin{bmatrix} b_1 \\ b_2 \end{bmatrix} = \begin{bmatrix} S_{11} & S_{12} \\ S_{21} & S_{22} \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \end{bmatrix}$$

In this case, we use the S matrix or scattering parameters.  $\mathbf{a}_1$  and  $\mathbf{b}_1$  present the normalized incident and reflected waves at port 1.  $\mathbf{a}_2$  and  $\mathbf{b}_2$  present the corresponding waves at port 2. There are direct relationships between  $\mathbf{a}_1$ ,  $\mathbf{b}_1$ ,  $\mathbf{a}_2$ ,  $\mathbf{b}_2$  and the corresponding  $I_1$ ,  $V_1$ ,  $I_2$ ,  $V_2$ .

In the case of a two port network, the I and V as a function of a and b is given by the following:

$$\begin{aligned}V_1 &= (a_1 + b_1) / \sqrt{R_1} \\I_1 &= (a_1 - b_1) / \sqrt{R_1} \\V_2 &= (a_2 + b_2) / \sqrt{R_2} \\I_2 &= (a_2 - b_2) / \sqrt{R_2}\end{aligned}$$

For the same case, a and b as a function of I and V is given by the following:

$$\begin{aligned}a_1 &= \frac{V_1 + R_1 I_1}{2\sqrt{R_1}} \\b_1 &= \frac{V_1 - R_1 I_1}{2\sqrt{R_1}} \\a_2 &= \frac{V_2 + R_2 I_2}{2\sqrt{R_2}} \\b_2 &= \frac{V_2 - R_2 I_2}{2\sqrt{R_2}}\end{aligned}$$

where  $R_1$  and  $R_2$  are the reference impedances of ports 1 and 2 respectively.

The S-parameters are widely used to characterize high frequency circuits, mainly because they present no difficulty in measurements while the other parameters are difficult to measure. The scattering parameters are not unique; they are defined for a given reference impedance for each port. The reference impedance  $R_0$  is usually  $50\Omega$  for all ports to facilitate measurement (standard coaxial cable has  $50\Omega$  characteristic impedance).

In general, an n-port circuit has an  $n \times n$  scattering matrix of the following form:

$$\begin{bmatrix} b_1 \\ \vdots \\ b_n \end{bmatrix} = \begin{bmatrix} S_{11} & \dots & S_{1n} \\ \vdots & \ddots & \vdots \\ S_{n1} & \dots & S_{nn} \end{bmatrix} \begin{bmatrix} a_1 \\ \vdots \\ a_n \end{bmatrix}$$

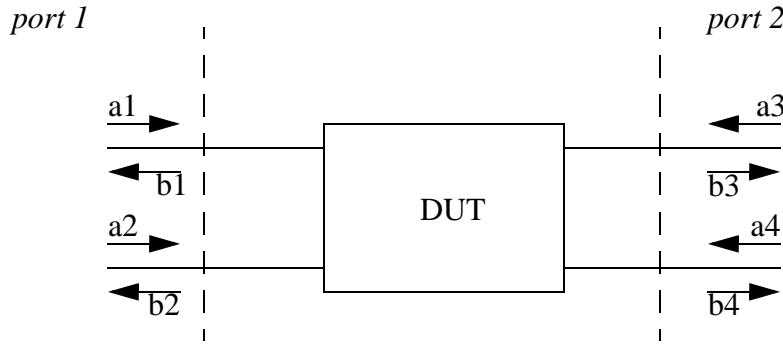
## Mixed-Mode S Parameters

Bockelman and Eidenstadt<sup>1</sup> developed a theory for combined differential and common normalized power waves (in terms of even and odd mode). Then it is now possible to characterize multiport networks at high frequencies, especially such device which are simulated by common-mode or differential-mode source, by using the extended S parameter definition. This adaptation, called “mixed-mode S parameter”, addresses differential and common-mode operation, as well as the conversion between the two modes operation.

---

1. David E. Bockelman William R. Eisenstadt, “Combined Differential and Common-Mode Scattering Parameters: Theory and Simulation” July 1995.

According with this new definition, we can see that a two port S parameters form a 4x4 matrix containing the mixed-mode S parameters (differential-mode, common mode and cross-mode S parameters). Consider the following differential circuit, each port can support the propagation of differential-mode and common-mode waves



The response of this differential circuits to a stimulus can be expressed with the mixed-mode S parameter matrix:

$$\begin{bmatrix} b_{d1} \\ b_{d2} \\ b_{c1} \\ b_{c2} \end{bmatrix} = \begin{bmatrix} S_{dd11} & S_{dd12} & S_{dc11} & S_{dc12} \\ S_{dd21} & S_{dd22} & S_{dc21} & S_{dc22} \\ S_{cd11} & S_{cd12} & S_{cc11} & S_{cc12} \\ S_{cd21} & S_{cd22} & S_{cc21} & S_{cc22} \end{bmatrix} \begin{bmatrix} a_{d1} \\ a_{d2} \\ a_{c1} \\ a_{c2} \end{bmatrix}$$

where the partition labeled  $S_{dd}$  are the differential-mode S parameters,  $S_{cc}$  are the common-mode S parameter, and  $S_{cd}$  and  $S_{dc}$  the cross-mode S parameters. The  $a_{di}$  and  $b_{di}$  are the normalized differential-mode stimulus and response waves;  $a_{ci}$  and  $b_{ci}$  are the normalized common mode stimulus and response waves. The definition of these normalized waves are:

$$a_{d1} = \frac{1}{\sqrt{2}}(a_1 - a_2)$$

$$a_{c1} = \frac{1}{\sqrt{2}}(a_1 + a_2)$$

$$b_{d1} = \frac{1}{\sqrt{2}}(b_1 - b_2)$$

$$b_{c1} = \frac{1}{\sqrt{2}}(b_1 + b_2)$$

$$a_{d2} = \frac{1}{\sqrt{2}}(a_3 - a_4)$$

$$a_{c2} = \frac{1}{\sqrt{2}}(a_3 + a_4)$$

$$b_{d2} = \frac{1}{\sqrt{2}}(b_3 - b_4)$$

$$b_{c2} = \frac{1}{\sqrt{2}}(b_3 + b_4)$$

In general, an n-port has a  $2n \times 2n$  mixed-mode S parameter matrix of the following form:

$$\begin{bmatrix} b_{d1} \\ \dots \\ b_{dn} \\ b_{c1} \\ \dots \\ b_{cn} \end{bmatrix} = \begin{bmatrix} S_{dd11} & \dots & S_{dd1n} & S_{dc11} & \dots & S_{dc1n} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ S_{ddn1} & \dots & S_{ddnn} & S_{dcn1} & \dots & S_{dcnn} \\ S_{cd11} & \dots & S_{cd1n} & S_{cc11} & \dots & S_{cc1n} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ S_{cdn1} & \dots & S_{dcnn} & S_{ccn1} & \dots & S_{ccnn} \end{bmatrix} \begin{bmatrix} a_{d1} \\ \dots \\ a_{dn} \\ a_{c1} \\ \dots \\ a_{cn} \end{bmatrix}$$

## Implementation Issues

Eldo uses three methods to simulate an S-Model given by S, Y, or Z parameters. These methods are briefly described below:

1. Complex Pole Fitting (CPF) technique is a method based on complex-pole fitting of the original dependence. During an initial “fitting” stage, the model’s given dependence is represented as a sum of simple first-order components, each one defined by its complex pole and residue. The result of fitting is re-usable; once generated, the list of poles and residues is stored in a \*.pls file and can be used repeatedly for simulations without the need to re-fit. This file has the same name and location as the original data (Touchstone) file.

The representation of frequency dependence created by fitting allows fast and accurate transient simulation of the S-Model. Simulation progresses linearly in time, using an effective, recursive, convolution-based algorithm. Although poles/residues can be used to build an equivalent circuit, this is not needed or recommended, for performance reasons: the model-evaluation time in CPF is less by a factor of 5-7 than that for the equivalent circuit built from the same poles.

Due to the very nature of fitting, the CPF method always results in a causal solution. It also has a delay-extraction capability useful when simulating transmission lines or connectors.

2. Digital Signal Processing (DSP) technique is an alternative approach that transforms the frequency-domain data into time domain parameters via inverse FFT, Hilbert transform and convolution. Important modifications were made to these basic algorithms to allow

both periodic and non-periodic dependencies, to ensure the causality of the system, to account for singularities in matrix representation and to ensure high-speed convolution.

If the circuit frequency response is naturally periodic (as for delay-containing operators) and is given only in a fraction of a single period, the DSP method is recommended to ensure accurate simulation. In this case, the last point given in the input data file should correspond to the half-period of the dependence.

3. System identification (SI) technique is a third method that represents S-parameters in the form of a rational function in ‘s’ (Laplace variable). These functions are then converted into systems of linear differential equations so that Eldo can solve them during the transient analysis.

With the above three methods, Eldo can efficiently solve a wide variety of problems. Frequency dependencies can be either quite smooth or with a large number of sharp resonant peaks (up to many hundred). The user may specify input data either with equidistant frequency points, starting from zero or not; or give them in any other way, (for example, logarithmically spaced) relevant to the method of data acquisition.

Of course, one can expect accurate simulations only if the original data is complete and accurate. The frequency dependence given should completely encompass the range of interest, from the lowest to the highest operation frequency. For example, high accuracy at DC is unlikely if the data starts from non-zero frequency. Similarly, accurate simulation of short transitions, lasting for hundred ps, is impossible if the highest point is far below tens of GHz. Also, the data points should be given with good resolution, sufficient to reproduce the shape of the dependencies. For example, many more points are needed to describe a dependence with many sharp peaks than for a smooth one, even if they are both defined in the same frequency range. Finally, the input data should be causal, so that the real and imaginary parts of the frequency dependence satisfy the dispersion Kronig-Kramers relation. In reality, data becomes slightly non-causal due to unavoidable measurement/simulation errors, especially (typically) at higher frequencies. However, serious measurement errors (like taking the negated phase) cause catastrophic non-causality that will lead to improper simulation results.

## Applications

S-parameters can originate from the following:

- Simulation of passive networks using an electromagnetic solver.
- Measurements from a passive network (interconnects, transmission lines, passive filters, etc.).
- Frequency-domain simulation of passive networks (AC analysis).
- Data sheets of active or passive devices.

The major applications of S-Model are as follows:

- Transient simulation of microwave linear circuits originally described by its response in frequency domain.
- Transient simulation of interconnects together with non-linear drivers/loads, either for VLSI or GaAs integrated circuits, using either their calculated or measured scattering parameters.
- Lossy transmission-line transient simulation in the presence of either linear or non-linear drivers/loads.
- Transient simulation of arbitrary (or user-defined) passive microwave circuits after extracting their S-parameters (using a standard electromagnetic solver).

## Functionality

### Basic Functionality

The S-model implemented in Eldo is a building block that makes possible DC, AC, and Transient simulation of circuits with any number of N-ports described by their S (Scattering), Y (Admittance), or Z (Impedance) parameters, in the form of tabulated data in the frequency domain.

The tabulated data is contained in an ASCII data file in the Touchstone® format. This format is briefly described on [page 15-18](#). When giving the instance of the model in the Eldo netlist file, the user can specify the data file either by setting an index associated with the file's name, or by explicitly defining the name of the file. The optimal of the three algorithms discussed above—Complex Pole Fitting (CPF), Digital Signal Processing (DSP), and System Identification (SI)—is selected by the internal Eldo monitor that allows great flexibility of simulation. The method can also be specified directly by the user.

### Detailed Functionality

- Simulation of N-ports with no restriction on N. N is determined automatically by the Eldo parser according to the number of pins.
- Any number of instances of the same model, any number of different models.
- All SST and MODSST simulation types are supported.
- DC Simulation.
- AC Simulation. When the CPF algorithm is chosen either by the monitor or the user, the frequency response of the system is computed based on the found poles/residues, to enable smooth and causal simulation results. In the case of the DSP method, AC simulation is performed by interpolation and extrapolation of the tabulated parameters, with the response made symmetrical around the maximal tabulated frequency point, and the obtained spectrum is made periodic. With the SI algorithm, AC simulation produces the frequency response of the transfer function that best fits the tabulated frequency

points. If AC simulation is performed without transient simulation and without any method forced by the user, the response is obtained through interpolation of the given frequency-domain data points.

- Transient Simulation. Transient simulation is obtained through the application of the most appropriate of the CPF, DSP, or SI algorithms.
- Input data can be specified as S, Y, Z, G, H, T or A parameters.
- Tabulated data may have linear, logarithmic, or irregular distribution in its frequency range. Frequency values may start from 0 Hz or any positive value. Any number of points is allowed.
- The choice of the CPF, DSP, or SI algorithm can be forced through a parameter of the model. All algorithms allow any kind of spacing. However, since internally DSP requires linearly spaced  $K^2+1$  data points starting at zero frequency, it uses interpolation and extrapolation of the input data to satisfy this requirement.
- Simulation is possible with the Eldo default options.
- For an N-port block ( $N>1$ ), port impedances can be identical, or different for each port.
- Speed-optimized C-FAS model.

## Instantiating a Block Defined by S-Parameters

```
.MODEL FBLOCK MACRO LANG=C
YNAME FBLOCK PARAM:
+ [ M=VAL ]
+ [ IDX_M=VAL ]
+ [ NO_DELAY=VAL ]
+ [ GROUPFIT=VAL ]
+ [ SYMMETRY=VAL ]
+ [ FORCE_PASSIVITY=VAL ]
+ [ FORCE_REFIT=VAL ]
+ [ EXTRAP_TO_DC=VAL ]
+ [ POLE_REDUCTION=VAL ]
+ [ HIGH_PRECISION=VAL ]
+ [ MAXCOL=VAL ]
+ [ MAXROW=VAL ]
+ [ IDX_F=VAL ]
+ STRING: FILENAME
+ PIN: IP1 IN1 ... IPN INN
```

The first line (**.MODEL**) is the reference to the C-FAS model, where the entity is called **FBLOCK**. The other lines are the model parameters.

---

### Note

This **.MODEL** line is optional. This declaration is not a requirement of Eldo when referencing embedded Eldo FBLOCK models.

---

The keyword **PARAM:** precedes the list of parameters, (each one shown in brackets as they are all optional). Most of the options, except for **M**, **IDX\_M** and **NO\_DELAY**, are specific to the CPF method.

**M** is a device multiplier parameter, simulating the effect of multiple S-block elements in parallel. Default value is 1.

**IDX\_M** is a parameter that forces a specific algorithm to be used (**IDX\_M=0** forces the CPF algorithm, **IDX\_M=1** specifies DSP, **IDX\_M=2** specifies SI). The default value of **IDX\_M** is 0 (CPF).

**NO\_DELAY** is used to allow or prevent delay extraction in the CPF or DSP methods. **NO\_DELAY=0** allows delay extraction, **NO\_DELAY=1** forbids it. The default value is 0.

**GROUPFIT=1** is used in CPF to force group fitting instead of individual for every matrix component. As a rule, with this option, fitting requires less effort but this might compromise accuracy. By default, its value is 0 that corresponds to individual fitting.

**SYMMETRY=0** disables the default assumption (**SYMMETRY=1**) made in CPF on the fitting stage that the original S (or Y or Z) matrix is symmetric. Matrix symmetry is a valid assumption as long as the S-model describes a reciprocal subcircuit. We cannot simply rely on symmetry of the matrices in the input data. Very often, the input matrices generated by field-solvers or measured from reciprocal systems, are not strictly symmetric, however they should be handled as symmetric.

**FORCE\_PASSIVITY=1** enables passivity enforcement on the fitting stage of the CPF method. The default value is 0 (no passivity enforcement). Passivity enforcement is recommended for all passive devices. Without this option, even for the passive system, fitting may result in a non-passive model if the input data is defined within a limited frequency range. If non-reciprocal linear active devices (such as amplifiers or filters) are to be simulated, both **SYMMETRY** and **FORCE\_PASSIVITY** should be disabled.

**FORCE\_REFIT=1** forces fitting in CPF regardless to whether the corresponding pls file is already present or not. This might be needed if we want to redo fitting with different options, such as **FORCE\_PASSIVITY**. However, the user should be careful in using such option, it should be disabled after the desired fit is built. By default, **FORCE\_REFIT** is disabled.

**MAXROW=VAL** sets the limit (**val/2**) to the frequency points of the original dependence used in fitting. By default, **MAXROW=40000**, that corresponds to 20,000 points.

**MAXCOL=VAL** sets the limit (**val/2**) to the maximal order of complexity for fitting in CPF. By default, **MAXCOL=1500**, that corresponds to a order of complexity of 750. For very complicated (sharp, irregular) dependencies it is sometimes reasonable to reduce the order of complexity, especially if we have reasons not to entirely trust the input data at higher frequencies. As a rule, reducing order of complexity is a better strategy than reducing the number of points to consider (**MAXROW**).

**HIGH\_PRECISION=1** increases fitting accuracy by allowing more poles than in regular mode (with default value 0). This option can be useful for verification purposes, for example if a “reference solution” is required. However, it is not recommended if the input data itself is not very accurate. Also, since high-precision fitting produces more poles, it makes simulation slower.

**POLE\_REDUCTION=1** (default value) enables the mode of transient simulation in which some of the fitted poles (that are too fast, too slow or too small) are removed in order to speed-up the solution. This mode typically gives up to 30-50% reduction in solution time when the step of the transient solution is fixed. The decision about pole reduction is made from considering the solution step (pole is too “fast”), or the duration of the simulation interval (pole is too “slow”). Therefore, the set of actually used poles is defined “dynamically” from considering the parameters of the **.TRAN** command. The generated list of poles/residues (\*.pls) file remains unaffected. Pole reduction does not considerably affect the solution accuracy. However, if the precise simulation is needed, the option can be disabled by setting **POLE\_REDUCTION=0**.

**EXTRAP\_TO\_DC=1** restores a missing point at zero frequency (DC) by extrapolating the curve from low frequency points given in the Touchstone file. If the DC point is present in the input data, this option has no effect. Compared to the default case (**EXTRAP\_TO\_DC=0**) it allows, as a rule, to achieve better accuracy in DC simulation when the point at zero frequency is not given.

The keyword **STRING:** is to define the name of the touchstone file, containing the input data. Path definition is allowed. Another way of defining the data file is using the parameter **IDX\_F**. Note that the parameter **IDX\_F** should be defined under the keyword **PARAM:** together with all other parameters, not under the **STRING:** keyword. This parameter defines the index (integer number) VAL associated with the S parameter file (**IDX\_F=VAL** implies that the input parameter file is named **SBVAL.PAR**).

The  $2 \times N$  pins of the N-port model will be connected to nodes IP1 IN1 ... IPN INN.

Any model may be instantiated as many times as required with the same or different input data file.

Any **FBLOCK** instance will contribute to the global noise results of **.NOISE** and **.SSTNOISE**. If the Touchstone format file contains noise parameters then they will be used to compute the noise contribution, otherwise the simulator will use the Twiss formula, as defined below:

$$C_y = 2kt(Y + Y^H)$$

Where:

$C_y$  = Noise Correlation Matrix

$k$  = Boltzmann Constant

$t$  = Temperature

$Y$  = Y Parameter Matrix

$H$  = Hermitian Matrix (complex conjugate transpose)

The **FBLOCK** file parameter is searched with the same methodology as searching library files, see “[Searchpath priorities](#)” on page 10-135. This means that if the FBLOCK file is not found in the current directory, the library where the corresponding FBLOCK instance was found is searched first if FBLOCK was actually read from a library. If not found, the directories are searched in the order specified by the option **SEARCH**.

## Examples

```
.model dio D rs=4.68 bv=6.1 cjo=246p
.model Fblock macro lang=c
vin 1 0 dc 5 ac 1 pulse(0 5 1n 1n 1n 5n 10n)
rin 1 2 50

ytline Fblock param:
+ force_passivity=1
+ string: C:\s-parameterdata\lowpassfilter.s2p
+ pin: 2 0 3 0

dout 3 0 dio
.ac dec 10 1 10meg
.tran 10n 100n
.plot ac vdb(2) vdb(3)
.plot tran v(1) v(2) v(3)
.end
```

In the above example, we define a block **ytline**. By default, the CPF method runs. Delay extraction is allowed (if feasible), fit is set to individual, the model is assumed symmetric, passivity enforcement is set for pre-fit stage; refit, extrapolation to DC, and high precision flags are disabled, and the parameters **MAXCOL**, **MAXROW** are set to their default values, 1500 and 40,000 respectively. The file name is given in conventional form, by using the keyword “**string:**”.

```
.subckt sparam_2p p1 p2 grnd
.model Fblock macro lang=c
    y2port FBLOCK param:
+ idx_f=4
+ idx_m=1
+ no_delay=1
+ pin: p1 grnd p2 grnd
.ends sparam_2p
```

In this example, a block **y2port** refers to an S-parameter file **sb4.par** (since **idx\_f=4**). Here, the S-block is described as a sub-circuit. We choose the DSP method and prevented delay extraction.

## Touchstone® Data Format

The Touchstone data format file is an ASCII text file in which data appears line by line: N lines for each data point of N ports. The data points are stored in increasing order of frequency.

The first of these N lines consist of a frequency value and N pairs of values for S, Y, Z, G, H, T or A parameters.

The (N-1) following lines contain N pairs of values.

Values are separated by one or more spaces or tabulations.

Example of S parameters for three ports:

```
F  SR11  SI11  SR12  SI12  SR13  SI13
    SR21  SI21  SR22  SI22  SR23  SI23
    SR31  SI31  SR32  SI32  SR33  SI33
```

---

### Note

 Two ports may also be represented in single line format but will have a different parameter order (notice that **s12** and **s21** are swapped), see below.

Two ports on a single line:

```
Freq S11  S21  S12  S22
```

Two ports on dual lines:

```
Freq S11  S12
      S21  S22
```

---

Comment lines begin with an exclamation mark (!). The first un-commented line in the file must be a specification line. An optional specification line begins with the number symbol (#) followed by a space. Then, several optional parameters are specified in the following order:

- Frequency Unit (Hz, kHz, MHz, GHz). Default value is Hz.
- Parameter type (S, Y, Z, G, H, T, A). Default value is S.
- Data format (MA, DB, RI). Default value is RI. MA means Magnitude in Volts, and the phase in degrees. DB means Magnitude in dB, and phase in degrees. RI means Real and Imaginary parts.
- Reference impedance of each port (when all the ports have the same reference impedance, only one may be specified). Default value is all ports with the same 50 Ω reference impedance.

```
# [Hz|kHz|MHz|GHz] [S|Y|Z|G|H|T|A] [RI|MA|DB]
+ [R Val|R1 Vall ... Rn Valn]
```

- The two-port noise parameters (NFMIN, GAMMA\_OPT\_MAG, PHI\_OPT, RNEQ) can be used when the user has specified a **.NOISE** command in the netlist and when the circuit to be analyzed is a two-port circuit. NFMIN is the minimal noise figure of the

two-port. GAMA\_OPT is the magnitude of the optimal reflection coefficient associated with the minimum noise figure. PHI\_OPT is the angle of the optimal reflection coefficient associated with the minimum noise figure. RNEQ is the equivalent noise resistance.

## Example

```
# khz s ri r 50
```

Frequency values are in kHz, the data are S-parameter data, they are stored in the format Real and Imaginary part and the reference impedance is  $50\Omega$  for each Port.

## Mixed Mode S Parameter Extraction

When extracting mixed mode S parameters the contents of the Touchstone output data file will change. For example the file header for a 2-port network may appear as follows:

```
! Data from foo
! S11 = SDD11
! S12 = SDS12
! S13 = SDC11
# HZ S RI  R1 1.000000E+01  R2 1.000000E+00
```



# Chapter 16

## Speed and Accuracy

---

### Introduction

This chapter describes the algorithms in Eldo and their control options. Most of the information relates to the transient analysis, as it is the most time-consuming analysis, and also the most frequently used analysis.

The most relevant trade-off that users are interested in the speed/accuracy trade-off. Circuit-level transient simulation is indeed the numerical resolution of an algebraic differential system of equations. As such, it is not an exact procedure (unlike the linear AC or NOISE analyses), and many ‘switches’ can be used to control the accuracy of the results. Usually, improved accuracy comes with an increased CPU time.

Eldo includes three different algorithms, namely NR (Newton-Raphson), OSR (One Step Relaxation) and IEM (Integral Equation Method). Each of these algorithms has its own set of properties.

- NR is the most general and robust algorithm, and it is always used by default. It is very accurate, and applicable to all kind of circuits, without restriction. NR is used by the vast majority of ‘SPICE’ commercial simulators, because of its generality.
- OSR is efficient for the analysis of large, loosely-coupled circuits, typically digital CMOS. It is reasonably accurate, and the CPU time grows almost linearly with the size of circuit, whereas the CPU time growth rate of NR is super-linear. However OSR works really well only if the loose-coupling assumption is verified, thus it is much less general than NR. For example OSR is not effective with bipolar circuits. Further details on OSR can be found in the [OSR algorithm](#) section of this chapter.
- IEM is yet another completely different algorithm, unique to Eldo. It is useful when very high accuracy is desired and/or when NR shows stability issues. Some components cannot be formulated in a way that is compatible with IEM, thus it is also less general than NR. Further details on the use of IEM can be found in the [IEM algorithm](#) section in this chapter and also the [Integral Equation Method \(IEM\)](#) chapter.

Unless explicitly triggered, by the user with commands and/or switches in the netlist, neither OSR nor IEM are used by Eldo. By default, only NR is used, for the whole circuit.

Eldo is also able to partition a circuit into different parts that are simulated with different algorithms. For example, some partitions can be simulated with classical NR, whereas others are simulated with OSR or IEM. Each partition can use its own accuracy control parameters.

## Speed and Accuracy in Eldo

Eldo has numerous control parameters to choose the most appropriate speed/accuracy compromise. This chapter attempts to present these parameters, covering the most important ones, and some of the less important ones.

The system of equations that represent the behavior of a circuit cannot be solved analytically, apart from trivial cases. Thus a simulator has to use numerical algorithms to find an approximate solution.

In the case of transient analysis, the problem to solve is to find the solution of a DAE (Differential Algebraic Equations) system. To simplify, and deliberately using loose notations, the ‘solution’ that the simulator tries to find is a set of N voltage waveforms  $v_n(t)$ , where n ranges from 1 to N, N being the number of nodes in the circuit, and t represents the time variable. These voltages are the solution of  $f(v(t), q'(v(t)), t)=0$ , where  $f()$  and  $q()$  are non-linear functions. Equation  $f()=0$  is nothing but the expression of the Kirchoff law, i.e. the sum of all currents entering a node is null, at any time. Function  $q()$  models the dynamic part of the circuit, i.e. the generally bias-dependent charges in the circuit.

This system is differential and non-linear. To solve it numerically, time is discretized, and the equations are solved at discrete time points. The time-derivatives are approximated using a so-called integration scheme or method, using current and previous values of the solution  $v(t_i)$ ,  $v(t_{i-1}), \dots, v(t_{i-m})$ . The number m of previous time points used to approximate the time-derivatives depends on the ‘order’ of the integration method. In all cases, an approximation error is introduced in this process.

Once the time-derivatives have been eliminated, the system to solve is ‘simply’ non-linear. Many numerical methods exist to solve this numerically. They are usually requiring a certain number of iterations, starting from an initial guess  $v_0$ , and each iteration providing, hopefully, a better estimate  $v_j$  of the solution  $v_{\text{exact}}$ . This iterative process will normally converge. Some criterion are needed to decide when to stop the iterative process and accept the last estimate as the ‘solution’ at the current time point. Again an approximation error is introduced here. One of the commonly used methods to solve non-linear systems of equations is the so-called Newton-Raphson method. It is commonly used mainly because of its generality and also for its relative robustness.

As a summary, mainly two types of errors are involved in the resolution of the system:

- Errors due to the numerical integration process
- Errors due to the numerical resolution of non-linear equations

## Integration methods

As explained before, the process of ‘eliminating’ the time-derivatives in the original circuit equations is a source of error. These time-derivatives are replaced with finite differences, which

only approximate the true time-derivatives. One of the most basic approximation scheme is the Backward-Euler method, which approximates  $v'(t_i)$  using the finite difference  $(v(t_i) - v(t_{i-1})) / (t_i - t_{i-1})$ . Intuitively, it is easy to understand that the smaller the time step  $h = t_i - t_{i-1}$ , the smaller the error. More sophisticated schemes exist, which provide less error with the same time step. For example the so-called trapezoidal and Gear methods are such methods.

The numerical resolution of differential equations is a vast subject, and there exists abundant literature about the subject. Dozens of methods have been proposed and studied in depth, some of them having very attractive properties, particularly allowing to use very large time steps without running into stability issues. However, in the context of circuit simulation, many of these methods are not practically applicable. The differential equations that model an electrical network (either IC or PCB) dynamics have unfortunately ‘bad’, undesirable characteristics. For example, they are implicit (in most if not all formulations used in commercial simulators) and very often ‘stiff’ (which means that the individual time constants involved in the system can routinely exhibit orders of magnitude differences). The cost of evaluation of the non-linear functions is also very high. This unfortunate situation leaves very few good practical candidates as integration methods.

Eldo implements three methods, namely the simple Backward-Euler (BE) method, the trapezoidal method (TRAP) and the Gear method (GEAR). The trapezoidal method is the default in Eldo. It provides a very good speed/accuracy compromise. TRAP and GEAR are both second-order method, whereas BE is a first-order method. The accuracy of BE is theoretically one order of magnitude worse than TRAP or GEAR, so it is usually reserved for cases where speed is the most important criterion, and accuracy can be somewhat sacrificed. In many cases, the speedup obtained with BE is not ‘spectacular’, although the loss in accuracy can be significant. This is because the GEAR and TRAP methods are still relatively simple methods (the derivative approximations are not too complicated). Thus the accuracy improvement with TRAP or GEAR over BE is generally worth the extra cost of CPU time.

Although TRAP and GEAR have similar theoretical accuracy, they still have their own characteristics. TRAP has the very undesirable tendency to generate numerical ‘ringing’, particularly on the current variables. Ringing shows up as obviously non-physical oscillations of small (or large!) amplitude riding on top of a correct and accurate average value. The oscillations have the rhythm of the time steps, they don’t belong to the circuit intrinsic time constant(s). They usually dampen over time, if the simulator properly controls the time step. This does not happen systematically with all test cases, but it is a well-known artifact of the TRAP method. All simulators, including Eldo, implement some code that tries to eliminate or reduce these oscillations, but sometimes, it may be very difficult to get rid of them entirely.

From the user point of view, there are not many options. Clamping the maximum allowed time step is usually the most radical solution. However this may slowdown the whole simulation a lot. Increasing the accuracy may also help.

If these oscillations happen, and they are a problem (for example because the testbench attempts to measure currents with high accuracy), then the solution is to switch to the GEAR method. Much cleaner current waveforms will be produced by GEAR. However, everything has a price,

and the price here is that GEAR has its own undesirable property to artificially damp natural oscillations of the circuit. Thus, for the analysis of pure oscillators, or when trying to identify local or global instability problems in a design, GEAR is not recommended, because it will tend to artificially ‘stabilize’ any circuit. BE is even worse with that respect.

If you want to get a feeling of how these methods behaves with respect to damping of natural oscillations, try the following: create a pure parallel LC network between a node you will initialize at 1Volt (using a .IC statement) and the ground (0). Pick L and C and the transient analysis duration T so that you can see about one hundred periods of the waveform ( $T=100.\sqrt{L\cdot C}\cdot 2\pi$ ). Try it with TRAP, then with GEAR, then with BE.

## Time Step Control

As explained in the background section, the resolution of the system of equations uses discrete time steps, and errors are unavoidable in all realistic cases. Time step control designates the set of methods used by the simulator to select these time steps, so that the accuracy of the solution is maintained within predefined tolerances.

### Time Step Control Algorithms Overview

To simplify, the time step can be controlled in three different ways. Eldo can use either:

- Local Truncation Error control (LTE)
- Rate-of-change control (DVDT)
- Iteration Count control (IC)

Some variants on these schemes are also available and detailed below, but these are the main three strategies.

By default, Eldo controls the local truncation error (LTE), and determines the time steps it can take based on estimations of this error. When a solution at time  $t$  has been accepted, to progress in time, Eldo will compute the value  $h$  of the largest time step it can take while still maintaining an acceptable LTE. Note that this is just a guess. The solution at time point  $t+h$  is predicted using the previous solutions at the previous time points. This serves as the initial guess. Then Eldo tries to achieve convergence at the new point  $t+h$ . If convergence cannot be achieved, the time step is reduced using a smaller time step  $h'$  ( $h' < h$ ), and a new resolution is attempted. If convergence is achieved, an estimate of the LTE is computed. If it is acceptable, the solution at time  $t+h$  is accepted, and a new cycle begins. If the error is found to be too large, then the time step is reduced in a way that should satisfy the truncation error constraint, and a new solution is computed. This process is reiterated until a time step and solution satisfying the truncation error criterion is found.

Thus when using truncation error control, there are two reasons why the time step may have to be reduced in the course of the transient simulation. Time step reduction can occur either because the Newton iterations do not converge, or because the estimated error on the time-

derivative expression is too large. These two dimensions (Newton convergence and LTE) are usually highly intricate.

To monitor this, a very useful option of Eldo is the NEWACCT option. If .option NEWACCT is set, Eldo reports interesting (and readable) statistics about the iteration counts, average number of iterations per time step, number of time step rejections etc. This can be used to double-check whether the simulator runs ‘normally’ or not. These statistics are reported at the very end of the output .chi file. Eldo also reports some statistics in the original SPICE style, but the newacct reporting is much more readable and useful.

Convergence failure in the Newton iterations has several possible reasons. The initial guess may be simply too distant from the solution. This might happen if the chosen time step is ‘over-optimistic’ or a sharp change in the circuit’s state occurs within the time step. Note that the simulator anticipates sharp changes in the stimuli, and all ‘break’ points such as the edges in PWL or PULSE signals force coinciding time points. Another reason for convergence failure can be discontinuities in the model equations, or simply strong non-linearities.

Controlling the local truncation error (i.e. the error made while approximating the time derivatives with finite differences) is the most conservative and rigorous way to control the time steps. It will usually provide more reliable and accurate results. This is the method selected by Eldo by default.

The rate-of-change control method uses the rate of change of the voltages to control the time steps. The idea behind this method is to control the time steps used so that the voltages do not change ‘too fast’. It is simpler than the LTE method, but also less accurate. There is no direct general relationship between the rate of change of the voltage and the actual truncation error, at least not under all conditions. The method can however provide accurate results if the rate of change is forced to remain small enough.

The iteration count method attempts to control the time step by monitoring only the rate of convergence of the Newton iterations. The idea being that if convergence is obtained rapidly, with just a few iterations, it probably means that the initial predicted guess was ‘good’, and conversely if many iterations are required, it probably means that the guess was incorrect. There is no attempt to estimate the truncation error. The control is entirely indirect, though the monitoring of the iteration count. This method is the least reliable of all and not necessarily any faster.

## Time Step Control – Algorithm Selection with the LVLTIM parameter

To select one of the time step control methods which we described previously, the optional parameter LVLTIM is used. It can be set as a ‘.option’ in the netlist:

```
.option lvltim=1|2|3|4
```

**Note**

If you are not truly comfortable with these notions, you should simply leave everything by default.

---

- **LVLTIM=0**

With **LVLTIM=0**, Eldo controls the time step based on the rate of convergence of the Newton iterations. This is called “Iteration Count” (IC) control. The time step control algorithm in this case is really simple and only depends on three parameters, namely **FT**, **ITL3** and **ITL4**. The default values of these control parameters can be altered with .option statements. The algorithm is as follows:

If convergence is obtained in less than **ITL3** iterations, then the time point is accepted, and the next time step is increased by a factor 2 at most. If convergence is obtained in less than **ITL4** iterations then the time point is accepted, and the next time step is kept unchanged. If more than **ITL4** iterations would be needed, the time step is reduced (divided by **FT**), a new prediction is made, and then a new attempt to reach convergence begins.

As apparent from the above, no estimation of the truncation error is calculated. This algorithm is usually faster, but it is also the less reliable.

Default values are **FT=8**, **ITL3=3**, **ITL4=8**.

**Note**

If you decide to experiment with **LVLTIM=0**, it is highly recommended to start with the default values of **FT**, **ITL3** and **ITL4**, and to be careful when altering these parameters.

---

- **LVLTIM=1 DVDT=0**

This triggers the rate-of-change time step control. The time step is adjusted depending on the parameters **ABSVAR** and **RELVAR** and not on the local truncation error estimation. The time step is controlled so that no voltage changes by more than **ABSVAR** (in absolute value) nor by more than **RELVAR** (in relative change). The default value of **ABSVAR** is 200mV. The default value of **RELVAR** is 0.15.

- **LVLTIM=1 DVDT=-1**

This hybrid variant actually uses the same algorithm as in the case of the default **LVLTIM=2**, except that time steps are not rejected when truncation error is too large. In other words, truncation error estimation is used only to predict the time step that can be used, but if the iterations converge, the time step is accepted without LTE control. The simulation may be faster though less accurate.

- **LVLTIM=2**

This is the default time step control algorithm. It is by far the most reliable algorithm, and yields the highest accuracy. In this mode, Eldo estimates the local truncation error

(LTE) and adjusts the time step accordingly. The LTE estimate is used for both time step prediction and rejection control.

A prediction of the largest time step is made, based on the LTE estimation of the previous time point. If convergence cannot be reached with this time step, the time step is reduced (divided by **FT**), a new prediction is made, and then a new attempt to reach convergence begins. If convergence is reached, the time point is not accepted right away. Instead, a new estimation of the LTE is calculated. If the LTE is within the tolerance, the time point is accepted (i.e. all state variables are updated, and Eldo proceeds with the next time point). If the estimated LTE is too large, the time step is rejected, and a smaller time step is retried.

The truncation error can only be estimated, and the estimation algorithms are different depending on the integration algorithm (BE, TRAP or GEAR). It happens that many times, the truncation error estimate provided by the estimation algorithm/formula is significantly larger than the exact error (this has been experimented with simple circuits for which the differential equations can be solved analytically, and thus the exact error can be computed). Thus, in most circuit simulators – and Eldo makes no exception - a correction factor can be used to compensate this ‘over-estimation’ and not force unnecessary small time steps.

The compensation of the truncation error is controlled with the **TRTOL** parameter. This parameter can range from 1 to 7 (the default value is 7). As **TRTOL** is made smaller, the estimated truncation error is made larger, and thus the time steps are globally reduced.

Note than in 99% of cases there is absolutely no reason for you to play with **TRTOL**. The default value is correct for the vast majority of circuits and conditions.

In general, the acceptable LTE threshold is controlled by the global EPS parameter. More on LTE control can be found in the [Control of the Local Truncation Error \(LTE\)](#) section.

- **LVLTIM=3**

With **LVLTIM** set to 3, the time step is controlled in the same way as for **LVLTIM=2**, i.e. LTE estimates are used to predict and control the time steps. However, additional time points are calculated based on the **TPRINT** value in the **.TRAN** statement.

For example if using:

```
.option lvltim=3
.TRAN 1N 1U
```

the time step control is based on the same LTE considerations as with **LVLTIM=2**, but additional time points are forced every 1ns, regardless of any LTE considerations.

Obviously, if the **TPRINT** parameter of the **.TRAN** statement is very small, this will slow down the simulation significantly compared to **LVLTIM=2**, but will probably improve the accuracy (there are not many examples where computing extra time points

can reduce accuracy). If on the other hand, the TPRINT is anyway larger on average than what is required from an LTE point of view, this will have little impact on speed, and may be useful for post-processing or FFT purposes.

## Control of the Local Truncation Error (LTE)

When **LVLTIM** equals 2 or 3 (see previous section), LTE estimates are used to monitor the time step selection. Truncation error is the part of the solution error which originates from the approximation of the time-derivative terms (for example, in the case of a simple capacitor, the  $dv/dt$  term in the  $I=C.dv/dt$  equation) with finite-differences expressions. LTE can be estimated and monitored using the  $dv/dt$  terms or the  $v$  term itself.

When using LTE control to determine the acceptable time steps, Eldo may use the voltage quantities and/or the charge/flux quantities.

By default Eldo uses voltage quantities for LTE estimates. There are however situations where LTE on charge/flux is used:

- If **OPTION QTRUNC** is specified, LTE will be computed on charges/flux, not on voltages. This corresponds to the way SPICE operates. This is sometimes more efficient and/or accurate for PCB applications, but often seems rather indirect to IC designers.
- If Eldo detects ‘PCB-like’ devices such as those shown in the list below, then LTE will be computed on charges/flux:
  - Current sources above 1A
  - Voltage sources above 500V
  - Inductors above 0.1H
  - Negative capacitors
  - Magnetic core devices

If option QTRUNC is set, LTE is computed using charges and flux quantities. In this case parameters (resp.) NGTOL, CHGTOL and FLUXTOL designate the absolute tolerances on (resp.) voltages, charges and fluxes, and RELTRUNC is the associated relative tolerance.

## More about time step control

Eldo can also ‘clamp’ the time step to both a minimum value and a maximum value.

To clamp the maximum time step you want to allow, use the HMAX parameter. Eldo will do all its normal time step control as explained previously, but still not take any step larger than HMAX. Use this option carefully, as setting HMAX to a smaller value than required may force long simulation times. Unless very clear suspicion exist that the result is corrupted because the selected time steps are too large, this option should not be used.

To clamp the time step to a minimum value, use the **HMIN** parameter. This is a rather dangerous option, because it basically prevents Eldo to use the time step it should use to maintain the accuracy within the specified tolerances. Unexpected results or failures are possible if using an inappropriate **HMIN** value. The default value for **HMIN** is 1ps. Unless you are desperately looking for speed improvements, it is best not to play with **HMIN**.

## Using a fixed time step

A fixed internal time step can be specified by the **STEP** parameter; this may be useful when simulating certain classes of circuits (e.g. switched-capacitor circuits), however, care must be taken. Overall, it is not safe, and may cause excessively long simulation times.

If the requested **STEP** value is unreasonably small, the simulator may be fooled into believing that some nodes are not changing and will set them into “latency.” A badly chosen fixed time step can thus sometimes have unwelcome results.

A **STEP** value that is too large may similarly cause problems. Consider a bipolar circuit when a large transition of an input signal occurs between two steps. This will nearly impossible to handle for the simulator.

Overall, using this option in the case of regular IC circuit analysis is highly discouraged.

## Newton Iterations Accuracy Control

The main parameters controlling the accuracy of the Newton iterations are **RELTOL**, **VNTOL**, **ABSTOL** and **CHGTOL**

Whenever Newton iterations are used to solve the non-linear system of equations, the following convergence criterion for voltages, currents and charges are used:

For node voltages:

$$|V(i) - V(i-1)| < \text{RELTOL} * \max(|V(i)|, |V(i-1)|) + \text{VNTOL}$$

where  $V(i)$  is the value at current iteration  $i$  and  $V(i-1)$  the value of the previous iteration.

For branch currents:

$$|I(i) - I(i-1)| < \text{RELTOL} * \max(|I(i)|, |I(i-1)|) + \text{ABSTOL}$$

where  $I(i)$  is the value at current iteration  $i$  and  $I(i-1)$  the value of the previous iteration.

For charges:

$$|Q(i) - Q(i-1)| < \text{RELTOL} * \max(|Q(i)|, |Q(i-1)|) + \text{CHGTOL}$$

where  $Q(i)$  is the value at current iteration  $i$  and  $Q(i-1)$  the value of the previous iteration.

The same RELTOL parameter controls the relative tolerance for voltages, currents and charges. However, when voltages, currents or charges become ‘small’ in absolute value, a relative tolerance becomes useless, thus absolute tolerances are required. Of course the typical orders of magnitude of voltages, currents and charges are quite different, thus the necessity to have specific absolute tolerances (VNTOL, ABSTOL, CHGTOL).

As a rule of thumb, if using the default settings, a simulation that goes well should use at most three or four Newton iterations per time step. This can be less if the circuit is almost linear, or if the settings such as the RELTOL value are relaxed. If many more iterations are needed, it means that either the time step control options have been relaxed a lot (and potentially the truncation error is large), or there are strongly non-linear characteristics in the circuit, which are difficult to solve. The number of Newton iteration will also increase if the RELTOL value is reduced to obtain more accuracy.

## Global Tuning of the Accuracy—EPS

A global parameter, **EPS**, is used as a general controller to set the required accuracy of a simulation. When this parameter is changed, a collection of lower-level parameters are adjusted accordingly, affecting both the Newton convergence tolerances and the time step control tolerances.

It is always possible to set these low-level parameters individually. However using EPS guarantees that the adjustments to the low-level parameters are consistent, which is not always easy to achieve for beginners or users not too familiar with circuit simulation tricks...

The parameter adjustments induced by EPS are shown in the table below (the global TUNING parameter which appears in the first line of the table is explained in the next section).

**Table 16-1. Global Tuning of Accuracy**

TUNING			STANDARD			ACCURATE		VHIGH	
<b>EPS</b>	1e-1	1e-2	<b>1e-3</b>	1e-4	1e-5	<b>1e-6</b>	1e-7	<b>1e-8</b>	1e-9
<b>RELTOL</b>	5e-2	5e-3	<b>7.5e-4</b>	5e-4	2.5e-4	<b>1e-4</b>	5e-5	<b>1e-5</b>	1e-6
<b>VNTOL</b>	1e-6	1e-6	<b>1e-6</b>	1e-6	1e-6	<b>1e-6</b>	1e-7	<b>1e-8</b>	1e-9
<b>CHGTOL</b>	1e-9	1e-9	<b>1e-14</b>	1e-14	1e-14	<b>1e-15</b>	1e-16	<b>1e-18</b>	1e-18
<b>FLUXTOL</b>	1e-11	1e-11	<b>1e-11</b>	1e-11	1e-11	<b>1e-11</b>	1e-11	<b>1e-11</b>	1e-11
<b>ITOL</b>	1e-3	1e-3	<b>1e-3</b>	1e-3	1e-3	<b>1e-3</b>	1e-3	<b>1e-3</b>	1e-3
<b>ABSTOL</b>	1e-12	1e-12	<b>1e-12</b>	1e-12	1e-12	<b>1e-12</b>	1e-13	<b>1e-14</b>	1e-15
<b>RELTRUNC</b>	5e-2	5e-3	<b>7.5e-4</b>	5e-4	2.5e-4	<b>1e-4</b>	5e-5	<b>1e-5</b>	1e-6
<b>CHGTRUNC</b>	1e-9	1e-9	<b>1e-14</b>	1e-14	1e-14	<b>1e-15</b>	1e-16	<b>1e-18</b>	1e-18

\* EPS=1e-3 is the default value. It is increased to 5mV if the lowest voltage source found in the circuit is above 5 Volts.

\*\* When set indirectly through EPS, RELTRUNC always gets the same value as RELTOL. RELTRUNC is only used if LTE is controlled on charges/flux, i.e. when .option QTRUNC is set.

\*\*\* When set indirectly through EPS, CHGTRUNC always gets the same value as CHGTOL. CHGTRUNC is only used if LTE is controlled on charges/flux, i.e. when .option QTRUNC is set.

Setting EPS automatically changes the low level parameters shown in the table above. However it is still possible to override some of them. For example, if using “.option eps=1e-3 reltol=1e-8”, RELTOL will be set to 1e-8, instead of 1e-3. VNTOL will be set to 1e-6, indirectly through EPS. This would not be too consistent probably, but it is still allowed.

It is important to notice that RELTOL, which is one of the main parameters defining the global accuracy, does NOT scale linearly with EPS.

## Global Tuning of the Accuracy—TUNING

Another global parameter, **TUNING**, may be used as a general controller to define the accuracy of a given simulation. With TUNING, the user does not provide numerical values for the accuracy control switches. Instead, a qualitative flag is specified for the desired accuracy, and similarly to the effect of EPS, a number of adjustments are activated. Actually, ‘accuracy’ in this context should be understood as ‘speed/accuracy compromise’.

The TUNING parameter can take four values, namely FAST, STANDARD, ACCURATE and VHIGH.

The names are self-explanatory. FAST can be used when the number one concern is to accelerate a simulation, and you are ready to sacrifice some accuracy to achieve this.

The FAST setting is equivalent to EPS=1e-3, overridden with VNTOL=10e-6, ABSTOL=100e-12, RELTOL=1.25e-3 and CHGTOL=1e-12.

Thus FAST is not entirely equivalent to any given value of EPS. It is an adequate setting if you are not too worried about picoAmps and nanoVolts accuracy, and you want mostly a short runtime.

STANDARD, ACCURATE and VHIGH are completely equivalent to specific values of EPS (see table in the previous section)

STANDARD actually corresponds to the Eldo default settings, i.e. is equivalent to setting EPS=1e-3. If you do not set any option in the netlist file, this is what Eldo will use. Thousands of test cases covering all possible IC technologies have shown that it was the best compromise

for what Eldo tries to achieve by default, i.e. reliable and accurate results in the shortest CPU time.

Finally ACCURATE and VHIGH (which stands for Very HIGH accuracy) will alter the tolerance switches to achieve higher degrees of accuracy, usually to the expense of longer simulation times of course. ACCURATE is equivalent to setting EPS=1e-6, and VHIGH is equivalent to EPS=1e-8.

The VHIGH setting must be used carefully, as it can lead to excessively long simulation times. It is however sometimes necessary, particularly for the cases of the startup phase of sensitive oscillators.

Using EPS settings smaller than 1e-9/1e-10 is usually un-reasonable and not recommended.

Setting the TUNING flag simply uses the .OPTION mechanism. For example:

```
.option TUNING=ACCURATE
```

## IEM algorithm

With IEM, the differential system is first transformed into a system of integral equations, which is then transformed into an algebraic system by series expansion. The truncation error results from the finite number of terms in the series. In all cases, truncation error is also a function of the time step size. The IEM method is described in more detail in a dedicated chapter [Integral Equation Method \(IEM\)](#).

Once the non-linear algebraic system is obtained, it is solved by an iteration loop. IEM targets improvements of the accuracy of the solution (compared to Newton). It does not specifically target large circuits, so it is mostly used for cell or macro-block simulations where accuracy is critical.

IEM is interesting for cases where high accuracy is required and/or when the default Newton method runs into numerical stability issues due to the integration methods (trapezoidal ringing for example). IEM however does not support all devices, macro-models etc. that Newton supports. Some devices cannot be efficiently formulated in a way that is compatible with the IEM algorithm. Thus the applicability of IEM is somewhat limited. It is mostly used for cell characterization applications. The [Integral Equation Method \(IEM\)](#) chapter describes the limitations of the IEM method.

## Integration method

When using IEM, the notion of integration method like BE, TRAP or GEAR, is irrelevant. The equations are cast to an integral form prior to the resolution. Thus there are no choices about a numerical integration method to use.

## Time step control

When using IEM, the time step control algorithm uses local truncation error (LTE) control. Only the LVLTIM=2 and LVLTIM=3 methods are selectable together with IEM. The other methods are not reliable enough when used together with IEM, and they will be refused by Eldo.

## Accuracy control

When using IEM, the same accuracy control parameters as those used for the default Newton method are available. Thus mainly the RELTOL and ABSTOL parameters will control the accuracy of the resolution

## OSR algorithm

The One Step Relaxation (OSR) algorithm of Eldo is dedicated to the simulation of large MOS circuits showing weak local couplings (large-scale feedback loops are not a problem). It is a unique algorithm to Eldo. Its primary usage is the fast transistor-level simulation of large digital CMOS circuits, for which the weak local coupling assumption is generally valid. When the conditions for its applicability are met, its speed and accuracy are excellent. In many cases, it is just as accurate as the Newton method, but much faster, and also the growth of the CPU time with the circuit size is almost linear, which allows simulating larger networks. Although its capacity is still less than that of fast-SPICE timing simulators such as Mach TA, it provides a quite interesting point in the speed/accuracy plane.

OSR is not particularly efficient with tightly coupled analog circuits. Better results will be obtained using either the default Newton-Raphson or IEM.

OSR can be activated explicitly or indirectly. The most straightforward way to activate OSR is to use “.OPTION OSR”. If this option is set, Eldo will attempt to simulate the whole circuit with OSR. However, if the circuit contains elements that prevent OSR from being effective, or if its connectivity is such that OSR will fail, Eldo will revert back to the default Newton algorithm, for part or all of the circuit. A warning message is then displayed.

Actually, Eldo is able to simulate a circuit with certain parts handled by OSR, and other parts handled by Newton. More details on this possibility are given in the section ‘Combined OSR/Newton simulation’

Additional information related to each of the cases above can be found in the appropriate sections of the [Simulator and Control Options](#) chapter.

## OSR and the notion of latency

When OSR is used, Eldo places the nodes for which the surrounding nodes do not show any voltage variation greater than **EPS** volts into ‘latency’. Thereafter, Eldo effectively bypasses

the calculation of such nodes. This allows significant CPU time savings, especially for large digital circuits where quite often only a fraction of the nodes are changing over one time step.

Due to the formulation of OSR, latency exploitation is very natural and effective. Latency is much more complicated to control in a reliable way in the context of the regular Newton method.

Even with OSR, latency and above all ‘wake-up from latency’ is however always tricky. When slowly varying signals are applied to a circuit with high capacitances, Eldo may not detect any voltage variation within one time step, resulting in nodes being placed in latency. Depending on the tolerances (EPS), this may cause incorrect simulation results. In case latency is potentially the source of problems, the **.OPTION NOLAT** command may be used to suppress the use of latency. If this option is set, Eldo will not attempt to use latency, and will solve all nodes, whatever their activity.

## Integration method

When using OSR, the same integration methods as for the case of Newton (namely BE, TRAP and GEAR) can be used (using **.OPTION METHOD=**).

## Time step control

When using OSR, the default time step control algorithm is the same as in the case of Newton (LVLTIM=2), and thus it uses local truncation error (LTE) control. Only the LVLTIM=1, LVLTIM=2 and LVLTIM=3 methods are selectable together with OSR. The other method (LVLTIM=0 is meaningless in the context of OSR (there are no Newton iterations when using OSR...)).

## Accuracy control

The accuracy control when using OSR is much simpler than with Newton. The global parameter **EPS** is used to control everything. Furthermore, the actual value of **EPS** (in Volts) is directly used for the convergence control.

Accuracy of the relaxation loop and the inner loop is directly controlled by **EPS**.

- Relaxation loop is stopped when:  
 $|V(i_{relax}) - V(i_{relax-1})| < \text{EPS}$  for all nodes
- Inner loop  
 $|V(iter) - V(iter-1)| < \text{EPS}/50$  on the current node

## Combined OSR/Newton simulation

Eldo can ‘partition’ a circuit so that some blocks are simulated OSR and other blocks are simulated using the standard Newton algorithm. Typical candidate examples would include circuits with large digital CMOS blocks driving and/or driven by analog blocks such as voltage regulators, bandgap references, high-gain operational amplifiers etc.

In these cases, the speed and capacity of OSR is used to handle the large weakly-coupled sections of the circuits, and the accuracy and ability to handle tightly coupled devices of Newton is used for the analog blocks.

The partitioning can be left entirely to the discretion of Eldo, or the user can try to help and indicate which blocks (subcircuits) must be simulated with OSR or Newton. In this latter case, the partitioning is always ‘indicative’ only, and Eldo may choose to alter the boundaries of the partitions to accommodate the requirements of the OSR algorithm.

There are several ways to trigger these mixed OSR/Newton simulations:

- **.OPTION BLOCKS=NEWTON**

This first partitioning method is automatic, based on the degree of coupling between devices. A global option is set (**.OPTION BLOCKS=NEWTON**). Eldo will identify the blocks consisting of tightly coupled devices, and place them in partitions that will be solved using the Newton method. The loosely coupled nodes are placed in the OSR partition. The global circuit is solved using a relaxation loop. ‘Simply’, the relaxation loop operates with individual nodes and also group of tightly coupled nodes (the Newton partitions). Note that the identification of tightly coupled devices is not based on the netlist hierarchy (.subckt etc.).

- Flag (**ANALOG**) given on **SUBCKT** definition

This partitioning methods ‘tags’ the subcircuit definitions. The subcircuit definitions tagged with (ANALOG) will be solved using the Newton method. This is useful when the netlist hierarchy does reflect the nature and coupling level of the blocks. All nodes which are not inside user-defined (**ANALOG**) **SUBCKT**, and which are connected only to MOSFET, grounded capacitances, or low-value floating capacitances (the threshold value for floating capacitance is set with **.OPTION CAPANW=**), will be solved by OSR, all other nodes being solved by Newton. See the subcircuit definition syntax “**.SUBCKT**” on page 10-306.

- **.OPTION NWBLOCK** [**RELTOL=** ] [**VNTOL=** ] <list\_of\_nodes>

This method is the most ‘manual’ one. Each Newton block is defined explicitly, with a list of nodes. Hierarchical names and wildcards in such names are allowed. Optionally the required Newton accuracy can be redefined for each block, using the **RELTOL** and **VNTOL** parameters. See the description of the **.NWBLOCK** option in the [Simulator and Control Options](#) for additional details.

- **.OPTION SWITCH**, or some (**SWITCH**) flag on some subcircuit calls.

If the SWITCH option is used or if some subcircuit calls are tagged with a (SWITCH) flag, then the Eldo-XL algorithm will be activated if possible. Please refer to the *Eldo-XL* documentation for further details regarding the switch models, Eldo-XL and its relation with OSR. Please note that this technology (Eldo-XL) is not enhanced in any aspect any longer.

With any of the partitioning methods described above, OSR is ‘implicitly activated’. OSR becomes the default method, and Eldo (or the user) defines what still has to be simulated with Newton.

Whenever OSR and Newton are activated simultaneously, the (resp.) OSR and Newton rules for accuracy control apply to the (resp.) OSR and Newton partitions. For example the RELTOL and VNTOL parameters still define the accuracy of the nodes inside the Newton partitions, whereas EPS defines the accuracy of the OSR nodes.

## Simulation of Large Circuits

The Eldo circuit simulator is capable of handling very large circuits. To handle such circuits, containing several hundreds of thousands of components, memory usage becomes an important consideration.

For more information on memory requirements and dimensioning, please refer to the *AMS Installation Guide*.

For the simulation of large circuits, the following suggestions may help.

- Explore the manual...

There are indeed several dozens options and features which can significantly impact speed, and help with the simulation of large circuits. Some of them are extremely specific and relevant only in the presence of such or such element. A good place to start is the [Simulator and Control Options](#) chapter, which lists all available .OPTION, grouped by categories. Particularly the [Simulation Speed, Accuracy & Efficiency Options](#) section complements the present chapter with reference data.

- Use workstation with sufficient RAM and SWAP.

As with any software, an attempt to run a simulation with real memory far lower than that required to contain the circuit can result in a phenomena known as “thrashing.” When this occurs, CPU time is almost totally taken up in swapping data in and out of main memory, with little or no useful computation being made. Any tool that monitors process activity will report CPU usage, real memory usage and swap usage. When running large simulations, it is highly recommended to use of these tools to monitor the simulation processes.

- Use the 64 bit versions of Eldo.

If more than 2 GB of real memory is needed, consider using the 64 bit versions of Eldo. This is activated by using “`eldo -64b -i circuit.cir`” on the command line.

- Specify node number via **MAXNODES** and **MAXTRAN** options.

If you have an idea of the number of nodes and components (grounded capacitors not included) your circuit has, it is wise to specify these options:

```
.OPTION MAXNODES=VALUE MAXTRAN=VALUE
```

In this case, Eldo directly allocates the correct amount of memory, reducing the requirement of using the expensive C function `realloc()`.

- Collapse the intrinsic MOS transistor nodes

When simulating large circuits containing mostly MOS transistors, the number of nodes grows very quickly with the number of MOS itself. By default, Eldo creates explicit internal nodes for the drain and source parasitic access resistors. Thus each MOS transistor with non-zero access resistors adds two internal nodes to the system. Although these nodes do not connect to anything else, and the sparse techniques used to solve the matrices greatly reduce the impact of these additional anodes upon the CPU time, they still contribute to create a much larger system. For example if 50,000 MOS are instantiated, the typical size of the circuit could be 30,000 or 50,000 ‘true’ nodes, but grows to 130,000 or 150,000 with two internal nodes per MOS transistor. Even for Eldo, this is quite a change in problem size...

There exists an option to ‘collapse’ these resistors into the drain current calculation, thus avoiding explicit creation of these intrinsic nodes. This option is **NONWRMOS**. With this option, the effect of the parasitic resistors upon the drain current is still taken into account, although not as accurately as when the nodes are explicitly created.

Particularly, some of the overlap capacitances in the MOS model become connected to the external drain/source, whereas they really are connected to the internal nodes. This may change results slightly.

However, experience has shown that many times the accuracy degradation is barely measurable, particularly when these resistors remain ‘small’. So, when having to simulate huge circuits, this is definitely something to try. It is however recommended to ‘calibrate’ the accuracy degradation caused by the option with a reasonably small circuit first. And then to decide whether this degradation is acceptable or not for the large circuit.

This option, as its name indicates, only affects MOS transistors. It has no effect upon the handling of the internal base, collector and emitter nodes of bipolar transistors, neither upon diodes.

- Optimize the density of time points.

Is not always possible to directly alter the density of time points that the simulator must compute to maintain a given accuracy. However it is highly recommended to get familiar with some of the options which affect the number of time points. Particularly,

the OUT\_STEP, INTERP, FREQSMP and OUT\_RESOL might have very significant impacts. Detailed discussion of these options can be found in “[Limiting the Size of Transient Output Files](#)” on page 10-218.

- Use the Periodic Circuit Speedup (**PCS**) option.

Used to increase the simulation speed for circuits with periodic or nearly periodic nature. PLLs in near-lock state belong to this category. The amount of speedup depends on the design nature. The speedup will be more significant with relatively large circuits. With circuits showing no periodicity at all, the option will not usually provide any speedup. Periodic Circuit Speedup is invoked by setting **PCS=1|2** for BSIM3v3 models only (**PCS=1|2** represent two possible speed optimization methods, **PCS=1** is more recommended) or **PCS=3** for BSIM4 and BSIM3v3 models (with same speed optimization as **PCS=1**). Default is 0. This option only supports the BSIM3v3 and BSIM4 models.

BSIM4, unlike BSIM3, has many parasitic configurations: gate resistors, body resistors network, bias dependent access resistors, etc. These parasitic configurations are controlled by a set of instance and model parameters (Rdsmod, Rbodymod and Rgatemod). As the complexity of the parasitic configuration around the core model increases, the gain expected by the PCS decreases.

This option should not affect the accuracy of the results.

- Try suppressing DC analysis for large digital circuits.

This is achieved by setting the **UIC** option of the **.TRAN** command. Very large circuits are often digital circuits for which an accurate and time consuming DC analysis may not be necessary. The logical initialization performed by Eldo prior to any analysis should ensure that the transient analysis that follows will start with nodes correctly preset. Initial voltage conditions are specified using the **.IC** command.

- Decrease the accuracy for MOS digital circuits.

For MOS digital circuits, accuracy can often be decreased to 10mV instead of the default value of 5mV. This can save a large amount of CPU time.

- Employ the **.USE** command.

It may be useful to split the circuit into a number of blocks, find DC solutions for each block and then find the DC solution of the circuit as a whole using the separate solutions as a starting point.

# Tips for troubleshooting and/or improving convergence and performance

## Operating Point Calculation

Before any kind of analysis is carried out by Eldo, a DC or operating point for the circuit in question is usually carried out<sup>1</sup>. DC and operating point convergence times can vary greatly, depending on the type of circuit simulated. Eldo provides a number of commands which may be used to speed up the convergence process. Each command is briefly described below.

### **.IC V<NN>=VALUE**

When used, Eldo fixes the specified node voltages for the duration of the DC analysis. If the **UIC** parameter is also present (in the **.TRAN** command), no DC analysis is performed and the voltages are initialized as defined in the **.IC** command. All other voltages on nodes not initialized in the **.IC** command are determined by Eldo itself. During subsequent analysis (transient), the node voltages are freed of their initial values, and may therefore assume different values.

---

**i** See “[.IC](#)” on page 10-125 for further details.

---

### **.NODESET V<NN>=VALUE**

This command is used to help calculate the DC operating point by initializing selected nodes during the first DC operating point calculation. After the first calculation has been completed the node values are “released” and a second DC operating point calculation is started. This command is useful when the whereabouts of the DC operating point is known, enabling the simulator to converge directly to it and also for bistable circuits or circuits with more than one operating point.

---

**i** See “[.NODESET](#)” on page 10-179 for further details.

---

### **.GUESS V<NN>=VALUE**

Enables the user to set the initial voltages to specific nodes for the first iteration of a DC operating point run.

---

**i** See “[.GUESS](#)” on page 10-122 for further details.

---

1. Unless the **UIC** parameter is present in the **.TRAN** command.

## **.RAMP**

This command can be used when no DC operating point has been found using the above methods. Two ramping facilities are provided.

DC convergence algorithms are automatically used by Eldo when the standard DC algorithm fails to converge. The sequence of algorithms used is as follows: Newton, Gmin ramping, DC ramping, Transient ramping, PSTRAN, T° ramping, and DPTRAN. The DPTRAN algorithm is used as a last attempt at convergence.

---

### **Note**



See “[.RAMP](#)” on page 10-269, [.OPTION GRAMP](#) on page 11-33, [.OPTION PSTRAN](#) on page 11-56, and [.OPTION DPTRAN](#) on page 11-57 for further details.

---

## **.OPTION VMIN, VMAX**

By default, Eldo looks for the DC operating point within the limits of the circuit power supply. In the case of a circuit which contains elements that create energy, such as voltage or current controlled sources, it is possible that Eldo can look for solutions outside the expected bounds. If so, Eldo displays the message:

Searching operating point between [x1,x2]

The `x1` and `x2` limits can be controlled by the user in order to speed up the DC convergence process using the **VMIN** and **VMAX** parameters. It must be stressed, however, that unrealistic limits can also cause convergence problems.

## **.SAVE, .USE & .RESTART**

A circuit may need to be simulated several times. In each case, the operating point must be determined although it may be the same for each run or may be only slightly different; in any case a large change of operating point is unlikely. To avoid the wasteful re-calculation of the operating point, a system is provided to store and reload the result of the DC analysis from the simulator; this significantly increases performance since operating point calculation time is thereby reduced.

The save, use and restart system is based on the commands:

```
.SAVE [FNAME] DC | END | TIME=VALUE [REPEAT] [TEMP=VALUE] [STEP=VALUE]
.USE FNAME [IC | NODESET | GUESS]
.RESTART FNAME [FILE=TMPCOU]
```

All commands listed above must be placed in the circuit description file.

The **.SAVE** command, as its name implies, is used to save analysis data to a file.

The **.USE** command is intended to be used to load DC or operating point information for one or more parts of the circuit to be simulated; its application is thus in cases where a large or complex circuit is to be simulated and the DC or operating point data for parts of the circuit is already known. Multiple **.USE** commands may be used in a simulation.

The **.RESTART** command is different from the **.USE** command in several ways. Firstly, whereas the **.USE** information may apply only to part of the circuit, **.RESTART** data always applies to the whole circuit. In fact, the **.RESTART** command first checks that the node names specified in the file used to “restart” the circuit, exactly correspond to those specified in the circuit description file where the **.RESTART** command is placed. The **.RESTART** command may be used to either load the operating (DC) point data and thereafter perform a transient simulation, or it may be used to restart a transient simulation from a specific point where the simulation had been interrupted in an earlier run. Obviously, to carry out a transient analysis from a specific point, the **.TRAN** command parameters need to be changed to specify the new simulation times.

---

**Note**



You only need to change the **.TRAN** command if you want to change its STOP time.  
You don't need to change its START time.

---

The following sections describe the use of the above. First, here is a short synopsis.

```
.SAVE [FNAME] DC|END|TIME=VALUE [REPEAT] [TEMP=VALUE] [STEP=VALUE]  
.USE [FNAME]  
.RESTART [FNAME] [FILE=TMCOU]
```

**FNAME**      Filename, including extension, where the data is to be stored. If not specified, Eldo uses *<cirname>.iic* where *<cirname>* is the circuit filename.

**DC**      Save the DC data.

**END**      Save the complete simulation environment at the end of a transient analysis run, to allow a restart of the simulation from that point.

**TIME=VALUE**  
Save the simulation status after the specified CPU time has elapsed.

**REPEAT**      Save the simulation status after each interval of CPU time has elapsed, as specified in **TIME=VALUE**. The save file is overwritten each time a run is saved. This is similar to the **.OPTION SAVETIME=VALUE** command, except that the latter command is in simulation time as opposed to elapsed time units.

**TEMP=VALUE**  
Save the simulation status only when simulation temperature is equal to the value provided as parameter to **TEMP**. This is useful when sweeping through several temperatures.

**STEP=VALUE**

Save the simulation status when the **STEP** parameter value, which is being swept as the result of a **.STEP** command, equals the value specified in the **.SAVE** command.

**FILE=TMPCOU**

Binary output file. If the restarted simulation is to continue writing binary output data to a *.cou* file, appending the new simulation data to the end of the previous *.cou* file, then the previous file must be renamed or else it will be lost. The **FILE** parameter of the **.RESTART** command provides the new filename, TMPCOU, for the old *.cou* file. The system then concatenates the old *.cou* file data with the new *.cou* data in the original *.cou* file.

## **.SAVE DC in Combination with RESTART**

The DC or operating point values for a circuit can be saved to a file and used later to greatly shorten this part of the analysis, when the circuit is re-simulated. This is achieved by employing the **.SAVE DC** and **.USE NODESET** commands, provided that circuit node names remain the same. The following steps are used, assuming that the circuit is stored in the circuit description file *<cname>.cir*.

1. Add the following line to the circuit description file:

```
.SAVE <FNAME> DC
```

If no filename *<fname>* is given, the default used is *<cname>.iic*.

2. Find the operating point of the circuit. This is stored in the file as specified in the **.SAVE** command.
3. Now change the circuit description file to include any other analysis required. Do not specify **dc**, **uic** or operating point calculation. Add to the circuit description file one of the lines:

<b>.RESTART &lt;FNAME&gt; NODESET</b>	to use <b>.NODESET</b> or alternatively
<b>.RESTART &lt;FNAME&gt; IC</b>	to use the <b>.IC</b> system

The simulator will attempt to use the operating point information stored in the *<FNAME>* file.



To find out the differences between **.NODESET** and **.IC** strategies, see the “Operating Point Calculation” on page 16-19.

## **.SAVE DC in Combination with USE**

Eldo provides a system of combining the DC values of parts of a circuit to speed up the operating point calculation of the complete circuit. The DC state of the circuit sections which

are available are stored in separate files. Such “circuit parts” may of course be whole subcircuits. In the general case, the DC data will be available on several files, called here <OPDATA\_SS>. To “use” such for the DC calculation of a complete circuit, add the following lines to the circuit description file, and for each <OPDATA\_SS> file:

<code>.USE &lt;OPDATA_SS&gt; NODESET</code>	to use <code>.NODESET</code> or
<code>.USE &lt;OPDATA_SS&gt; IC</code>	to use the <code>.IC</code> system

The simulator will attempt to use the operating point information stored in all the <OPDATA> files.



To find out the differences between `.NODESET` and `.IC` strategies, refer to “[.SAVE, .USE & .RESTART](#)” on page 16-20 of this chapter.

---

## **.SAVE END in Combination with RESTART**

It is possible to run a transient simulation and save the final state of the run. Thereafter, it is possible to restart the transient simulation from the time the last simulation was ended. To achieve this, the `.SAVE` command must be used in combination with the `END` parameter.

To restart the simulation, the circuit description file must be modified. The original `.SAVE` command must be removed and the `.TRAN` command must be modified to reflect the new end time. Furthermore, a `.RESTART` command must be included in the circuit description file.

`.RESTART <FNAME>`

Using the above procedure, the simulation will produce a new binary output file (`.cou` file) which contains data starting from the end of the last simulation. If it is desired to concatenate old and new `.cou` files, then the old `.cou` file should be renamed to say <TMPCOU> and the above `.RESTART` command modified as follows:

`.RESTART <FNAME> FILE=<TMPCOU>`

The system then concatenates the old `.cou` file data with the new `.cou` data.

## **More Sophisticated SAVE & RESTART Procedures**

The complete `.SAVE` syntax shown in the line below, includes provision to save the simulation state either periodically or at predefined times and/or temperature or sweep (`STEP`) parameter conditions:

`.SAVE [ FNAME ] {DC | END | TIME=VALUE} [ REPEAT ] [ TEMP=VALUE ] [ STEP=VALUE ]`

It is possible to save the result of the simulation periodically (with `REPEAT`), to save the run for a particular temperature (with `TEMP`) or for a particular swept parameter (with `STEP`).



The previous section contains a description of each of the parameters of the **.SAVE** command.

In each case, the rule to restart the simulation from a predefined state is the same. In each case, to restart the simulation, the restart command needs to be added to the circuit description file and, of course, the save commands contained therein should be removed.

The transient simulation parameters in the **.TRAN** command may also need to be modified to reflect the new desired simulation time stop.

## Aborting simulation and saving current state

If a transient simulation run is interrupted by the user with a Control-C, Eldo prompts the user, inquiring if the simulation status should be saved. If the user answers in the affirmative, the simulation data is saved in a file called *<cirname>.sav* where *<cirname>* is the name of the circuit description file with no extension.

If the user wishes to restart the software from this interrupted point, then the circuit description file should be modified to include the command:

```
.RESTART
```

The simulation will then be restarted automatically from the time it was interrupted. The **.TRAN** command needs to be updated. Options and/or stimuli might be changed as well.

# Chapter 17

## Integral Equation Method (IEM)

---

### Introduction

The Integral Equation Method (IEM) is a simulation algorithm developed for transient analog circuit simulation. IEM provides better performance than classic algorithms based on the Newton-Raphson method combined with a multi-step discretization scheme (e.g. Backward Euler, trapezoidal, gear, etc.).

Tests have shown IEM to be superior to Newton methods in terms of stability and accuracy, which has a favorable incidence on speed. The improvements appear particularly for circuits requiring high precision or those which exhibit tight coupling or stability problems.

IEM's performance enhancement derives from the use of a semi-analytic approach combined with efficient numerical techniques, to produce a new advanced simulation technology capable of removing the limits of the Newton-Raphson method.

### Application Domains

IEM is invoked for TRANSIENT analysis of ANALOG circuits only. DC and AC analyses can only be performed using Newton-Raphson methods.

Large, loosely coupled digital circuits are simulated (by default) using the One Step Relaxation (OSR) algorithm which is more efficient for these types of circuit. If, however, these circuits contain analog (tightly coupled) blocks, then these blocks can be simulated using IEM (or Newton) together with OSR for the remaining part of the circuit (see the “[Use of IEM, Tolerance Parameters](#)” on page 17-2).

In case Eldo is combined with another digital simulator through a backplane then IEM is not supported.

### Circuit Elements Supported—Limitations

As a rule, all circuit elements accepted by Eldo are supported by IEM except objects described by HDL-A, FAS or GUDM.

To be more explicit, elements actually supported by IEM are listed below.

- All electrical device models, except:
  - BNR HICUM bipolar transistor

- R, L, and C defined by the **VALUE** statement
- S and Z domain filters
- All independent or controlled sources are supported, except:
  - Controlled sources defined by either the **VALUE** or **TABLE** statements (Note: Linear and polynomial controlled sources *are* supported)
- All digital and mixed-signal macromodels are supported
- Only comparator and delay elements are supported among analog macromodels
- Only the switch element is supported among switched capacitor macromodels
- Accusim magnetic models are supported but not those of Eldo.

In case a circuit or block contains an element not supported by IEM, this circuit or block is simulated using Newton-Raphson and a warning message is issued.

## Use of IEM, Tolerance Parameters

The following simulation algorithms may be used on a circuit depending on the setting of the **.OPTION** line in the netlist and/or on the use of the (**ANALOG**) attribute on subcircuits:

<b>.OPTION NEWTON</b>	In this case the whole circuit is simulated in a single block, using the Newton method.
<b>.OPTION IEM</b>	In this case the whole circuit is simulated in a single block, using IEM.
<b>.OPTION BLOCKS=NEWTON</b>	In this case, the circuit is partitioned (whenever possible) into one or more blocks. Simulation inside each block is performed using the Newton method and relaxation is undertaken between blocks using OSR. OSR is also used to simulate the non-Newton part of the circuit.
<b>.OPTION BLOCKS=IEM</b>	In this case, the circuit is partitioned (whenever possible) into one or more blocks. Simulation inside each block is performed using IEM and relaxation is undertaken between blocks using OSR. OSR is also used to simulate the non-IEM part of the circuit.

---

**Note**



The default simulation option is Newton, i.e. **.OPTION BLOCKS=NEWTON**

---

If a block contains either a GUDM, HDL-A or FAS model, then simulation inside that particular block will be performed using the Newton method. If no IEM blocks are created (because all

blocks contain unsupported objects or the circuit is loosely coupled and can be simulated with OSR) then the IEM option is removed and a warning message is issued.

**RELTOL** and **VNTOL** parameters can be used in the same way as for SPICE-like simulators. These control both convergence of iteration loops *and* step size.

---

**Note**

 For Newton-Raphson, **RELTOL** does *not* control the step size.

---

If **RELTOL** or **VNTOL** are not explicitly specified in the netlist, then **EPS** can be used to control their values, as follows:

- $\text{RELTOL} = (1.0 \times 10^{-9} \times \text{EPS})^{1/4}$   
**RELTOL** is then limited in the range  $1.0 \times 10^{-2}$  to  $1.0 \times 10^{-5}$
- $\text{VNTOL} = (1.0 \times 10^{-7} \times \text{EPS})^{1/2}$   
**VNTOL** is then limited in the range  $1.0 \times 10^{-3}$  to  $1.0 \times 10^{-9}$

For IEM, the option **LVLTIM** is by default set to 2, i.e. truncation error is used to control the time step. It is possible to set **LVLTIM=3** in the netlist, which will also allow the truncation error to control the time step. Additionally, it will impose calculated points at **TPRINT** values in the **.TRAN** command. Other settings of **LVLTIM** are not allowed since they produce less accurate results which would be meaningless with IEM.

All other SPICE compatible options (e.g. **ABSTOL**, **ITOL**, etc.) behave in the same way as for Newton-Raphson. This of course does not include integration scheme options (**BE**, **TRAP**, **GEAR**, ...) which do not apply to IEM.

In order to maintain compatibility with previous Eldo versions and Newton usage, **EPS** may still be used in the Eldo sense, i.e. as a means to control all precision parameters.

## Efficient Usage of IEM

This section describes some details to help users achieve the best accuracy and performance results with IEM, used alone or in combination with OSR.

- IEM alone (**.OPTION IEM**)

As already discussed above, this setup allows IEM to simulate the whole circuit in a single block. This is required for all-analog designs, or at least when the analog part of the circuit is much larger than the logic circuitry (i.e. approximately 75% of devices or more). This option is particularly recommended for circuits requiring high precision and/or those manifesting stability problems. IEM intrinsically gives very good accuracy and doesn't require tightening of accuracy parameters. Hence, the default **EPS** value is usually enough to achieve high quality simulation results. Lowering **EPS** will give even higher precision, but at the expense of CPU time.

- IEM + OSR (**.OPTION BLOCKS=IEM**)

This setup allows Eldo to partition the circuit, assigning tightly coupled blocks into IEM, the remaining part being simulated with OSR. By default, MOS and grounded capacitors are assigned to OSR, hence *it is recommended to use this setup in conjunction with a correct imposition of the attribute (**ANALOG**) for relevant circuits*. This way, BJTs, resistors, and all MOS analog subcircuits will be simulated by IEM while other MOS blocks will be considered as logic ones and efficiently simulated by OSR (whenever possible).

This setup provides advantages of both methods and optimizes simulation of very large circuits where a meaningful section is made of digital MOS cells. Furthermore, it is possible of course to use XL acceleration procedures; MOS devices (the ones simulated by OSR) can be replaced by their linearized switch models imposing subcircuits as (**SWITCH**) or globally using the **USERSWITCH** option (this doesn't affect MOS in **ANALOG** blocks). With a small loss of accuracy on digital cells, a great gain in efficiency is achieved.

As usual, it is possible to increase accuracy by reducing **EPS**, however, as in all-IEM simulation this is not particularly recommended.

---

**Note**



The combination of IEM (a highly precise algorithm) with OSR (a fast but less precise algorithm) may not be justified in some cases.

---

## Examples for IEM

### Introduction

This section demonstrates the advantages of IEM, using the following set of examples:

- Stability
  - *bjtinv\_iem.cir* and *bjtinv\_nrm.cir*
  - *ivdd\_iem.cir* and *ivdd\_nrm.cir*

Newton-Raphson (**nrm**) versions are included for comparison with the *\*iem.cir* files.

- Accuracy
  - *oscil\_iem.cir*, *oscil\_nrm.cir* and *oscil\_ref.cir*
  - *moy1\_iem.cir*, *moy1\_nrm.cir* and *moy1\_ref.cir*

Newton-Raphson (**nrm**) and reference (**ref**) versions are included for comparison with the *\*iem.cir* files.

- Speed

- o *ladder\_iem.cir* and *ladder\_nrm.cir*
- o *opamp\_5pin\_iem.cir* and *opamp\_5pin\_nrm.cir*

Newton-Raphson (**nrm**) versions are included for comparison with the *\*iem.cir* files.

The listings for these examples can be found in the following directory included with your software:

```
$MGC_AMS_HOME/eldo/$eldover/examples/eldo
```

---

**Note**



For more examples of using Eldo please refer to the [Examples](#) appendix and the [Tutorials](#) chapter.

---

## Stability

IEM is inherently stable as opposed to, for example, the Trapezoidal scheme which may sometimes exhibit numerical instabilities. These instabilities can be suppressed by reducing tolerances. Backward-Euler scheme also offers numerical damping but tolerances should be greatly decreased in order to obtain an acceptable precision. In both cases, this increases the CPU time. For IEM, these numerical instabilities do not appear even at default precision. The following examples (*bjtinv\_iem.cir* and *ivdd\_iem.cir*) demonstrate this effect.

### Example 1—bjtinv

#### Complete Netlist

---

**Note**



The netlist is also provided as a Newton-Raphson version (*bjtinv\_nrm.cir*) in the examples directory for comparison. The netlist is identical to *bjtinv\_iem.cir* with the exception of **.OPTION NEWTON** selected as opposed to **.OPTION IEM**.

---

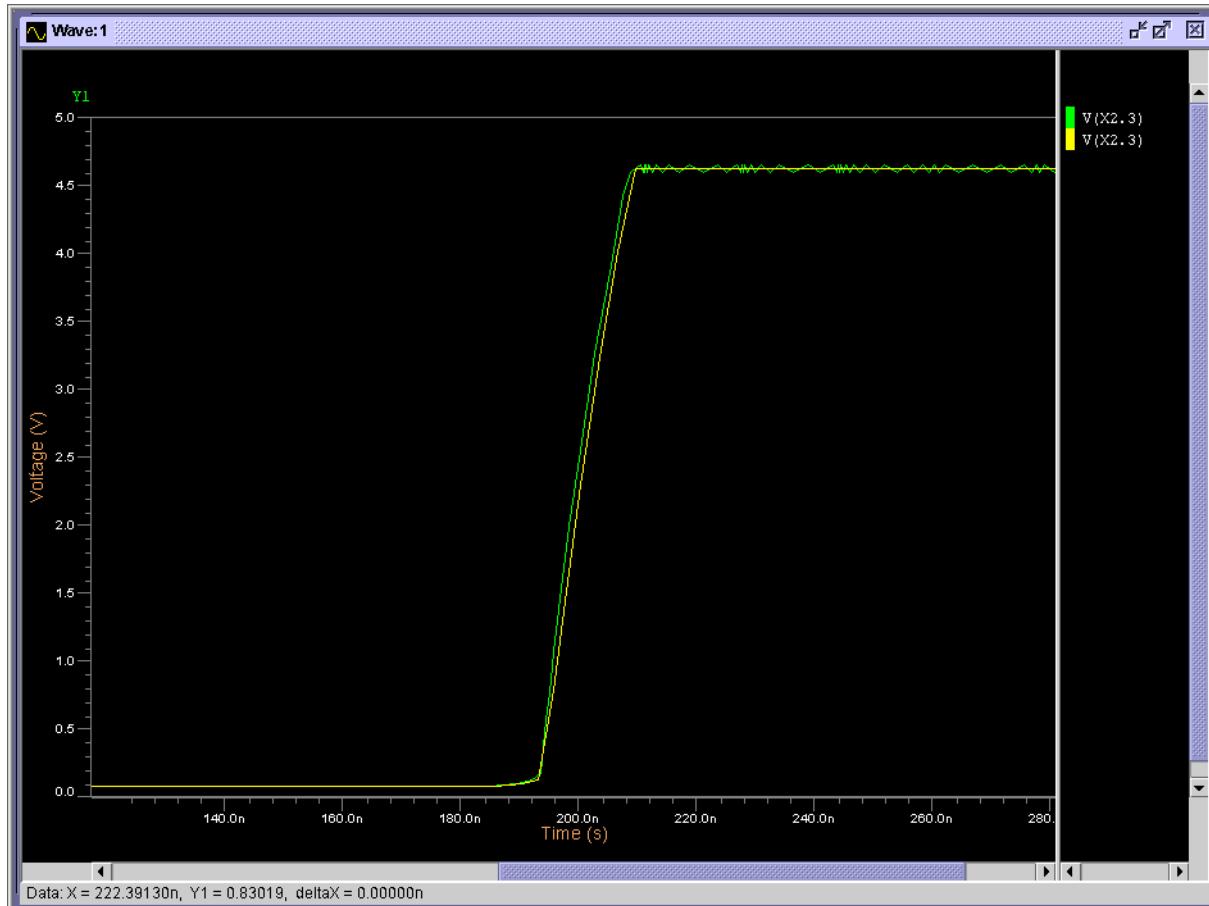
```
bjtinv_iem.cir
vcc 5 0 dc 5.
vin 1 0 pulse(0 5 10ns 40ns 40ns 110ns 300ns)
.subckt inv 5 1 6
q1 3 2 0 x33
q2 6 4 0 x33
rc1 5 3 1k
rc2 5 6 1k
rb1 1 2 10k
rb2 3 4 10k
.ends inv
x1 5 1 6 inv
x2 5 6 10 inv
x3 5 10 11 inv
x4 5 11 12 inv
```

```
x5 5 12 13 inv
x6 5 13 14 inv
.print tran v(1) v(11) v(14)
.plot tran v(1) v(x2.3) v(11) v(14)
.tran 2ns 300ns
.model x33 npn(bf=80 rb=100 va=50 tf=.3e-9 tr=6.e-9)
.option iem accusim2
.end
```

## Simulation Results

Figure 17-1 compares the waveform produced by the Newton-Raphson method (shown in green) compared with the waveform produced by the IEM method (shown in yellow).

**Figure 17-1. Example 1—bjtinv**



## Example 2—ivdd

**Note**



Included in this directory are *ivdd\_xxx.cir* examples which are intended for use with Eldo-ST. Eldo-ST is available for STMicroelectronics licensees only. Non-license holders may still simulate these examples but multiple warning messages will be generated.



Please refer to “[How to Invoke Eldo-ST](#)” on page H-1.

### Complete Netlist

**Note**



The netlist is also provided as a Newton-Raphson version (*ivdd\_nrm.cir*) in the examples directory for comparison. The netlist is identical to *ivdd\_iem.cir* with the exception of **.OPTION NEWTON** selected as opposed to **.OPTION IEM**.

```
ivdd_iem.cir
.option XA= 1.500000e-06
*-----* Main
Circuit Netlist:
*-----C25 Z GND
8.5188e-16
C24 Z VDD 8.1332e-16
C23 N2 GND 6.3258e-15
C22 N2 VDD 6.79986e-15
C21 N2 Z 4.05478e-15
C20 A GND 9.072e-16
C19 A VDD 8.3745e-16
C18 A N2 8.05086e-16
M9 VDD A N2 VDD EP3 W=1.05571e-05 L=5e-07 AD=1.14175e-11
+ AS=1.1425e-11 PD=3.3e-06 PS=1.31e-05
M16 Z N2 VDD VDD EP3 W=1.07571e-05 L=5e-07 AD=5.22e-12
+ AS=1.0495e-11 PD=1.3e-06 PS=2.7e-06
M14 Z N2 VDD VDD EP3 W=1.07571e-05 L=5e-07 AD=5.22e-12
+ AS=1.0495e-11 PD=1.3e-06 PS=2.7e-06
M12 Z N2 VDD VDD EP3 W=1.07571e-05 L=5e-07 AD=5.22e-12
+ AS=1.0495e-11 PD=1.3e-06 PS=2.7e-06
M10 Z N2 VDD VDD EP3 W=1.07571e-05 L=5e-07 AD=5.22e-12
+ AS=1.14175e-11 PD=1.3e-06 PS=3.3e-06
M17 VDD N2 Z VDD EP3 W=1.07571e-05 L=5e-07 AD=1.7695e-11
+ AS=5.22e-12 PD=1.59e-05 PS=1.3e-06
M15 VDD N2 Z VDD EP3 W=1.07571e-05 L=5e-07 AD=1.0495e-11
+ AS=5.22e-12 PD=2.7e-06 PS=1.3e-06
M13 VDD N2 Z VDD EP3 W=1.07571e-05 L=5e-07 AD=1.0495e-11
+ AS=5.22e-12 PD=2.7e-06 PS=1.3e-06
M11 VDD N2 Z VDD EP3 W=1.07571e-05 L=5e-07 AD=1.0495e-11
+ AS=5.22e-12 PD=2.7e-06 PS=1.3e-06
M0 GND A N2 GND EN3 W=5.4e-06 L=5e-07 AD=5.9025e-12 AS=6e-12
+ PD=3.4e-06 PS=8.4e-06
M7 Z N2 GND GND EN3 W=5.25711e-06 L=5e-07 AD=2.87e-12
+ AS=5.445e-12 PD=1.3e-06 PS=3.2e-06
```

```

M5 Z N2 GND GND EN3 W=5.25711e-06 L=5e-07 AD=2.87e-12
+ AS=5.445e-12 PD=1.3e-06 PS=3.2e-06
M1 Z N2 GND GND EN3 W=5.25711e-06 L=5e-07 AD=2.87e-12
+ AS=5.9025e-12 PD=1.3e-06 PS=3.4e-06
M3 Z N2 GND GND EN3 W=5.25711e-06 L=5e-07 AD=2.87e-12
+ AS=5.445e-12 PD=1.3e-06 PS=3.2e-06
M8 GND N2 Z GND EN3 W=5.25711e-06 L=5e-07 AD=9.645e-12
+ AS=2.87e-12 PD=1.14e-05 PS=1.3e-06
M6 GND N2 Z GND EN3 W=5.25711e-06 L=5e-07 AD=5.445e-12
+ AS=2.87e-12 PD=3.2e-06 PS=1.3e-06
M4 GND N2 Z GND EN3 W=5.25711e-06 L=5e-07 AD=5.445e-12
+ AS=2.87e-12 PD=3.2e-06 PS=1.3e-06
M2 GND N2 Z GND EN3 W=5.25711e-06 L=5e-07 AD=5.445e-12
+ AS=2.87e-12 PD=3.2e-06 PS=1.3e-06
*****
.model en3
+ nmos      level=3
+ dmut = 1.474    dvt =-0.0011
+ eox = 11.3n    dl = 0.130u      dw = -0.100u
+ vt0 = 0.540    kb1 = 0.624      kb2 = 0.690
+ phil = 0.831    vc = 1.221      k0 =0.654E-04
+ tg = 0.064     racc = 411.2u    tw = -0.001u
+ a2 = -0.0074u   a3 = -0.076u    a1 = -0.260u
+ b2 = 0.004u     b3 = 0.032u    b1 = 0.055u
+ del= 0.294     gl = -0.255u   gw = 0.002u
+ delvg= -0.040   glvg = 0.152u  gwvg = 0.001u
+ ke =0.250E+08   10 = 1.977u   eps = 0.018u
+ n1 = 0.330     n2 = 1.233   an =0.506E+06
+ bn =0.455E+08   cjs=8.02e-04 mjs=0.35
+ cjp=2.90e-10   mjp=0.09    cjc=1.37e-10
+ rec=0.04u      recl=0.1u    is=7.05e-04
+ ip=4.3e-09     xti=3.0     nd=2.0
.model ep3
+ pmos      level=3
+ dmut = 1.330    dvt =-0.0010
+ eox = 11.3n    dl = 0.140u      dw = -0.10u
+ vt0 = 0.600    kb1 = 0.680      kb2 = 0.680
+ phil = 0.829    vc = 0.000      k0 =0.152E-04
+ tg = 0.122     racc = 1071.2u    tw = -0.002u
+ a2 = -0.036u   a3 = -0.154u    a1 = -0.154u
+ b2 = 0.009u     b3 = 0.036u    b1 = 0.036u
+ del= 0.311     gl = -0.191u   gw = 0.010u
+ delvg= -0.036   glvg = 0.040u  gwvg = -0.011u
+ ke =0.370E+08   10 = 0.574u   eps = 0.052u
+ n1 = 0.432     n2 = 1.069   an =0.219E+06
+ bn =0.734E+08   cjs=7.93e-04 mjs=0.33
+ cjp=2.29e-10   mjp=0.14    cjc=1.1e-10
+ rec=0.04u      recl=0.1u    is=8.00e-04
+ ip=4.0e-09     xti=3.0     nd=2.0
*****
Vdd vdd 0 dc 3.6
Vin A 0 pwl (0 0 4n 0 4.25n 3.6 9n 3.6 9.25n 0)
*****
Vdum Z Z1 dc 0
*****
Cload Z1 0 3.2p
*****
.ttemp 25

```

```

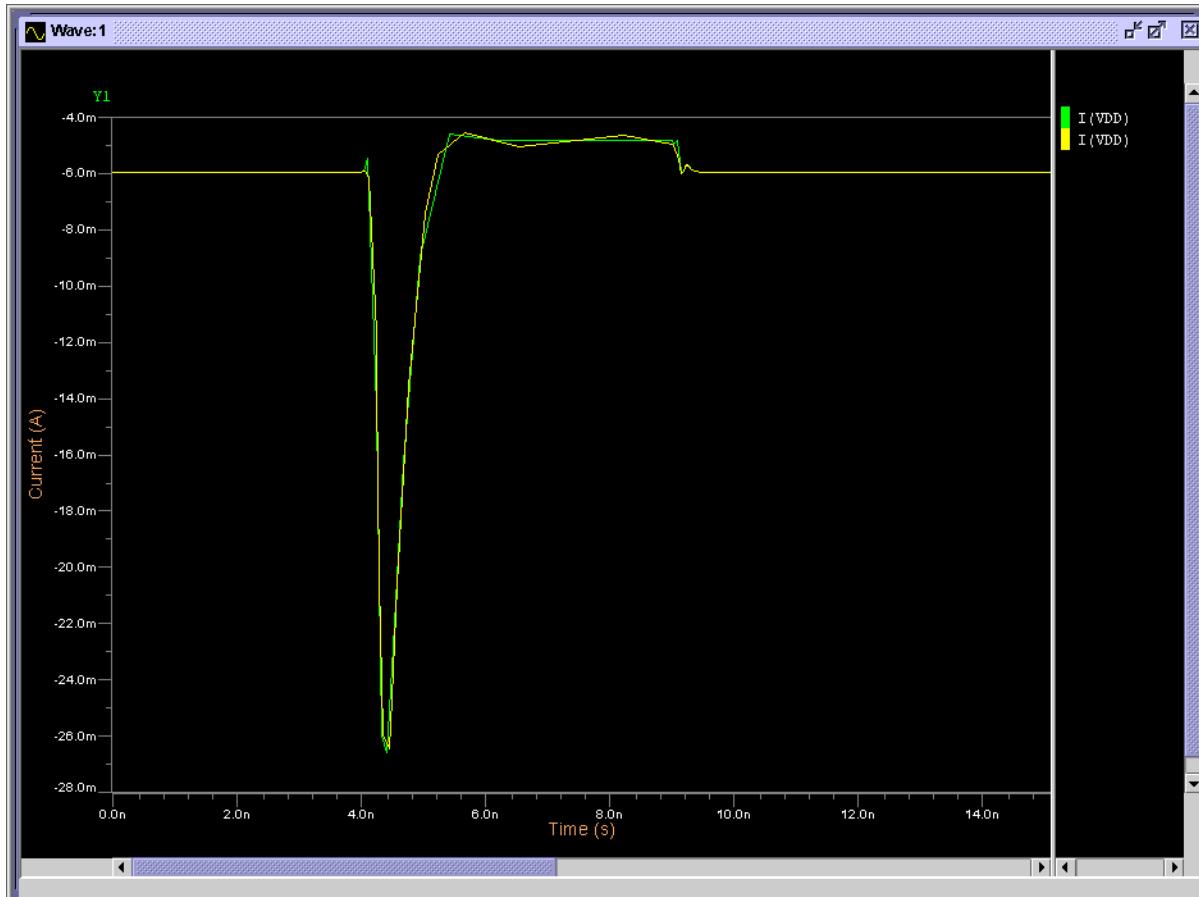
.tran 0.005n 32n
.connect 0 GND
.option accusim2
.option iem !stable
*.option newton !trapezoidal exhibits numerical oscillations
.plot tran i(vdd)
.plot tran i(vdum)
.defwave wivdd=abs(i(vdd))
.defwave wvdum=abs(i(vdum))
.extract integ(w(wivdd))
.extract integ(w(wvdd))
.end

```

## Simulation Results

Figure 17-2 compares the waveform produced by the Newton-Raphson method (shown in yellow) compared with the waveform produced by the IEM method (shown in green).

**Figure 17-2. Example 2—ivdd**



## Accuracy

The precision is measured by comparing the results for each algorithm to a reference. The reference can be obtained using either of the two algorithms, by setting extremely low values on tolerance parameters. By default, IEM gives results at relatively high precision. In order to attain the same accuracy using Newton-Raphson, tolerance parameters should be tightened, which results in an increase in the CPU. The performance gain is measured by the CPU ratio between both algorithms when the output results are at the same level of accuracy.

### Example 3—oscil

For this oscillator, a jitter lower than 2%, can be obtained by IEM at default settings, or by Newton-Raphson at **EPS=3.0e-7**. At this level of accuracy, the speed-gain for IEM is about 3x.

### Complete Netlist

#### Note



The netlist file is also provided as Newton-Raphson and reference versions (*oscil\_nrm.cir* and *oscil\_ref.cir* respectively) in the examples directory. *oscil\_ref.cir* is the reference model to which the *iem* and *nrm* circuits are compared. The netlists are identical to *oscil\_iem.cir* with the exception of **.OPTION NEWTON** selected for *oscil\_nrm.cir* and **.OPTION NEWTON EPS=1.e-8** selected for *oscil\_ref.cir*.

```
oscil_iem.cir
.MODEL PBSIM2 PMOS  level=11
+ TNOM=2.70000E+01 RSH=6.50000E+01  TOX=1.50000E-02
+ VDD=5.00000E+00 VGG=5.00000E+00  VBB=-5.00000E+00
+ JS=2.00000E-06 XPART=0.00000E+00  CGSO=2.00000E-10
+ CGDO=2.00000E-10 CGBO=0.00000E+00  PB=6.75000E-01
+ MJ=4.30000E-01 PBSW=0.10000E+00  MJSW=4.30000E-01
+ CJ=3.80000E-04 CJSW=1.20000E-10  DL=2.75240E-01
+ DW=2.15737E-01 MU0=4.88029E+02  NO=1.40000E+00
+ LNO=0.00000E+00 WN0=0.00000E+00  NB=5.00000E-01
+ LNB=0.00000E+00 WNB=0.00000E+00  ND=0.00000E+00
+ LND=0.00000E+00 WND=0.00000E+00  PHI=7.71089E-01
+ LPHI=1.66784E+00 WPHI=-3.12807E-01 MU0B=0.00000E+00
+ LMU0B=0.00000E+00 WMU0B=0.00000E+00 K1=8.10175E-01
+ LK1=0.00000E+00 WK1=0.00000E+00  K2=4.67768E-02
+ LK2=0.00000E+00 WK2=0.00000E+00  ETA0=-1.78327E-02
+ LETA0=2.74604E-02 WETA0=0.00000E+00 U10=-1.46747E-02
+ LU10=2.66063E-01 WU10=0.00000E+00 VFB=-9.74625E-01
+ LVFB=-2.01651E+00 WVFB=4.68098E-01 ETAB=0.00000E+00
+ LETAB=0.00000E+00 WETAB=0.00000E+00 U1B=0.00000E+00
+ LU1B=0.00000E+00 WU1B=0.00000E+00 MU20=1.32844E+00
+ LMU20=1.75239E+00 WMU20=0.00000E+00 MU2B=0.00000E+00
+ LMU2B=0.00000E+00 WMU2B=0.00000E+00 MU30=1.29954E+00
+ LMU30=-9.23397E+00 WMU30=0.00000E+00 MUS0=5.30129E+02
+ LMUS0=9.17775E+00 WMUS0=0.00000E+00 MUSB=0.00000E+00
+ LMUSB=0.00000E+00 WMUSB=0.00000E+00 MU2G=-4.77583E-01
```

```

+ LMU2G=0.00000E+00 WMU2G=0.00000E+00 MU3B=0.00000E+00
+ LMU3B=0.00000E+00 WMU3B=0.00000E+00 MU3G=1.17696E+00
+ LMU3G=0.00000E+00 WMU3G=0.00000E+00 MU40=-8.24961E-01
+ LMU40=7.86998E-01 WMU40=0.00000E+00 MU4B=0.00000E+00
+ LMU4B=0.00000E+00 WMU4B=0.00000E+00 MU4G=-1.19857E-01
+ LMU4G=0.00000E+00 WMU4G=0.00000E+00 UA0=2.89822E-02
+ LUA0=2.68543E-01 WUA0=-5.76159E-02 UAB=-1.12017E-02
+ LUAB=0.00000E+00 WUAB=0.00000E+00 UB0=1.22936E-02
+ LUB0=-1.08792E-02 WUB0=1.58744E-03 UBB=2.34752E-04
+ LUBB=0.00000E+00 WUBB=0.00000E+00 U1D=0.00000E+00
+ LU1D=0.00000E+00 WU1D=0.00000E+00 VGHIGH=2.00000E-01
+ LVHIGH=0.000E+00 WVHIGH=0.000E+00 VOFO=1.80000E+00
+ LVOFO=0.00000E+00 WVOFO=0.00000E+00 VOFB=0.00000E+00
+ LVOFB=0.00000E+00 WVOFB=0.00000E+00 VOFD=0.00000E+00
+ LVOFD=0.00000E+00 WVOFD=0.00000E+00 AIB=0.00000E+00
+ LAIB=0.00000E+00 WAIB=0.00000E+00 BIB=0.00000E+00
+ LBIB=0.00000E+00 WBIB=0.00000E+00 VGLOW=-1.50000E-01
+ LVGLOW=0.000E+00 WVGLOW=0.0000E+00 DELL=0.00000E+00
+ BEX=-1.50000E+00 TCV=1.00000E-03 AI0 = -2.70194E-02
+ LAI0= 1.03485E-01 WAI0= 0.00000E+00 BI0 = 6.95477E-01
+ LBIO= 5.20197E-01 WBI0= 0.00000E+00
.MODEL NBSIM2 NMOS
.level=11
+ TNOM=2.70000E+01 RSH=6.50000E+01 TOX=1.50000E-02
+ VDD=5.00000E+00 VGG=5.00000E+00 VBB=-5.00000E+00
+ JS=2.00000E-06 XPART=0.00000E+00 CGSO=2.00000E-10
+ CGDO=2.00000E-10 CGBO=0.00000E+00 PB=6.75000E-01
+ MJ=4.30000E-01 PBSW=0.10000E+00 MJSW=4.30000E-01
+ CJ=3.80000E-04 CJSW=1.20000E-10 DL=2.75240E-01
+ DW=2.15737E-01 MU0=4.88029E+02 N0=1.40000E+00
+ N0=0.00000E+00 WN0=0.00000E+00 NB=5.00000E-01
+ LNB=0.00000E+00 WNB=0.00000E+00 ND=0.00000E+00
+ LND=0.00000E+00 WND=0.00000E+00 PHI=7.71089E-01
+ LPHI=1.66784E+00 WPHI=-3.12807E-01 MU0B=0.00000E+00
+ LMU0B=0.00000E+00 WMU0B=0.00000E+00 K1=8.10175E-01
+ LK1=0.00000E+00 WK1=0.00000E+00 K2=4.67768E-02
+ LK2=0.00000E+00 WK2=0.00000E+00 ETAO=-1.78327E-02
+ LETAO=2.74604E-02 WETA0=0.00000E+00 U10=-1.46747E-02
+ LU10=2.66063E-01 WU10=0.00000E+00 VFB=-9.74625E-01
+ LVFB=-2.01651E+00 WVFB=4.68098E-01 ETAB=0.00000E+00
+ LETAB=0.00000E+00 WETAB=0.00000E+00 U1B=0.00000E+00
+ LU1B=0.00000E+00 WU1B=0.00000E+00 MU20=1.32844E+00
+ LMU20=1.75239E+00 WMU20=0.00000E+00 MU2B=0.00000E+00
+ LMU2B=0.00000E+00 WMU2B=0.00000E+00 MU30=1.29954E+00
+ LMU30=-9.23397E+00 WMU30=0.00000E+00 MUS0=5.30129E+02
+ LMUS0=9.17775E+00 WMUS0=0.00000E+00 MUSB=0.00000E+00
+ LMUSB=0.00000E+00 WMUSB=0.00000E+00 MU2G=-4.77583E-01
+ LMU2G=0.00000E+00 WMU2G=0.00000E+00 MU3B=0.00000E+00
+ LMU3B=0.00000E+00 WMU3B=0.00000E+00 MU3G=1.17696E+00
+ LMU3G=0.00000E+00 WMU3G=0.00000E+00 MU40=-8.24961E-01
+ LMU40=7.86998E-01 WMU40=0.00000E+00 MU4B=0.00000E+00
+ LMU4B=0.00000E+00 WMU4B=0.00000E+00 MU4G=-1.19857E-01
+ LMU4G=0.00000E+00 WMU4G=0.00000E+00 UA0=2.89822E-02
+ LUA0=2.68543E-01 WUA0=-5.76159E-02 UAB=-1.12017E-02
+ LUAB=0.00000E+00 WUAB=0.00000E+00 UB0=1.22936E-02
+ LUB0=-1.08792E-02 WUB0=1.58744E-03 UBB=2.34752E-04
+ LUBB=0.00000E+00 WUBB=0.00000E+00 U1D=0.00000E+00
+ LU1D=0.00000E+00 WU1D=0.00000E+00 VGHIGH=2.00000E-01
+ LVHIGH=0.000E+00 WVHIGH=0.000E+00 VOFO=1.80000E+00

```

```
+ LVOF0=0.00000E+00 WVOF0=0.00000E+00 VOFB=0.00000E+00
+ LVOFB=0.00000E+00 WVOFB=0.00000E+00 VOFD=0.00000E+00
+ LVOFD=0.00000E+00 WVOFD=0.00000E+00 AIB=0.00000E+00
+ LAIB=0.00000E+00 WAIB=0.00000E+00 BIB=0.00000E+00
+ LBIB=0.00000E+00 WBIB=0.00000E+00 VGLOW=-1.50000E-01
+ LVGLOW=0.0000E+00 WVGLOW=0.0000E+00 DELL=0.00000E+00
+ BEX=-1.50000E+00 TCV=1.00000E-03 AI0=-2.70194E-02
+ LAI0= 1.03485E-01 WAIO = 0.0000E+00 BI0= 6.95477E-01
+ LBIO= 5.20197E-01 WBIO = 0.0000E+00

Vdd vdd 0 dc 5
Vss vss 0 dc 0

.subckt inv in out vss vdd
r1 vdd vdd1 10
r2 vss vss1 10
M1 out in vss1 vss nbsim2 L=2U W=6U AD=90E-12 AS=90E-12 PD=24u
+ PS=24u
M2 out in vdd1 vdd pbsim2 L=2U W=10U AD=90E-12 AS=90E-12
+ PD=24u PS=24u
.ends inv

Xin1 in1 out1 vss vdd inv
c1 out1 0 0.1f
Xin2 out1 out2 vss vdd inv
c5 out2 0 0.1f
Xin3 out2 out3 vss vdd inv
c2 out3 0 0.1f
Xin4 out3 out4 vss vdd inv
c3 out4 0 0.1f
Xin5 out4 in1 vss vdd inv
c4 in1 0 0.1f

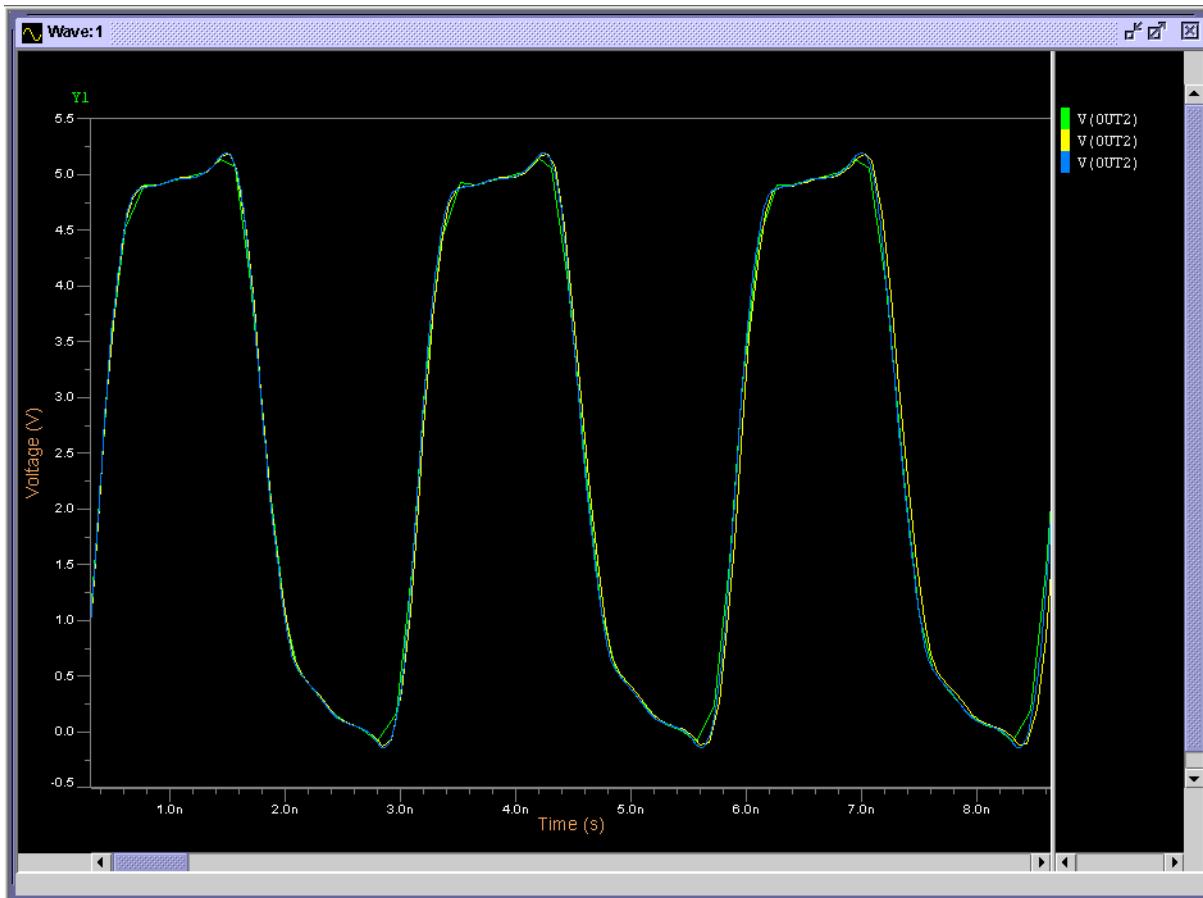
.tran ln 100n uic

.ic v(out1)=5 v(out2)=0 v(out3)=5
.plot tran v(out1)
.plot tran v(out2)
.plot tran v(out3)
.plot tran v(out4)
.option accusim2
.option iem
.end
```

## Simulation Results

Figure 17-3 compares the waveform produced by the Newton-Raphson method (shown in yellow) compared with the waveform produced by the IEM method (shown in green) and the waveform produced by the reference model (shown in blue).

**Figure 17-3. Example 3—oscil**



## Example 4—moy1

The matching considerations in transistor current sources are showed in this circuit. By default, IEM gives an error in the average current difference of 32 nA. The same precision is attained by Newton-Raphson at **EPS=1.0e-6**. At this level the speed-gain for IEM is about 2×.

### Complete Netlist

---

#### Note



The netlist file is also provided as Newton-Raphson and reference versions (*moy1\_nrm.cir* and *moy1\_ref.cir* respectively) in the examples directory. *moy1\_ref.cir* is the reference model to which the *iem* and *nrm* circuits are compared. The netlists are identical to *moy1\_iem.cir* with the exception of **.OPTION NEWTON** selected for *moy1\_nrm.cir* and **.OPTION NEWTON EPS=1.e-8** selected for *moy1\_ref.cir*.

---

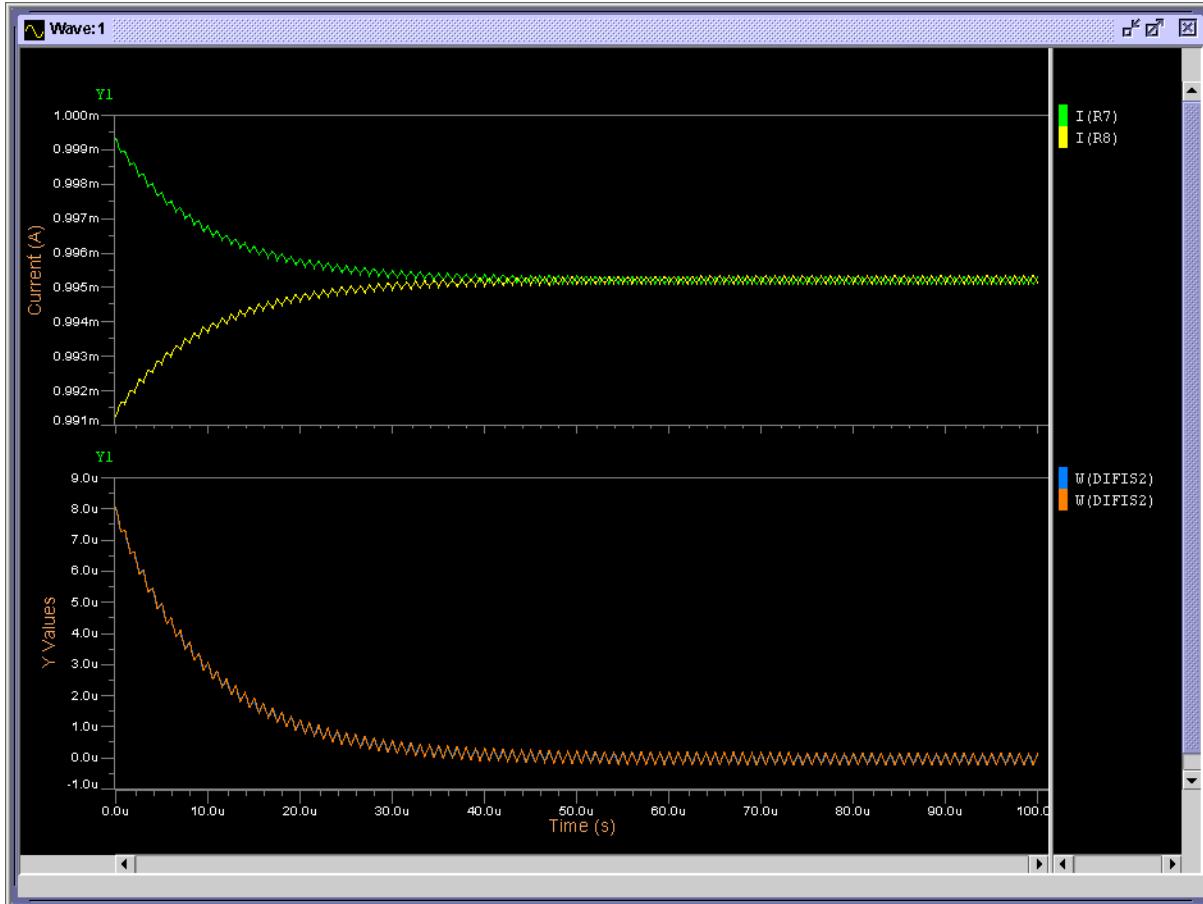
```
moy1_iem.cir
.MODEL MN NMOS LEVEL= MERCK2 EOX= 200E-10 MU0= 500
+ DPHIF= 0.8 DW= 1.0E-6 DL= 0.1E-6 VT0= 0.75
+ KB= 0.6 REC= 0.1E-6 TG= 0.08 VL= 1.0E5 GL= 0.5E-6
+ KL= 0.3E-6 KW= 0.2E-6 DINF= 0.3 GW= 0.4E-6
+ LDIF= 3.5E-6 CDIFS0= 1.4E-4 CDIFP0= 8E-10
+ VE= 20E4 LMIN= 1.2E-6 WMIN= 3.4E-6 RSH= 500

mr vp vp r 0 mn w=200u l=2u
rr r 0 1k
m1 3 vp 1 0 mn w=200u l=2u
r1 1 0 1k
m2 4 vp 2 0 mn w=200u l=2u
r2 2 0 1.01k
m3 5 hb 3 0 mn w=200u l=2u
m4 6 hb 4 0 mn w=200u l=2u
m5 6 h 3 0 mn w=200u l=2u
m6 5 h 4 0 mn w=200u l=2u
m7 7 a 5 0 mn w=200u l=2u
m8 8 a 6 0 mn w=200u l=2u
r7 s1 7 1k
c7 7 0 10n
r8 s2 8 1k
c8 8 0 10n
* alims
vs1 s1 0 dc 10
vs2 s2 0 dc 10
va a 0 dc 7
ipol 0 vp 1m
.param d=2n
.param thold=490n+d
vh h 0 pulse( 3 4 10n 10n 10n thold 1u)
vhb hb 0 pulse( 4 3 10n 10n 10n thold 1u)
* simuls
.dc
.op
.tran 100n 100u
.defwave difis2=i(r7)-i(r8)
.plot tran i(r7) i(r8)
.plot tran w(difis2)
.width out=80
.option accusim2
.option iem
*.option newton eps=1.e-6
.option noascii
.extract average(w(difis2),98u,100u)
.end
```

## Simulation Results

Figure 17-4 compares the waveform produced by the Newton-Raphson method (shown in green) compared with the waveform produced by the IEM method (shown in yellow).

**Figure 17-4. Example 4—moy1**



## Speed

Speed gain ratio depends on the nature of the circuit simulated as well as the level of activity. Slowly varying circuits may manifest no gain. In the following circuits, a certain level of activity was present such as to show IEM advantage at default settings for both IEM and Newton-Raphson.

## Example 5—ladder

The speed-gain for IEM is about 2.3× (as compared to Newton-Raphson).

### Complete Netlist

#### Note

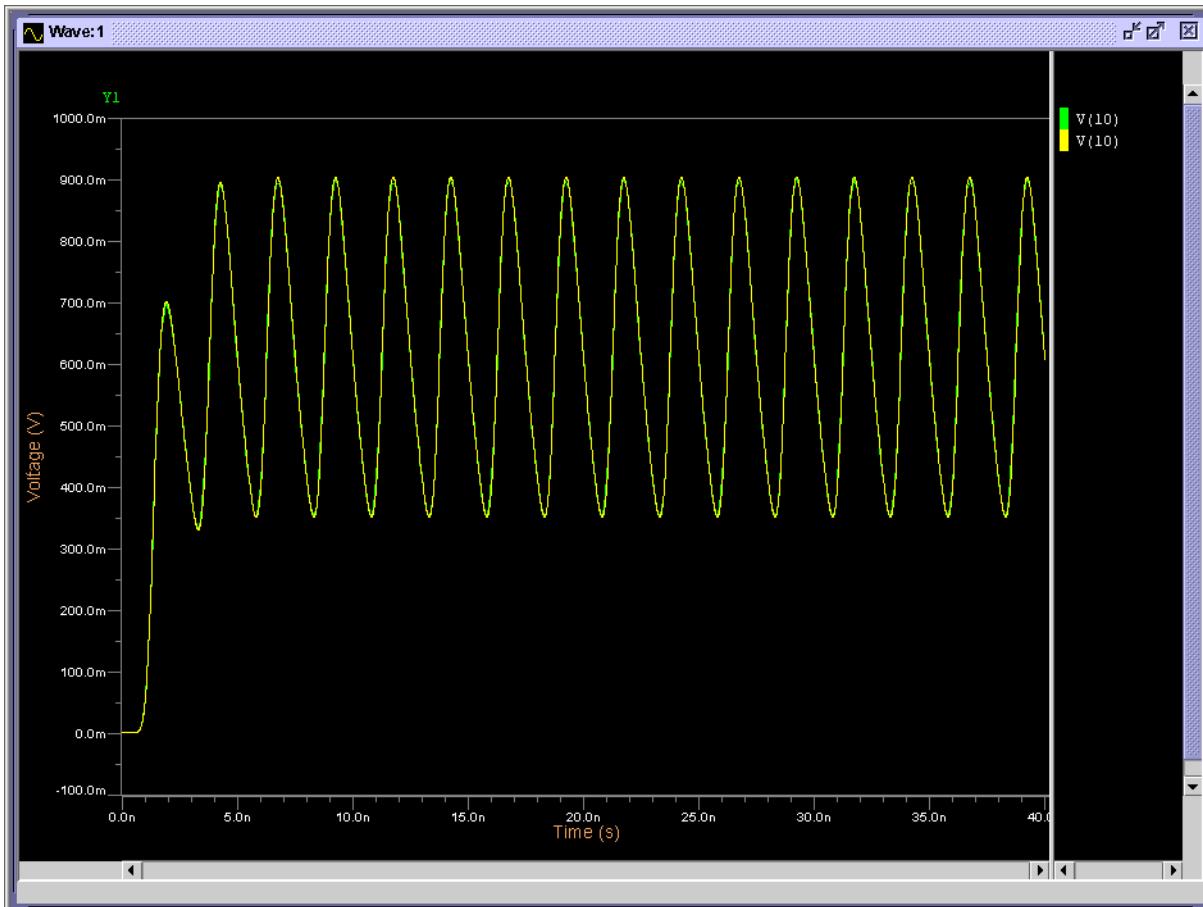
The netlist file is also provided as a Newton-Raphson version (*ladder\_nrm.cir*) in the examples directory for comparison. The netlist is identical to *ladder\_iem.cir* with the exception of **.OPTION NEWTON** selected for *ladder\_nrm.cir*.

```
Simple R-Ladder Circuit
.param r=1K
.subckt r2r 1 2
R1 1 2 r
R2 2 0 {2*r}
D 0 2 DMOD OFF
.MODEL DMOD D CJO=1P
.ends r2r
*VIN 1 0 pulse 0 1024 0 1n 1n 100U 1
vin 1 0 sin 1 1024 0.4g
x1 1 2 r2r
x2 2 3 r2r
x3 3 4 r2r
x4 4 5 r2r
x5 5 6 r2r
x6 6 7 r2r
x7 7 8 r2r
x8 8 9 r2r
R1 9 10 r
ROUT 10 0 r
.width out=80
.tran 0.1N 40N
.plot tran v(10)
.option accusim2
.option iem
*.option newton
.end
```

## Simulation Results

Figure 17-5 compares the waveform produced by the Newton-Raphson method (shown in green) compared with the waveform produced by the IEM method (shown in yellow).

**Figure 17-5. Example 5—ladder**



## Example 6—opamp\_5pin

For this circuit, the speed-gain for IEM is about 2.3× (as compared to Newton-Raphson).

### Complete Netlist



#### Note

The netlist file is also provided as a Newton-Raphson version (*opamp\_5pin\_nrm.cir*) in the examples directory for comparison. The netlist is identical to *opamp\_5pin\_iem.cir* with the exception of **.OPTION NEWTON** selected for *opamp\_5pin\_nrm.cir*.

---

```

opamp_5pin_iem.cir
.option TNOM=27
VIZ11 4 0 10V
VIZ9 0 5 10V
VDZ0 6 0 SIN 0 1 1000 0 0
*.dc vdz0 1 -1 -0.01
*.plot dc v(3)
RIZ16 2 3 10K

```

```
RIZ14 6 1 10K
RIZ13 6 0 10K
RIZ12 0 2 10K
RIZ3 3 0 10K
XA1 2 1 3 4 5 LM107
* OPAMP MACROMODEL $LM107
* NATIONAL LINEAR P3-140, 1982
* SUPPLY VOLTAGE(S): 15
* USAGE: XNAME <-> <+> <OUT> <VCC> <VEE> $LM107
.SUBCKT LM107 2 3 6 7 4
V1 1004 0 1
DX1 0 1000 DMOD
DX2 0 1001 DMOD
DX3 1000 1002 DMOD
DX4 1001 1003 DMOD
VSINK 1002 1004 0
VSOURCE 1003 1004 0
FOUT 1000 1001 VO 1
FSOURCE 7 126 VSOURCE 1
FSINK 296 4 VSINK 1
IXX 6 0 0
VO 158 6 0
EVCC 126 5 7 5 1
EVEE 296 5 4 5 1
VLM2 5 115 55.7517
VLM1 116 5 55.7517
DD1 16 5 DD OFF
DD2 5 16 DD OFF
DCLP 158 112 DMOD OFF
DCLN 113 158 DMOD OFF
DLM1 106 116 DMOD OFF
DLM2 115 106 DMOD OFF
GDM 16 5 8 9 31.4165U
GCM 15 5 12 5 -21.2896M
GB 15 5 16 5 427.5729
JP 117 4 4 MMOD
EGND 5 4 7 4 0.5
ELIM 105 5 15 6 31.6228
VCHAIN 7 117 0
HCMR1 7 108 VCHAIN 222.535
HCMR2 110 4 VCHAIN 667.605
HCLP 126 112 VCHAIN 965.444
HCLN 113 296 VCHAIN 965.444
FEE 12 110 VCHAIN 1.4242M
FLIM 15 5 V40 1
V40 105 106 0
CE 12 5 1.25P
C1 8 9 2.5P
C2 15 16 5P
Q1 8 2 10 QM1
Q2 9 3 11 QM2
RE1 10 12 11.0128K
RC1 8 108 31.8304K
RE2 11 12 11.0128K
RC2 9 108 31.8304K
REE 12 5 1.0042G
*.opt abstol=1P vntol=1U reltol=1M
R2 5 16 0.1MEG
```

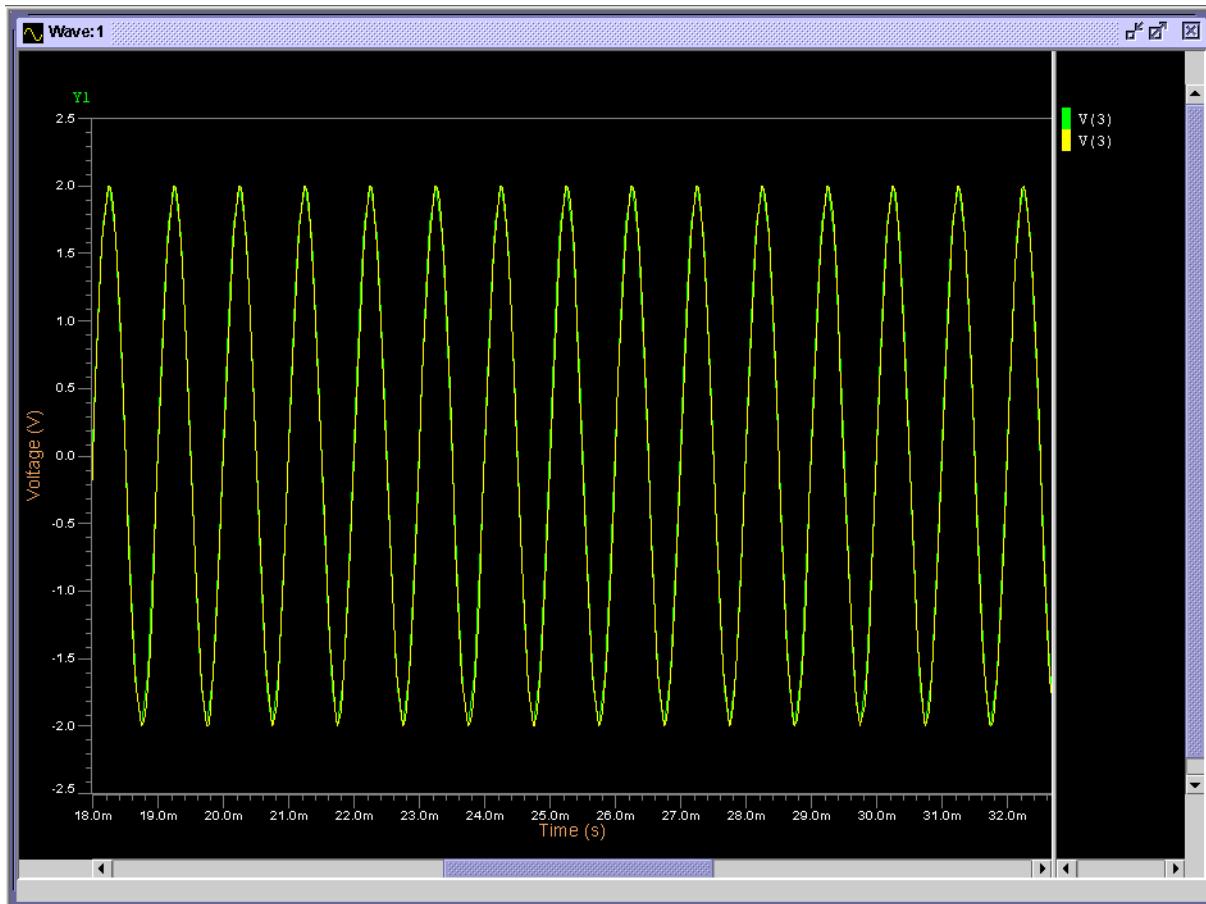
```
RO2 5 15 134
RO1 158 15 66
.MODEL QM1 NPN (IS=0.8F BF=40.6504)
.MODEL QM2 NPN (IS=0.8216F BF=42.735)
.MODEL DMOD D (N=1)
.MODEL DD D (N=30.1931)
.MODEL MMOD NJF (VTO=-0.6 BETA=4.993M)
.ENDS LM107
* BEGIN ELDO
.temp 27
.op

.TRAN 0.001 0.05 0
.plot tran v(3)
.plot tran v(6)
.width out=80
.option accusim2
.option iem
*.option newton
.end
```

## Simulation Results

Figure 17-6 compares the waveform produced by the Newton-Raphson method (shown in green) compared with the waveform produced by the IEM method (shown in yellow).

Figure 17-6. Example 6—opamp\_5pin



# Chapter 18

## Pole-Zero Post-processor

---

### Introduction

Eldo is able to obtain the locations of poles and zeros for analog circuits following a small signal analysis. This data is useful for obtaining stability and phase margin information as well as predicting closed loop performance from an open loop response. As a lot of closely spaced poles or zeros are not easily discernible via a Bode plot, the interactive post processor and its textual output clearly help to identify such singularities and evaluate the most important areas in a circuit's behavior.

The pole-zero data is obtained by linearizing the circuit about an operating point and outputting information about circuit matrix description and AC response. This data will be analyzed by the Eigenvalue QZ algorithm, one of the most accurate techniques available.

The pole-zero post-processor allows the designer to detect all the roots of the analyzed circuit and to simplify this information extracting the most meaningful of these roots. Moreover, a high level model of the reduced circuit—equivalent for AC and small signal analysis—is calculated and given in the form of an FNS device which can replace the original circuit when used as part of a more complex design.

The pole-zero post-processor may be activated for circuits which have been simulated using the **.PZ** and **.AC** commands. The **.PZ** command can take either the voltage at a node, between two nodes or the current through a voltage source as parameters. The pole-zero analysis determines the transfer function relationship—expressed as complex poles and zeros—between the input where the AC voltage source is applied and the output specified as a parameter in the **.PZ** command.

Pole-zero only works on quasi-static devices; pole-zero would give incorrect results on HDL-A models which are not linear with OMEGA ( $\exp(\text{OMEGA})$  for instance). However, for HDL-A models that make use of derivatives of order higher than 2 it is possible to declare additional IMPLICIT states and make the model linear in OMEGA.

### Example

In ‘pseudo’ HDL-Av1 code, convert:

```
b = ddt(a)
c = ddt(b);
```

into:

```
EQUATION (b,c) FOR ac,dc,transient =>
b == ddt(a);
c == ddt(b);
```

## Running the Dialog for Pole-Zero Analysis

Before running the pole-zero post-processor, the user must make sure that an AC analysis has been performed on the circuit of interest and that a **.PZ** command has been included in the netlist. This has the effect of producing a file called *<circut>.pz* in the output directory. After this has been done, type the following command to physically start the pole-zero dialog:

```
pz <circut>.pz
```

The first result is the impression of all the roots found for the circuit, ordered by magnitude but with zeros divided in negative and positive real parts, as follows for a typical example:

Poles	Modulus	Real Part	Imaginary Part
1	6.264953E+00	-6.264953E+00	0.000000E+00
2	3.853866E+06	-3.824006E+06	-4.788095E+05
3	3.853866E+06	-3.824006E+06	4.788095E+05
4	4.396770E+06	-4.396770E+06	0.000000E+00
5	5.423444E+06	-5.423444E+06	0.000000E+00
6	6.554762E+06	-3.126871E+06	-5.760866E+06
7	6.554762E+06	-3.126871E+06	5.760866E+06
8	6.584083E+06	-5.150248E+06	-4.101840E+06
9	6.584083E+06	-5.150248E+06	4.101840E+06
10	1.343013E+07	-1.343013E+07	0.000000E+00
11	1.531050E+07	-1.531050E+07	0.000000E+00
12	2.217833E+07	-2.217833E+07	0.000000E+00
13	4.927351E+07	-4.927351E+07	0.000000E+00
14	6.564085E+07	-6.564085E+07	0.000000E+00
15	1.228260E+08	-1.228260E+08	0.000000E+00
Zeros	Modulus	Real Part	Imaginary Part
1	3.853866E+06	-3.824006E+06	-4.788095E+05
2	3.853866E+06	-3.824006E+06	4.788095E+05
3	4.396770E+06	-4.396770E+06	0.000000E+00
4	5.423444E+06	-5.423444E+06	0.000000E+00
5	6.435558E+06	-5.867538E+06	-2.643560E+06
6	6.435558E+06	-5.867538E+06	2.643560E+06
7	6.554762E+06	-3.126871E+06	-5.760866E+06
8	6.554762E+06	-3.126871E+06	5.760866E+06
9	1.343013E+07	-1.343013E+07	0.000000E+00
10	1.531050E+07	-1.531050E+07	0.000000E+00
11	2.217833E+07	-2.217833E+07	0.000000E+00
12	6.564085E+07	-6.564085E+07	0.000000E+00
13	1.228260E+08	-1.228260E+08	0.000000E+00
14	3.857487E+06	3.857487E+06	0.000000E+00
15	2.908742E+09	2.908742E+09	0.000000E+00

At this point it is possible to simplify the results, since in practice many poles and zeros will be output and some of them will be almost superimposed giving a negligible resultant effect.

Moreover, sometimes the circuit will work in a well determined frequency range over which we

will focus our attention, while also all the parasitic and less meaningful roots will be detected. That is why the following reduction mechanisms are available.

## Frequency Limit

The program asks the user if he wants to limit the analysis up to a certain frequency. Here is such a dialog:

```
Do you want to set an upper frequency limit for the selected poles and
zeros? [yes/no]:
y
So give the highest frequency to be considered:
5e7
Roots up to 0.5000E+08 Hz will be examined.
```

If the roots of your circuit are spanned up to a frequency quite a lot higher than the upper frequency of your AC analysis, this limit can be very useful in simplifying textual output and FNS models without meaningful loss of accuracy.

## Pole-Zero Cancellation by Threshold

Very close poles and zeros can be deleted as their influence on the circuit behavior compensates. Big circuits with a lot of roots often have closely spaced poles and zeros which can be deleted to show the most meaningful remaining roots. A prompt reminds the user about the threshold mechanism, showing the condition to delete a pole **P** and a zero **Z**. The condition is that:

$$\text{abs}(\text{RE}[P] - \text{RE}[Z]) < \text{abs}(\text{RE}[P]) \diamond \text{TH} + \text{TH}$$

and the same for the imaginary part.

Therefore, if a threshold of say, 0.1 is given, there are two possibilities:

1. For very low frequency roots, say fractions of 1 Hz, **TH** has the meaning of maximum difference between the real parts of the pole and the zero, and the same for the imaginary part.
2. In most cases, there are roots at quite high frequencies so the first term on the right side of the relation dominates and **TH** takes the meaning of maximum ratio between the distance amongst the roots and their value, so in this case a 10% tolerance.

At this point, a table of remaining poles and zeros (after the two elimination steps) is given.

## Hand Selection

After these steps, the user can still express a further choice between the resulting roots. A prompt asks:

Do you agree with the Selected POLES [Yes/No]?

If the selected poles and zeros are Ok, then press **Y** and the analysis goes on. If only a certain number of poles and zeros are required, press **N** to reveal the following prompt:

Poles selection by their index in INCREASING order

Enter a negative index to end the selection.

## Select Index

The user is asked to select which poles are to be included in the final transfer function. The selection is made via the index to the values—the index being the first number which appears in the tables above. Once the values to use have been chosen, the selection is terminated by entering a negative number. The same dialog then starts for zeros.

---

### Note



Make sure the selections are made in increasing integer order including the negative terminator otherwise required values may be lost.

---

The remaining set of poles and zeros are printed, followed by another table showing the difference between the actual and the now approximated model, swept in frequency and giving both the magnitude (dB) and the phase (degree) differences. This is shown below:

Hz	dB's Error	Degrees Error
1.000000E+01	6.525140E-04	1.020462E-02
1.000000E+02	6.201058E-04	1.096881E-02
1.000000E+03	5.805557E-04	1.102087E-02
1.000000E+04	5.806112E-04	1.107192E-02
1.000000E+05	5.791327E-04	3.599889E+02
1.000000E+06	5.815470E-04	1.444244E-02
1.000000E+07	1.179731E-03	4.588650E-02
1.000000E+08	5.953287E-02	3.609858E-01
1.000000E+09	4.490041E+00	2.134625E+00

## FNS Model

The reduced circuit is now modeled as an S-domain transfer function (say an FNS device in Eldo syntax) which can replace the original circuit with the introduced approximations for AC and small signal analysis—this can be very useful for developing complex designs, allowing the replacement of even a complex block with its equivalent model and great simplification of simulation of the higher level system. The output format is shown below:

```
FNS1 IN OUT
+
+      498057.650621 !
+ 0.0271151244556 ! * s^(-1)
+ 4.88425501019e-10 ! * s^(-2)
+ 2.65330374224e-18 ! * s^(-3)
+ -3.22583377778e-27 ! * s^(-4)
+ 8.84119411e-37 ! * s^(-5)
+
+
+      1
+ 0.0315272851854 ! * s^(-1)
+ 1.08957721435e-09 ! * s^(-2)
+ 1.85561098689e-17 ! * s^(-3)
+ 2.03791858143e-25 ! * s^(-4)
+ 1.03556420914e-33 ! * s^(-5)
```

If too many poles and zeros are left after the reduction steps, FNS evaluation could fail due to too high coefficients (a message is printed in such a case). However, the efficiency of the reduction algorithm normally allows a reduction of at most 6-7 poles which can give a very good approximation of the original circuit.

The PZ program produces a *<circut>.cpz* output file to be visualized with Xelga which shows the Bode diagrams of gain and phase for the original and the reduced circuit. Also, a file *<circut>.mpz* is written, showing a scatter map of the circuits roots in the complex plane (to be seen with Xelga releases starting from 2.7). And finally, a subcircuit containing FNS is stored in file *<circut>.pzck*, which can be included by an upper level netlist.

After all these steps, the user is asked whether they wish to repeat the analysis, typically to alter some parameters such as threshold, frequency limit etc. thereby modifying the accuracy/simplicity ratio of the whole analysis.

When “no” is the answer, the program is exited leaving an ASCII output file *<circut>.pzs* containing a trace of all the program outputs and the dialog.



# Chapter 19

## Monte Carlo Analysis

---

### Introduction

Monte Carlo (MC) analysis is a series of DC, AC, or TRAN analyses, where one or more circuit or model parameter(s) follows a probability distribution. The distribution is uniform, Gaussian, or user-defined. This kind of analysis can be useful for yield analysis. Eldo generates a probability series using a pseudo-random number generator with a seed value for use for each MC variable.

### Usage

#### Define the .MC Command

The first step in MC is to define the **.MC** command.

```
.MC RUNNO [OUTER] [OV] [SEED=i] [NONOM] [ALL]
+ [VARY=LOT|DEV] [IRUN=n] [NBBINS=val] [ORDMCS] [MCLIMIT]
+ [PRINT_EXTRACT=NOMINAL|ALL|run_number]
```

- **RUNNO**

Number of simulation runs (probability samples). Integer.

- **OUTER**

When both **.STEP** and **.MC** commands are given, Eldo performs MC analysis for each point of the **.STEP** command. If **OUTER** is specified, then Eldo will perform a **.STEP** for each point of the **.MC** command, inverting the normal behavior.

- **OV**

Compute the standard deviation of the quantity **ov** and output to the ASCII output file. The format of **ov** is either:

**I(Vxx[, Vyy])**, which specifies the difference in current between voltage sources **Vxx** and **Vyy**. If **Vyy** and the comma are omitted, then this will return just the current through **Vxx**; or

**V(n1[, n2])**, which specifies the voltage potential between nodes **n1** and **n2**. If **n2** and the comma are omitted, then this will return the potential between node **n1** and ground.

Other analysis output formats are available for AC analysis. Please refer to the **.PRINT** documentation.

- SEED=i

The integer `i` is used to initialize the pseudo-random sequence of numbers for the probability distribution. Running MC twice with the same seed value will give the same results.

- ALL

When specified, the results of every MC simulation run are stored in the output files (one set per `runno`). Without this, only the nominal, minimum, and maximum results are saved. This includes both ASCII and binary wave results.

- IRUN=n

When specified, will run just the nth MC simulation of a series. For example,

```
.MC 10 IRUN=3
```

tells Eldo to run the 3rd MC analysis of the 10 point series. Integer.

- VARY=LOT | DEV

Determines whether MC variations are independent or correlated for model MC variables. When specifying `DEV`, then `DEV` and `DEVX` variation is taken into account. Only one `VARY` specification can be set. By default, Eldo applies `LOT`, `DEV` and `DEVX` variation.

- NBBINS=VAL

Specifies the number of bins for the histogram produced when Monte Carlo analysis is used with `.EXTRACT` statements. Default is 10.

- ORDMCS

Determines whether multiple MC parameters in the simulation share the same pseudo-random probability values or not.

For example, without `ORDMCS`, Eldo generates a single probability stream  $p_1, p_2, p_3, \dots, p_n$ . Each probability is between 0 and 1. These probability series are shared among the variables. For two MC variables, A and B, the assignment is:

Run 1:  $A = p_1, B = p_2$   
Run 2:  $A = p_3, B = p_4$   
Run 3:  $A = p_5, B = p_6$   
Run n:  $A = p_{[2n-1]}, B = p_{[2n]}$

For four MC variables, A, B, C, and D, the assignment is:

Run 1:  $A = p_1, B = p_2, C = p_3, D = p_4$   
Run 2:  $A = p_5, B = p_6, C = p_7, D = p_8$   
Run 3:  $A = p_9, B = p_{10}, C = p_{11}, D = p_{12}$   
Run n:  $A = p_{[4n-3]}, B = [4n-2], C = [4n-1], D = [4n]$

With `ORDMCS`, each variable gets its own probability series. For two MC variables, the assignment is:

Run 1:  $A = p_{a1}, B = p_{b1}$   
Run 2:  $A = p_{a2}, B = p_{b2}$

Run 3: A = pa3, B = pb3

Run n: A = pa[n], B = pb[n]

For four MC variables, the assignment is:

Run 1: A = pa1, B = pb1, C = pc1, D = pd1

Run 2: A = pa2, B = pb2, C = pc2, D = pd2

Run 3: A = pa3, B = pb3, C = pc3, D = pd3

Run n: A = pa[n], B = pb[n], C = pc[n], D = pd[n]

- **MCLIMIT**

Specifies that all parameters with statistical distribution (**DEV**, **DEVX**, or **LOT**) will have their distribution modified to one of two deviation values. These values correspond to the maximum deviation of the original distribution as defined by the option **SIGTAIL**. For example, a parameter with a nominal value of 1.0, a statistical deviation of **DEV/GAUSS=5%**, and with option **SIGTAIL** at its default value of 4, the two values will be calculated as follows:

$$1 - (4 * 0.05) = 0.8, \quad 1 + (4 * 0.05) = 1.2$$

This functionality can be useful to force Monte Carlo runs to use maximum deviation combinations. **MCLIMIT** affects all statistical parameters whatever their original distribution.

- **PRINT\_EXTRACT=NOMINAL | ALL | run\_number**

Specifies for which run extracted values should be printed and written to output files.

**NOMINAL**

Only the nominal extracted value is printed (default).

**ALL**

Extracted values are printed for all runs.

**run\_number**

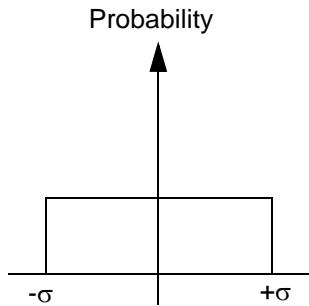
Extracted values are printed only for the specified run\_number (0 is the nominal run).

## Assign Nominal Parameter Values

In the second step, each MC parameter is assigned a nominal value, a standard deviation (expressed in an absolute value or a relative percentage of the nominal), and LOT/DEV correlation. These are the available probability functions:

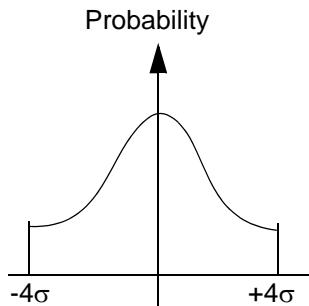
### Uniform

Uniform probability, as it sounds, spreads the probability evenly over the sample space. The probability of a sample occurring outside of  $\pm\sigma$  of the nominal value is zero. This is the default probability function.



## Gaussian

A Gaussian probability function resembles a bell curve. In Eldo, the curve is truncated at  $\pm 4\sigma$ . Note that the probability of a sample landing  $\pm\sigma$  of nominal is 68.3%, and within  $\pm 3\sigma$  of nominal is 99.99%.



## User-defined

An arbitrary model can be defined through the **.DISTRIB** command. Please refer to “[.DISTRIB](#)” on page 10-77 for more details.

## LOT/DEV Correlation

MC parameters can be correlated or not. For example, capacitors in an IC design may have correlation, e.g. their capacitance values tend to rise or fall together. This is LOT correlation. However, components on a printed circuit board tend not to be correlated, which is DEV correlation.

Parameters can have both LOT and DEV variation. Using both means that affected devices have a degree of independence, even though they are correlated. If a parameter is not a primitive, but depends on parameters with no LOT/DEV specification, then a LOT /DEV specification can be set on that parameter.

A 20% LOT setting with a 5% DEV setting for a uniform distribution means that the parameter may not exceed 25% off of nominal. Furthermore, devices that share that parameter are assigned the 20% LOT variation together (s1); then, individual 5% DEV variations are assigned based on s1, instead of the nominal value.

**Note**

 DEV variation specified with **.MODEL** statements (or **.MCMOD**) can refer to dimensions of the current object directly, without any need to encapsulate models into subcircuits. The syntax for accessing an instance parameter is for example:  
**E(\*,<instance\_parameter\_name>)**  
The \* character is used here to refer to the current instance.

---

A **LOTGROUP** can be defined to share the same distribution between dissimilar elements. Once a **LOTGROUP** is defined, it is used in the same way as **LOT** or **DEV**.

```
.LOTGROUP group_name[/distrib_type]=val[%]
```

See below for an example.

## Define MC Parameters

MC parameters are defined through **.MODEL**, **.PARAM** with **.MCMOD**, or **.PARAM** statements.

### **.MODEL**

Individual model parameters can be given different probability functions. To invoke this, a **DEV** and/or **LOT** parameter must immediately follow the model parameter. For example:

```
.model n nmos vto=0.6 dev=0.4 tox=1.5e-7
```

defines a NMOS model with **vto** set to 0.6V and **tox** set to 1.5e-7. The model parameter **vto** has a uniform distribution with  $\sigma=0.4$ V, but **tox** is a constant. The following specification is equivalent:

```
.model n nmos vto=0.6 dev=66.67% tox=1.5e-7
```

In order to use a Gaussian distribution, the **DEV** parameter is changed to **DEV/GAUSSIAN**. Using the same example, changing the **vto** distribution to Gaussian would alter the **.MODEL** card as follows:

```
.model n nmos vto=0.6 dev/gauss=0.4 tox=1.5e-7
```

If a **.LOTGROUP** is defined, then that can be used, as well. The Gaussian distribution example can be expressed as:

```
.LOTGROUP group_a/gauss=0.4
```

```
.model n nmos vto=0.6 lotgroup=group_a
.model n2 nmos vto=2.5 lotgroup=group_a
```

A different parameter of a different model using LOTGROUP group\_a would have the same distribution. For example, the same number, between +0.4 and -0.4, will be used for both vto parameters.

## MCMOD

Modifying **.MODEL** lines for MC analysis can be inconvenient. Eldo offers the **.MCMOD** command to assign probability functions for **.MODEL** parameters without changing any part of the **.MODEL** statement.

```
.MCMOD model [(list_of_instances)] param1 LOT|DEV=val1
+ [{param2 LOT|DEV=val2 param3 LOT|DEV=val3 ...}]
```

The **.MODEL** name must be specified, followed by one or more pairs of parameters and LOT and/or DEV values. Optionally, a list of instances will apply the MC probabilities to only instances in that list.

Using the example above, the MC parameter may be written as:

```
.model n nmos vto=0.6 tox=1.5e-7
.mcmod n vto dev=0.4
```

## PARAM

The syntax for MC **.PARAM** parameters is different than for **.MODEL** parameters. There are two ways to use **.PARAM**. The simpler of the two ways is outlined; please refer to “[.PARAM](#)” on page 10-205 for additional details.

- **.PARAM param1=UNIF|AUNIF|GAUSS|AGAUSS|LIMIT({option\_list})**
  - **UNIF(nom, rel)**  
Defines a uniform distribution. The parameter param1 varies uniformly between nom - nom×rel and nom + nom×rel.
  - **AUNIF(nom, abs)**  
Defines a uniform distribution. The parameter param1 varies uniformly between nom - abs and nom + abs.
  - **GAUSS(nom, rel, sigcoef)**  
Defines a Gaussian distribution. The curve is centered around nom, and the standard deviation is given by  $\sigma = (\text{nom} \times \text{rel}) \div \text{sigcoef}$ .
  - **AGAUSS(nom, abs, sigcoef)**  
Defines a Gaussian distribution. The curve is centered around nom, and the standard deviation is given by  $\sigma = \text{abs} \div \text{sigcoef}$ .

- **LIMIT(nom, abs)**

Defines a limit distribution. Each sample can take the value of nom + abs or nom - abs only, with equal probability.

When defining a parameter using MC distribution variation on the parameter differs depending on where the parameter is specified. **LOT** variation is used when a defined parameter affects model parameters, this means the same random value will be used each time it affects a separate model parameter. **DEV** variation is used when a parameter affects instance parameters, an independent random value is calculated each time the parameter is specified.

## Correlation

A correlation coefficient can be set up between parameters. Please refer to “[.CORREL](#)” on page 10-42 for details.

## Output

MC analysis, when used with **.EXTRACT** statements, will produce histograms of the **.EXTRACT** results in the *.chi* file. This will typically look like:

```
Distribution of IR1
      Range [-3.30511   -631.74897M]
      Nominal value: -2.00000
      Average value: -1.99378
      Standard Deviation: 412.77233M

[ -3.50000    -3.21318 ] NB=   3 FREQ=3.00e-01% |
[ -3.21318    -2.92635 ] NB=   9 FREQ=9.00e-01% |
[ -2.92635    -2.63952 ] NB=  44 FREQ=4.40e+00% | **
[ -2.63952    -2.35270 ] NB=137 FREQ=1.37e+01% | *****
[ -2.35270    -2.06588 ] NB=242 FREQ=2.42e+01% | *****
[ -2.06588    -1.77905 ] NB=261 FREQ=2.61e+01% | *****
[ -1.77905    -1.49223 ] NB=196 FREQ=1.96e+01% | *****
[ -1.49223    -1.20540 ] NB=  84 FREQ=8.40e+00% | ****
[ -1.20540   -918.57500M] NB=   20 FREQ=2.00e+00% | *
[ -918.57500M -631.75000M] NB=    4 FREQ=4.00e-01%
```

A summary of the extracted MC distribution results can be written to the **.AEX** file and the file specified in the **.EXTRACT** command using the option **DUMP\_MCINFO**, see [page 11-45](#) for more information.

**LBOUND** and/or **UBOUND** parameters can be specified on the **.EXTRACT** command to provide lower and upper bound values. A Monte Carlo analysis can use this information to determine whether the extracted value remains in the range [LBOUND, UBOUND] during MC analysis, and a report will be displayed at end of the ASCII output (*.chi*) file. For example:

```
.extract label = toto yval(v(s),25n) lbound = 0.43 ubound = 0.44
```

The result may look something like the below:

```
Distribution of TOTO
  Range [ 420.84593M 461.95279M]
  Nominal value: 440.36822M
  Average value: 443.83611M
  Standard Deviation: 12.80183M
  Passed      : 4 ( 20.00000 %)

*** MC runs which passed all extract: 4 ( 20.00000 %)

MC run 5 OK
MC run 9 OK
MC run 12 OK
MC run 17 OK
```

## Examples

The effects of MC analysis can be seen easily in a simple passive low-pass filter.

```
.model c cap dev=20%
.model r res lot=20% dev=5%

.mc 20 all

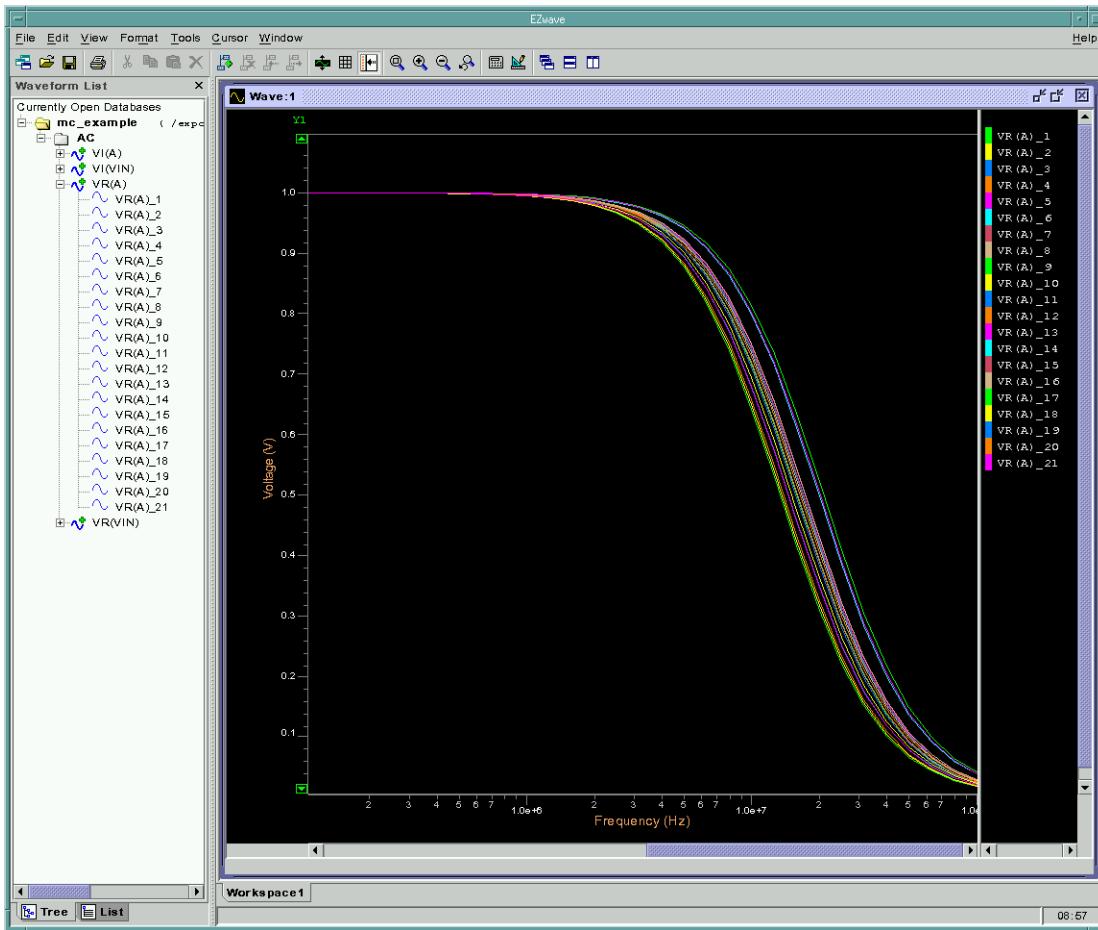
v1 vin 0 ac=1 dc=1
r1 vin a r 1k
c1 a 0 c 10p

.ac dec 10 100 1e8
.probe v
.end
```

The capacitor model has DEV of 20%, so all capacitors using this model are uncorrelated and vary up to 20% from the nominal value specified in the element instantiation. Capacitor C1 varies uniformly between 8 pF and 12 pF. The resistor model produces resistors that are correlated with a 20% deviation. After Eldo applies the 20% LOT variation, each resistor has a 5% non-correlated variation.

The ALL argument to **.MC** tells Eldo to save the result from every MC iteration. Otherwise, only the average, maximum, and minimum values are saved.

Running this simulation and viewing the voltage at node a produces the following graph.

**Figure 19-1. Monte Carlo Analysis Example**


Device sizes can be subject to a probabilistic distribution, simulating process variation, through the use of **.PARAM** statements. In this example, the NMOS transistor's width follows a Gaussian curve, centered around  $20\text{e-}6$  with  $\sigma = 0.2 \times 20\text{e-}6 = 4\text{e-}6$  as defined by parameter **nwidth**. This parameter will also be subject to **DEV** variation. A **.EXTRACT** measures the propagation time through the inverter. The **.chi** file contains statistical information about the outcome of the **.EXTRACT**.

```

.model n nmos level=1
+ vto=1.2
+ kp=2.5e-5
+ gamma=1.5

.model p pmos level=1
+ vto=-1.5
+ kp=1.2e-5
+ gamma=1.2

.param nwidth=gauss(20u,0.2,1)
m1 out in vss vss n w=nwidth l=15u
m2 out in vdd vdd p w=30u l=15u

```

```
cout out 0 0.1p

vdd vdd 0 5
vss vss 0 0

vin in 0 pwl (0 0 50n 0 51n 5)

.mc 20 all
.tran 1n 100n
.probe tran v(out)

.extract tran label=tpd tpdud(v(in), v(out))

.end
```

The **.PARAM** usage may be applied to model parameters, as well. The **vto** of the NMOS transistor can be replaced with a Gaussian parameter. In this case, the standard deviation  $\sigma$  is 0.24. This technique may be more useful if the parameter needs to be a mathematical function of several variables.

```
.model n nmos level=1
+ vto=nvto
+ kp=2.5e-5
+ gamma=1.5

.model p pmos level=1
+ vto=-1.5
+ kp=1.2e-5
+ gamma=1.2

.param nvto=gauss(1.2,0.2,1)
m1 out in vss vss n w=20u l=15u
m2 out in vdd vdd p w=30u l=15u

cout out 0 0.1p

vdd vdd 0 5
vss vss 0 0

vin in 0 pwl (0 0 50n 0 51n 5)

.mc 20 all
.tran 1n 100n
.probe tran v(out)

.extract tran label=tpd tpdud(v(in), v(out))

.end
```

Each instance of a transistor using the NMOS model will have the same value of **vto** because **nvto** affects a model parameter and therefore **LOT** variation is used.

# Chapter 20

## Optimizer in Eldo

---

### Introduction

#### Overview

The Eldo optimizer is a general-purpose electrical circuit optimization program, that aims to find the values of parameters (the optimization variables) in the circuit such that the behavior or the characteristics of the circuit conform as close as possible to the specifications. The optimizer can achieve a simultaneous improvement in AC, DC, transient domain, steady-states and modulated steady-states analyses.

The designer specifies the design objectives and the optimizer will adjust the component parameters of the circuit (such as resistor or capacitor values, the  $\beta$  value of a transistor, widths and lengths of a MOSFET) in order to meet a specified electrical performance. Optimization can be applied to:

- Circuit parameters
- Model parameters
- Element parameters
- Device lengths, widths, areas and peripheries.

The parameter values must conform to the manufacturing limits, process limits, or discrete device values. To achieve this, restrictions can be specified on the design parameters. Various constraints (or inter-relations) can be specified, for example, circuit parameters must be non-negative, or must not violate upper boundaries. In addition, more complicated constraints can be specified, for example, how components physically interact, producing non-linear relations.

The process of identifying the *objective*, *variables*, and *constraints* for a specific problem is known as *modeling*. The construction of an appropriate model is the first step (sometimes the most important) in the optimization process. If the model is too simplistic, it will not generate useful insights into the practical problems. If too complex, it may become difficult to solve. Thus, the designer's task is to discover what is the most appropriate model for their requirements.

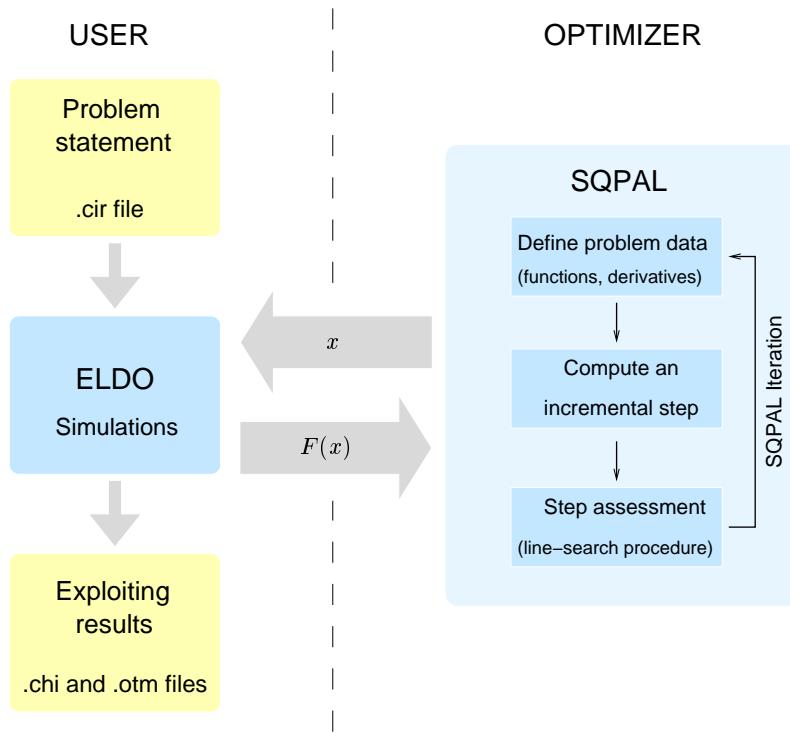
## Performing an Optimization

The optimization process is illustrated in [Figure 20-1](#). The yellow boxes represent the tasks the designer has to perform, while the blue boxes form the simulator/optimizer pair. As suggested, this process uses a client/server paradigm to organize the computations. It means that,

- the task of evaluating the functions  $F(x)$  (the extracted measures) remains in the hands of the simulator, and the optimizer is only required to provide a new approximate solution  $x$ .
- The simulation and the optimization processes are *totally* distinct.

In nonlinear optimization, it is possible to generate an improved solution just by knowing the current iterate and the numerical values of the function and derivatives at the current point.

**Figure 20-1. Operating diagram of the Optimization process**



In order to use the Eldo optimizer, the designer must provide the following information:

- The *nominal circuit*, identical to the circuit provided for simulation in netlist format.
- The *design variables*, many of the component values in the circuit may be fixed. The designer must specify those parameters which may be optimized by the Eldo optimizer in its search for an optimal solution. The designer's selection of variables is accomplished by making minor modifications to the working netlist.

- The *design objectives* may be any quantity, or combination of quantities that can be represented by a single number. Note that, the design objective is not mandatory. It is possible to only look for *feasible* points that satisfy a set of implicit relations. For example, a profile describing the frequency response desired for a filter circuit. Eldo provides a number of methods for specifying a waveform to match, or by simply defining point(s) that correspond to the desired behavior. These methods may also be combined.

Before continuing our presentation, we give some useful definition and mathematical notations. The design variables are denoted by:

$$\mathbf{x} = (x^{(1)}, x^{(2)}, \dots, x^{(N)})$$

a vector of real numbers of dimension  $N$ . Therefore the space of variables is denoted by  $\Re^N$ . The scalar product is denoted by  $\langle u|v \rangle = \sum_i u_i v_i$  onto the space  $\Re^N$ , and its associated norm is  $\|u\|$ .

The information the user has to provided are referred to as the *problem statement*. The designer provides a model for the optimization problem, which is simply the minimization or maximization of a function  $x \rightarrow f(x)$  subject to the *constraints* on its variable.

For example, the specification of the length  $l$  and the width  $w$  of a MOS transistor will be done through a **.PARAMOPT** command:

```
.PARAMOPT
+ l=(10u,2u,100u)
+ w=(60u,2u,200u)
```

Using our mathematical notations, the vector of design variables is:

$$\mathbf{x} = (x^{(1)}, x^{(2)}) = (l, w)$$

and the **.PARAMOPT** commands can be expressed as:

$$2 \times 10^{-6} \leq l \leq 10^{-4}$$

and

$$2 \times 10^{-6} \leq w \leq 2 \times 10^{-4}$$

The first value given in the command **.PARAMOPT l=(10u,2u,100u)** is used to initialize the optimization algorithm. On the other hand, the objective function will be specified by complementing a **.EXTRACT** or a **.MEAS** command. For example, the minimization of the noise at frequency 500MHz (the function we denote  $x \rightarrow f(x)$ ) for an amplifier will be specified with:

```
.EXTRACT noise
+ LABEL=snf_500 yval(snf, 500meg)
+ GOAL=MINIMIZE
```

The problem we have specified enters in the class of bound constrained problems. Using the notation of Optimization (or Mathematical Programming), we will write this problem as follows:

Minimize	$f(x)$
Subject to	$x_l \leq x \leq x_u$

We might also have decided to minimize the noise for an amplifier at frequencies 500MHz and 700MHz:

```
.EXTRACT noise
+ LABEL=snf_500 yval(snf, 500meg)
+ GOAL=MINIMIZE

.EXTRACT noise
+ LABEL=snf_700 yval(snf, 700meg)
+ GOAL=MINIMIZE
```

yielding two extracted functions,  $F(x) = (F^{(1)}(x), F^{(2)}(x))$ . The associated optimization problem would be:

$$\begin{aligned} \text{Minimize} \quad & f(x) = F^{(1)}(x) + F^{(2)}(x) \\ \text{Subject to} \quad & x_l \leq x \leq x_u \end{aligned}$$

Once we have defined the design parameters and the design objective, the user must add the **.OPTIMIZE** command to the netlist file, and starts the execution of Eldo.

After the optimization algorithm has been applied to the model, the designer must be able to recognize whether it has succeeded in its task of finding a solution. In many cases, there are optimality conditions for checking that the current set of unknowns is a solution of the problem. If the optimal conditions are not satisfied, they may give useful information on how the current estimate of the solution can be improved. The designer will find this kind of information within the output results of Eldo.

From the optimizer's point of view, it is important to understand that the only way to check optimality (or success) is to compute some measure of optimality and feasibility at the current iterate. There are no other criterion. If this optimality measure is less than some specified tolerance, the current point is considered as optimal. The fundamental notions of optimality and feasibility are considered in a semi-rigorous way in the section "[Robust Optimization Using Corners](#)" on page 20-8. The section presents and motivates the ideas as concise as possible, it is not mandatory for users who wish to avoid technical details.

## Designing a Low Noise Amplifier Example

The Low Noise Amplifier (LNA) architecture is a fully balanced dual-gain amplifier to achieve gain, linearity and the noise specifications for a Zero-IF receiver architecture. Output power matching is not required since the output load is given by the integrated mixer.

We aim at showing how the Eldo optimizer can be used on this example. The architecture of the LNA is based on the 2.45GHz Switched-Gain CMOS LNA [9]. The high-gain stage is a fully balanced cascade configuration with an integrated LC tank as the load. This design reduces the Miller effect and improves isolation (-S12). the low-gain stage consists of two NMOS devices used as switches to achieve the required linearity and insertion loss.

### Problem definition

The main characteristics are: Voltage Gain (AV), Input Power Matching (S11), Noise Figure (NF), and Input Third Order Intercept Point (IIP3).

The design parameters are:

- Input Network Model (CPIN1, CPIN2, LSIN,CSIN)
- Source Inductor (associated with serial resistance)
- NMOS Width (composed of 10mm length fingers)

The input network consists of a serial capacitor (CSIN) used to isolate the LNA DC from the input. A  $\pi$ -network (consists of CPIN1, CPIN2, LSIN) enables simultaneous input power and noise matching. Changing the source inductor (LS) and the width of the NMOS through the number of fingers (N1) will improve noise matching, input power matching and linearity characteristics such as IIP3 (the current and the load values are fixed to 6mA and  $200\Omega$  respectively).

### Analyses

Using a single test bench, different analyses were simulated to extract the following key specifications:

- An AC analysis in order to extract Return Loss and the Voltage Gain.
- A NOISE analysis for extracting the Noise Figure.
- A multi-tone SST analysis for the IIP3.

```
.PARAM VG=0.6 VD=1.8
.PARAM FUND1=2.45G FUND2=2.46G PIN=-50
.PARAM IS=6m ROUT=200 RS=LS/1n

VIN IN 0 RPORT=50 IPOR1T=1 AC 1 FOUR FUND1 FUND2 PDBM (1,0) PIN -90
+ (0,1) PIN -90
VOUT OUT 0 RPORT=ROUT IPOR1T=2
```

```
.DC
.AC LIN 11 2G 3G
.SST FUND1=FUND1 NHARM1=5 FUND2=FUND2 NHARM2=5
.NOISE V(OUT) VIN 10
.SNF INPUT=(VIN) OUTPUT=(VOUT)

.PLOT AC SDB(1,1)
.PLOT NOISE DB(SNF) DB(NFMIN)
.DEFWAVE AV_DB=VDB(OUT)-VDB(IN)
.PLOT AC W(AV_DB)
```

The **.DEFWAVE** command was used to define the voltage gain (dB).

## Design Variables

For the optimization analysis, each parameter has an initial value, together with lower and upper bounds. For example, the capacitor CPIN1 has an initial value of 0.1p with lower and upper bounds of 0.10p and 10p respectively. The variables LS and N1 are specified using the fourth argument of the **.PARAMOPT** command. It means that these parameters are allowed to lie only on a discretized grid (the fourth parameter gives the step of this grid). Refer to “[Discretization of Design Parameters](#)” on page 20-26 for details on this feature. Our problem has bound constraints on variables:

.PARAMOPT CPIN1=(0.1p,0.10p,10p)	$! 0.1 \times 10^{-12} \leq x^{(1)} \leq 10 \times 10^{-12}$
.PARAMOPT CPIN2=(0.1p,0.10p,10p)	$! 0.1 \times 10^{-12} \leq x^{(2)} \leq 10 \times 10^{-12}$
.PARAMOPT LSIN=(0.1n,0,30n)	$! 0 \leq x^{(3)} \leq 30 \times 10^{-9}$
.PARAMOPT CSIN=(1p,0.10p,10p)	$! 0.1 \times 10^{-12} \leq x^{(4)} \leq 10 \times 10^{-12}$
.PARAMOPT LS=(0.25n,0,3n,0.25n)	$! 0 \leq x^{(5)} \leq 3 \times 10^{-9}$
.PARAMOPT N1=(30,10,50,5)	$! 10 \leq x^{(6)} \leq 50$

The degenerative integrated inductor (LS) would typically come from a design kit library. In our context, it is useful to specify the unit step (0.25n) to obtain the optimal matching inductor.

## Design Objective and Extracted Measures

To specify design objectives, the keyword **GOAL=<goal>** is used on the **.EXTRACT** command, this describes the ideal target we would like to reach for this specification with the optimization process.

```
.EXTRACT AC LABEL=AV_db@2.4GHz           ! AC analysis, measure F(1)(x)
+ YVAL(WR(AV_DB),2.4G)                   ! Residual function r(1)(x) = F(1)(x) - 20
+ GOAL=20

.EXTRACT AC LABEL=AV_db@2.5GHz           ! AC analysis, measure F(2)(x)
+ YVAL(WR(AV_DB),2.5G) GOAL=20

.EXTRACT AC LABEL=S11_db@2.4GHz          ! AC analysis, measure F(1)(x)
+ YVAL(SDB(1,1),2.4G) GOAL=-15
```

```

.EXTRACT AC LABEL=S11_dB@2.5GHz
+ YVAL(SDB(1,1),2.5G) GOAL=-15

.EXTRACT NOISE LABEL=NF_dB@2.4GHz           ! Noise analysis, measure F(5)(x)
+ YVAL(DB(SNF),2.4G)
+ GOAL=MINIMIZE

.EXTRACT NOISE LABEL=NF_dB@2.5GHz
+ YVAL(DB(SNF),2.5G)
+ GOAL=MINIMIZE

.EXTRACT NOISE LABEL=NFMIN_dB@2.4GHz
+ YVAL(DB(NFMIN),2.4G)
+ GOAL=MINIMIZE

.EXTRACT NOISE LABEL=NFMIN_dB@2.5GHz
+ YVAL(DB(NFMIN),2.5G)
+ GOAL=MINIMIZE

```

The specified design objectives finally lead to the following optimization problem:

$$\begin{aligned}
 \text{Minimize} \quad f(x) &= \sum_{i=1}^{\text{NO}} f^{(i)}(x) + \frac{1}{2} \sum_{i=1}^{\text{NR}} r^{(j)}(x)^2 \\
 (\text{P}) \quad \text{Subject to} \quad x_l^{(i)} \leq x^{(i)} \leq x_u^{(i)}, \text{ for } i = \{1, \dots, N\}
 \end{aligned}$$

where we have NR=4 residual functions and NO=4 functions to minimize.

The optimization process stops when the accuracy on the design variables need not be improved further, or more precisely, when the current point  $\bar{x}$  satisfies the *optimality condition*. This optimality property can be checked in the Eldo output (.chi file):

```

*****
***** OPTIMIZATION RESULTS *****
Objective function      =    4.0823097e+00
Optimality              =    1.3731107e-05
Feasibility             =    0.0000000e+00
Circuit simulations     =          226
Number of iterations    =          30

Exit in normal mode with return code = 0

Diagnostics: The optimization has been succesful.
The required accuracy has been achieved.

*****

```

The value labeled “Optimality (gradient)” is the norm of the projected gradient of the objective function. A small value indicates that optimality condition is satisfied.

The design parameters found by the optimizer provides a voltage gain close to 20dB, a return loss of 12dB and an IIP3 of 0dBm. The value of the Noise Figure (NF) is similar to the value of the Minimum Noise Figure (NFMIN) thus the noise matching is correct.

***** MEASUREMENTS *****					
NAME	GOAL VALUE	LOWER BOUND	UPPER BOUND	WEIGHT VALUE	FINAL VALUE
AV_DB@2.4GHZ	2.00000e+01			1.00000e+00	21.7
AV_DB@2.5GHZ	2.00000e+01			1.00000e+00	19.8
S11_DB@2.4GHZ	-1.50000e+01			1.00000e+00	-12.5
S11_DB@2.5GHZ	-1.50000e+01			1.00000e+00	-10.0
NF_DB@2.4GHZ				1.00000e+00	1.0
NF_DB@2.5GHZ				1.00000e+00	1.0
NFMIN_DB@2.4GHZ				1.00000e+00	971.3M
NFMIN_DB@2.5GHZ				1.00000e+00	1.0

## Robust Optimization Using Corners

This example illustrates the combination of optimization and **.ALTER** commands. For this purpose we developed a new architecture based on the concepts of multi-context of simulation and multi-netlist optimization. This more general architecture will replace the current one for future releases.

### Problem definition

Nominal optimization focuses on finding the *best* design parameters for one *nominal* operating condition of the circuit. Typically the power supply, the ambient temperature and the process technology are given their nominal value, and the best set of design parameters is found by the optimizer, given its targets. But if the circuit has to operate under a variety of operating conditions, nothing guarantees that this will still be the case, if the design parameters are simply set to these optimal nominal values. Maybe if the power supply level is slightly changed, the circuit will fail.

‘Robust optimization’ focuses instead on finding the ‘best’ set of parameters that fulfill the specifications across a certain range of operating conditions. For example, if a circuit is supposed to operate between 1.7 and 1.9V, and from -25C to +100C, and accommodate variations in the process (defined by ‘corner’ device model libraries) you might want to optimize the circuit so that its performance is such or such whatever the power supply level and the temperature, and the corner. In this case the optimization targets might be upper or lower bounds on certain characteristics (for ex. the DC consumption has to be ‘lower than 50uA’, in all operating conditions). Or, targets might be set as targets for the average value of a given specification.

Operating conditions are conveniently defined with **.ALTER** sections in Eldo. In each **.ALTER** section, a specific combination of parameters defining the operating conditions (typically the

power supply level and the temperature) and corner device model library, can be defined. The optimization commands (design variables definition, design objective definitions, and optimize command from the main netlist are then interpreted to span the main netlist conditions *and* the various combinations defined through the .alter sections. Obviously these types of optimizations are usually more costly than simple nominal optimizations. Eldo can distribute the necessary simulation on multi-processor machines, thus potentially accelerating the process.

## Circuit statements

The following netlist is available in the directory: `$anacad/eld0/$eldover/examples/optimizer`

```
* Operational amplifier
m16 vdd bias m17d vdd pch_33 w='ws*4.8u' l=0.8u
m17 m17d m17d vss vss nch_33 w='ws*3.2u' l=0.8u
m15 m15d m17d vss vss nch_33 w='ws*3.2u' l=0.8u
m13 m13d m17d vss vss nch_33 w='ws*3.2u' l=0.8u
m14 vdd m15d m15d vdd pch_33 w='ws*1u' l=0.8u
m12 vdd m13d m13d vdd pch_33 w='ws*4.8u' l=0.8u
m3 vdd m13d m1d vdd pch_33 w='ws*4.8u' l=0.8u
m4 vdd m13d m2d vdd pch_33 w='ws*4.8u' l=0.8u
m11 vdd m13d out vdd pch_33 w='ws*16u' l=0.8u
m1 mld inp com com nch_33 w='ws*0.8u' l=0.6u
m2 m2d inn com com nch_33 w='ws*0.8u' l=0.6u
m9 com m17d vss vss nch_33 w='ws*3.2u' l=0.8u
m5 m2d m15d m7d vdd pch_33 w='ws*2.4u' l=0.8u
m6 m1d m15d m8d vdd pch_33 w='ws*2.4u' l=0.8u
m7 m7d m7d vss vss nch_33 w='ws*1.6u' l=0.8u
m8 m8d m7d vss vss nch_33 w='ws*1.6u' l=0.8u
cx m8d out c_comp ! lousy compensation cap.
cl out 0 1p ! load cap.
m10 out m8d vss vss nch_33 w='ws*13u' l=0.8u

.connect out inn ! follower connection

.op

.tran 1n 400n
.plot tran v(inp) v(out)

.option tuning=vhigh noascii nomod

vdd vdd 0 1.65V
vss vss 0 -1.65V
vinp inp 0 pulse -0.9 0.9 10n 1n 1n 200n 400n ac 1 0
vb bias 0 0.55V

* Typical model for 3.3V devices
.lib ./cln90g_1k.eld0 TT_33
```

## ALTER sections

We use the following corners:

```
* SS_33 : Slow NMOS Slow PMOS model for 3.3V devices
```

```
.alter label=SS_33
.lib ./cln90g_1k.eldo SS_33

* FF_33 : Fast NMOS Fast PMOS model for 3.3V devices
.alter label=FF_33
.lib ./cln90g_1k.eldo FF_33

* SF_33 : Slow NMOS Fast PMOS model for 3.3V devices
.alter label=SF_33
.lib ./cln90g_1k.eldo SF_33

* FS_33 : Fast NMOS Slow PMOS model for 3.3V devices
.alter label=FS_33
.lib ./cln90g_1k.eldo FS_33
```

## Analyses and design objectives

Two analyses were specified to define the following design objectives

- An transient analysis in order to extract the output voltage.
- A DC analysis for extracting the IDS current.

```
.extract tran label=rslope slope(v(out),0,0,140n)
+ goal=80e6
+ weight=1e3

.extract dc    label=ibias id(m16)
.param scal_i=1e4
.extract dc label=scal_ibias scal_i*id(m16)
+ goal=minimize
+ weight=1
```

We used the constant parameter `scal_i` in order to rescale the extracted measure `ibias`. The values taken by `rslope` and `scal_ibias` have the same order of magnitude. The specified weight for the measure `rslope` was introduced to find a solution that minimize the rise time at the expense of larger `ibias` values. Users can experiment different choices of weights.

## Design Variables

We use a shrink parameter `ws` for the width of each MOS instantiated in the circuit netlist. The capacitor is also optimized.

```
.PARAMOPT
+ ws=(2,0.5,4)
+ c_comp=(1p,0.01p,100p)
```

## Results of optimization

This example must be run with the following commands:

```
.optimize tol_opt=1e-2
.option opseldo_alter opseldo_output=1
```

where the option **opseldo\_alter** informs Eldo that optimization must be performed on all **.ALTER** sections. The results of optimization are placed in a new file with extension **.mtm**. We give here the sections containing the final extracted measures:

```
***  
***      Status Of Design Objectives  
***  
  
***  
***      Minimize/Maximize Objectives  
***  
  
Name      Current Value      Weight  
*****  
SCAL_IBIAS 3.2064693e-01 1.0000000e+00  
  
SCAL_IBIAS 4.3131626e-01 1.0000000e+00  
  
SCAL_IBIAS 2.3031284e-01 1.0000000e+00  
  
SCAL_IBIAS 4.0472609e-01 1.0000000e+00  
  
SCAL_IBIAS 2.4545281e-01 1.0000000e+00  
  
***  
***      Goal Objectives  
***  
  
Name      Current Value      Goal Value      Weight  
*****  
RSLOPE   7.4257089e+07 8.0000000e+07 1.0000000e+03  
  
RSLOPE   9.9268448e+07 8.0000000e+07 1.0000000e+03  
  
RSLOPE   5.3697962e+07 8.0000000e+07 1.0000000e+03  
  
RSLOPE   9.2769193e+07 8.0000000e+07 1.0000000e+03  
  
RSLOPE   5.7590289e+07 8.0000000e+07 1.0000000e+03
```

## Basic Notions and Definitions in Optimization

Optimization involves the study of optimality criteria for problems, the determination of algorithmic methods of solution, the study of the structure of such methods, and computer experimentation. When optimizing a circuit, the mathematical aspects are not great concern. However, since constrained optimization problems are more complicated than unconstrained problems. It is important to have some notions or basic knowledge of the theory.

An optimization problem is the minimization or maximization of a function subject to a set of constraints on its variables. A formal definition of Optimization could be:

Given a *feasible* set  $X$  and the *objective* function ( $x \in X \rightarrow f(x) \in \mathfrak{R}$ ), the aim is to find  $\bar{x} \in X$  such that, for all  $x \in X$ , there holds  $f(\bar{x}) \leq f(x)$ .

## Notion of Feasibility

In the Eldo optimizer, only the case where the set  $X$  is a subset of the Euclidean space  $\Re^N$  is considered. The feasible set  $X$  is defined by the constraints, i.e. given a number  $M$  of functions  $x \in X \rightarrow c_j(x) \in \Re$ . The *feasible* set is defined as:

$$X = \{x \in \Re^N \mid x_l \leq x \leq x_u, c_l \leq c(x) \leq c_u\}$$

If any point  $X'$  satisfies all the constraints, it is said to be a *feasible point*. The relation  $x_l \leq x \leq x_u$  must be understood component-wise, where  $x_l$  and  $x_u$  are  $N$ -vectors, i.e. the  $i$ th component of the design variables  $x$ , has upper and lower boundaries:

$$x_l^{(i)} \leq x^{(i)} \leq x_u^{(i)}$$

Some components of  $x$  or  $c(x)$  may lack upper or lower boundaries, these instances are handled by setting the appropriate components of  $x_l$ ,  $x_u$ ,  $c_l$  and  $c_u$  to  $-\infty$  or  $+\infty$  values.

One may illustrate the role of the constraints when dimension  $N=2$  by drawing the zero contour of each constraint function. For an equality constraint, the line itself is the set of feasible points; for an inequality constraint the line marks the boundary of the feasible region and the infeasible side is conventionally shaded. This is shown in [Figure 20-2](#). Case (1) has constraints:

$$0 \leq x^{(1)} \leq \infty$$

$$0 \leq x^{(2)} \leq \infty$$

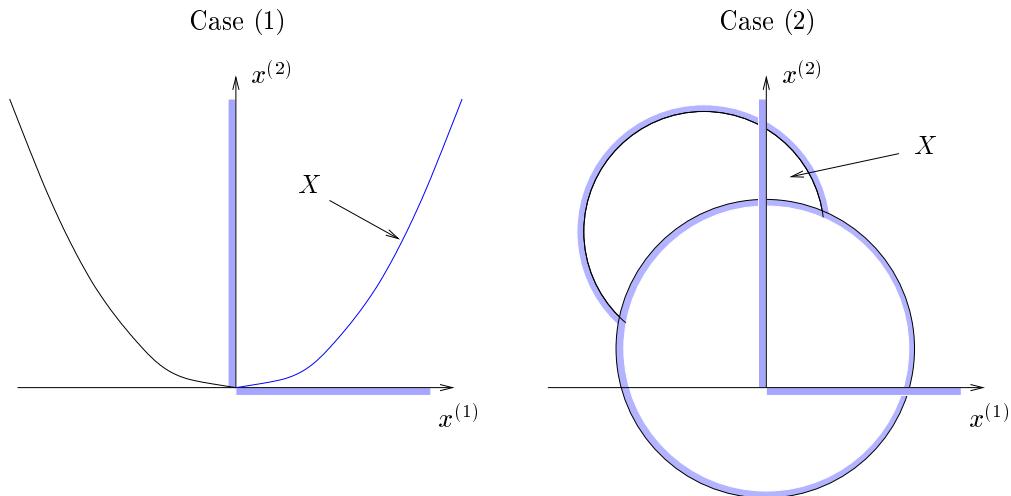
$$x^{(2)} = (x^{(1)})^2$$

The last constraint (the parabola) can be written as  $0 \leq c^{(1)}(x) \leq 0$  where we have defined the constraint  $c^{(1)}(x) = (x^{(1)})^2 - x^{(2)}$ .

It means that equality constraints are simply considered as special cases of inequality constraints.

Case (2) illustrates a slightly more complicated situation. The feasible set lies at the intersection of the interior of the smallest circle and the exterior of the biggest circle. This domain is also cut off by the bound constraints  $x \geq 0$ .

**Figure 20-2. Examples of feasible regions**



## Active or Binding Constraints

Active constraints at any point  $x$  are defined by the index set:

$$A(x) = \left\{ i \in I; c^{(i)}(x) = c_l^{(i)} \text{ or } c^{(i)}(x) = c_u^{(i)} \right\}$$

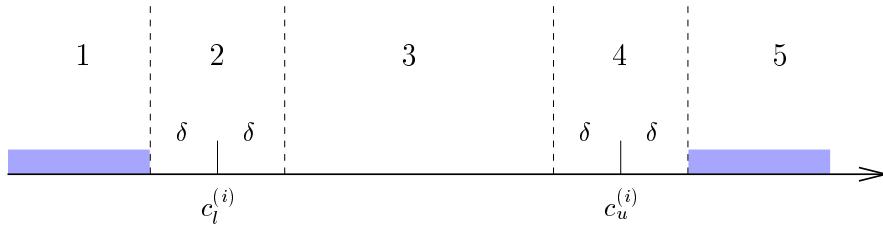
so that  $x$  is on the *boundary* of its feasible region. In many cases the function values will be the result of extensive computation, possibly involving an iterative procedure that can provide rather few digits of precision at reasonable cost. For instance, suppose that a constraint function  $c^{(i)}(x)$  is computed for some relevant  $x$  and if the first 6 digits are known to be correct. A constraint should be considered as active at its upper bound (or lower bound), if the magnitude of the difference between the values  $c^{(i)}(x)$  and  $c_u^{(i)}$  (or  $c_l^{(i)}$  respectively) is less than some tolerance of order  $1.0 \times 10^{-6}$ .

This tolerance, say  $\delta$ , specifies how accurately the constraints should be satisfied. It defines the maximum absolute violation in nonlinear constraints at a feasible point.

A constraint is considered satisfied if its violation does not exceed the tolerance  $\delta$ .

In [Figure 20-3](#), we illustrate the feasible region for the constraints  $c_l^{(i)} \leq c_I^{(i)}(x) \leq c_u^{(i)}$ .

**Figure 20-3. Illustration of the constraints  $c_l^{(i)} \leq c^{(i)}(x) \leq c_u^{(i)}$**



The constraints are considered satisfied if  $c^{(i)}(x)$  lies in region 2, 3 or 4, and inactive if  $c^{(i)}(x)$  lies in region 3. The constraint  $c_l^{(i)} \leq c^{(i)}(x)$  is considered active in region 2, and violated in region 1. Similarly,  $c^{(i)}(x) \leq c_u^{(i)}$  is active in region 4, and violated in region 5. For equality constraints  $c_l^{(i)} = c_u^{(i)}$ , regions 2 and 4 are the same, and region 3 is empty. The default value is appropriate when the constraints contain data to about that accuracy.

Note that specifying an appropriate tolerance on feasibility may lead to several savings, by allowing the optimization procedure to terminate when the difference between function values along the search direction becomes as small as the absolute error in the values. Please refer to the [.OPTIMIZE command](#) described in “[The Optimization Command](#)” on page 20-38, for the specification of the **TOL\_FEAS** argument.

## Identification of Active Constraints

When inequality constraints are present, the task of optimization is considerably more difficult than the case of constrained problems having only equalities. It is simply that, one does not know in advance which inequalities will play a role at the optimal point.

Determining which of the constraints  $c_1 \leq c(x) \leq c_u$  are *active* at a solution that gives  $3^M$  possibilities (the number 3 comes from the fact that for one inequality there are three possibilities  $c_i(x) = c_l^{(i)}$  or  $c_i(x) = c_u^{(i)}$ , or  $c_l^{(i)} < c(x) < c_u^{(i)}$ ). Since we do not know the optimum a priori, it follows that there exist  $3^M$  ways to put the inequality constrained problem into the form of an equality constrained problem. Despite the ‘relative simplicity’ of solving equality constrained problems, we cannot face the exponential number of  $3^M$  different cases.

## First Order Optimality Conditions

Let us recall that the gradient of a function  $x \in \Re^N \rightarrow f(x) \in \Re$  is denoted by  $\nabla f(x)$ , and defined as the vector of partial derivatives:

$$\nabla f = \left( \frac{\partial f}{\partial x^{(1)}}, \frac{\partial f}{\partial x^{(2)}}, \dots, \frac{\partial f}{\partial x^{(N)}} \right)$$

The concept of a stationary point for which  $\nabla f(\bar{x}) = 0$  is fundamental to the subject of unconstrained optimization, and is a necessary condition for a local minimizer.

For unconstrained minimization the necessary conditions  $\nabla f(\bar{x}) = 0$  and  $(s, \nabla^2 f(\bar{x}) \cdot s) \geq 0$  for all direction  $s$ , illustrate the requirement for zero slope and non-negative curvature in any direction at point  $\bar{x}$ , that is to say, there is no descent direction at  $\bar{x}$ .

In constrained optimization, there is an additional complication of a feasible region. Suppose that we are solving the following problem:

	Minimize	$f(x)$
(P)	Subject to	$c(x) \geq 0$

A local minimizer must be a feasible point, and satisfy a set of other conditions that illustrate the need for the non existence of a feasible descent directions at point  $\bar{x}$ . There is no direction  $s$  such that:

- Any feasible incremental step  $d$  lies along a feasible direction  $s$  which satisfies  $(s, \nabla c_i(\bar{x})) < 0$  for all  $i \in \bar{A} = A(\bar{x})$ ,
- $f(x)$  has negative slope along  $s$ , that is  $(s, \nabla f(\bar{x})) < 0$ .

These conditions are satisfied if only both the conditions:

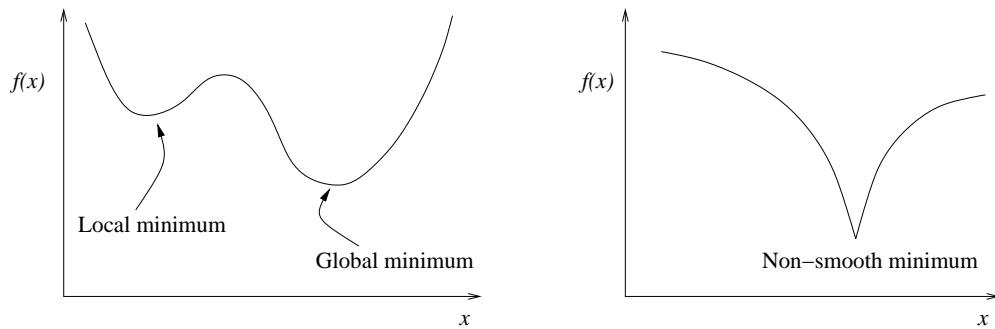
$$\nabla f(\bar{x}) = \sum_{i \in \bar{A}} \bar{\lambda}_i \cdot \nabla c_i(\bar{x}) \text{ and } \bar{\lambda}_i \geq 0 \text{ hold at point } \bar{x}.$$

The vector  $\lambda = (\lambda_1, \lambda_2, \dots, \lambda_N) \geq 0$  is the vector of Lagrange multipliers. These two relations constitute the first order optimality conditions. An optimum point satisfies these conditions necessarily.

## Global and Local Optimization

The fastest optimization algorithms only search for a *local* solution, at a point at which the objective function is smaller than all other feasible points in its vicinity. They do not always find the best of all such minima, that is, the *global* solution (see [figure 20-4](#)). The reason for this is that practical methods for finding global optima are (at present time) too expensive in all but the most specialized cases.

**Figure 20-4. Different types of minima**



General non-linear problems may possess local solutions that are not global solutions. Global solutions are highly desirable, but they are usually difficult to identify and even more difficult to locate. Unless very strong assumptions are made about the functions that define the optimization problem in question, characterizations of global minima are almost never possible, so even if one is found, we may never know.

## Smooth and Non-smooth Problems

A difficult situation arises when the functions  $f$  and  $c_i$  do not have continuous derivatives, this is referred to as non-smooth or non-differentiable optimization. In this case, methods for smooth problems (such as the SQPAL algorithm) are not appropriate, because non-smooth minima (shown in figure 20-4) do not satisfy the same conditions as smooth minima.

It is assumed in problem (P), that the functions  $x \rightarrow c_i(x)$  are continuous. It is also assumed that  $x \rightarrow f(x)$  is continuous for all  $x \in X$  and preferably for all  $x \in \Re^N$ .

## Discrete Optimization

In some optimization problems, the variables only make sense if they take on integer or discrete real values. The obvious strategy of ignoring the integrality requirement, solving the problem with real variables, and then rounding all the components to the nearest integer is not guaranteed to give a solution that is close to optimal. Problems of this type should be handled using the tools of discrete optimization. The mathematical formulation is changed by adding the constraint “ $x_i$  integer for all  $i \in K$ ”, where  $K$  is a subset of  $\{1, \dots, N\}$ .

Continuous optimization problems are easier to solve, because the smoothness of the functions makes it possible to use the *objective* function and constraint information at a particular point  $x$  to deduce information of the function's behavior at all points close to  $x$ . The same statement cannot be made about discrete problems, where points that are close in some sense may have markedly different function values. Moreover, the set of possible solutions is too large to make an exhaustive search for the best value in this finite set. When models contain variables allowed

to vary continuously and others that can attain only integer values. We will refer to these as mixed-integer programming problems (MIP).

Minimize	$f(x)$
Subject to	$x_l \leq x \leq x_u$
	$c_l \leq c(x) \leq c_u$
	$x_i$ integer for all $i \in K$

Please refer to “[Discretization of Design Parameters](#)” on page 20-26, for details about this feature.

## Algorithmic Material

### The SQPAL Algorithm

The Eldo optimizer uses the SQPAL solver, which is based on a *sequential quadratic programming* method (SQP), using an augmented Lagrangian approach (AL) [2] for solving the sequence of quadratic problems (QP) involved in the SQP iterations. It is efficient for solving small and medium scale problems belonging to the following classes:

- unconstrained problems,
- bound constrained problems,
- systems of nonlinear equations,
- general (equality and inequality) constrained problems.

The next sections have been written in order to present the basic concepts of algorithm, and specifically the Newtonian methods in Optimization. Users who wish to consult recent books giving a broader view of optimization techniques should refer to Nocedal and Wright [7], and Fletcher [5]. For more advanced books one may consult Bonnans, Gilbert, Lemarechal and Sagastizabal [1] and Gill, Murray and Wright [6].

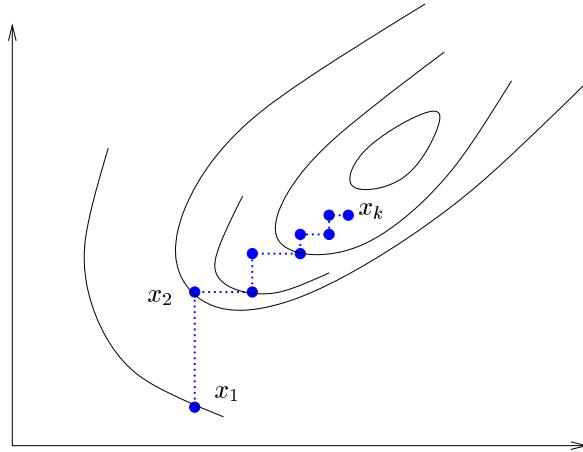
### The Need for Specific Optimization Methods

The designer is invited to think about the practical situation, where these types of problems are solved from scratch. The designer should consider some ad-hoc methods. For instance, if the problem is only two or three variables, a rough method is to generate points  $\{x_1, x_2, x_3, \dots\}$  at random in a given region, and select the one that gives the optimum value over a large number of trials. The designer can easily believe that the amount of circuit simulations would increase exponentially, and that this method has no reason to find an *optimal* point.

Another simple method would be the *alternating variables* method in which on iteration  $k$ , the  $k$ th component of the variable vector  $x$  is changed in an attempt to reduce the objective function, and the other variables are held fixed. When all the variables have been changed, the whole

cycle is repeated until convergence occurs. In practice, the *alternating variables* method is very inefficient and unreliable. Unfortunately, the process of the iteration is characterized by the oscillatory behavior shown in [Figure 20-5](#). The reason is that this method ignores the possibility of correlation between the variables, that causes the search direction parallel to the second component  $x^{(2)}$  to destroy the property that  $x_2$  is the minimizer in the direction parallel to the first component  $x^{(1)}$ . Moreover, it is possible to find problems for which this method fails to converge.

**Figure 20-5. The ad-hoc method of alternating variables**



This discussion shows that the choice of the algorithmic approach in the Eldo optimizer depends on the general features of the circuit optimization problems. In this category of problems, it is usual that most of the CPU time is spent in the circuit simulations. Thus, when developing the algorithm, the principal argument for the chosen approach is to keep the number of simulations as small as possible.

The optimizer is based on a sequential quadratic programming method (SQP). This category of methods has proved to be effective in practice. They usually require fewer simulations than some other methods, at the expense of solving a complicated subproblem.

## Extension of Newtonian Methods

General purpose circuit simulation programs such as Eldo provide a variety of analysis types including DC analysis, time-domain transient analysis, AC and noise analysis. Most of these analyses involve computing the solution of a system of coupled nonlinear differential-algebraic equations. This type of equations arise as a result of the properties of general electronic circuits. A numerical integration method converts a set of nonlinear differential equations into a set of nonlinear algebraic equations. The most commonly used method to solve nonlinear equations is the Newton method [8][3].

Consider a mapping  $z \rightarrow F(z)$ . We want to solve numerically for  $z \in \Re^N$  the system with  $N$  equations in the  $N$  unknowns:

$$F(z) = 0$$

The Newton method generates a sequence  $\{z_k\}$  by the recurrence formula:

$$z_{k+1} = z_k + s_k$$

where the step  $s_k$  solves the equation  $F(z) = 0$  linearized at point  $Z_k$ :

$$F(z_k) + F'(z_k) s_k = 0$$

In circuit terms, the Newton method replaces each nonlinear device in the circuit by a linearized model based on operating point information. This process converts the nonlinear circuit into a linear equivalent network.

The previous equation has a unique solution if  $F'(z_k)$  is non-singular. In this case:

$$s_k = -F'(z_k)^{-1} F(z_k)$$

The process of optimization is similar, since it relies on algorithmic techniques associated with the concept of *descent methods* and the extension of Newtonian methods. The numerical optimization method converts the problem statement:

Minimize	$f(x)$
Subject to	$x_l \leq x \leq x_u$
	$c_l \leq c(x) \leq c_u$

into a set of nonlinear *equations* and *inequations*, characterizing the *first order optimality conditions* [5] [6].

## Simplified Description of the Algorithm

For instance, to solve the unconstrained problem:

Minimize	$f(x)$
----------	--------

a possibility is to solve its optimality condition  $\nabla f(x) = 0$ , taking  $F = \nabla f$ . The Newton equation can be written:

$$\nabla f(x_k) + \nabla^2 f(x_k) s_k = 0$$

This equation is the first-order optimality condition of the *quadratic problem*:

(QP) <sub>k</sub>	$\text{Minimize } q_k(s) = \langle \nabla f   s \rangle + \frac{1}{2} \langle H_k s   s \rangle$
-------------------	--

So the main work of the  $k$ th iteration consists in solving a quadratic problem in order to get an incremental step  $s_k$ . The linear part of the cost function  $\Phi_k(s)$  is obtained by using the gradient objective function, and the quadratic term is a quasi-Newtonian approximation (see Dennis and More [4]) of the objective curvatures.

Based on the previous remarks, we can outline a simplified structure of the  $k$ th iteration:

- Step Computation: determine a direction of search  $s_k$  by solving a tangent quadratic subproblem  $(QP)_k$ ,
- Step Assessment: find  $\alpha_k > 0$  in order to minimize a chosen merit function  $x \rightarrow \Phi(x)$  along the line  $\alpha \rightarrow x_k + \alpha s_k$ ,
- Update:  $x_{k+1} = x_k + \alpha_k s_k$ .

This approach is referred to as a descent method since the search direction  $s_k$  satisfies a descent property:

$$\langle s_k | \nabla \Phi_k \rangle < 0$$

where the vector  $\nabla \Phi_k$  represents the gradient of the chosen merit function (which is simply the objective function here  $\Phi = f$ ). The role of the line search algorithm is to *damp* the displacement  $s_k$ . But the most important feature of this algorithmic process is the following:

During optimization, simulations are required at two distinct stages:

- the computation of *derivatives* of the functions ( $F^{(1)}, \dots, F^{(NF)}$ ) involved in the problem statement,
- the step assessment procedure, or *line search* algorithm.

From the previous description of the SQPAL algorithm, it should be clear that one of the most demanding phase in terms of simulation is the computation of derivatives. We use finite differences in Eldo optimizer. Finite differencing is an approach to the calculation of approximate derivatives whose motivation comes from Taylor's theorem. Like many software package, OpsEldo performs automatic calculation of finite differences whenever the simulator is unable (or unwilling) to supply code to compute exact derivatives.

A popular formula for approximating the partial derivative:

$$\frac{\partial F}{\partial x^{(j)}}$$

at a given point is the *forward differences* or *one-sided differences*. We use a parameter, denoted by  $h_f$ , that controls the interval used to estimate gradients of the  $F^{(i)}$  functions by forward differences:

$$\begin{bmatrix} \frac{\partial F^{(1)}}{\partial x^{(j)}} \\ \frac{\partial F^{(2)}}{\partial x^{(j)}} \\ \vdots \\ \frac{\partial F^{(NF)}}{\partial x^{(j)}} \end{bmatrix} \approx \begin{bmatrix} \frac{F^{(1)}(x + h_f^{(j)}) - F^{(1)}(x)}{h_f^{(j)}} \\ \frac{F^{(2)}(x + h_f^{(j)}) - F^{(2)}(x)}{h_f^{(j)}} \\ \vdots \\ \frac{F^{(NF)}(x + h_f^{(j)}) - F^{(NF)}(x)}{h_f^{(j)}} \end{bmatrix}$$

One-sided difference estimates are used to ensure feasibility with respect to an upper or lower bound on  $x$ . If  $x$  is close to an upper bound, the trial intervals will be negative. The final interval is always positive.

- An approximation to the derivative (jacobian matrix) of  $F$  can be obtained by evaluating the mapping  $F$  at  $N + 1$  points and performing some elementary arithmetic.
- The resulting gradient estimates should be accurate to  $O(h_f)$  unless the functions are badly scaled.

## Known Limitations

This version of the optimizer has been enhanced to ensure more robustness regarding the treatment of constraint feasibilities. The SQPAL algorithm is able to make explicit allowance for infeasible constraints when computing the incremental steps obtained from a sequence of subproblems. This phenomenon is referred to as local infeasibility. However, some problems may still occur.

In constrained optimization, a state where no feasible solution exists can occur. In this state the constraints are inconsistent and the problem is infeasible. If all the constraints are bounds on the variables:

$$x_l^{(i)} \leq x^{(i)} \leq x_u^{(i)}$$

It is simple to determine whether a feasible point exists, since the  $i$ th is only inconsistent when:

$$x_l^{(i)} > x_u^{(i)}$$

Such infeasibilities are automatically trapped when parsing the **.PARAMOPT** command. Conversely, with general constraints there are no simple characteristics that identify whether a feasible solution exists.

The output solution from optimizer will (but not always) be a local optimum if the problem is feasible. When the problem is infeasible (or a difficulty arises) during the optimization phase, the previous version of the optimizer returned the best point obtained so far. This was considered as a failure of the algorithm. Such behavior is not very helpful since the user often requires an answer, even when none exists. Ideally an approximate solution that is feasible and produces a good objective value would be generated.

In order to cope with this limitation, the optimizer can run in one of two different modes:

- *normal* mode
- *feasible* or *relaxed* mode

The new algorithm enters the relaxed mode when a situation of local infeasibility is detected. This mode performs a shift of the general bounds on constraints. This may be considered as a relaxation of the initial problem, where the new objective is complemented by adding the minimization of the “sum of infeasibilities”. If a “near feasible” point can not be found, this is an indication that the model from which the problem was derived may be poor. A similar situation occurs when a circuit optimization problem has a solution whereas the mathematical problem does not, this caused by inaccuracies in the model. In both cases the *relaxed* mode should be prove valuable.

Consider the original problem:

	Minimize	$f(x)$
(P)	Subject to	$c(x) \geq 0$

The *relaxed* mode will consist in the following modification of the original problem,

	Minimize	$v \rightarrow \Pi(v)$
(P)	Subject to	$c(x) + v \geq 0$
		$v \geq 0$

where the slack variables are positive artificial variables  $v \geq 0$ . We allow constraint violations, since we will have  $c(x) \geq -v$ . The function  $v \rightarrow \Pi(v)$  is a penalty function, so that we allow the constraints to be violated but add a penalty to the objective for doing so.

Note that this approach has very interesting properties, for instance when the optimization problem is infeasible. In this case, optimal solution often violates only a small number of the constraints. Therefore, we have computed a point that satisfies many of the constraints, i.e., we have identified a large subset of constraints that is feasible. This is more informative for the user, than finding that the constraints are mutually feasible (this approach is closely related to the L1-norm regularization technique).

# Modeling Capabilities in Eldo

The modeling capabilities implemented in Eldo are designed to match the optimization functionalities. The designer's request for optimization, the definition of the parameters to be optimized, the design objectives and the goals that they should satisfy is done in the circuit netlist file. The optimization problem that will be solved by the Eldo optimizer is introduced in the next section.

## Optimization Variables Specification

The design parameters must be specified through the **.PARAMOPT** command:

```
.PARAMOPT variable=(  
+ init_value,  
+ [lower_bound | lower_percent% ],  
+ [upper_bound | upper_percent% ]  
+ [, increment])
```

### Parameters

**variable**      Name of the design variable(s). This parameter can be one of the following:

```
parameter_name |  
P(parameter_name) |  
P(subckt_name,subckt_parameter) |  
E(device,parameter) |  
M(model_name,model_parameter) |  
EM(device_name,model_parameter)
```

The character strings `parameter_name`, `subckt_name` and `model_name` may contain wildcards characters `*` and `?`. These features allow a *set* of parameters to be manipulated, rather than a *unique* parameter i.e. the **.PARAMOPT** command has two distinct behaviors; it can define the vector of design parameters `x`, using a *component-wise* specification, or a *generic* specification. These features are explained below:

**initial\_value**

Initial value of the design variable. Optional in some situations, please refer to “[Optional Initial Value](#)” on page 20-27.

**lower\_bound, upper\_bound**

Lower and upper bounds specified for the design variable. Unbounded (or free variables) must be specified using the star character `*`, for example:

```
.PARAMOPT var1=(1.0, 0.0, *)
```

specifies a non-negative  $0 \leq x \leq \infty$  parameter `var1` with the initial value set to 1.0. Free variables  $-\infty \leq x \leq \infty$  are specified using the triplet

```
.PARAMOPT variable=(init_value, *, *)
```

**lower\_percent%, upper\_percent%**

Percentages of the initial value. For example, the command:

.PARAMOPT var1=(5.0, 10%, 35%), specifies that the effective lower and upper bounds are  $x_l^{(1)} = 5.0 \times (1 - 0.1)$  and  $x_u^{(1)} = 5.0 \times (1 + 0.35)$ .

increment      Specifies a discretization for the final value of the design parameter.  
Optional. Please refer to “[Discretization of Design Parameters](#)” on page 20-26 for more information.

---

### **Caution**



The .PARAMOPT command supports fixed variables:  $x_l^{(i)} = x^{(i)} = x_u^{(i)}$ . For instance, it is possible to fix the number of fingers N1 when optimizing an NMOS device by specifying: .PARAMOPT N1 = (30, 30, 30). For efficiency, the command .PARAM N1=30 should be used for fixing design variables. The reason for introducing this feature in the .PARAMOPT command will be useful for future version in the context of sensitivity analysis.

---

## **Example 1**

To optimize the length  $l$  and width  $w$  of a MOS transistor. The parameter specification can be:

```
.PARAMOPT
+ l=(10u,2u,100u,0.01u)
+ w=(60u,2u,200u,0.01u)
```

Here, the optimization is initiated with a length of 10 microns and a width of 60 microns. The length is allowed to change between 2 microns and 100 microns by steps that are multiples of 0.01 microns. Similarly the width can take values between 2 microns and 200 microns, that differ from the initial value (60 microns) by a multiple of 0.01 microns.

## **Generic Specification of Design Parameters**

Suppose that an optimization has to be performed on the following circuit:

```
.MODEL NMOD1.1 NMOS level=53
+ lmin=0.1u lmax=0.2u
+ wmin=1u wmax=100u
+ vth0=0.2

.MODEL NMOD1.2 NMOS level=53
+ lmin=0.2u lmax=0.3u
+ wmin=1u wmax=100u
+ vth0=0.3

.MODEL NMOD1.3 NMOS level=53
+ lmin=0.3u lmax=1u
+ wmin=1u wmax=100u
+ vth0=0.4
```

```

vd d 0 dc 1.5
vs s 0 dc 0
vb b 0 dc 0
vg g 0 dc 3

rd1 d d1 1
rd2 d d2 1
m1 d2 g s b NMOD1 w=10u l=0.25u
m2 d1 g s b NMOD1 w=10u l=0.35u

.DC vd 3 1 0.1

.PLOT dc i(rd1) i(rd2)
.EXTRACT dc label=EX1 max(i(rd1)) GOAL=2m

```

Adding the following commands will *only* change the length of **M2**:

```

.OPTIMIZE
.PARAMOPT E(M2,l)=(0.35u,0.1u,1u,*)

```

The length of **M1** and **M2** can be changed with:

```

.OPTIMIZE
.PARAMOPT E(M*,l)=(*,0.1u,1u,*)

```

The initial values specified on the instances are kept unchanged. The following command will change lengths of **M1** and **M2**, the initial values are both set to  $0.35\mu$ .

```

.OPTIMIZE
.PARAMOPT E(M*,l)=(0.35u,0.1u,1u,*)

```

The following sequence of examples will affect the parameter **vth0**.

```

.OPTIMIZE
.PARAMOPT EM(M2,vth0)=(0.3,0.1,1,*)

```

Changes the parameter **vth0** of the model used by **M2**.

#### **Note**

For this kind of parameter, the initial value is ignored.

Using wildcards, the parameter **vth0** can be changed for both **M1** and **M2**, this can be achieved with:

```

.OPTIMIZE
.PARAMOPT EM(M*,vth0)=(*,0.1,1)

```

The following commands:

```

.OPTIMIZE
.PARAMOPT M(NMOD1.3,vth0)=(0.3,0.1,1,*)

```

and:

```
.OPTIMIZE
.PARAMOPT M(NMOD1.* ,vth0)=(0.3,0.1,1,*)
```

Will change the parameter `vth0` of the model `NMOD1.3` and all models `NMOD1.*` respectively.

**Note**

For this kind of parameter, the initial value is ignored.

## Discretization of Design Parameters

The [Figure 20-6](#) illustrates such situations. The continuous box represents the feasible domain specified by the upper and lower bounds, and the black bullets are the feasible points where the final parameters are allowed to lie. For example, when the lower bound is a finite number, the set of discretized points are as follows:

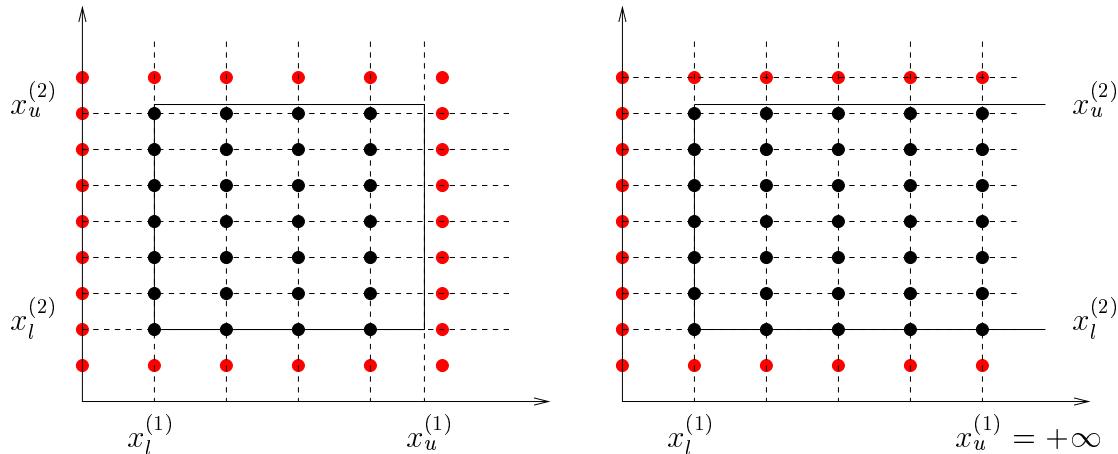
$$\{x_l^{(i)}, x_l^{(i)} + \delta_i, x_l^{(i)} + 2\delta_i, \dots\}$$

Where  $\delta_i > 0$  is the given increment. When the lower bound is infinite ( $x_l^{(i)} = \infty$ ), the grid is started from the upper bound if this one is finite. The set of discretized points is then:

$$\{x_u^{(i)}, x_u^{(i)} - \delta_i, x_u^{(i)} - 2\delta_i, \dots\}$$

When a design parameter is unbounded ( $x_l^{(i)} = -\infty$  and  $x_u^{(i)} = \infty$ ), the requirement of discretization is not considered.

**Figure 20-6. Discretization of final parameters**



The obvious strategy of ignoring the requirement of discretization is used, solving the problem with real variables, and then rounding all the components to the nearest point onto the grid. This strategy is not guaranteed to give solutions that are close to optimal. Increments on design parameters should only be used with design parameters for which rounding the optimized parameters is necessary.

## Optional Initial Value

By default, the **.PARAMOPT** command requires an initial value for the parameter to be optimized. However, it is possible to omit the parameter `initial_value`, and take the value that is available in the netlist. This can be achieved in two ways:

- Globally, the option **PARAMOPT\_NOINITIAL** can be specified to inform Eldo that initial values are not specified. The following statement must be used:

```
.PARAMOPT variable=
+ lower_bound, upper_bound
+ [, increment])
```

- For a single parameter, the following can be specified, and the initial value will be obtained from the netlist.

```
.PARAMOPT variable=
+ *,
+ lower_bound, upper_bound
+ [, increment])
```

## Specification of Correlated Parameters

Element parameters can be correlated during the optimization process. The parameters are specified through the command:

```
.CORREL expr element={expression}
```

Where `element` can take the form of:

```
E(element_name,element_parameter)
EM(element_name,model_parameter)
M(model_name,model_parameter)
parameter_name
```

### Note

The meaning of **.CORREL EXPR** is completely different to **.CORREL** in a Monte Carlo analysis.

The character string `expression` can contain references to **E()**, **EM()** and **M()** these quantities must be associated with the **.PARAMOPT** command. Therefore, in the following case, `p2` will not be affected by `p1`:

```
C1 1 0 p1
R1 1 0 p2
.PARAMOPT p1=(1n,0.1n,10n)
.CORREL expr p2=E(C1,c)
```

In the following case, `p2` will be affected by `p1`:

```
C1 1 0 1n
R1 1 0 p2
.PARAMOPT E(C1,C)=(1n,0.1n,10n)
.CORREL expr p2=E(C1,C)
```

The purpose of the **.CORREL** command is to avoid editing files when an optimization is required to be performed on a non-parametric netlist. For example:

```
LCR Parallel Network
.OPTION eps = 1.0e-6 numdgt=10
vin 1 0 ac 10
r2 1 2 50
r3 2 3 50k
r5 3 0 50
l1 2 3 100u
c4 2 3 1n
vin2 a 0 ac 10
ra a b 50
rb b c 50k
rc c 0 50
la b c 100u
ca b c 1n
.ac dec 10 1 1g
.PLOT ac vdb(2)
.PLOT ac vdb(b)
.EXTRACT ac LABEL=WIDTH
+ xdown(VDB(2),14.2,1)-xup(VDB(2),14.2,1)
+ GOAL=1e6
.EXTRACT ac LABEL=WIDTHB
+ xdown(VDB(B),14.2,1)-xup(VDB(B),14.2,1)
* Optimization commands
.OPTIMIZE
.PARAMOPT E(c4,C)=(1n,1n,20n)
.CORREL expr E(ca,C)='3* E(c4,C)/(2+1)'
```

## Problem Statement and Measurements Specification

We will refer to our general optimization as (P). Given the N-vector of variables  $x$ , our problem can be written as follows:

$$\begin{aligned}
 \text{Minimize} \quad f(x) &= \sum_{i=1}^{\text{NO}} \pm f^{(i)}(x) + \frac{1}{2} \sum_{j=1}^{\text{NR}} \mu_r^{(j)} r_j(x)^2 \\
 (\text{P}) \quad \text{Subject to} \quad x_l^{(i)} &\leq x^{(i)} \leq x_u^{(i)}, \text{ for } i = \{1, \dots, N\} \\
 c_E^{(i)}(x) &= e^{(i)}, \text{ for } i = \{1, \dots, NE\} \\
 c_l^{(i)} &\leq c_I^{(i)}(x) \leq c_u^{(i)}, \text{ for } i = \{1, \dots, NI\}
 \end{aligned}$$

In this problem the vectors  $(f^{(1)}, \dots, f^{(NO)})$  and  $(f^{(1)}, \dots, f^{(NR)})$  represent respectively, the NF-vector of objectives functions to be minimized (or maximized), and the NR-vector of error functions (or residuals). The vectors  $(c^{(1)}, \dots, c_E^{(NE)})$  and  $c^{(1)}, \dots, c^{(NE)}$  are the NE and NI-vectors of equality and inequality constraints.

#### **Note**



Another type of condition that is not included in problem (P), is a constraint of the form  $c_{(i)} > 0$ . It is difficult to handle constraints of this form in an active set method such that the SQPAL algorithm, because the feasible region is not closed and the constraint cannot be active at a solution. However, it can be useful to include them in the problem via the transformation  $c_i(x) \geq \varepsilon > 0$ , possibly solving a sequence of problems in which  $\varepsilon \rightarrow 0$  if the constraints happen to be active. The reason for doing this might be to prevent or dissuade  $f(x)$  being evaluated at an infeasible point at which it is not defined.

---

## Optimization with the .EXTRACT Command

Excepting the bounds on design variables, the constraint and objective functions are *always* specified by complementing a **.EXTRACT** or a **.MEAS** command. The specification for objective and constraints functions can be written as:

```
.EXTRACT
+ [EXTRACT_INFO] [LABEL=NAME] [FILE=FNAME] [VECT]
+ [CATVECT] $MACRO|FUNCTION
+ [GOAL=MINIMIZE | MAXIMIZE | goal_value] |
+ [EQUAL=equ_value] |
+ [[LBOUND=lower_value] [UBOUND=upper_value]]
```

It is important to note that some of these complementing arguments are mutually exclusive. For instance, a constraint cannot be specified at the same time as an equality and as an inequality. On the other hand, a constraint is never minimized or maximized, this has no mathematical sense.

## Objective Specification

The *objective* function  $x \rightarrow f(x)$  is composite, that is,  $f$  is given by the algebraic summation of two categories of elemental objective functions. The functions belonging to the *first category* (A) has to be specified by complementing the **.EXTRACT** command as follows:

```
.EXTRACT { . . . } [GOAL=MINIMIZE | MAXIMIZE]
```

This syntax allows multiple specifications of *objective* functions. If several measures are qualified as **GOAL=MINIMIZE** or **=MAXIMIZE**, the effective objective function will be the summation of all these particular measures. Hence, the sum will be optimized:

$$f_A(x) = \sum_{i=1}^{NO} \pm f^{(i)}(x)$$

Where  $x \rightarrow f_i(x)$  corresponds to a particular measure to be optimized. Note that, we always minimize, thus “maximizing  $x \rightarrow -f_i(x)$ ” will be handled as “minimizing  $x \rightarrow f_i(x)$ ”. The dimension NF is the total number of objective functions.

In its essence, this feature represents the scalarization of a multi criterion problem, attaching the same weight to the  $i$ -th objective  $x \rightarrow f_i(x)$ . In a multi criterion problem, an optimal point  $\bar{x}$  satisfies  $f_i(\bar{x}) \leq f_i(y), i = 1, \dots, NO$  for every  $y$ . In other words,  $\bar{x}$  is simultaneously optimal for each of the scalar problems

	Minimize	$f^{(i)}(x)$
(P) <sub>i</sub>	Subject to	$x_l \leq x \leq x_u$
		$c_l \leq c(x) \leq c_u$

for  $i = 1, \dots, NO$ . When there is an optimal point, we say that the objectives are *not competing*, since no compromises have to be made among the objectives. Each objective is as small as it could be made, even if the others were ignored.

The second category (B) of elemental functions is made up with the vector of error functions. These functions are handled using the non-linear least squares approach, that is, by minimizing a weighted Euclidean norm of the vector  $r(x)$ :

$$f_B(x) = \frac{1}{2} \sum_{j=1}^{NR} \mu^{(j)} r^{(j)}(x)^2$$

With the weights  $\mu^{(j)} > 0$ . By default, these weights are initialized to one. The functions belonging to this second category have to be specified by complementing the **.EXTRACT** command as follows:

**.EXTRACT { . . . } [GOAL=goal\_value] [WEIGHT=weight\_value]**

By denoting the goal value as  $\bar{F}^{(j)}$ , the residual (or error function) will be defined as:

$$r^{(j)}(x) = F^{(j)}(x) - \bar{F}^{(j)}$$

where  $F^{(j)}(x)$  represents the actual value of the extracted measure.

## Example 2

In this example a noise analysis is performed for an amplifier at frequencies 500MHz and 700MHz, and the spot noise figure at both frequencies is concurrently minimized with respect to the resistance `rportin`. The lower bound 1.0 for the spot noise figure is used as target for two goal values.

```

.INCLUDE ampli.ckt
X1 in load vdd ampli_ckt
Vdc vdd 0 3.3
Vout load 0 rport=50 iport=2
Vin in 0 rport=rportin iport=1

.AC list 500meg 700meg
.NOISE v(load) Vin 1
.SNF input=Vin output=vout

.OPTIMIZE
.PARAMOPT rportin=(50,1,1k)
.EXTRACT noise
+ LABEL=snf_500 yval(snf, 500meg)
+ GOAL=1.
.EXTRACT noise
+ LABEL=snf_700 yval(snf, 700meg)
+ GOAL=1.

```

Replacing the `.GOAL`=value commands by `.GOAL=minimize`, will produce a different formulation of this problem:

```

.EXTRACT noise
+ LABEL=snf_500 yval(snf, 500meg)
+ GOAL=MINIMIZE
.EXTRACT noise
+ LABEL=snf_700 yval(snf, 700meg)
+ GOAL=MINIMIZE

```

The optimizer output for this example are shown in “[Exploiting Output Results](#)” on page 20-42.

---

### **Note**



Unlike equality constraints, the goal values of error functions are not guaranteed to be achieved at the optimum point. The equality (and inequality) constraints are most of the time more relevant than the above least squares approach.

---

## Inequality Constraints Specification

In the Eldo optimizer, designers can define their optimization problem using two types of constraints.

Inequalities on measures correspond to  $c_l \leq c_I(x) \leq c_u$ , these constraints are specified by complementing the `.EXTRACT` command as follows:

```
.EXTRACT { . . . }
+ [LBOUND=lower_value]
+ [UBOUND=upper_value]
```

These types of constraints are more relevant than the above equality constraints. The SQPAL algorithm also handles them directly. It is important to quote the following; the iterates (the successive values of design parameters taken during optimization) are by no means guaranteed to be feasible with respect to these constraints. If the optimization problem appears to be feasible, they will be satisfied at solution, or in the vicinity of a solution.

Equalities and inequalities specifications are mutually exclusive. This means that the constraint specifications must satisfy the following:

```
.EXTRACT { . . . }
+ [EQUAL=equ_value] |
+ [[LBOUND=lower_value] [UBOUND=upper_value]]
```

### Example 3

This example uses an amplifier with a fundamental frequency 900MHz and 10 harmonics that is optimized for the power efficiency and the power added efficiency to be between 0.4 dB and 0.6 dB.

```
* Power efficiency: Pout at fund1 / Pdc
.EXTRACT fsst LABEL=pe
+ yval(pm(rL), fund1)/yval(pm(vdc), 0)
+ LBOUND=0.4 UBOUND=0.6

* Power added efficiency: (Pout-Pin) at fund1 / Pdc
.EXTRACT fsst LABEL=pae
+ (yval(pm(rL),fund1)-yval(pm(vin), und1))/yval(pm(vdc),0)
+ LBOUND=0.4 UBOUND=0.6

* Steady-state analysis definition and plots
.sst fund1=900MegaHz nharm1=10

.OPTIMIZE
```

## Equality Constraints Specification

Equality constraints on measures are formulated as  $c_E(x) = e$ . They must be specified component-wise by complementing the **.EXTRACT** command as follows:

```
.EXTRACT { . . . } [EQUAL=value_equ]
```

The SQPAL algorithm handles arguments directly, but they are satisfied only at the solution.

## Example 4

Here a steady-state analysis of an amplifier with fundamental frequency 900MHz and 10 harmonics is optimized for the power efficiency and the power added efficiency to be 0.5dB. The quasi-periodic circuit steady state is computed with the harmonic balance simulator Eldo RF. This example demonstrates the availability of optimizing functionalities within the Eldo RF simulator.

```

* Power efficiency: Pout at fund1 / Pdc
.EXTRACT fsst LABEL=pe
+ yval(pm(rL), fund1)/yval(pm(vdc), 0)
+ EQUAL=0.5

* Power added efficiency: (Pout-Pin) at fund1 / Pdc
.EXTRACT fsst LABEL=pae
+ (yval(pm(rL),fund1)-yval(pm(vin), und1))/yval(pm(vdc),0)
+ EQUAL=0.5

* Steady-state analysis definition and plots
.ssst fund1=900MHz nharm1=10

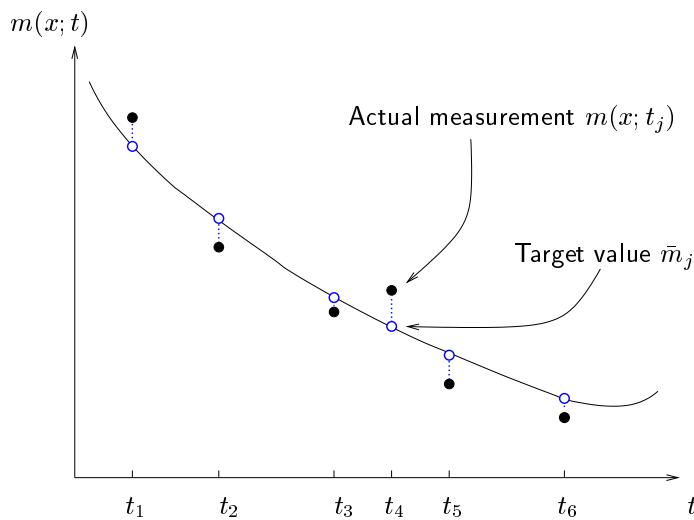
.OPTIMIZE

```

## Data and Curve Fitting

We may wish to find the design parameters so that some measurements agree as closely as possible with a set of target values. This is known as the *nominal optimization* problem.

**Figure 20-7. Deviation between the model and the extracted measures**



To improve the match of some design objectives with their specifications. In Figure 20-7, measurement samples are drawn at specific times (or frequencies), tabulating the time  $t_j$  and measurements  $m_j(x) = m(x; t_j)$ . The symbol  $x$  refers to the parameters of the circuit. The smooth

curve in [Figure 20-7](#) (plotted as a function of  $t$ ) is the constructed model (or the target curve) used to define the target values  $\bar{m}_j$  for measurements.

The differences between the model predictions and the observed values (the actual measurements  $m(x; t_j)$  given the set of current design parameters) are combined in the following least-squares function:

$$\frac{1}{2} \sum_{i=1}^{n_i} [\bar{m}_j - m(x; t_j)]^2$$

Graphically, each term in this summation represents the square of the vertical distance between the smooth curve (defining the target values  $\bar{m}_j$ ) and the point  $(t_j, \bar{m}_j)$ .

## Command Syntax

The following native syntax is used to perform data (or curve fitting).

```
.EXTRACT [analysis] LABEL=measure
+ FITTING
+ tvar mvar
+ [SCALE=linear|log|log10]
+ [WEIGHT=weight]
+ [FROM=value_from] [TO=value_to]
+ [INCR=increment|DEC=npoints]
```

## Parameters

**tvar** Name of the discretized curve to fit. It can be the name of a previously defined wave or the reference to a variable in a **.DATA** statement (see the examples below).

**mvar** Name of a valid extracted measure.

**SCALE** Selects the scaling for the error function calculation:

```
scale=linear     $\bar{m}_j = (\text{tvar})_j$  and  $m_j(x) = (\text{mvar})_j$ 
scale=log10     $\bar{m}_j = \log 10((\text{tvar})_j)$  and  $m_j(x) = \log 10((\text{mvar})_j)$ 
scale=log       $\bar{m}_j = \ln((\text{tvar})_j)$  and  $m_j(x) = \ln((\text{mvar})_j)$ 
```

**INCR | DEC** increment and npoints are only meaningful for **.DATA** defined target curves only. increment is the step in the analysis sweep variable to be used for generating target values using a piece wise linear interpolation of data provided by the **.DATA** command. npoints is the number of points per decade in the analysis sweep variable to be used for generating target values using a logarithmic interpolation of the provided data points. The

corresponding simulated values are obtained by interpolation of the available simulated values.

---

**Note**

 If this option is not specified, only the data provided by the **.DATA** statement are used as target points.

---

## Example 5

In this example, some **.DC** operating point analyses are performed for every value of **vds** from 0 to 40 with a step increment of one. and every value of **vgs** between 4 and 16 with a step increment of two. Three measurements are defined for the variable **IDS** of **.DATA mosfit**, corresponding to values of **vgs** equal to 4 and the values of **vds** equal to 1, 2 and 3 respectively. The current at node **vdrain** is optimized to fit the values of **IDS**. The values of **vds** and **vgs** for the measurements correspond to simulation points as specified in the **.DC** command and interpolation of the **i(vdrain)** simulated values is therefore avoided.

```
.OPTIMIZE RESULTS=res1

.DC vds 0 40 1 vgs 4 16 2

.EXTRACT DC LABEL=res1
+ FITTING
+ mosfit(IDS) i(vdrain)

.DATA mosfit
+ VDS VGS IDS
+ 1.000E+00 4.000E+00 1.574E+00
+ 2.000E+00 4.000E+00 1.806E+00
+ 3.000E+00 4.000E+00 1.826E+00
.ENDDATA
```

---

**Note**

 As a rule of thumb the designer is advised to set analyses commands to avoid interpolation of simulated values (that is, such that measurement points be also simulation points) in order not to introduce the additional interpolation error in the optimization process.

---

---

**Note**

 Although many operating points are computed, only those corresponding to the measurement points are used for the optimization, and it would be worth replacing the analysis command by the sequence.

---

```
.PARAM vds=4.0
.DC vds 1 3 1
```

## Example 6

A small signal analysis is performed for frequencies in the range [100 kHz, 1 GHz] with 2 points per decade. The voltage magnitude at node 16 is optimized to fit the values of the data labelled `v16` on `.DATA vfit`. There are three measurement points defined at 100 kHz, 30 MHz and 1 GHz respectively. The values of `vm(16)` at these frequencies are obtained by interpolation of the values simulated according to the `.AC` analysis command.

```
.OPTIMIZE

.AC DEC 2 100k 1g

.EXTRACT AC LABEL=VM16
+ FITTING
+ vfit(v16) vm(16)
+ SCALE=linear

.DATA vfit
+ freq v16
+ 100k 200
+ 30Meg 200
+ 1g 1m
.ENDDATA
```

## Example 7

This example optimizes the voltage values at node `vout` to fit the four values of `vv` provided in the `.DATA` statement named `datav`.

As the simulation points in the transient analysis are dynamically set by Eldo for accuracy purposes, the measurement points are not necessarily simulation points and it is therefore likely that the simulated values be interpolated to the measurement points. This is a peculiarity of the optimization of targets computed during transient analyses.

```
.OPTIMIZE

.TRAN 0 200n

.EXTRACT TRAN LABEL=v1
+ FITTING
+ datav(vv) v(vout)
+ SCALE=linear

.DATA datav
+ tt vv
+ 0 0.1
+ 80n 0.1
+ 120n 5.0
+ 200n 5.0
.ENDDATA
```

**Note**

 Some transient simulation points may be imposed in **.OPTION TIMESTEP= step**

---

## Interpolated Measurements

The option [**INCR=increment** | **DEC=npoints**] can be used to generate additional measurement points by interpolation. The corresponding simulated values are obtained by interpolation of available simulated values.

```
.OPTIMIZE
.AC DEC 2 100k 1g
.EXTRACT AC LABEL=VM16
+ FITTING
+ vfit(v16) vm(16)
+ SCALE=linear
+ DEC=2
.DATA vfit
+ freq v16
+ 100k 200
+ 30Meg 200
+ 1g 1m
.ENDDATA
```

A small signal analysis is performed for frequencies in the range [100 kHz, 1 GHz] with two points per decade. The voltage magnitude at node 16 is optimized to fit a set of interpolated values of the data labelled **v16** in **.DATA vfit**. The measurement data provided in **vfit** defines a piece wise linear curve from which two measurement points per decade are generated by logarithmic interpolation. The number of points per decade is chosen so that the original and interpolated measurement points are also simulation points (the latter being defined by the **.AC** command).

## Data Driven Analyses

It is possible to define simulation points in the analysis command using the **.DATA** command.

[Example 8](#) demonstrates what happens when the data is also used to define the measurement and targets, when used in conjunction with the **INCR=increment** or **DEC=npoints** options of the **.EXTRACT** command.

### Example 8

In this example, only the operating point analyses corresponding to the couples of values **VDS** and **VGS** provided in **.DATA mosfit** are performed and for each of these there is a target **IDS**.

```
.OPTIMIZE RESULTS=res1

.DC DATA=mosfit
.EXTRACT DC LABEL=res1
+ FITTING
+ mosfit(IDS) i(vdrain)

.DATA mosfit
+ VDS VGS IDS
+ 1.000E+00 4.000E+00 1.574E+00
+ 2.000E+00 4.000E+00 1.806E+00
+ 3.000E+00 4.000E+00 1.826E+00
.ENDDATA
```

## The Optimization Command

This section is concerned with the optimization command acting on the SQPAL algorithm parameters. In some specific cases, designers can perform optimization using the *dichotomy* and *passfail* methods. Please refer to [Example 9](#) to set-up a measurement with Eldo using this method.

## Command Syntax

The optimization specification acting on all the analyses specified in the circuit netlist is done using the following command:

```
.OPTIMIZE
+ [QUALIFIER=value {, QUALIFIER=value}]
+ [PARAM=list_of_variables|*]
+ [RESULTS=list_of_measures|*]
```

Where

**qualifier**      The name of the corresponding configuration argument (see the list and definitions in the [Arguments](#) section).

**list\_of\_variables**  
List of comma-separated variables to be tuned, specified with the **.PARAMOPT** command.

**list\_of\_measures**  
List of comma-separated measures to be optimized, specified with **.EXTRACT** commands and/or **.MEAS** commands.

### **Note**



If the specification of tuning parameters with **PARAM=** is omitted or if the character \* is specified in place of an explicit list, all the design parameters specified by a **.PARAMOPT** command are optimized. If the specification of targets with **RESULTS=** is omitted or if \* is specified in place of an explicit list. All the targets specified by a **.EXTRACT** or a **.MEAS** command are optimized.

## Arguments

The performance of the solver is controlled by a number of parameters. Each option has a default value that should be appropriate for most problems. The defaults are given below. For specific situations it is possible to specify non-standard values for some or all of the parameters. If experimentation is necessary, we recommend changing just one option at a time.

**MAX\_ITER**=*i*value

Maximum number of iterations permitted for optimization. Default 1000.

**MAX\_SIMUL**=*i*value

Maximum number of circuit simulations allowed. Default 1000.

**TOL\_OPT**=*r*value

Tolerance on optimality conditions. This parameter specifies the accuracy to which the user wishes the final iterate to approximate a solution of the problem. We may consider **TOL\_OPT** as indicating the number of correct figures desired in the design functions at the solution. Specifying only the value **TOL\_OPT** has the same effect that setting separately both the tolerances **TOL\_GRAD** and **TOL\_FEAS**. Default value =  $10^{-4}$ .

**FD\_FW RD\_INT**=*r*value

Relative design-variable change for the computation of the perturbation used for gradient evaluations. The perturbations are taken as **FD\_RELCHG** times the maximum of the absolute design parameter. Default value =  $10^{-6}$ .

**TOL\_FEAS**=*r*value

Tolerance on feasibility conditions. An iterate is said to satisfy the feasibility conditions if  $v(x) \leq \delta \cdot (1 + \|\bar{x}\|)$  where  $\delta$  is the optional parameter **TOL\_FEAS**. Default value is  $10^{-6}$ .

**TOL\_GRAD**=*r*value

Tolerance on the measure of *criticality* of the current iterate. An iterate is said to be *critical* if  $\|P[\bar{x} - \nabla f(\bar{x}) - \langle A | \bar{\lambda} \rangle; x_l, x_u] - \bar{x}\| \leq \gamma \cdot (1 + \|\bar{\lambda}\|)$  where  $\gamma$  is the optional parameter **TOL\_GRAD**. Default value is  $10^{-4}$ .

**PRN\_MAJOR**=[ YES | NO ]

Controls the output of the optimization algorithm. A single line of information is written to the *.otm* file during a major iteration. Default YES. When set to NO the information will not be written to the file.

## Example 9

.OPTIMIZE specifies that the optimization will be performed using the default configuration.

```
.OPTIMIZE MAX_ITER=20 MAX_SIMUL=100 TOL_OPT=1e-3
```

Specifies that the optimization will finish whenever the optimality and the feasibility are less than  $10^{-2}$ , or 20 iterations or 100 circuit simulations have been performed.

## Dichotomy and Pass/Fail Methods

Eldo can perform an optimization without using the SQPAL algorithm. This can be applied when there is only one goal (specified through a .EXTRACT GOAL=value command) and one parameter to optimize.

## Command Syntax

The optimization specification acting on unique analyses specified in the circuit netlist is achieved using the following command:

```
.OPTIMIZE
+ METHOD = PASSFAIL | DICHOTOMY|SECANT
+ [MAX_ITER=max_iter]
+ [TOL_RELPAR=relpar_value]
+ [TOL_RELTARG=relout_value]
+ [PARAM=parameter]
+ [RESULTS=measure]
```

Where:

**MAX\_ITER**=max\_iter

The maximum number of iterations allowed for the algorithms.

**TOL\_RELPAR**=relpar\_value

Convergence criterion on the relative tuning parameters variation. The optimization is stopped whenever all the design variables relative changes at the current iteration are less than **RELPAR**. Default value is  $10^{-3}$ .

**TOL\_RELTARG**=reltarg\_value

Convergence criterion on the relative change of the sum of squares of the misfits to the measurements. Default  $10^{-3}$ .

**PARAM**=parameter

Name of parameter to be tuned, specified with the .PARAMOPT command.

**RESULTS**=measure

is the list of comma-separated targets to be optimized, specified with .EXTRACT commands and/or .MEAS commands.

## Set-up Time Computation

The goal here is to use Eldo to compute the set-up time of a flip-flop. The definition of set-up time is: time between input and the clock (`TIC`) so that propagation time between clock and output (`TCO`) is 10% above nominal value (computed when there is a large time between input and the clock).

To do this, suppose that the nominal value of `TCO` is already known.

$$\text{TCO\_TARGET} = 1.1 \times \text{TCO\_NOMINALS}$$

A very general method, available even with older versions of Eldo, would be to sweep `TIC`, measure `TCO`, and extract from `TCO(TIC)` the value of `TIC` so that `TCO=TCO_TARGET`.

```
.PARAM TIC=200n
.STEP param TIC 205n 195n 0.1n
.EXTRACT LABEL=TCO
+ (XUP(V(Q),0.48,200n,300n,1) - XUP(V(CP),0.48,200n,300n,1))
.EXTRACT sweep xycond(XAXIS, meas(TCO)=1.1*TCO_NOMINAL)
```

Simulation time can be improved with use of `.OPTION AUTOSTOP=2`. Accuracy is proportional to the parameter sweep increment, simulation time is proportional to the number of steps in the swept interval. From Eldo v5.8, it is possible to use two new methods of optimization that are suitable for the computation of set-up the time. The `PASSFAIL` method computes the maximum value for which an extract can be measured. This is not exactly the definition of set-up, but it can be used as a first step. This would give:

```
.OPTIMIZE METHOD=PASSFAIL RESULTS=TCO
.PARAMOPT TIC=(201n,199n,205n)
```

The `DICHOTOMY` method, finds the value of one optimization parameter so that one extract yields a target value, provided that the extract values for the `Min` and `Max` values of the optimization parameter bracket the target.

In this example, the `Min` value of the optimization parameter `TIC` has been computed with `PASSFAIL`. This would give:

```
.EXTRACT tran LABEL=DCP
+ (XUP(V(Q),0.48,200n,300n,1) - XUP(V(CP),0.48,200n,300n,1))
+ GOAL={1.1*TCO_NOMINAL}

.PARAMOPT TIC=(201n,200.07n,205n) !MIN provided by PASSFAIL
```

From Eldo v5.9, it is possible to combine both `PASSFAIL` and `DICHOTOMY` methods in one step with the following syntax:

```
.OPTIMIZE METHOD=DICHOTOMY
.EXTRACT tran LABEL=DCP
+ (XUP(V(Q),0.48,200n,300n,1) - XUP(V(CP),0.48,200n,300n,1))
+ GOAL={1.1*TCO_NOMINAL}
.PARAMOPT TIC=(201n,195n,205n)
```

The **SECANT** method search algorithm is more advanced than the plain **DICHOTOMY**. Many problems suitable for bisection (one design parameter, one target) are actually fairly linear, i.e. the target value is almost proportional to the design parameter. In these conditions, using the plain **DICHOTOMY** is clearly sub-optimal in terms of the number of iterations required. Using the **SECANT** algorithm will greatly diminish the number of iterations. This feature is useful for designers wanting to accelerate their setup/hold simulations.

## Exploiting Output Results

### Binary Output

```
.PLOT OPT WOPT(label_name) [(LOW, HIGH)] [(VERSUS)]
+ {OVN [(LOW, HIGH)]} [(SCATTERED)]
```

### Parameters

OPT	Optimization analysis. The waves produced by this analysis can only be specified using the keyword <b>WOPT</b> .
WOPT	Only available for extracting waves generated during an optimization process. These are implicitly declared waves which have the same name as the extract label they refer to. They represent extract results versus the index of the run.

It is possible to plot extracted waves generated during an optimization process. These are implicitly declared waves which have the same name as the extract label they refer to. They represent extract results versus index of the run.

### Example

```
.EXTRACT tran label=FOO yval(v(node),3n) !Define the extract
.PLOT OPT WOPT(FOO)
```

### ASCII Output

A typical Eldo optimizer output is given below. It corresponds to a constrained optimization problem we created specifically for this purpose.

The following ASCII output can be retrieved from the **.chi** file or the standard output. It contains three parts: the first gives final diagnostics after optimization, while the others give respectively the status and values of the parameters and functions (objective and constraints).

\*\*\*\*\* OPTIMIZATION RESULTS \*\*\*\*\*

Objective function	=	1.000000e-00
Optimality	=	1.1733333e-10
Feasibility	=	8.9150909e-13
Circuit simulations	=	13
Number of iterations	=	3

Exit in normal mode with return code = 0

Diagnostics: The optimization has been successful.  
The required accuracy has been achieved.

\*\*\*\*\*

\*\*\*\*\* PARAMETERS \*\*\*\*\*

NAME	INITIAL VALUE	LOWER BOUND	UPPER BOUND	INCREMENT VALUE	CURRENT VALUE
X1	1.00000e-01	0.00000e+00	NONE	0.00000e+00	0.0
X2	7.00000e-01	0.00000e+00	NONE	0.00000e+00	0.0
X3	2.00000e-01	0.00000e+00	NONE	0.00000e+00	1.0000E-00

\*\*\*\*\*

\*\*\*\*\* MEASUREMENTS \*\*\*\*\*

NAME	GOAL VALUE	LOWER BOUND	UPPER BOUND	WEIGHT VALUE	FINAL VALUE
OBJECTIVE				1.00000e+00	1.0000E-00
CONSTR1	0.00000e+00			1.00000e+00	3.5103E-10
CONSTR2		0.00000e+00		1.00000e+00	1.0000E-00

\*\*\*\*\*

The value of the objective function is the final value of the composite objective function to be minimized in the problem statement.

Small values of the optimality and feasibility measurements indicate that an optimum has been found.

## Final Diagnostic Conditions

Users who are unfamiliar with the concepts of optimization (such as optimality or feasibility) should refer to “[Robust Optimization Using Corners](#)” on page 20-8. When the optimizer has finished, a Return Code is printed in the OPTIMIZATION RESULTS to summarize the final diagnostic. In order to distinguish between the normal mode and the relaxed mode, an

offset value of 100 will be added when the optimizer has finished in the relaxed mode. For example, a return code of 101 (1+100) means that the iteration limit has been reached.

It is essential to check the value of the return code and its associated message.

These messages and their meaning are described in the following table:

Code	Message and meaning
0	<p>The optimization has been successful. The required accuracy has been achieved.</p> <p>The iterates have converged to a point <math>\bar{x}</math> that satisfies the first-order optimality conditions to the accuracy requested by the optional parameters <code>TOL_OPT</code> or <code>TOL_GRAD</code> and <code>TOL_FEAS</code> (refer to “<a href="#">The Optimization Command</a>” on page 20-38). The user should check whether the following two conditions have been satisfied:</p> <ul style="list-style-type: none"><li>• the final value of Optimality (gradient) is significantly small.</li><li>• the final values of Feasibility (equality) and Feasibility (inequality) are significantly small. Note, these values are set to zero if there are no equality or inequality constraints.</li></ul>
1	<p>Optimization was not successful. The iteration limit was reached.</p> <p>The limiting number of iterations, determined by the optional parameter <code>MAX_ITER</code> (see “<a href="#">The Optimization Command</a>” on page 20-38) has been reached.</p> <p>Check the iteration log contained in the <code>.otm</code> file. If the optimizer appears to be making progress, <code>MAX_ITER</code> may be too small. If so, increase its value and rerun the optimizer, possibly using the values of the design variables obtained so far (this may consider as a basic <i>warm start</i>).</p>
2	<p>The current point cannot be improved. A sufficient decrease in the merit function could not be attained during the final line search.</p> <p>The final iterate <math>\bar{x}</math> does not satisfy the first-order optimality conditions, and no improved point for the merit function could be found during the final line search.</p> <p>A sufficient decrease in the merit function could not be attained during the final line search. This sometimes occurs because an overly stringent accuracy has been requested, i.e. <code>TOL_OPT</code> is too small. In this case the user should verify the two conditions describe under <code>code=0</code> above to determine whether or not the final solution is acceptable.</p>
4	Optimization was not successful. The maximum number of simulation runs has been reached.

- The limiting number of simulations, determined by the optional parameter **MAX\_SIMUL** (see “[The Optimization Command](#)” on page 20-38) has been reached.
- 6 The current point is feasible and quite optimal but the required accuracy is not achieved.  
The optimizer is within  $1 \times 10^{-2}$  of satisfying the **TOL\_OPT** (or **TOL\_GRAD**) optimality tolerance. For next runs, the user should check that **TOL\_OPT** is not too small.
- 7 The objective function may be unbounded on the feasible set, or alternatively the scaling of the problem may be very poor.
- 8 Optimization was not successful. The objective function or the constraints seem to be undefined at starting point. The optimization was abandoned.

## Optimizer Output Control Options

- **OPSEUDO\_ABSTRACT**

Generates a summary table of simulations containing parameter and extract values for each run.

- **OPSEUDO\_DETAIL=[NONE | ALL]**

If this option is set to **NONE**, only the last run will be stored in generated files (**.wdb**, **.cou**, **.aex**) and no other simulation information will be displayed. When set to **ALL**, simulation information for all runs will be stored. Default is **NONE**. (This option was named **OPSEUDO\_NODETAIL** in pre-v6.6 Eldo versions and was not enabled.)

- **OPSEUDO\_DISPLAY\_GOALFITTING**

Displays the intermediate goals when splitting DC measurements during optimization. Each **.DATA** point is optimized as an independent goal.

- **OPSEUDO\_FORCE\_GOALFITTING**

Forces Eldo to split DC fitting extracts in independent goals even if data points are correctly ordered (as opposed to **OPSEUDO\_NOGOALFITTING**). It can also split AC fitting extracts.

Disables the splitting of DC measurements during optimization. By default, each **.DATA** point is optimized as an independent goal.

- **OPSEUDO\_NETLIST**

Generates a netlist modified from the original input file, which contains the optimized parameter values but also every parameter set under #ifdef statements.

- **OPSELDO\_NOGOALFITTING**

Disables the splitting of DC measurements during optimization. By default, each **.DATA** point is optimized as an independent goal only if they are not correctly ordered.

- **OPSELDO\_OUTER**

Allows a reverse behavior of optimization and sweep simulations (**.TEMP**, **.DATA** or **.STEP**). A full optimization will be performed for each set of sweep parameters.

- **OPSELDO\_OUTPUT**

This option controls results of optimization (Opsim or bisection method).

**OPSELDO\_OUTPUT=0**

Print results in a simplified format: parameter name, goal, optimized value

**OPSELDO\_OUTPUT=1**

Print results using simplified format (as for **OPSELDO\_OUTPUT=0**) and generate a **.ops0** file using the same format.

**OPSELDO\_OUTPUT=2**

Print results in a detailed format: parameter name, minimum value, maximum value, weight, goal, optimized value

**OPSELDO\_OUTPUT=3**

Print results using detailed format (as for **OPSELDO\_OUTPUT=2**) and generate a **.ops0** file using the same format.

## References

1. J.F. Bonnans, J. Ch. Gilbert, C. Lemarechal, C. Sagastizabal. *Numerical Optimization -Theoretical and Practical Aspects*. Springer Verlag, Berlin, 2003.
2. F. Delbos and J. Ch. Gilbert. *Global linear convergence of an augmented Lagrangian algorithm for solving convex quadratic optimization problems*. Journal of Convex Analysis, 12, 2005.
3. J. E. Dennis and R. B. Schnabel. *Numerical methods for Unconstrained Optimization and Nonlinear Equations*. Prentice-Hall, Englewood Cliffs, 1983.
4. J. E. Dennis and J. More. *Quasi-Newton methods, motivation and theory*. SIAM Review, 19, 46-89, 1977.
5. R. Fletcher. *Practical Methods of Optimization* (second edition). John Wiley & Sons, Chichester, 1987.
6. P. E. Gill, W. Murray, and M. H. Wright. *Practical Optimization*. Academic Press, New York, 1981.
7. J. Nocedal and S. W. Wright. *Numerical Optimization*. Springer Series in operations Research, Springer, New York, 1999.

8. J. M. Ortega and W. C. Rheinboldt. *Iterative Solution of Nonlinear Equations in Several variables*. Academic Press, New York, 1971.
9. R. Point, M. Mendes and W. Foley. *A differential 2.4GHz Switched-Gain CMOS LNA for 802.11b and Bluetooth*. 2002 IEEE Radio and Wireless Conference.



# Chapter 21

## Reliability Simulation

---

### Introduction

Reliability models have been implemented in Eldo's UDRM interface (User Defined Reliability Model).

The objective of reliability simulation is to be able to model the gradual damage, which occurs to the devices in a certain design causing degradation in the performance of that design. It is required to evaluate the amount of degradation occurring in a certain period of operation and examine the circuit performance after this period. The modeled damage could follow one or more of several damage mechanisms, which show gradual degradation in device performance. Other mechanisms, which cause sudden and complete damage of the device, (like the oxide breakdown for example) are not targeted in this scope.

The reliability simulations follows the model defined by the user using the UDRM interface. A simple Hot-carrier and NBTI reliability model is provided as an example.

Please refer to *Eldo UDRM User's Manual* for more information.

### Running Reliability Simulation in Eldo

Reliability simulation in Eldo can be used with any normal netlist provided that it contains a **.TRAN** analysis. This **.TRAN** analysis will be the analysis used for all the fresh and aged simulations, and its outputs will be available also for all the fresh and aged simulations. The reliability simulation can be invoked and controlled through the below Eldo commands.

---

#### Note



Age analysis is applied only to specific instances having the instance parameter AGE=1 specified in the instance line. Setting AGE=0 on the instance line will remove the device from the Age analysis. Default value for AGE is 0.

---

### Reliability Commands

The reliability commands are detailed on the following pages.

## .AGE

### Age Analysis

```
.AGE [TAGE=value] [NBRUN[S]=value]
+ [LIN[={YES|ON|1}|{NO|OFF|0}]] [LOG[={YES|ON|1}|{NO|OFF|0}]]
+ [TSTART=value] [TSTOP=value]
+ [MODE=AGESim | AGEload | AGESave] [AGELIB=file_name]
+ [AGEALL[={YES|ON|1}|{NO|OFF|0}]]
+ [ASCII[={YES|ON|1}|{NO|OFF|0}]]
+ [COMPUTE_LAST[={YES|ON|1}|{NO|OFF|0}]]
+ [PLOT={FRESH_FINAL|ALL}]
+ [TUNIT=year|month|day|hour|min]
+ [AGEDSIM={YES|ON|1}|{NO|OFF|0}]
+ [LOGMODE_MINEXP=<VALUE>]
```

This command activates the Age analysis.

### Parameters

- **TAGE**

The final time after which reliability simulation needs to be performed. (The time after which the circuit performance needs to be measured.) This parameter is not active in case **MODE=AGEload**.

- **NBRUN[S]**

Specifies the number of time intervals Tage will be divided into in case of repetitive scheme. This parameter is not active in case **MODE=AGEload**.

- **LIN**

Defines the rule that is applied in repetitive scheme to split **TAGE** into several intervals to be linear. If **LIN={YES|ON|1}** is specified, the runs are equally spaced in time. If **LIN** is specified without a value, this is equivalent to **LIN=YES**.

- **LOG**

Defines the rule that is applied in repetitive scheme to split **TAGE** into several intervals to be logarithmic. If **LOG={YES|ON|1}** is specified, runs crowd at start following a log scale divisions. If **LOG** is specified without a value, this is equivalent to **LOG=YES**.

- **TSTART**

The stress integral start time. Default is the start time of the **.TRAN** command.

- **TSTOP**

The stress integral stop time. Default is the stop time of the **.TRAN** command.

- **MODE**

Defines the simulation mode.

**AGESim:** Eldo will perform a normal aged simulation.

**AGEload:** Stress values will be loaded from the file specified in **AGELIB** statement and a single simulation will be performed. This file should be the output from a previous run with **MODE=AGESave**.

**AGESave:** Eldo will perform a normal aged simulation and the stress values will be saved at the end of the process.

- **AGELIB**

Defines the name of the file to which stress values will be dumped (**MODE=AGESave**) or from which they will be read (**MODE=AGEload**).

- **AGEALL**

Changes the default of the instance parameter **AGE**. For instance, if **AGEALL={YES|ON|1}** is specified, age analysis will be performed to all BSIM3v3 or BSIM4 MOSFET devices. Only the devices having **AGE=0** instance parameter will skip this age analysis. The default is zero. If **AGEALL** is specified without a value, this is equivalent to **AGEALL=YES**.

- **ASCII**

Specifies whether the file created in **AGESave** mode is readable or not. By default, this file is in binary format to reduce its size. If **ASCII={YES|ON|1}** is specified, the file will be generated in readable ascii format. If **ASCII** is specified without a value, this is equivalent to **ASCII=YES**.

- **COMPUTE\_LAST**

Specifies whether the stress computation is to be done for the last run or not. By default, it will not be done. If the user wants to enable these calculations (for output purpose), **COMPUTE\_LAST={YES|ON|1}** should be specified. If **COMPUTE\_LAST** is specified without a value, this is equivalent to **COMPUTE\_LAST=YES**.

- **PLOT**

Specifies which results will be saved in the output files (for **.PRINT** and **.PLOT** commands) in the repetitive scheme. If **PLOT=FRESH\_FINAL** is specified, only the curves for the fresh simulation and the final reliability run will be saved in the output files. If **PLOT=ALL**, the curves for the fresh simulation and all reliability runs will be saved. The default is **PLOT=FRESH\_FINAL**.

- **TUNIT**

Specifies the units for the parameter **TAGE**. Can be specified as years, months, days, hours or minutes.

- **AGEDSIM**

Setting this parameter to **0** (or **NO**) allows the user to specify that the age simulation will not be performed. This is useful if the user wishes to create an **AGELIB** library without having to run the age simulation. Default value is **1** (or **YES**).

- **LOGMODE\_MINEXP=<VALUE>**

Specifies the limit between the exponential calculation and the linear calculation of the aging factor. If TAGE is greater than this limit, the factor will be calculated using:

$$\exp\left(\text{index} \cdot \log\left(\frac{1.0 + \text{tage}}{\text{nbrun}}\right)\right) - 1.0$$

If TAGE is smaller than this limit, the factor will be calculated using:

$$\left(\exp\left(\text{index} \cdot \log\frac{1.0 + 1.0e6}{\text{nbrun}}\right) - 1.0\right) \cdot \frac{\text{tage}}{1.0e6}$$

## .AGEMODEL

### Reliability Model Parameter Declaration

**.AGEMODEL MODEL=model\_name [parameter=value]**

This command allows the user to specify the model parameters related to reliability calculations. Parameters must be defined in the UDRM library.

#### Parameters

- **MODEL**

Specifies the name of the MOS model to associate the following reliability parameters to. Should be a valid BSIM3v3 or BSIM4 model.

## .SETKEY

### Set Reliability Model Key (Password)

**.SETKEY** [ MODEL=model\_name ] KEY=key\_value

This command defines a key (password), which is associated to a specific model or to all the models of a library. This statement is encrypted using the same encryption method as the **.MODEL** cards. It is used to restrict external access to model's parameters' values. The UDRM will not be able to access the values of these parameters unless the user provides the same key in his netlist by using the **.USEKEY** command shown below.

#### Parameters

- **MODEL**

Defines the model for which the key is valid. If this statement is not given, the key will be used for all encrypted models.

- **KEY**

A combination of any characters (case sensitive).

## .USEKEY

### Use Reliability Model Key (Password)

```
.USEKEY [ MODEL=model_name ] KEY=key_value
```

This command provides the encryption key (password) to be used to allow the UDRM API to retrieve encrypted model parameter's value.

#### Parameters

- **MODEL**

Defines the model for which the key is valid. If this statement is not given, the key will be used for all encrypted models.

- **KEY**

A combination of any characters (case sensitive). This should be the non-encrypted version of the **key\_value** used in the **.SETKEY** command.

## Monitoring STRESS Values

Eldo allows the user to monitor the values of the stress applied on the devices in his output files, through the normal output commands. The following output quantities are defined:

### STRESS(<device\_name>,<index>)

This quantity represents the value of a specific instantaneous stress vector component (the stress component with the specified index from the stress vector).

### STRESS(<device\_name>)

This quantity represents the total instantaneous stress (sum of STRESS(<device\_name>,i) with i between [0, <stress vector size -1>] ).

These quantities can be used in **.PLOT**, **.PRINT**, **.EXTRACT**, or **.DEFWAVE** cards.

#### Example

```
.plot tran STRESS(M1, 1)
```

### ACC\_STRESS(<device\_name>)

This quantity represents the cumulative amount of stress in the device.

It can be used in **.PLOT**, **.PRINT**, **.EXTRACT**, or **.DEFWAVE** cards, to be monitored versus the run index.

#### Example

```
.plot sweep(age) ACC_STRESS(M1)
```

## Reliability Simulation Limitations

Reliability simulation (aging) can be applied only to MOSFET devices.

For the Current implementation, only BSIM3v3 and BSIM4 MOS models support Reliability simulation (aging).

Sweep analysis types (**.MC**, **.TEMP**, and **.STEP**) cannot be used along with the **.AGE** command, for **MODE=AGESave**, or **MODE=AGEload**.

When reliability simulation is applied to devices inside a subcircuit (where the **.MODEL** and **.AGEMODEL** is defined inside the subcircuit), the **.AGEMODEL** parameters cannot depend on subckt parameters.

# Chapter 22

## Post-Processing Library

---

### Introduction

The Post-Processing Library (PPL) is a tool that allows interfacing between Eldo and a set of predefined, user defined DSP or mathematical functions. User Defined Functions (UDF) are written using the scripting language Tcl. This chapter describes both the commands used from Eldo to call a PPL function and the extensions of the Tcl language for Post-Processing abilities.

Various examples illustrate as many configurations as possible. All can be found in the directory: `$MGC_AMS_HOME/eldo/$eldover/examples/ppl`

### Registering User Defined Functions inside Eldo

#### .USE\_TCL Command—Use Tcl File

##### Syntax

```
.USE_TCL FILENAME
```

This command will load the Tcl file FILENAME into Eldo's Tcl interpreter, making all declared functions to be recognized inside Eldo. Since these functions will have to be identified during the parsing of the netlist, the `.USE_TCL` commands must be placed before any call to one of these functions.

After loading the Tcl file Eldo can use all the functions written in this file as if they were macros. This means that these functions can accept any argument, are defined for the whole netlist and can return both numbers and/or waves.

### Macro-like Usage of UDF

The first example shows how two resistors get their values directly from a UDF function or through a parameter (files are *realvalue.cir* and *realvalue.tcl*). See [Table 22-4](#) on page 22-29 for more information on the example files.

*realvalue.tcl* contains:

```
# Function which returns a real value
proc GETVALUE { rname rval } {

# Use native Tcl syntax for expression calculation
set newvalue [expr 2*$rval + log($rval)]
```

```
    puts "The value of resistor $rname is $newvalue"
    return $newvalue
}
```

**Note**

 The name of the function *must* be in uppercase or Eldo will not be able to identify it.

*realvalue.cir* contains:

```
* This example demonstrates the use of a simple Tcl function
* returning a real value

* use_tcl commands must always be put before the first call
* to a Tcl function. Otherwise, Eldo will not be able to parse
* the expressions using these functions

.use_tcl realvalue.tcl

.param p1={GETVALUE( "R1" ,TEMPER ) }
.param p2=1

v1 1 0 sin(0 2 1g)
r1 1 2 p1
r2 2 0 {GETVALUE( "R2" ,p2 ) }

.tran 1n 10n
.plot tran i(r1)
.end
```

Except for the dump of the value the Tcl function GETVALUE is equivalent to:

```
.defmac GETVALUE(val) = 2*val+log(val)
```

UDF parameters can be real values, strings, waves, macros, UDF or keywords. Available keywords will be described later. It is important to notice that for this kind of use, the UDF may be called for each timestep. Thus wave parameters are not passed to the UDF as objects but as real values.

Files *samphold.cir* and *samphold.tcl* contain another example of UDF. The function uses a Tcl namespace to keep the previous value of a wave, and is called inside a **.DEFWAVE** to create a sampled representation of the input wave. See [Table 22-4](#) on page 22-29 for more information on the example files.

## Keyword Parameters

The following keywords can be used in any calls to UDF:

**XAXIS / TIME / FREQ** specifies the current timestamp or frequency

**TEMPER** specifies the current temperature

The following keywords can only be used in **.CALL\_TCL** commands:

<b>ICARLO</b>	specifies the current index of Monte-Carlo run
<b>IRUN</b>	specifies the current step index
<b>IALTER</b>	specifies the current alter index
<b>NBCARLO</b>	specifies the number of Monte-Carlo run
<b>NBRUN</b>	specifies the number of step
<b>NBALTER</b>	specifies the number of <b>.ALTER</b> statements

## Making Specific Calls to UDF inside Eldo

### **.CALL\_TCL** Command—Call Tcl Function

#### Syntax

```
.CALL_TCL [ANALYSIS] WHERE=trigger
+ [PLOT=YES|NO] [LABEL=alias_name] tcl_function_call
```

#### Parameters

- **ANALYSIS**  
specifies the type of analysis for which the call must be performed (TRAN, AC, ...)
- **WHERE**  
specifies when Eldo must call the Tcl function:
  - **START**  
only one call at the very beginning of the simulation
  - **START\_OF\_RUN**  
one call at the beginning of each run
  - **END\_OF\_RUN**  
one call at the end of each run
  - **END**  
one call at the very end of the simulation
- **PLOT**  
specifies that if the Tcl function returns a wave, Eldo will have to dump it in the output file.
- **LABEL**  
specifies a name which will be used to identify the resulting wave of the Tcl function in the output file.

- `tcl_function_call`

the Tcl function. Since it is a direct call to a Tcl function, the name of the function is case sensitive.

## Working with Waveforms inside UDF

### Waveforms in Arguments

It is very important to notice that in the context of `.CALL_TCL` commands, waves are not real values but vectors, or in a more general way, objects are generated by Eldo and given to the PPL. The argument UDF receives is only a reference to an object. Thus it is impossible to use directly these references in mathematical expressions such as:

```
proc WAVE_PLUS { wv_in } {set wv_out [expr $wv_in + $wv_in]
+ return $wv_out}
```

### Waveforms in Expressions

To allow operations on waveforms a set of commands has been registered inside the PPL environment. The following table lists the equivalence between common operators' syntax and syntax inside the PPL.

**Table 22-1. Equivalence between C and PPL syntax**

C syntax	Example	PPL syntax	Example
+	$a = b + c$	add	set a [add \$b \$c]
-	$a = b - c$	sub	set a [sub \$b \$c]
*	$a = b * c$	mult	set a [mult \$b \$c]
/	$a = b / c$	div	set a [div \$b \$c]
^	$a = b ^ c$	pow	set a [pow \$b \$c]
fmod	$a = \text{fmod}(b,c)$	mod	set a [mod \$b \$c]

These commands accept real, complex and/or wave arguments. For example, it is possible to add two waveforms, a waveform and a real or two reals.

Looking more closely at the Tcl syntax, it is obvious that it can quickly become very difficult to read when the expression is more complex.

In common language:

```
a = ((b+c)*(d+e))/(f+g)
```

Becomes in Tcl:

```
set a [div [mult [add $b $c] [add $d $e]] [add $f $g]]
```

For this purpose, two commands have been defined to simplify the syntax:

```
evalExpr
defineVec
```

Examples *expressions.cir* and *expressions.tcl* show the two different syntax. See [Table 22-4](#) on page 22-29 for more information on example files.

## evalExpr Command

Evaluate an expression in C-Like syntax:

```
evalExpr expression
```

The evalExpr command allows evaluation of an expression written using C-like syntax. It can contain both numbers and/or waves with quotes being mandatory. If the expression is purely numerical a single value is returned. Otherwise the result is a vector and is printed using the display command format (display command and output will be described later).

This command is more likely to be used with numerical expressions whereas the defineVec command is for wave calculations.

For example:

```
proc EVAL_EXPR { wv_in } {
    set a 3
    set numerical [evalExpr "2*$a + 4"]
    puts "numerical = $numerical"
    set vector [evalExpr "2*$wv_in + 4"]
    puts "vector = $vector"
}
```

A call to this function from Eldo command:

```
.call_tcl tran where=END eval_expr(v(1))
```

will produce the following output:

```
numerical = 10.00000000000000
vector = { 0 0.00000000000000 4.00000000000000 +
0.000000000000j }
{ 1 0.00000000002000 4.50133293425722 + 0.00000000000000j }
{ 2 0.00000000003900 4.97030313987119 + 0.00000000000000j }
{ 3 0.00000000007699 5.86034757905359 + 0.00000000000000j }
{ 4 0.00000000015297 7.27942102760290 + 0.00000000000000j }
{ 5 0.00000000022966 7.96738275669581 + 0.00000000000000j }
{ 6 0.00000000025000 8.00000000000000 + 0.00000000000000j }
{ 7 0.00000000029068 7.87006296906364 + 0.00000000000000j }
{ 8 0.00000000035325 7.18743864881722 + 0.00000000000000j }
```

## defineVec Command

Define a PPL object using C-Like syntax

```
defineVec <output_object> expression
```

Like the evalExpr command, defineVec allows evaluation of an expression written using C-like syntax but the result is stored in a PPL object named <output\_object> rather than displayed. An error will occur if an object using the same name already exists in the PPL.

This command is more likely to be used with wave calculations whereas the evalExpr command is for numerical expressions.

For example:

```
proc DEFINE_EXPR { wv_in } {
    set a 3
    defineVec numerical "2*$a+4"
    puts "numerical = [display numerical]"

    defineVec vector "$wv_in+4"
    puts "vector = [display vector]"
    return vector
}
```

A call to this function from the Eldo command:

```
.call_tcl tran where=END define_expr(v(1))
```

will produce the same output as function EVAL\_EXPR but the object *vector* can be reused in any expression or returned to Eldo in order to be dumped in the output file.

It is very important to notice that the return value of this function is not a Tcl variable but directly *vector*.

## Built-in Waveform Functions

**Table 22-2. Built-in Waveform Functions**

ABS	AVG	ACOS	ACOSH	ACOT
ACOTH	ASIN	ASINH	ATAN	ATANH
COMPRESS	COS	COSH	COT	COTH
DB	DEG	DELAY	DERIVE	DRV
EXP	FALLTIME	GMARGIN	HISTOGRAM	IIPX

**Table 22-2. Built-in Waveform Functions**

<b>IMAG</b>	<b>INTEG</b>	<b>INTEGRAL</b>	<b>INTERSECT</b>	<b>INVDB</b>
<b>LOG</b>	<b>LOG10</b>	<b>MAG</b>	<b>MAX</b>	<b>MIN</b>
<b>OIPX</b>	<b>PHASE</b>	<b>PHMARGIN</b>	<b>RAD</b>	<b>REAL</b>
<b>RELATION</b>	<b>RISETIME</b>	<b>RMS</b>	<b>SAMPLE</b>	<b>SETTLINGTIME</b>
<b>SIN</b>	<b>SINH</b>	<b>SQR</b>	<b>SQRT</b>	<b>SUM</b>
<b>TAN</b>	<b>TANH</b>	<b>TPD</b>	<b>TPDDD</b>	<b>TPDDU</b>
<b>TPDUD</b>	<b>TPDUU</b>	<b>VMAX</b>	<b>VMIN</b>	<b>WINDOW</b>
<b>XCOMPRESS</b>	<b>XVAL</b>	<b>XWAVE</b>	<b>YVAL</b>	

## **ABS**

Absolute wave.

### **Command Syntax**

```
set wv_out [abs $wv_in]
```

## **EXP**

Wave exponential.

### **Command Syntax**

```
set wv_out [exp $wv_in]
```

## **LOG, LOG10**

Wave logarithm.

### **Command Syntax**

```
set wv_out [log $wv_in]
set wv_out [log10 $wv_in]
```

## **SQRT**

Wave square root.

### **Command Syntax**

```
set wv_out [sqrt $wv_in]
```

## SQR

Wave square.

### Command Syntax

```
set wv_out [sqr $wv_in]
```

## SIN, COS, TAN, COT

Trigonometric wave functions.

### Command Syntax

```
set wv_out [sin $wv_in]
set wv_out [cos $wv_in]
set wv_out [tan $wv_in]
set wv_out [cot $wv_in]
```

## ASIN, ACOS, ATAN, ACOT

Inverse trigonometric wave functions.

### Command Syntax

```
set wv_out [asin $wv_in]
set wv_out [acos $wv_in]
set wv_out [atan $wv_in]
set wv_out [acot $wv_in]
```

## SINH, COSH, TANH, COTH

Hyperbolic wave functions.

### Command Syntax

```
set wv_out [sinh $wv_in]
set wv_out [cosh $wv_in]
set wv_out [tanh $wv_in]
set wv_out [coth $wv_in]
```

## ASINH, ACOSH, ATANH, ACOTH

Inverse hyperbolic wave functions.

### Command Syntax

```
set wv_out [asinh $wv_in]
set wv_out [acosh $wv_in]
set wv_out [atanh $wv_in]
set wv_out [acoth $wv_in]
```

## DB, INVDB

Conversion linear/dB magnitude.

### Command Syntax

```
set wv_out [db $wv_in]  
set wv_out [invdb $wv_in]
```

## DEG, RAD

Conversion radians/degrees.

### Command Syntax

```
set wv_out [deg $wv_in]  
set wv_out [rad $wv_in]
```

## DERIVE

Derivative of a wave at an x value.

### Command Syntax

```
set out_value [derive $wv_in $xvalue]
```

## DRV

Derivative of a wave with respect to the x (or scale) variable.

### Command Syntax

```
set wv_out [drv $wv_in]
```

## MAG

Linear magnitude from real and imaginary part.

### Command Syntax

```
set wv_out [mag $wv_in]
```

## PHASE

Phase (in degrees) from real and imaginary part.

### Command Syntax

```
set wv_out [phase $wv_in]
```

## REAL, IMAG

Real and imaginary part of a magnitude and phasewave pair (magnitude: linear, phase: in degrees).

### Command Syntax

```
set wv_out [real $wv_in]  
set wv_out [imag $wv_in]
```

## INTEG

Definite integral of the input wave with respect to the x (or scale) variable, global integral or interval integral the result has the unit {wave\_unit}\*{x\_unit}.

### Command Syntax

```
set out_value [integ $wv_in]  
set out_value [integ $wv_in $lower_bound $upper_bound]
```

- lower\_bound
  - x value after which measurement starts
- upper\_bound
  - x value after which measurement stops

## INTEGRAL

Indefinite integral of the input wave with respect to the x-axis variable. The result is a waveform.

### Command Syntax

```
set wv_out [integral $wv_in $first_y_value]
```

- first\_y\_value
  - Integration constant

## AVG

Average value of a wave.

### Command Syntax

```
set out_value [avg $wv_in]  
set out_value [avg $wv_in $lower_bound $upper_bound]
```

- lower\_bound
  - x value after which measurement starts

- **upper\_bound**  
x value after which measurement stops

## MIN, MAX

Minimum/maximum wave of two waves (min/max is evaluated point-by-point).

### Command Syntax

```
set wv_out [min $wv_in1 $wv_in2]
set wv_out [max $wv_in1 $wv_in2]
```

## VMIN, VMAX

Minimum/maximum value of a wave.

### Command Syntax

```
set out_value [vmin $wv_in]
set out_value [vmin $wv_in $lower_bound $upper_bound]
set out_value [vmax $wv_in]
set out_value [vmax $wv_in $lower_bound $upper_bound]
```

- **lower\_bound**  
x value after which measurement starts
- **upper\_bound**  
x value after which measurement stops

## RISETIME

Rise time of a wave, the result is of unit {x\_unit}.

### Command Syntax

```
set out_value [risetime $wv_in $lower_thr $upper_thr]
set out_value [risetime $wv_in $lower_thr $upper_thr $after]
```

- **lower\_thr**  
Lower threshold.
- **upper\_thr**  
Upper threshold.
- **after**

First transition between lower and upper thresholds is treated after this x value.

## FALLTIME

Fall time of a wave, the result is of unit {x\_unit}.

### Command Syntax

```
set out_value [falltime $wv_in $lower_thr $upper_thr]  
set out_value [falltime $wv_in $lower_thr $upper_thr $after]
```

- **lower\_thr**  
Lower threshold.
- **upper\_thr**  
Upper threshold.
- **after**  
First transition between lower and upper thresholds is treated after this x value.

## PHMARGIN

Phase margin is the phase distance to -180 degrees.

### Command Syntax

```
set out_value [phmargin $wv_in1 $wv_in2]
```

## GMARGIN

Gain margin is the linear gain distance to 1.0.

### Command Syntax

```
set out_value [gmargin $wv_in1 $wv_in2]
```

## YVAL

Wave value at x (or scale) value (linear interpolation to get the exact value).

### Command Syntax

```
set out_value [yval $wv_in $xvalue]
```

## XVAL

All x (or scale) values for one wave (or y) value.

### Command Syntax

```
set out_vector [xval $wv_in $yvalue]  
set out_vector [xval $wv_in $yvalue $slope]  
set out_vector [xval $wv_in $yvalue $slope $lower_bound $upper_bound]
```

- **slope**  
Find only x values, where wave crosses y with slope:
  - slope > 0 Positive slope
  - slope < 0 Negative slope
  - slope = 0 Positive and negative slope
- **lower\_bound**  
x value after which measurement starts
- **upper\_bound**  
x value after which measurement stops

## WINDOW

Selects a window (a, b) out of a wave, the points at the interval bounds are interpolated.

### Command Syntax

```
set wv_out [window $wv_in $lower_bound $upper_bound]
```

- **lower\_bound, upper\_bound**  
Begin and end of the wave window to select.

## INTERSECT

Get the intersection point (s) of two waves.

### Command Syntax

```
set out_vector [intersect $wv_in1 $wv_in2]
set out_vector [intersect $wv_in1 $wv_in2 $lower_bound $upper_bound]]
```

- **lower\_bound**  
x value after which measurement starts
- **upper\_bound**  
x value after which measurement stops

## RMS

Root mean square value of a wave.

```
rms=sqrt ((1/ (x_max-x_min))*integ(wave^2)).
```

### Command Syntax

```
set out_value [rms $wv_in]
```

```
set out_value [rms $wv_in $lower_bound $upper_bound]
```

- **lower\_bound**  
x value after which measurement starts
- **upper\_bound**  
x value after which measurement stops

## DELAY

Creates a delayed wave.

### Command Syntax

```
set wv_out [delay $wv_in $delay_value]
```

## XWAVE

Creates a wave with y = x values.

### Command Syntax

```
set wv_out [xwave $wv_in]
```

## SETTLINGTIME

Time required for the input wave to settle within a certain limit around the final value.

### Command Syntax

```
set out_value [settlingtime $wv_in $yvalue $eps]  
set out_value [settlingtime $wv_in $yvalue $eps $lower_bound $upper_bound]
```

- **yvalue**  
Value to reach.
- **eps**  
Tolerance value (yvalue +/- eps/2).
- **lower\_bound**  
x value after which measurement starts
- **upper\_bound**  
x value after which measurement stops

## HISTOGRAM

Generate a histogram of the input wave showing the waves magnitude probability density distribution. The input wave is first sampled to equidistant points if the optional argument <\$sample> is set.

### Command Syntax

```
set wv_out [histogram $wv_in $intervals]
set wv_out [histogram $wv_in $intervals $sample]
set wv_out [histogram $wv_in $intervals $lower_bound $upper_bound]
set wv_out [histogram $wv_in $intervals $lower_bound $upper_bound $sample]
```

- **intervals**  
the number of histogram intervals
- **sample**  
if 0 or not specified: don't sample
- **lower\_bound**  
x value after which measurement starts
- **upper\_bound**  
x value after which measurement stops

## SUM

Sums up the real part of a vector.

### Command Syntax

```
set out_value [sum $wv_in]
set out_value [sum $wv_in $lower_bound $upper_bound]
```

- **lower\_bound**  
x value after which measurement starts
- **upper\_bound**  
x value after which measurement stops

## RELATION

Generates a wave from the point-by-point comparison of two waves. It returns 1 whenever operator is 1 and wv\_in1 > wv\_in2, or operator is 0 and wv\_in1 == wv\_in2, or operator is -1 and wv\_in1 < wv\_in2. Otherwise it returns 0.

### Command Syntax

```
set wv_out [relation $wv_in1 $wv_in2 $operator]
```

## SAMPLE

Create a sampled wave.

### Command Syntax

```
set wv_out [sample $wv_in $sample_time]
set wv_out [sample $wv_in $sample_time $lower_bound $upper_bound $sample]
```

- `sample_time`  
sampling period
- `lower_bound`  
x value after which sampling starts
- `upper_bound`  
x value after which sampling stops

## RF Functions

### XCOMPRESS

Extracts the X-axis value of the wave at the point where the difference between the actual value of the wave and the linear extrapolation of the wave based on the computed slope value becomes greater than val.

### Command Syntax

```
set out_value [xcompress $wv_in $val]
```

### COMPRESS

Extracts the Y-axis value of the wave at the point where the difference between the actual value of the wave and the linear extrapolation of the wave based on the computed slope value becomes greater than val.

### Command Syntax

```
set out_value [compress $wv_in $val]
```

### IIPX

Returns the input referred intercept point of order x from the value of the circuit input and output, wv\_in and wave\_out respectively. wv\_in and wave\_out must be in dB or dBm. The intercept order is directly calculated from the inter modulation of freq1 and freq2.

### Command Syntax

```
set out_value [iipx $wv_in $wave_out $freq1 $freq2]
```

## OIPX

Returns the output referred intercept point of order x from the value of the circuit output wave, this must be in dB or dBm. The intercept order is directly calculated from the inter modulation of freq1 and freq2.

### Command Syntax

```
set out_value [oipx $wv_in $freq1 $freq2]
```

## TPD

Returns the propagation delay between wv\_in1 and wv\_in2.



For a complete description of the function and it's parameters, see [.EXTRACT command](#) on page 10-95 of the *Eldo User's Manual*.

---

### Command Syntax

```
set out_value [tpd $wv_in1 $wv_in2]
set out_value [tpd $wv_in1 $wv_in2 $vth]
set out_value [tpd $wv_in1 $wv_in2 $vthin $vthout]
set out_value [tpd $wv_in1 $wv_in2 $vthin $vthout $before $after]
set out_value [tpd $wv_in1 $wv_in2 $vthin $vthout $before $after $occur]
```

- **vth**  
Voltage defining a threshold or starting point.
- **vthin**  
Voltage defining a threshold or starting point.
- **vthout**  
Voltage defining a threshold or starting point.
- **before**  
Causes the function to be performed only if TIME < val .
- **after**  
Causes the function to be performed only if TIME > val .
- **occur**  
Computes the function for the VALth occurrence of the event.

## TPDUU

Returns the propagation delay(TPD) with WAVE1 and WAVE2 both rising.

## TPDUD

Returns the propagation delay(TPD) with WAVE1 rising, WAVE2 falling.

## TPDDU

Returns the propagation delay(TPD) with WAVE1 falling, WAVE2 rising.

## TPDDD

Returns the propagation delay(TPD) with WAVE1 and WAVE2 both falling.

# Built-in DSP Functions

**Table 22-3. Built-in DSP functions**

AUTOCOR	CONV	CORRELO	CT	FFT
HARMONICS	HDIST	IFFT	PERIODO	PSD
SAMPLER	SNR			

## AUTOCOR

Calculate the Auto correlation Function (AF) of a waveform.

The AF of a signal waveform is an average measure of its time domain properties and therefore especially relevant when the signal is random.

There are two methods available for calculating the Auto correlation, namely the Correlogram and Periodogram methods.



For more explanation refer to *Xelga User's Manual* page 2-53.

---

## Command Syntax

```
set wv_out [autocor $wv_in $method]
set wv_out [autocor $wv_in $method $nbpts]
```

- method

Select the AF calculation method.

0 for correlogram (default)

1 for periodogram

- **nbpts**  
Number of points for the AF result.

## CORRELO

Calculate the Power Spectral Density using correlogram method.



For more explanation refer to *Xelga User's Manual* [page 2-53](#).

---

### Command Syntax

```
set wv_out [correlo $wv_in $tstart $tstop $fs $nbpts $padding + $sample  
$fmin $fmax $normalized $ncor $nauto $npsd]
```

- **tstart**  
Start time for the signal.
- **tstop**  
Stop time for the signal.
- **fs**  
Sampling frequency.
- **nbpts**  
Number of sampling points.
- **padding**  
Selects the padding zeros type:
  - 0 No Padding
  - 1 Padding at the end
  - 2 Padding at the start
  - 3 Center wave
- **sample**  
Selects the sampling type:
  - 0 No sampling
  - 1 Interpolation
  - 2 Sample and Hold
- **fmin**  
Starting frequency used inside the FFT result window.
- **fmax**  
Last frequency used inside the FFT result window.

- **normalized**
  - 0 No normalization
    - 1 All real and imaginary parts of the result are divided by Nbpts/2 except for the first point, which is divided by Nbpts.
- **ncor**

Number of auto correlation points used for the PSD computation. Default value is Nauto.
- **nauto**

Number of points for the auto correlation result. Default value is Nbpts. Necessary condition is Nauto  $\geq 2$
- **npsd**

Number of points for the PSD result. Default value is Ncorr/2 + 1. Necessary condition is npsd  $\geq$  Ncorr/2 + 1.

## PERIODO

Calculate the Power Spectral Density using periodogram method.



For more explanation refer to *Xelga User's Manual* [page 2-53](#).

---

### Command Syntax

```
set wv_out [periodo $wv_in $tstart $tstop $fs $nbpts $padding + $sample  
$fmin $fmax $normalized $ncor $nauto $npsd]
```

- **tstart**

Start time for the signal.
- **tstop**

Stop time for the signal.
- **fs**

Sampling frequency.
- **nbpts**

Number of sampling points.
- **padding**

Selects the padding zeros type:

  - 0 No Padding
  - 1 Padding at the end
  - 2 Padding at the start
  - 3 Center wave

- **sample**  
Selects the sampling type:
  - 0 No sampling
  - 1 Interpolation
  - 2 Sample and Hold
- **fmin**  
Starting frequency used inside the FFT result window.
- **fmax**  
Last frequency used inside the FFT result window.
- **normalized**
  - 0 No normalization
  - 1 All real and imaginary parts of the result are divided by Nbpts/2 except for the first point, which is divided by Nbpts.
- **ncor**  
Number of auto correlation points used for the PSD computation. Default value is Nauto.
- **nauto**  
Number of points for the auto correlation result. Default value is Nbpts. Necessary condition is  $Nauto \geq 2$
- **npsd**  
Number of points for the PSD result. Default value is  $Ncorr/2 + 1$ . Necessary condition is  $npsd \geq Ncorr/2 + 1$ .

## CONV

Calculate the convolution of two signals.



For more explanation refer to *Xelga User's Manual* [page 2-53](#).

---

### Command Syntax

```
set wv_out [conv $wv_in1 $wv_in2 $nptsa $nptsb $fs]
```

- **nptsa**  
Number of points for wave1.
- **nptsb**  
Number of points for wave2.

- **fs**  
Sampling Frequency.

## CT

Calculate the Chirp Transformed (CT) wave.



For more explanation refer to *Xelga User's Manual* [page 2-52](#).

### Command Syntax

```
set wv_out [ct $wv_in $tstart $tstop $fs $nbpts $padding $sample $fmin  
$fmax $normalized $nzoom]
```

- **tstart**  
Start time for the signal.
- **tstop**  
Stop time for the signal.
- **fs**  
Sampling frequency.
- **nbpts**  
Number of sampling points.
- **padding**  
Selects the padding zeros type:
  - 0 No Padding
  - 1 Padding at the end
  - 2 Padding at the start
  - 3 Center wave
- **sample**  
Selects the sampling type:
  - 0 No sampling
  - 1 Interpolation
  - 2 Sample and Hold
- **fmin**  
Starting frequency used inside the FFT result window.
- **fmax**  
Last frequency used inside the FFT result window.

- **normalized**
  - 0 No normalization
  - 1 All real and imaginary parts of the result are divided by Nbpts/2 except for the first point, which is divided by Nbpts.
- **nzoom**
  - Number of points for the zooming.

## FFT

Calculate the Fast Fourier Transform (FFT) of a wave. FFT is the fastest and most efficient available algorithm for computing the DFT.

### Command Syntax

```
set wv_out [fft $tstart $tstop $fs $nbpts $padding $sample $fmin $fmax  
$normalized]
```

## HARMONICS

Calculate harmonics inside the interval [fmin, fmax] then generates the harmonics wave.

### Command Syntax

```
set wv_out [harmonics $wv_in $fundf]  
set wv_out [harmonics $wv_in $fundf $fmin $fmax]
```

- **fundf**
  - Fundamental Frequency.
- **fmin, fmax**
  - frequency band that should be taken for the computation.

## HDIST

Calculate the Total Harmonic Distortion (THD) of a signal.

### Command Syntax

```
set out_value [hdist $wv_in $fundf]  
set out_value [hdist $wv_in $fundf $fmin $fmax]
```

- **fundf**
  - Fundamental frequency.
- **fmin, fmax**
  - Frequency band that should be taken for the computation.

## IFFT

Calculate the Inverse Fast Fourier Transform.

### Command Syntax

```
set wv_out [ifft $wv_in $fstart $fstop $nbpts $padding $sample]
```

- **fstart**

Start frequency for the signal.

- **fstop**

Stop frequency for the signal.

- **npts**

Number of sampling points.

- **padding**

Selects the padding zeros type:

0 No Padding

1 Padding at the end

2 Padding at the start

3 Center wave

- **sample**

Selects the sampling type:

0 No sampling

1 Interpolation

2 Sample and Hold

## PSD

Calculate the Power Spectral Density (PSD) of a waveform.

The PSD of a signal waveform is an average measure of its time domain properties and therefore especially relevant when the signal is random.

There are two methods available for calculating the PSD, namely the Correlogram and Periodogram methods.



For more explanation refer to *Xelga User's Manual* [page 2-53](#).

---

### Command Syntax

```
set wv_out [psd $wv_in $method]
```

- ```
set wv_out [psd $wv_in $method $nbpts]
```
- **method**  
Select the PSD calculation method.
    - 0 for correlogram (default)
    - 1 for periodogram
  - **nbpts**  
Number of points for the PSD result.

## SAMPLER

Generate a sampled wave using some types of windowing.

### Command Syntax

- ```
set wv_out [sampler $tstart $tstop $fs $nbpts $padding $wtype $wparam]
```
- **tstart**  
Start time for the signal.
  - **tstop**  
Last time for the signal.
  - **fs**  
Sampling frequency.
  - **Nbpts**  
Number of sampling points.
  - **padding**  
Selects the padding zeros type:
    - 0 No Padding
    - 1 Padding at the end
    - 2 Padding at the start
    - 3 Center wave
  - **wtype**  
Window type:
    - 0 Rectangular (no wparam)
    - 1 Hamming (no wparam)
    - 2 Hanning (Alpha= w\_param, default is 0.5)
    - 3 Parzen (no wparam)

- 4 Welch (no wparam)
  - 5 Blackman (no wparam)
  - 6 Blackman7 (no wparam)
  - 7 Bartlett (no wparam)
  - 8 Kaiser (Beta= wparam, default is 10.056)
  - 9 Klein (no wparam)
  - 10 Tukey (no wparam)
  - 11 Dolph Chebyshev (Alpha=w\_param, default: 3.0)
- **wparam**  
Window optional parameter.

## SNR

Calculate the Signal to Noise Ratio in dBs of a noisy wave by using a specific frequency list.

### Command Syntax

```
set out_value [snr $wv_in $fmin $fmax $freq_list]
```

- **fmin, fmax**  
Frequency band that should be taken for the computation.
- **freq\_list**  
List of frequencies which should be used in the computation.

## Accessing Waves inside an External Database

### LOAD\_FILE

Load a Cou file in memory.

### Command Syntax

```
load_file $filename
```

This command returns a list of run identifiers. These identifiers are labelled RunX where X is an absolute counter starting from 0.

### Example

If a file *test.cou* contains one run:

```
puts "[load_file test.cou]"
```

Implies “Run0” will be used as the run identifier label.

If a file *test2.cou* contains three runs:

```
puts "[load_file test2.cou]"
```

Implies “Run0 Run1 Run2” will be used as the run identifier labels.

If both commands are called in the same UDF:

```
puts "[load_file test.cou]"
```

Implies “Run0” will be used as the run identifier label.

```
puts "[load_file test2.cou]"
```

Implies “Run1 Run2 Run3” will be used as the run identifier labels.

These identifiers are used to identify waves. For example if file *test.cou* contains waves v(1) and i(r1), their internal names inside the PPL will be Run0::v(1) and Run0::i(r1). These names can be used in all expressions.

## DISP\_FILE

Display the list of runs inside the PPL or the list of waves inside a given run.

### Command Syntax

```
disp_file  
disp_file $identifier
```

### Example

Considering that *test.cou* contains one run and *test2.cou* contains three runs,

```
puts "[load_file test.cou]"
```

Implies “Run0” will be used as the run identifier label.

```
puts "[load_file test2.cou]"
```

Implies “Run1 Run2 Run3” will be used as the identifier labels.

```
puts "[disp_file]"
```

Implies “Run0 Run1 Run2 Run3” will be used as the identifier labels.

```
puts "[disp_file Run0]"
```

Implies “V(1) I(R1)” will be used as the identifier labels.

## UNLOAD\_FILE

Remove all or a single run from memory.

### Command Syntax

```
unload_file all
```

- ```
unload_file $identifier
```
- **identifier**  
a valid run identifier or keyword “all” to remove all runs.

## Miscellaneous Commands

### DISPLAY

Write the content of PPL objects to the standard output.

#### Command Syntax

```
display $ppl_object
```

ppl\_object is a reference to a PPL object. Usually, references for waves start with WV and references for numbers start with DB.

### GETSIZE

Return the number of points in a wave.

#### Command Syntax

```
getsize $wv_in
```

### GETPOINT

Return the x value, real and imaginary parts of a wave point.

#### Command Syntax

```
getpoint $wv_in $ptindex
```

- **ptindex**  
Index of a point (between [0; size]).

### SETPOINT

Modify a wave point value.

#### Command Syntax

```
setpoint $wv_in $ptindex $xvalue $yvalue $yimgvalue
```

- **ptindex**  
Index of a point (between [0; size]).

- **xvalue**  
New x-axis value.
- **yvalue**  
New real part value.
- **yimgvalue**  
New imaginary part value.

## CREATEVECTOR

Create a vector from a list of (x,y) values.

### Command Syntax

```
createVector vectorName $xlist $ylist
```

- **vectorName**  
the name of the PPL object which will represent the vector in the library.
- **xlist**  
A list of ordered x values.
- **ylist**  
A list of values (same length as xlist).

### Example

```
createVector essai "0 1e-9 2e-9 3e-9 4e-9" "1.1 2.2 3.3 4.4
+ 5.5"
```

## Examples

Examples can be found in the directory: `$MGC_AMS_HOME/eldo/$eldover/examples/ppl`

**Table 22-4. Post-Processing Library Examples**

| File Names                                                     | Description                                                                                             |
|----------------------------------------------------------------|---------------------------------------------------------------------------------------------------------|
| <code>realvalue.cir</code> ,<br><code>realvalue.tcl</code>     | Demonstrate how to use UDF in a macro-like way to define a resistor value.                              |
| <code>samphold.cir</code> ,<br><code>samphold.tcl</code>       | Demonstrate how to use UDF in a macro-like way to define defwaves, and namespace in Tcl.                |
| <code>expressions.cir</code> ,<br><code>expressions.tcl</code> | Demonstrate how to use <code>.CALL_TCL</code> commands and how to work with expressions inside the UDF. |
| <code>wave_info.cir</code> ,<br><code>wave_info.tcl</code>     | Demonstrates how to use wave informations commands and loops in Tcl.                                    |

**Table 22-4. Post-Processing Library Examples**

| File Names                                     | Description                                                                                      |
|------------------------------------------------|--------------------------------------------------------------------------------------------------|
| <i>wave_meas.cir</i> ,<br><i>wave_meas.tcl</i> | Demonstrates how to use measurement functions in UDF.                                            |
| <i>asciigen.cir</i> ,<br><i>asciigen.tcl</i>   | Demonstrates how to use several <b>.CALL_TCL</b> commands to dump waves in an ascii output file. |

# Chapter 23

## IBIS Models support in Eldo

### Introduction

The IBIS (I/O Buffer Information Specification) is a fast and accurate behavioral method of modeling I/O buffers based on V/I curve data. It protects semiconductor vendor IP from giving away their proprietary technology process. IBIS models are used for signal integrity (SI) analysis that is becoming a key feature of the successful board design. IBIS libraries are widely available from most IC vendors on their web sites.

Information from IBIS files is read using a “golden parser”, developed through efforts of the IBIS Open Forum; the information is then processed, and the behavior of I/O buffers is emulated. Using these buffers is similar to using other Spice elements, such as transistors etc. It is necessary to give a name to the buffer, then to specify a list of nodes that are used to connect the buffer to the rest of the circuit and define the references to the IBIS file and the model for a buffer.

The I/O buffer (IBIS) model in Eldo is implemented as a new element. The name of this element starts with the letters `_IO_`.

---

#### Note

 The supported version of IBIS syntax is 2.1.

---

### General Syntax

```
_IO_xx NN {NN} file="path" component="component_name"  
+ model="model_name" | pin="pin_name" [power=on|off]  
+ [device=input|output|io|tristate|open_drain|open_source|  
+ open_sink|io_open_drain|io_open_source|io_open_sink|  
+ input_ecl|output_ecl|tristate_ecl|io_ecl]  
+ [interpol=linear|spline] [VI_Corner=typ|min|max}  
+ [C_comp_corner=typ|min|max}] [Package_Corner={typ|min|max}]  
+ [use_fall_wvf=0|1|2] [use_rise_wvf=0|1|2] [warn=on|off]  
+ [K_pullup=node K_pulldown=node_name] [C_comp_pu=value  
+ C_comp_pd=value C_comp_pc=value C_comp_gc=value]
```

This statement is used to instantiate an I/O buffer. This syntax will be used for the following model types:

- Input buffer;
- Output buffer;

- Tristate buffer;
- I/O buffer;
- Open Drain buffer;
- Open Source buffer;
- Open Sink buffer;
- I/O Open Drain buffer;
- I/O Open Sink buffer;
- I/O Open Source buffer;
- Input ECL buffer;
- Output ECL buffer;
- Tristate ECL buffer;
- I/O ECL buffer.

## Parameters

- **xx**  
I/O buffer name.
- **NN**  
Names of the nodes to be connected externally. These nodes should be listed in the correct order depending on the type of model instantiated.
- **file="path"**  
Specifies the path to the *.ibs* (IBIS) file that contains an IBIS model for an IBIS component. It can be either the full path (e.g. *"/user/test/Models/APEX.ibs"*), a relative path, or just a file name. Both *"/"* and *"\"* should be acceptable as separators in the path name (one for the UNIX and one for the NT version). An Eldo option **IBIS\_SEARCH\_PATH** is available to specify the path to the directory to search for the IBIS files.
- **component="component\_name"**  
Specifies the desired component model within the IBIS file.
- **model="model\_name"**  
Specifies the desired pin model within the component model. If model is set to *model\_name*, Eldo picks up all the parameters from the IBIS model description section and uses the package information from ([Package] section) for setting the package data. Therefore the model parameter specifies the desired driver model and helps Eldo to check the node list, set the appropriate voltage sources to the **PCR\_nd**, **GCR\_nd**, **PUR\_nd**, **PDR\_nd** nodes, and defines the component pin packages.

- **pin="pin\_name"**

Specifies the name of the pin within the component model. In this case the model name corresponding to the specified pin name is used. If the package parameters (R\_pin, L\_pin, C\_pin) are specified for this particular pin they override the package parameters L\_pkg, R\_pkg, and C\_pkg given in the [Package] section. If the parameters R\_pin, L\_pin and C\_pin are not specified the package parameters are used instead.

---

**Note**



The keywords **pin** and **model** are exclusive. It is an error to specify both these keywords in the same *Iocard*.

---

## Optional parameters

- **power=on|off**

The setting of **on** means that the voltage sources (shown in Fig. 1) are connected to the reference nodes PUR\_nd, PDR\_nd, PCR\_nd, GCR\_nd internally. The user should *not* connect them in this case. The default is **on**. **power=off** specifies that the user intends to connect the voltage sources themselves (possibly through other circuit elements such as resistors, inductors, transmission lines). No internal voltage sources will be created in this case.

- **device=input|output|io|tristate|open\_drain|open\_source|  
+ open\_sink|io\_open\_drain| io\_open\_source|  
+ io\_open\_sink|input\_ecl|output\_ecl|tristate\_ecl | + io\_ecl**

Specifies the desired device model to be used to check the I/O buffer type assignment against the buffer type is given in the IBIS file. It provides the user with additional check to guard the model assignment on I/O buffer instance card.

- **VI\_Corner=typ|min|max**

Specifies the IBIS corner to be used for the V-I curves and the reference voltages. Three families of curves corresponding to the typical, minimum and maximum values can be present in an IBIS file. The default is **typ**.

- **C\_comp\_Corner=typ|min|max**

Specifies the IBIS corner to be used for the component capacitance. The default value is **typ**.

- **Package\_Corner=typ|min|max|none**

Specifies the IBIS corner for the package model.

---

**Note**



The keyword **none** can be used to specify that the user does not want to use the package model specified in the IBIS file. This can be useful for debugging and when the user has a SPICE subcircuit for the package to use.

---

- **Corner={typ|min|max|fast|slow}**

This optional keyword can be used as a short-hand notation to simultaneously set **VI\_corner**, **C\_comp\_corner** and **Package\_corner**. If both the keyword **corner** and one or more of the keywords **VI\_corner**, **C\_comp\_corner**, **Package\_corner** are given the latter take precedence. The meaning of the keyword **corner** is given in the table below:

**Table 23-1. Corner Selection**

| corner | C_comp_corner | VI_corner | Package_corner |
|--------|---------------|-----------|----------------|
| typ    | typ           | typ       | typ            |
| min    | min           | min       | min            |
| max    | max           | max       | max            |
| fast   | min           | max       | min            |
| slow   | max           | min       | max            |

- **use\_fall\_wvf=0|1|2**

Specifies whether to use the ramp (0), one or two waveforms for the falling transition. The default behavior is to use the first two waveforms specified in the IBIS model and to use ramp if no waveforms have been specified.

---

**Note**

 This is not yet implemented. The keyword will be ignored.

---

- **use\_rise\_wvf=0|1|2**

Specifies whether to use the ramp (0), one or two waveforms for the rising transition. The default behavior is to use all the waveforms specified in the IBIS model and to use ramp if no waveforms have been specified.

---

**Note**

 This is not yet implemented. The keyword will be ignored.

---

- **warn=on|off**

Specifies if the warning message generated while parsing the IBIS model should be printed (**on**) or suppressed (**off**). The default setting is **on**.

---

**Note**

 The warning messages pertaining to the mismatch between the DC voltages and the initial /final voltages of the V-I curves should not be suppressed. Our experience indicates that this mismatch has been a source of multiple simulation problems.

---

- **K\_pullup=<node>, K\_pulldown=<node\_name>**

The purpose of these keywords is to display the values of the pullup and pulldown coefficients of the device as a function of time as the device goes through rising or falling transition. This is done by creating a time dependent voltage source connected between the <node> and ground.

- **C\_comp\_pu=value, C\_comp\_pd=value, C\_comp\_pc=value, C\_comp\_gc=value**

These four keywords are optional. If none of these keywords is specified the capacitor C\_comp is connected between the die of a device and ideal ground (0). However the user may specify these keywords to split C\_comp in up to four parts. If at least one of the parameters C\_comp\_pu=value, C\_comp\_pd=value, C\_comp\_pc=value, C\_comp\_gc=value is given, a capacitor is connected between the die of the device (IO\_nd) and the corresponding node, and no capacitor is connected between IO\_nd and 0. See the circuit diagram in Fig. 1b below. The value of this capacitor is C\_comp \* value. Values are dimensionless numbers between 0 and 1. Specifying a negative value is an error. If some of these keywords are not specified but at least one of them is specified the default is zero for the unspecified keywords.

#### **Note**

 Splitting of C\_comp is needed to model ground and power bounce. It is not yet in the IBIS standard but about to be made standard.

---

## Examples

```
_IO_1 die ctrl en
+ file="/user/test/My_Examples/IBIS/icxsource.ibs"
+ component = "apex20_k"
+ model="OUT" VI_Corner=typ
+ C_Comp_Corner=max
+ Package_Corner=min
```

Specifies a tristate IBIS driver. The power clamp, ground clamp, pullup and pulldown references are taken from the OUT IBIS model for the component apex20\_k. By default power=on, therefore Eldo will connect the voltage sources to the reference nodes.

```
_IO_2 in pc_node gc_node
+ file="c101_01.ibs"
+ component = "W48C101_01"
+ model="48c101_01_gesd_in"
+ power=off
```

Specifies an input buffer. The IBIS\_SEARCH\_PATH is not given. Eldo will try to find the IBIS file = "c101\_01.ibs" in the run directory. The power clamp and ground clamp nodes are connected externally to the voltage sources possibly through other circuit elements. These external voltage sources overwrite the IBIS reference voltages.

```
.option IBIS_SEARCH_PATH="/Central_lib/Ibis/models"  
...  
_IO_3 B3_input B3_d_out  
+ file="c101_01.ibs"  
+ component = "W48C101_01"  
+ model="3"
```

Specifies an input buffer. Eldo will try to find the IBIS file = "c101\_01.ibs" in the run directory first. If there is no such file Eldo will look at "/Central\_lib/Ibis/models" directory. The model references to the pin name for the component "W48C101\_01". The model parameter "3" maps a pin name to a specific input buffer model definition. The power clamp and ground clamp references will be taken from the IBIS input model. The \_IO\_3 buffer contains the digital output node B3\_d\_out. This external node generates the stimulus for other circuits in current design.

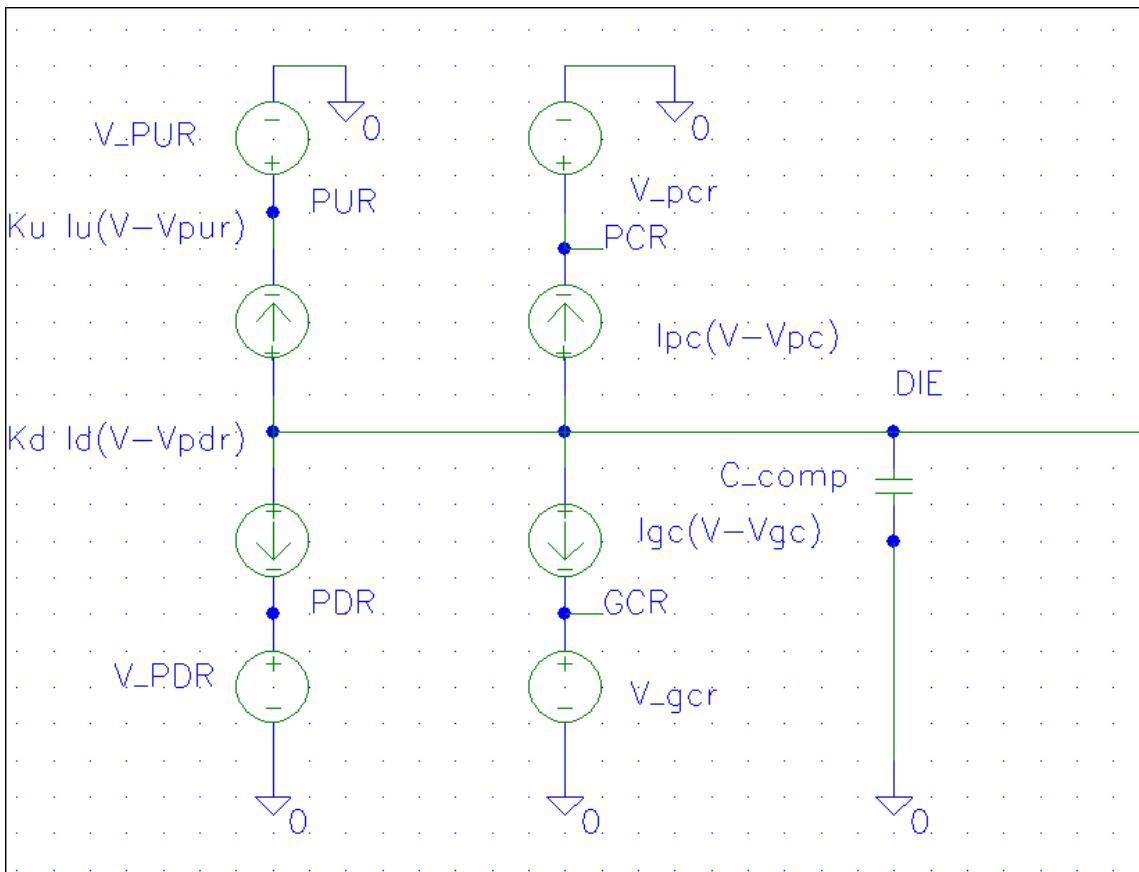
## Detailed Syntax

### Output Buffer

```
_IO_xx OUT CNTRL [PCR GCR PUR PDR]  
+ file="path" component="component_name"  
+ [[model="model_name"]|[pin="pin_name"]]  
+ [power=on|off] [device=output] [interpol=linear|spline]  
+ [[corner= {typ|min|max|fast|slow} |  
+ [VI_Corner=typ|min|max]  
+ [C_comp_corner=typ|min|max}  
+ [Package_Corner={typ|min|max}] ]  
+ [use_fall_wvf=0|1|2] [use_rise_wvf=0|1|2] [warn=on|off]
```

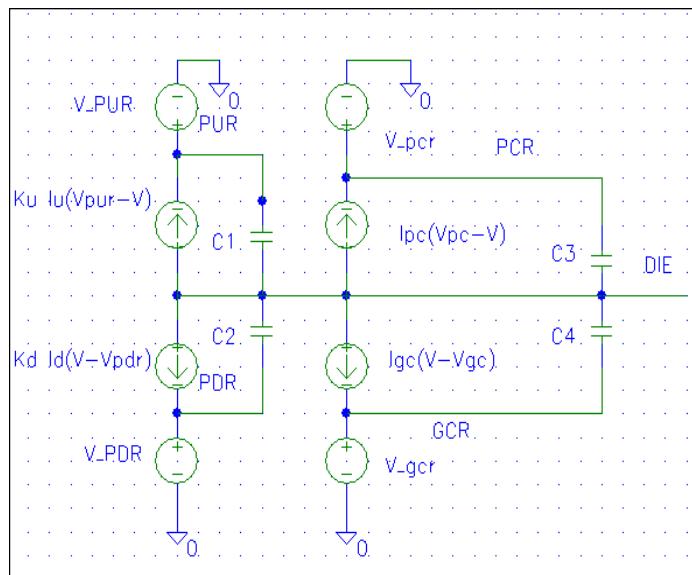
The model structure is based on IBIS macromodel with two representations for output (Fig.1) and input (Fig.2) pin models. The output pin model (Driver) on Fig.1 is valid only for 3-state pin and not for other output pin types. The structure of Driver can be divided into two parts. The first part, with time dependent pullup and pulldown voltage controlled current sources defines the output pin buffer. The second part, with powerclamp and groundclamp current sources represent the powerclamp and groundclamp that limit the signal overshoot and undershoot. The third part, which represents a package passive components R\_pkg, L\_pkg and C\_Pkg.

**Figure 23-1. Subcircuit for the nodal representation of an IBIS driver**



The IBIS driver consists of 4 voltage controlled current sources: the pullup current source connected between the nodes PUR and DIE, the pulldown current source connected between the nodes PDR and DIE, the power clamp connected between PCR and DIE, and the ground clamp connected between the nodes GCR and DIE. The pullup and pulldown current source depend both on time and voltage. There are also two digital inputs (CONTROL and ENABLE) not shown in this diagram. Therefore the pulldown and pullup components shown in this diagram will be implemented as a special kernel primitive (see below). The other two voltage sources are just VCCS (represented as G-element). Four voltage sources are used to set the potentials at the reference nodes (PUR, PDR, PCR, and DIE). In special cases when some of the potentials are zero or two potentials are equal less than 4 current sources will be needed.

**Figure 23-2. Splitting of C\_comp in case when all 4 keywords C\_comp\_pu, C\_comp\_pd, C\_comp\_gc, C\_comp\_pc are specified**



Note that:

$$C1 = C_{comp\_pu} * C_{comp}$$

$$C2 = C_{comp\_pd} * C_{comp}$$

$$C3 = C_{comp\_pc} * C_{comp}$$

$$C4 = C_{comp\_gc} * C_{comp}$$

$C_{comp\_pu}$ ,  $C_{comp\_pd}$ ,  $C_{comp\_gc}$ ,  $C_{comp\_pc}$  specify a dimensionless value between 0 and 1. It is expected they will sum to 1.

## Output ECL Buffer

```
_IO_xx OUT CNTRL [PCR GCR PUR PDR]
+ file="path" component="component_name"
+ [[model="model_name"] | [pin="pin_name"]]
+ [power=on|off][device=output_ecl] [interpol=linear|spline]
+ [[corner= [typ|min|max|fast|slow] |
+ [VI_Corner=typ|min|max]
+ [C_comp_corner=typ|min|max]]
+ [Package_Corner={typ|min|max}]]
+ [use_fall_wvf=0|1|2] [use_rise_wvf=0|1|2] [warn=on|off]
```

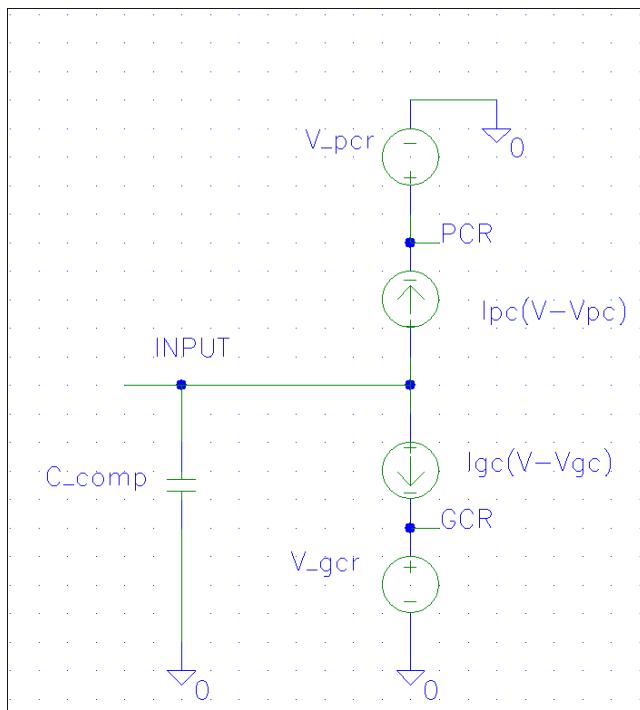
## Input Buffers

```
_IO_xx IN [ DO ] [ PCR GCR ]
+ file="path" component="component_name"
+ [[model="model_name"] | [pin="pin_name"]]
+ [power=on|off] [device=input] [interpol=linear|spline]
+ [[corner= [typ|min|max|fast|slow] |
+ [VI_Corner=typ|min|max]
+ [C_comp_corner=typ|min|max]
+ [Package_Corner={typ|min|max}]]
+ [warn=on|off]
```

The structure of Input buffer (Load) Fig.2 is quite similar to the Driver electrical model. The difference is the absence of the buffer. Only the powerclamp and groundclamp diodes are left as far as the active nonlinear part is concerned. The passive part is unchanged.

The node DO (Digital Output) is the digital value (0 or 1) representing the state (high or low) on the load.

**Figure 23-3. Subcircuit for the nodal representation of an IBIS load**



## input\_ecl

```
_IO_xx IN [DO] [PCR GCR]
+ file="path" component="component_name"
+ [[model="model_name"]|[pin="pin_name"]]
+ [power=on|off] [device=input_ecl] [interpol=linear|spline]
+ [[corner=[typ|min|max|fast|slow] |
+ [VI_Corner=typ|min|max]
+ [C_comp_corner=typ|min|max]
+ [Package_Corner={typ|min|max}]]
+ [use_fall_wvf=0|1|2] [use_rise_wvf=0|1|2] [warn=on|off]
```

## IO Buffer

```
_IO_xx OUT CNTRL EN [DO] [PCR GCR PUR PDR]
+ file="path" component="component_name"
+ [[model="model_name"]|[pin="pin_name"]]
+ [power=on|off] [device=input_output]
+ [interpol=linear|spline]
+ [[corner=[typ|min|max|fast|slow] |
+ [VI_Corner=typ|min|max]
+ [C_comp_corner=typ|min|max]
+ [Package_Corner={typ|min|max}]]
+ [use_fall_wvf=0|1|2] [use_rise_wvf=0|1|2] [warn=on|off]
```

## IO ECL Buffer

```
_IO_xx OUT CNTRL EN [DO] [PCR GCR PUR PDR]
+ file="path" component="component_name"
+ [[model="model_name"]|[pin="pin_name"]]
+ [power=on|off][device = io_ecl] [interpol=linear|spline]
+ [[corner=[typ|min|max|fast|slow] |
+ [VI_Corner=typ|min|max]
+ [C_comp_corner=typ|min|max]
+ [Package_Corner={typ|min|max}]]
+ [use_fall_wvf=0|1|2] [use_rise_wvf=0|1|2] [warn=on|off]
```

## tristate Buffer

```
_IO_xx OUT CNTRL EN [PCR GCR PUR PDR]
+ file="path" component="component_name"
+ [[model="model_name"]|[pin="pin_name"]]
+ [power=on|off] [device=three_state][interpol=linear|spline]
+ [[corner=[typ|min|max|fast|slow] |
+ [VI_Corner=typ|min|max]
+ [C_comp_corner=typ|min|max]
+ [Package_Corner={typ|min|max}]]
+ [use_fall_wvf=0|1|2] [use_rise_wvf=0|1|2] [warn=on|off]
```

## tristate ECL Buffer

```
_IO_xx OUT CNTRL EN [PCR GCR PUR PDR]
+ file="path" component="component_name"
+ [[model="model_name"]|[pin="pin_name"]]
+ [power=on|off] [device=three_state_ecl]
+ [interpol=linear|spline]
+ [[corner= {typ|min|max|fast|slow} |
+ [VI_Corner=typ|min|max]
+ [C_comp_corner=typ|min|max}]
+ [Package_Corner={typ|min|max}]]
+ [use_fall_wvf=0|1|2] [use_rise_wvf=0|1|2] [warn=on|off]
```



# Chapter 24 Tutorials

---

## Introduction

The most productive way of learning a simulation tool such as Eldo is to get ‘hands-on’ experience by sitting at a terminal and working through, stage by stage, a number of practical circuit simulation examples and tutorials. This chapter has been written to achieve this.

The tutorials cover a wide range of simple but concise circuit applications. Thus, they should be of interest not only to the novice user, but also to the experienced user wishing to learn more about specific simulation techniques within Eldo.

Upon completion, a wide range of techniques needed to perform efficient and productive analysis using Eldo should have been learnt.

Each tutorial starts with a brief description of the circuit in question together with a circuit diagram. A short summary of the Eldo commands used within the tutorial follows this, in order to aid users wishing to learn about a specific topic or the use of certain commands within Eldo. A complete circuit netlist is then shown, followed by a breakdown and explanation of each section. Actual output results from the circuit conclude each tutorial using EZwave, the Eldo waveform viewer.

A summary of the circuits used in this chapter, together with a brief description of the Eldo subject areas dealt with are listed below. Listings for these examples may be found in the following subdirectories included with your software:

`$MGC_AMS_HOME/eldo/$eldover/examples/eldo`

where `$MGC_AMS_HOME` is the directory where the software resides.

---

### Note



For more examples please refer to the [Examples](#) appendix and “[Examples for IEM](#)” on page 17-4.

---

**Table 24-1. Tutorials**

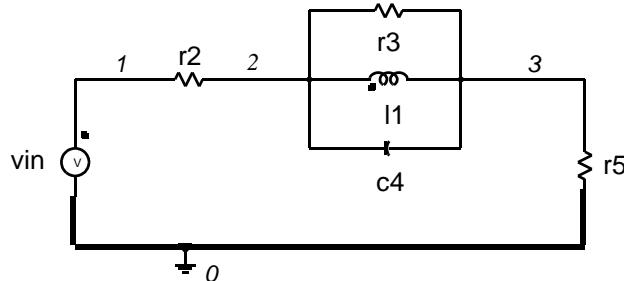
| Tutorial No. | Circuit Name            | Eldo Description                                 |
|--------------|-------------------------|--------------------------------------------------|
| 1            | <i>parallel_lcr.cir</i> | <a href="#">General introduction AC analysis</a> |
| 2            | <i>butterworth.cir</i>  | <a href="#">Transient &amp; AC analysis</a>      |

**Table 24-1. Tutorials**

| Tutorial No. | Circuit Name             | Eldo Description                                                                    |
|--------------|--------------------------|-------------------------------------------------------------------------------------|
| 3            | <i>bandpass.cir</i>      | AC & Noise analysis Model description                                               |
| 4            | <i>lowpass.cir</i>       | AC analysis. Subcircuit definition                                                  |
| 5            | <i>colpitts.cir</i>      | Transient analysis. Use of model library files                                      |
| 6            | <i>hv_cascade.cir</i>    | Transient analysis. Model description                                               |
| 7            | <i>noninvert_amp.cir</i> | AC & Monte Carlo analysis. Model description                                        |
| 8            | <i>bip_amplifier.cir</i> | DC sensitivity analysis                                                             |
| 9            | <i>sc_lowpass.cir</i>    | Transient and small signal AC analyses using the Z-domain switched capacitor models |

## Tutorial #1—Parallel LCR Circuit

This simple circuit simulation gives a general introduction to the syntax of Eldo by performing an AC analysis on a parallel LCR circuit. The complete netlist can be found in the file *parallel\_lcr.cir*.

**Figure 24-1. Parallel LCR Circuit**

Summary of Eldo Commands used in this Tutorial

- .AC—AC analysis
- .OPTION—Simulator configuration
- .PLOT—Plot simulator results

### Complete Netlist

```

LCR Parallel Network
vin 1 0 ac 10
r2 1 2 50
r3 2 3 50k
r5 3 0 50
l1 2 3 100u
c4 2 3 10n

```

```
.ac dec 10 1 1g
.option eps = 1.0e-6
.plot ac vdb(2) vdb(3) (-30,20)
.end
```

## Netlist Explanation

LCR Parallel Network

The above line is the circuit title. It must always be the first line of a simulation.

The first part of the Eldo netlist is a description of the circuit components.

```
vin 1 0 ac 10
```

The above line defines an AC voltage source **vin** connected between the nodes 1 and 0 of value 10V.

```
r2 1 2 50
r3 2 3 50k
r5 3 0 50
l1 2 3 100u
c4 2 3 10n
```

The above lines define the devices present in the circuit to be simulated. Each device instantiation gives the component name, the nodes to which the component is connected and the value of the component.

The next part of the netlist specifies the simulation control directives indicating what type of simulation Eldo should perform on the circuit.

```
.ac dec 10 1 1g
```

The above line indicates that an AC analysis should be performed on the circuit within the frequency range 1Hz to 1 GHz with 10 steps per decade.

```
.option eps = 1.0e-6
```

The above line increases the internal accuracy of Eldo from its default value of 5mV to 1 $\mu$ V. This is very important to achieve the best results for the simulation of most analog circuits.

```
.plot ac vdb(2) vdb(3) (-30,20)
```

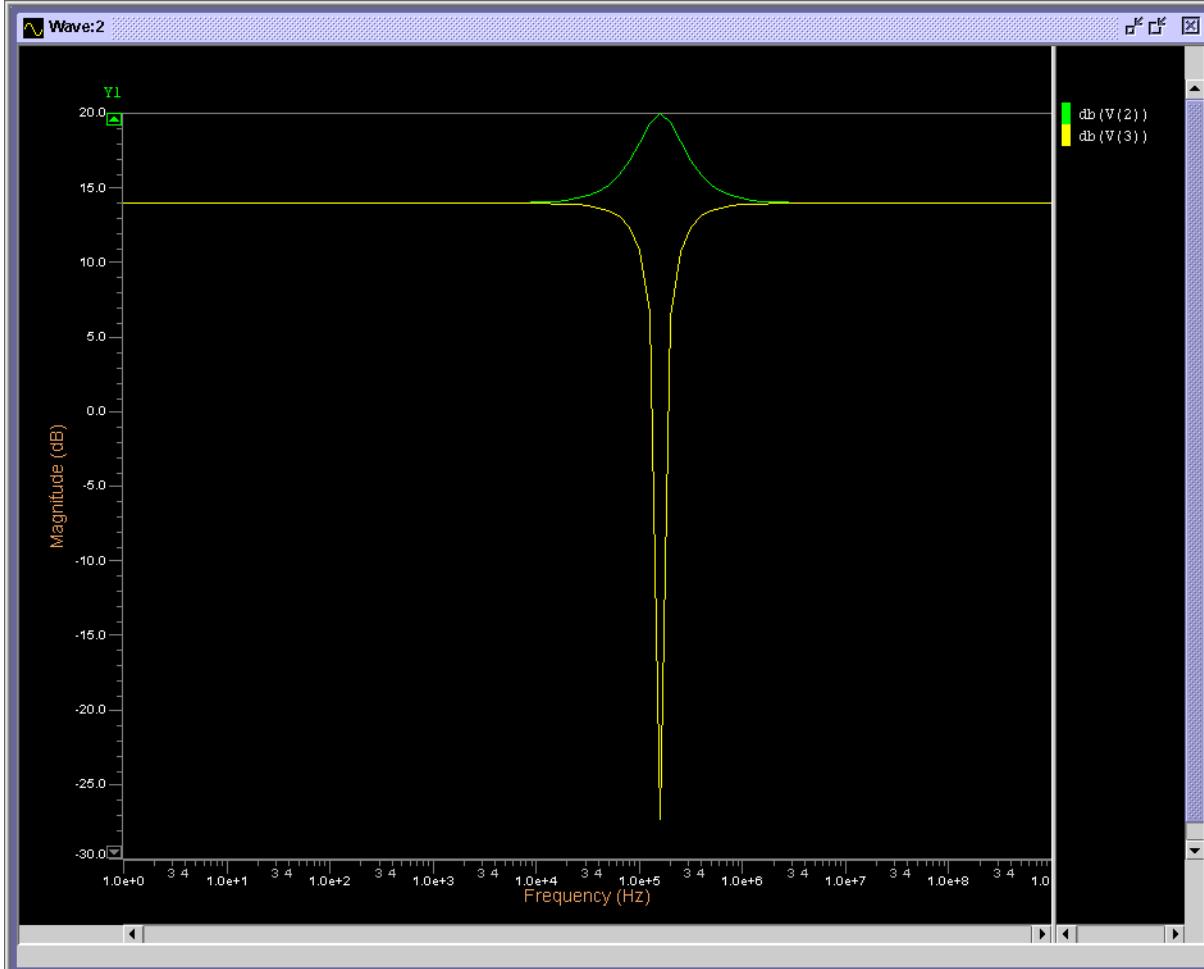
The above line specifies a dB/frequency plot of the nodes 2 and 3 on the same graph between the limits -30dB and +20dB. The results are stored in the *parallel\_lcr.wdb* file and can be displayed using the EZwave graphical results post-processor.

```
.end
```

The netlist must always be terminated with the above command.

## Simulation Results

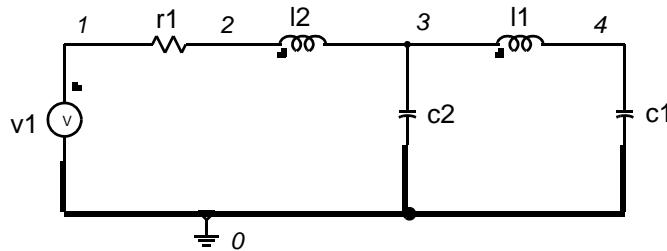
Figure 24-2. Tutorial #1—Simulation Results



## Tutorial #2—4th Order Butterworth Filter

This example simulates a 4th order Butterworth filter with a transient and an AC analysis being performed on the circuit. The complete netlist can be found in the file *butterworth.cir*.

**Figure 24-3. 4th Order Butterworth Filter**



Summary of Eldo Commands used in this Tutorial

- **.AC**—AC analysis
- **PLOT**—Plot simulator results
- **.TRAN**—Transient analysis

### Complete Netlist

```
4th Order Butterworth Filter
c1 4 0 1.5307n
c2 3 0 1.0824n
l1 3 4 1.5772u
l2 2 3 .38268u
r1 1 2 1
v1 1 0 ac 1 pwl (0 0 1u 0 2u 1 20u 1 20.1u 0)
.tran .2u 40u
.plot tran v(4) (-1,1.5)
.plot tran v(1) (0,1.5)
.ac dec 20 10000 100meg
.option eps=1.0e-6 be
.plot ac vdb(4) (-120,40)
.plot ac vp(4) (-200,200)
.end
```

### Netlist Explanation

```
4th Order Butterworth Filter
c1 4 0 1.5307n
c2 3 0 1.0824n
l1 3 4 1.5772u
l2 2 3 .38268u
r1 1 2 1
v1 1 0 ac 1 pwl (0 0 1u 0 2u 1 20u 1 20.1u 0)
```

The above line defines an AC source of 1V and a time dependent Piece Wise Linear function between the nodes 1 and 0. The **pwl** parameters describe a signal that stays at 0V until 1 $\mu$ s where it rises to 1V in 1 $\mu$ s. The signal stays at 1V until 20 $\mu$ s where it drops back to 0V in 0.1 $\mu$ s.



Refer to the output results for a pictorial representation of this signal.

```
.tran .2u 40u
```

The above line specifies that a transient analysis should be performed on the circuit lasting 40 $\mu$ s with a plotting increment for the line printer of 0.2 $\mu$ s.

```
.plot tran v(4) (-1,1.5)
.plot tran v(1) (0,1.5)
```

The above lines specify that voltage/time plots should be performed on separate graphs of the voltage at node 4 between the limits -1 and +1.5V, and of the voltage at node 1 between the limits 0 and +1.5V.

```
.ac dec 20 10000 100meg
```

The above line indicates that an AC analysis should be performed on the circuit within the frequency range 10000Hz to 100MHz with 20 steps per decade. This AC analysis statement replaces the transient analysis definition found earlier in the netlist.

```
.option eps=1.0e-6 be
```

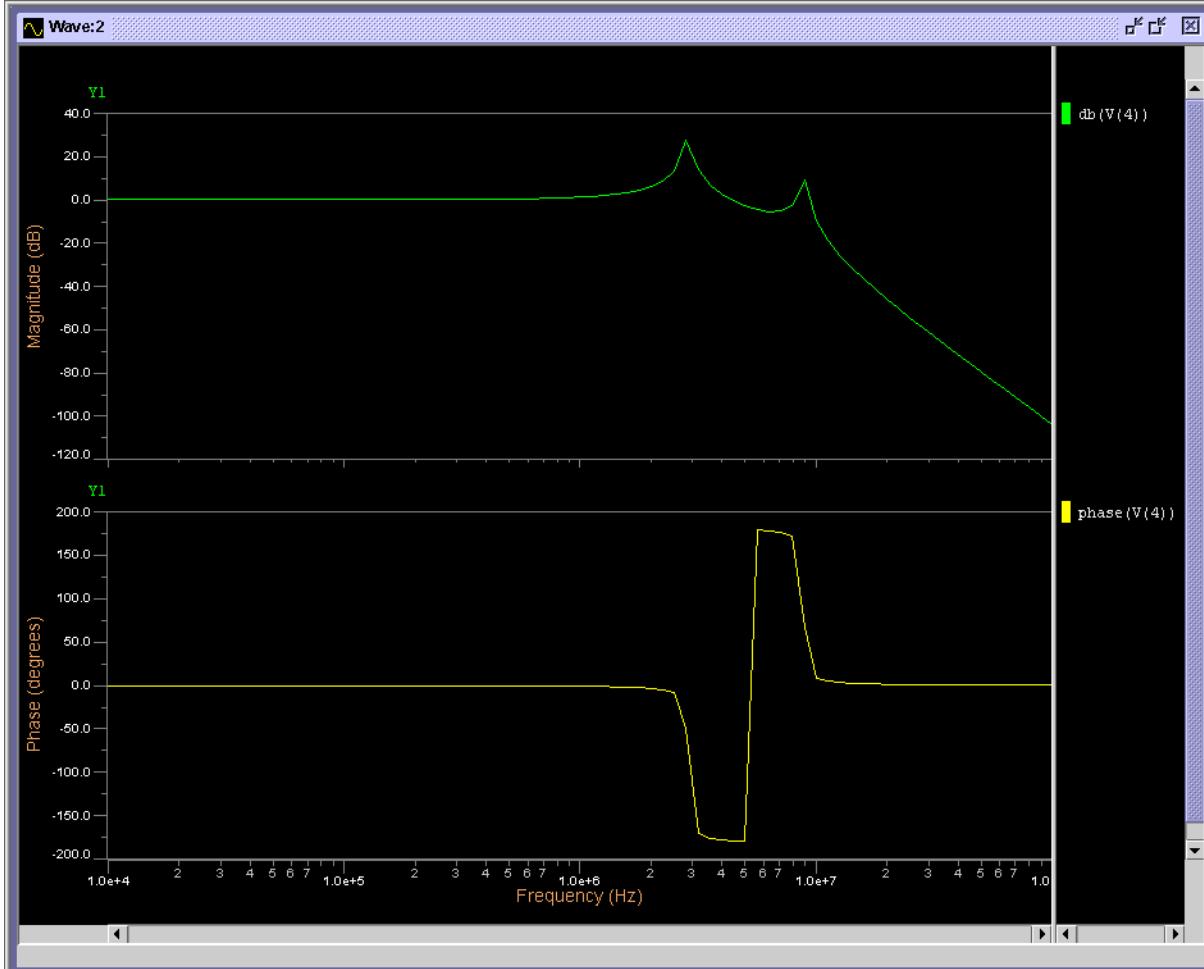
The above line sets the simulator accuracy together with the simulator algorithm as Backward Euler.

```
.plot ac vdb(4) (-120,40)
.plot ac vp(4) (-200,200)
```

The above lines specify that dB/frequency and phase/frequency plots should be performed of the voltage at node 4 between the limits -120dB and +40dB and -200 and +200 degrees respectively. These commands are added to the first simulation run netlist. The results are also added to the file *butterworth.wdb* and can be displayed as a second simulation page using the EZwave graphical post-processor.

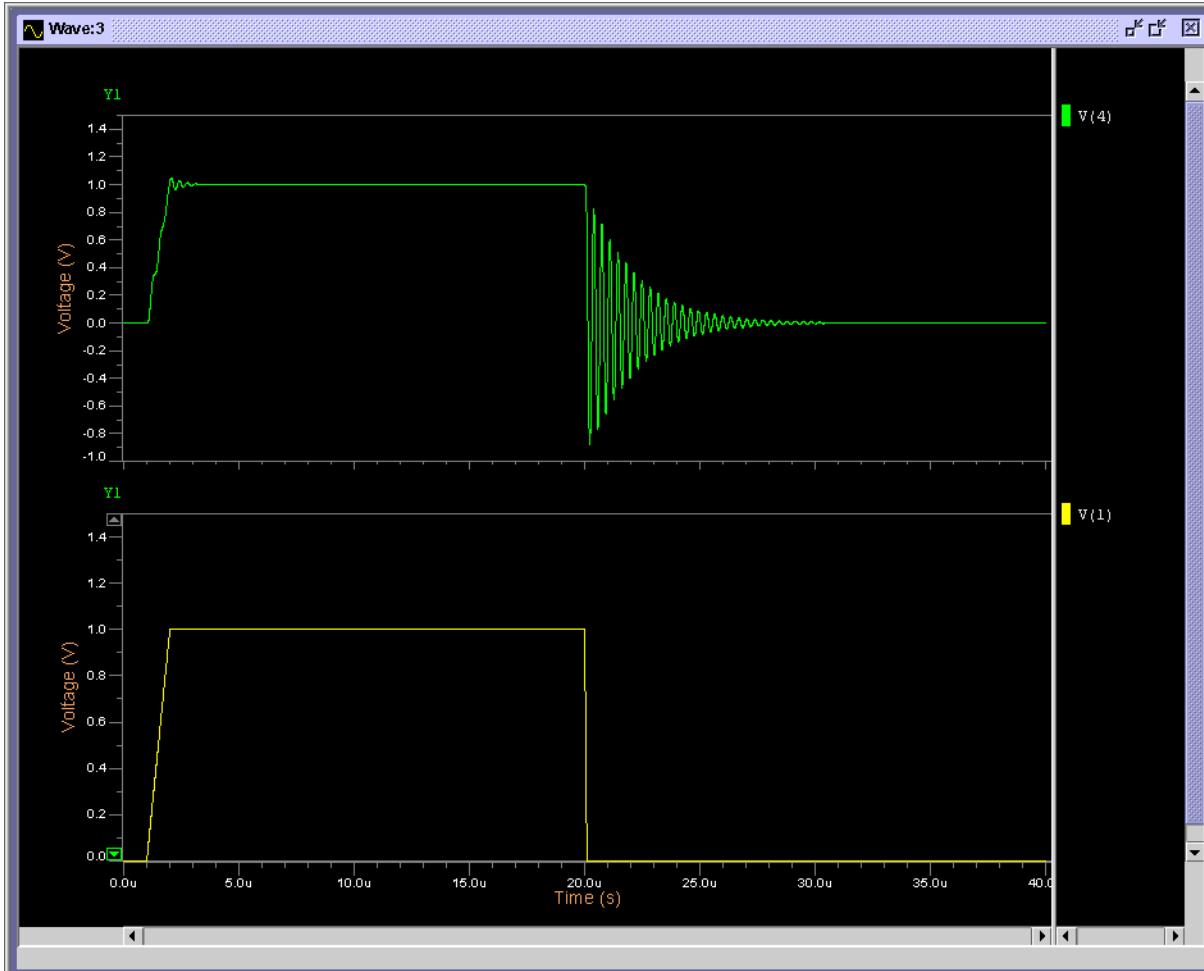
## Simulation Results—1

Figure 24-4. Tutorial #2—Simulation Results—1



## Simulation Results—2

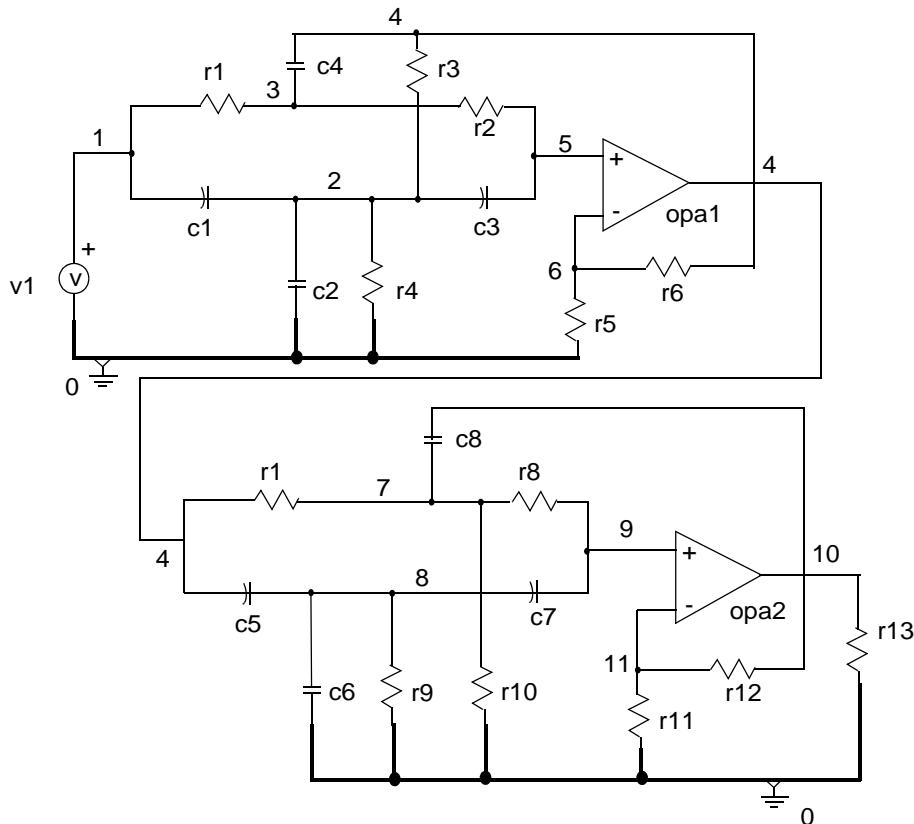
Figure 24-5. Tutorial #2—Simulation Results—2



## Tutorial #3—Band Pass Filter

This tutorial deals with an op-amp band pass filter. The simulation performs an AC analysis of the circuit, together with a noise analysis of the output stage of the filter. The complete netlist can be found in the file *bandpass.cir*.

**Figure 24-6. Band Pass Filter**



Summary of Eldo Commands used in this Tutorial

- MODEL**—Model definition
- NOISE**—Noise analysis

## Complete Netlist

```
Band-Pass filter
.model ampop modfas gain=10000.0 p1=5e3
r1 1 3 10k
r2 3 5 10k
r3 2 4 13.95k
r4 2 0 7.79k
r5 6 0 10k
r6 4 6 3.9k
r7 4 7 244.7k
r8 7 9 10k
r9 8 0 5k
r10 7 0 10.43k
r11 11 0 10k
r12 11 10 8.87k
r13 10 0 50
c1 1 2 3.27n
c2 2 0 16.73n
c3 2 5 20n
c4 3 4 40n
c5 4 8 3.17n
c6 8 0 9.5n
c7 8 9 12.7n
c8 7 10 25.3n
y1 opamp2 pin: 5 6 4 0 model: ampop
y2 opamp2 pin: 9 11 10 0 model: ampop
v1 1 0 ac
.ac dec 80 100 10k
.noise v(10) v1 80
.plot noise db(inoise)
.plot noise db(onoise)
.plot ac vdb(10) (10,-50)
.end
```

## Netlist Explanation

```
Band-Pass filter
.model ampop modfas gain=10000.0 p1=5e3
```

The above line describes the electrical parameters of the user defined model `ampop` based on the `opamp2` macromodel. The gain (`gain`) and dominant pole frequency (`p1`) of the model are set to 10000 and  $5 \times 10^3$ Hz.



For more details on the `opamp2` macromodel and its parameters, refer to [Analog Macromodels](#).

---

```

v1 1 0 ac
r1 1 3 10k
r2 3 5 10k
r3 2 4 13.95k
r4 2 0 7.79k
r5 6 0 10k
r6 4 6 3.9k
r7 4 7 244.7k
r8 7 9 10k
r9 8 0 5k
r10 7 0 10.43k
r11 11 0 10k
r12 11 10 8.87k
r13 10 0 50
c1 1 2 3.27n
c2 2 0 16.73n
c3 2 5 20n
c4 3 4 40n
c5 4 8 3.17n
c6 8 0 9.5n
c7 8 9 12.7n
c8 7 10 25.3n
yopa1 opamp2 pin: 5 6 4 0 model: ampop
yopa2 opamp2 pin: 9 11 10 0 model: ampop

```

The above lines instantiate two operational amplifiers `yopa1` and `yopa2` of macromodel type `opamp2` (linear 2-pole) connected between the nodes 5, 6, 4 and 0 and between the nodes 9, 11, 10 and 0 respectively. The electrical parameters of the macromodel are defined in the model `ampop`.

```
.ac dec 80 100 10k
```

The above line indicates that an AC analysis should be performed on the circuit within the frequency range 100Hz to 10kHz with 80 steps per decade.

```
.noise v(10) v1 80
```

The above line indicates that a noise analysis should be performed of the voltage at node 10 with the voltage source `v1` as input noise voltage. The analysis should be averaged over 80 frequency points.

```

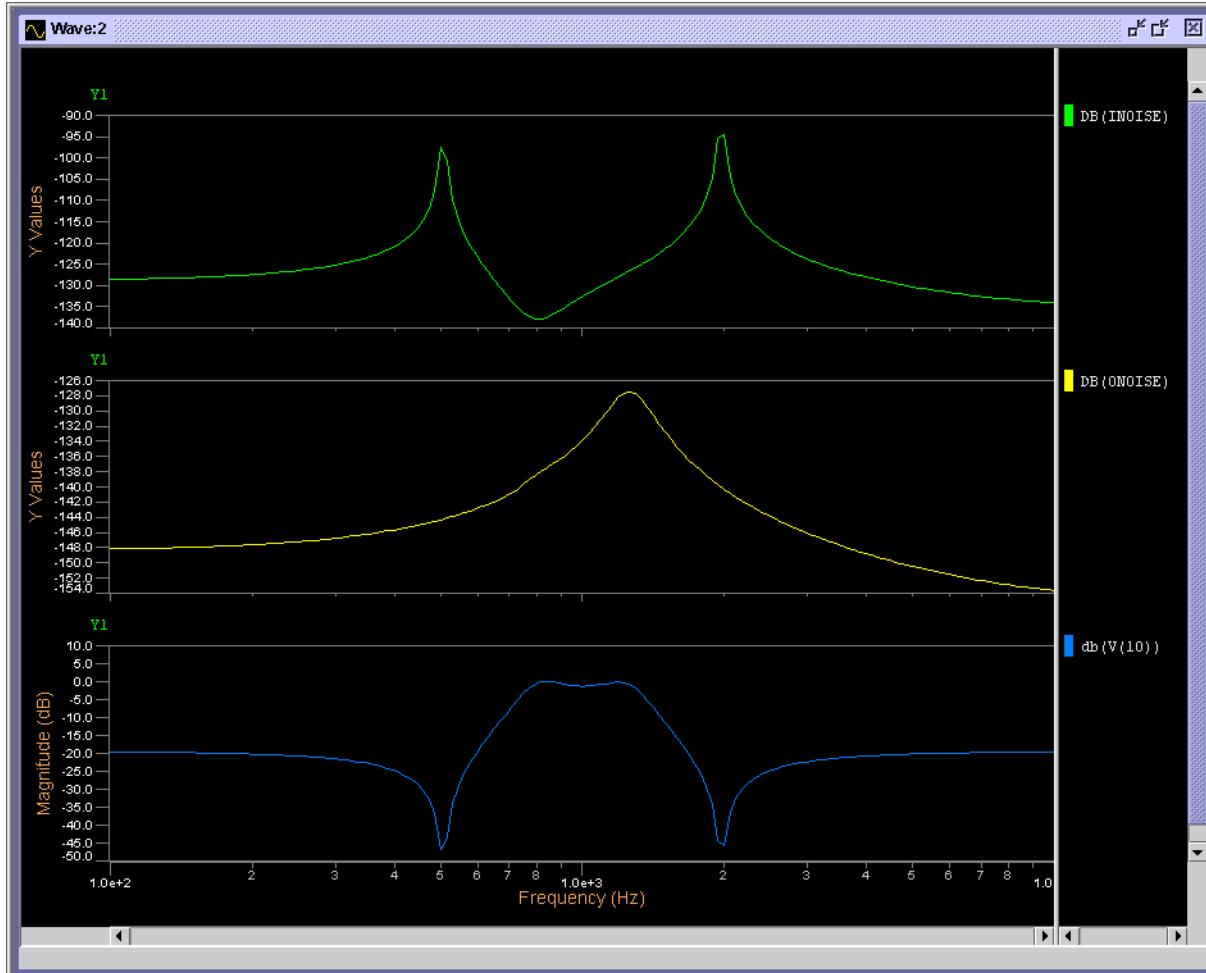
.plot noise db(inoise)
.plot noise db(onoise)
.plot ac vdb(10) (10,-50)

```

The above lines specify a dB/frequency plot to be performed on the input and output noise, together with a dB/frequency plot of the voltage at node 10, the output stage of the filter, between the limits 10 and -50dB.

## Simulation Results

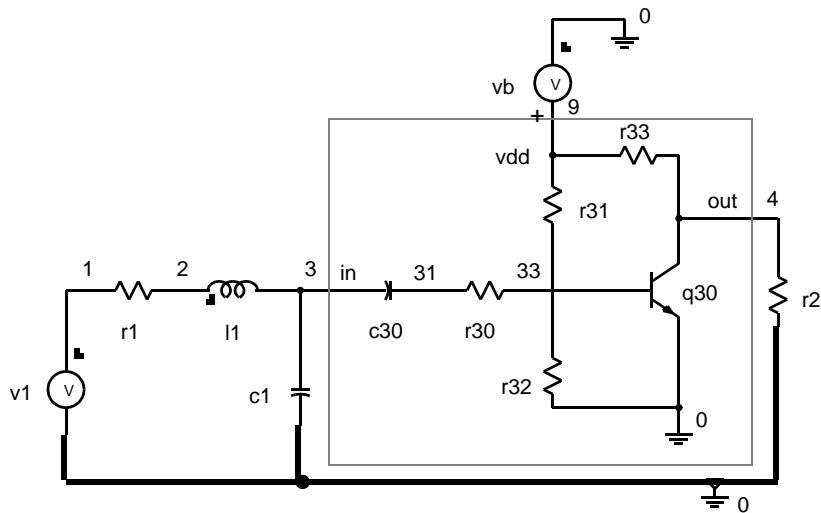
Figure 24-7. Tutorial #3—Simulation Results



## Tutorial #4—Low Pass Filter

This tutorial deals with the AC analysis of a low pass filter which incorporates a voltage amplifier, illustrating the use of Eldo's subcircuit capabilities, as the voltage amplifier part of the circuit itself is defined in this manner. The complete netlist can be found in the file *lowpass.cir*.

**Figure 24-8. Low Pass Filter**



Summary of Eldo Commands used in this Tutorial

- **.AC**—AC analysis
- **.MODEL**—Model definition
- **.PLOT**—Plot simulator results
- **.SUBCKT**—Subcircuit definition

### Complete Netlist

```

Low Pass Filter incorporating a Voltage Amplifier
* .MODEL definition
.model q2n2222 npn is=1.9e-14 bf=150 vaf=100 ikf=.175
+ ise=5e-11 ne=2.5 br=7.5 var=6.38 ikr=.012 isc=1.9e-13
+ nc=1.2 rc=.4 cje=26p tf=.5e-9 cjc=11p tr=30e-9 xtb=1.5
+ kf=3.2e-16 af=1.0
* Subcircuit definition
.subckt amp in out vdd
c30 in 31 47u
r30 31 33 390
r31 vdd 33 50k
r32 33 0 15k
q30 out 33 0 q2n2222
r33 vdd out 750
.ends amp

```

```

r1 1 2 10
l1 2 3 1.3m
c1 3 0 100n
r2 4 0 50k
x1 3 4 9 amp
vb 9 0 5
v1 1 0 ac 1
* Commands
.ac dec 10 1 1g
.plot ac vdb(3) vdb(4) (-250,50)
.plot ac vp(3) vp(4) (-200,200)
.end

```

## Netlist Explanation

Low Pass Filter incorporating a Voltage Amplifier

```

.model q2n2222 npn is=1.9e-14 bf=150 vaf=100 ikf=.175
+ ise=5e-11 ne=2.5 br=7.5 var=6.38 ikr=.012 isc=1.9e-13
+ nc=1.2 rc=.4 cje=26p tf=.5e-9 cjc=11p tr=30e-9 xtb=1.5
+ kf=3.2e-16 af=1.0
.subckt amp in out vdd

```

The above line indicates the start of the voltage amplifier subcircuit definition. The subcircuit is called `amp` and is connected between the nodes `in`, `out` and `vdd`.

```

c30 in 31 47u
r30 31 33 390
r31 vdd 33 50k
r32 33 0 15k
r33 vdd out 750
q30 out 33 0 q2n2222
.ends amp

```

The above line indicates the end of the definition of the subcircuit `amp`.

---

### Note

 All nodes used within the subcircuit are local nodes, in that they are only referenced within the subcircuit itself and that they do not have to correspond with the names of the nodes outside the subcircuit.

---

```

r1 1 2 10
r2 4 0 50k
l1 2 3 1.3m
c1 3 0 100n
x1 3 4 9 amp

```

The above line instantiates the subcircuit `x1` of type `amp` connected between the nodes `3`, `4`, and `9`. As explained earlier, these nodes correspond to the nodes `in`, `out` and `vdd` within the subcircuit.

```

vb 9 0 5
v1 1 0 ac 1

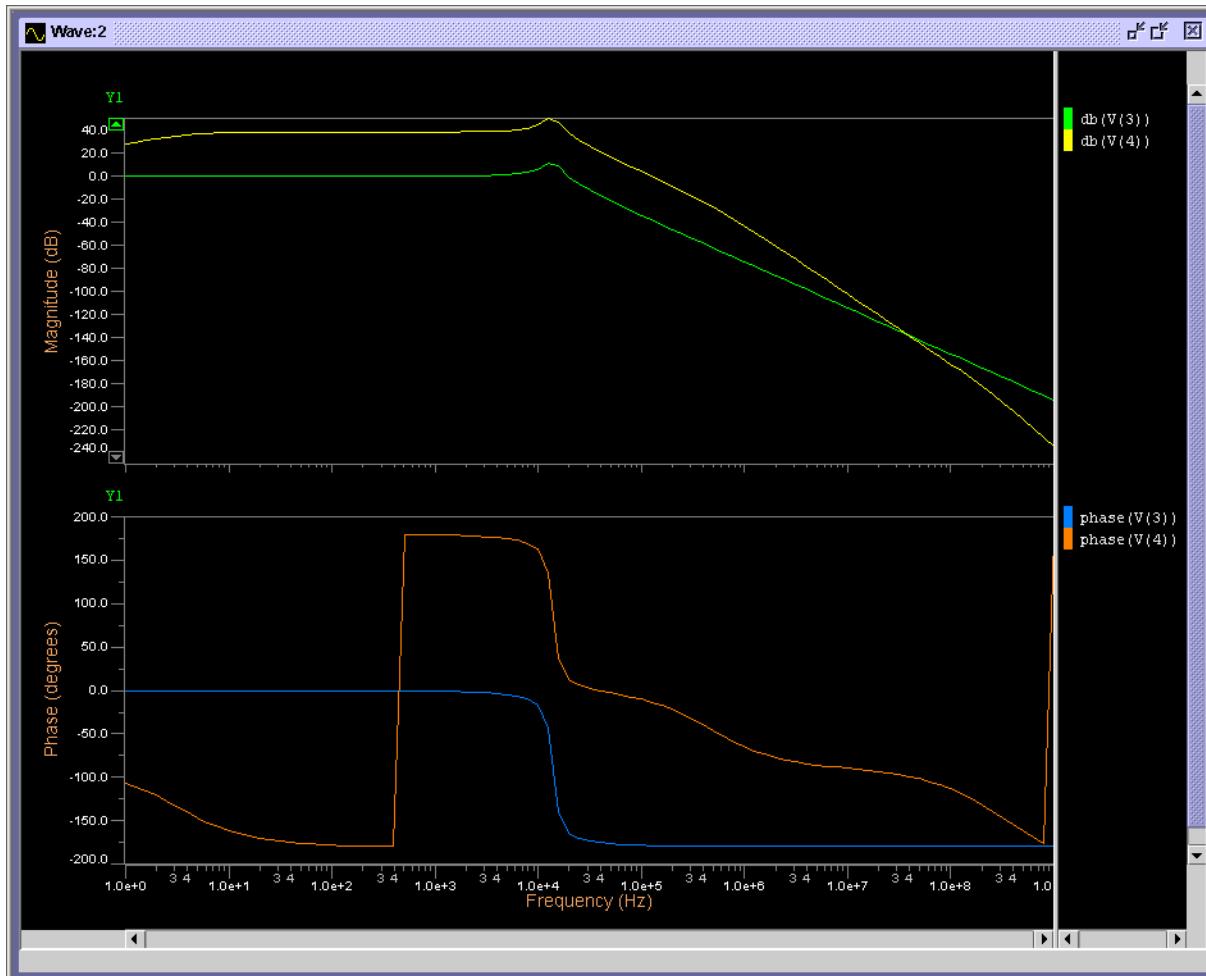
```

The above lines define the voltage sources in the circuit. An AC voltage source **v1** connected between the nodes 1 and 0 of value 1 V and a DC voltage source between the nodes 9 and 0 of value 5 V.

```
.ac dec 10 1 1g
.plot ac vdb(3) vdb(4) (-250,50)
.plot ac vp(3) vp(4) (-200,200)
.end
```

## Simulation Results

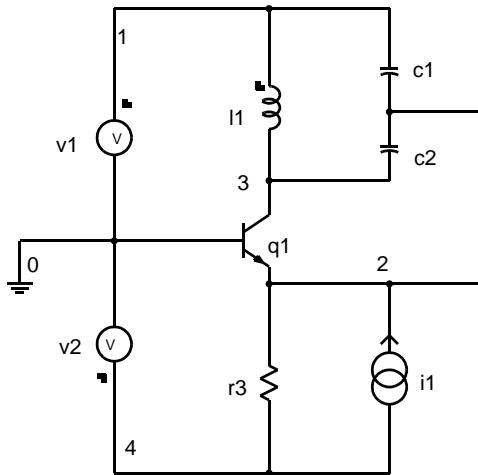
**Figure 24-9. Tutorial #4—Simulation Results**



## Tutorial #5—Colpitts Oscillator

This tutorial deals with the transient analysis of a simple oscillator circuit. It also illustrates making use of model library files found elsewhere in the system environment. The complete netlist can be found in the file *colpitts.cir*.

**Figure 24-10. Colpitts Oscillator**



Summary of Eldo Commands used in this Tutorial

- **.MODEL**—Model definition
- **.OPTION**—Simulator configuration
- **.PLOT**—Plot simulator results
- **.TRAN**—Transient analysis

### Complete Netlist

**.MODEL** definition in the library file: NBJT\_LIB

```
.model ts2 npn
+ bf=10 br=1 xtb=3 is=10f eg=1.11 rb=100
+ rc=10 vaf=50 tr=6n mjc=0.75 mje=0.33 vje=0.75
```

### Main Eldo Netlist

```
Colpitts Oscillator
11 1 3 5u
c1 1 2 2n
c2 2 3 100p
r3 2 4 2200
q1 3 0 2 ts2
v1 1 0 5
v2 4 0 -5
i1 2 4 pulse(0 10u 0 5n 5n 25n 50n)
```

```
.model lib nbjt_lib ts2
.option eps=1.0e-6
.tran 1u 12u
.plot tran v(1,3) (10,-10)
.end
```

## Netlist Explanation

**.MODEL** definition in the library file `NBJT_LIB`:

```
.model ts2 npn
+ bf=10 br=1 xtb=3 is=10f eg=1.11 rb=100
+ rc=10 vaf=50 tr=6n mjc=0.75 mje=0.33 vje=0.75
```

The above lines describe the electrical parameters of the `nPN` transistor model `ts2`.



For a detailed description of each of these parameters, refer to the [Device Models](#) chapter.

## Main Eldo Netlist

```
l1 1 3 5u
c1 1 2 2n
c2 2 3 100p
r3 2 4 2200
q1 3 0 2 ts2
```

The above line defines a transistor `q1` between nodes 3, 0 and 2 with electrical parameters defined by the model `ts2`.

```
v1 1 0 5
v2 4 0 -5
```

The above lines define the voltage sources in the circuit. A DC voltage source `v1` connected between the nodes 1 and 0 of value 5V and also a DC voltage source between the nodes 4 and 0 of value -5V.

```
i1 2 4 pulse(0 10u 0 5n 5n 25n 50n)
```

This line defines a time dependent pulse function between nodes 2 and 4 describing the following signal:

- 0 A at 0s (delay time is 0s)
- 0 A to 10 $\mu$ A in a rise time of 5ns
- 10 $\mu$ A from 5 to 30ns (pulse width is 25ns)
- 10 $\mu$ A to 0A in a fall time of 5ns
- 0 A at 35ns
- Cycle repeats starting from 50ns.

```
.MODEL LIB NBJT_LIB TS2
```

The above line indicates that the electrical parameters of the model TS2 are defined in the library file NBJT\_LIB as shown previously.

```
.option eps=1.0e-6  
.tran 1u 12u
```

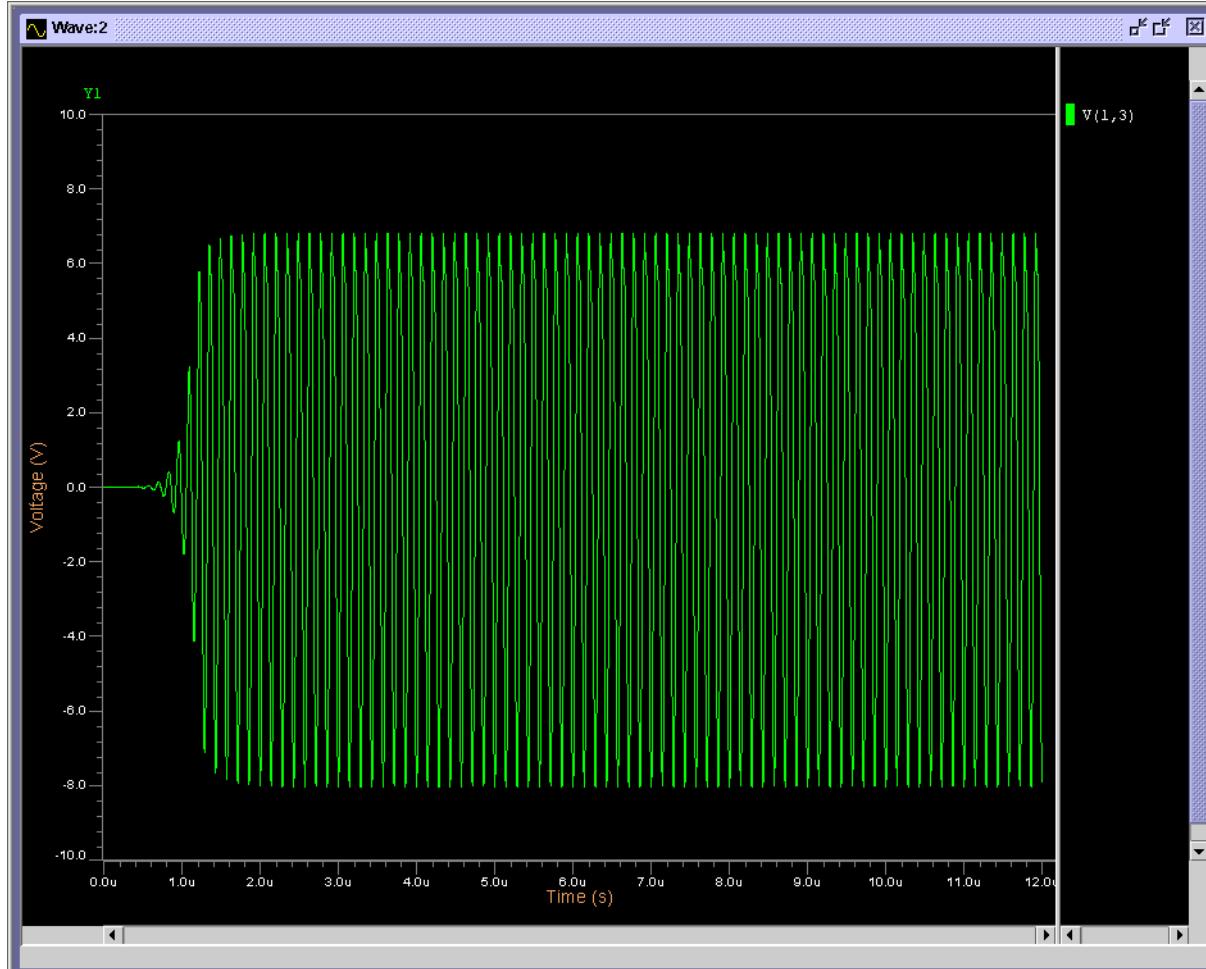
The above specifies a transient analysis is to be performed lasting 12 $\mu$ s with a plotting increment of 1 $\mu$ s.

```
.plot tran v(1,3) (10,-10)
```

The above line specifies a voltage/time plot to be performed of the voltage difference between the nodes 1 and 3 between the limits  $\pm 10V$ .

## Simulation Results

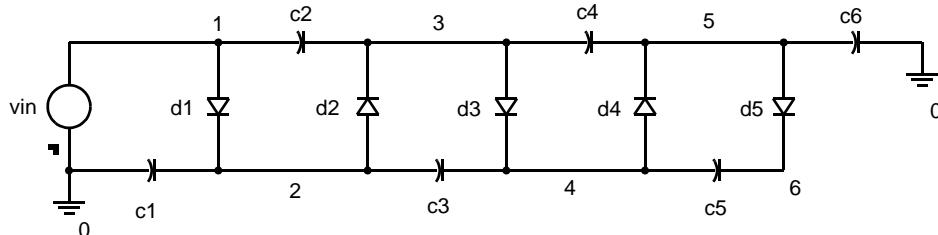
Figure 24-11. Tutorial #5—Simulation Results



## Tutorial #6—High Voltage Cascade

This tutorial deals with the transient analysis of a high voltage cascade circuit. The complete netlist can be found in the file *hv\_cascade.cir*.

**Figure 24-12. High Voltage Cascade**



Summary of Eldo Commands used in this Tutorial

- .MODEL—Model definition
- .OPTION—Simulator configuration
- .PLOT—Plot simulator results
- .TRAN—Transient analysis
- .PARAM—Global parameter setting

### Complete Netlist

```

high voltage cascade
.model dl1001 d rs=10 vj=0.8
d1 1 2 dl1001
d2 2 3 dl1001
d3 3 4 dl1001
d4 4 5 dl1001
d5 5 6 dl1001
c1 2 0 cap1
c2 1 3 cap1
c3 2 4 cap1
c4 3 5 cap1
c5 4 6 cap1
c6 5 0 cap2
vin 1 0 sin(0 2500 50k 0 0)
.param cap1=1n cap2=10n
.option reltol=0.01 vmax=100000 vmin=-100000
.tran 0.5m 5m
.plot tran v(5)
.plot tran v(1)
.end

```

## Netlist Explanation

```
high voltage cascade
.model dl1001 d rs=10 vj=0.8
```

The above line describes the electrical parameters of the diode model `dl1001`.



For a detailed description of each of these parameters, refer to the [Device Models](#) chapter.

```
d1 1 2 dl1001
d2 2 3 dl1001
d3 3 4 dl1001
d4 4 5 dl1001
d5 5 6 dl1001
c1 2 0 cap1
c2 1 3 cap1
c3 2 4 cap1
c4 3 5 cap1
c5 4 6 cap1
c6 5 0 cap2
vin 0 1 sin(0 2500 50k 0 0)
```

The above line defines a sinusoidal function between the nodes `1` and `0`, with a starting amplitude of 2500V and a frequency of 50kHz.

```
.param cap1=1n cap2=10n
```

The above line defines the global parameters `cap1` and `cap2` that are used in the capacitor and diode definitions to be 1nF and 10nF respectively.

```
.option reltol=0.01 vmax=100000 vmin=-100000
```

The above line sets the relative accuracy to a value of 0.01 and output voltage limits between 100,000 and -100,000V due to the high voltage levels present in this circuit.

```
.tran 0.5m 5m
```

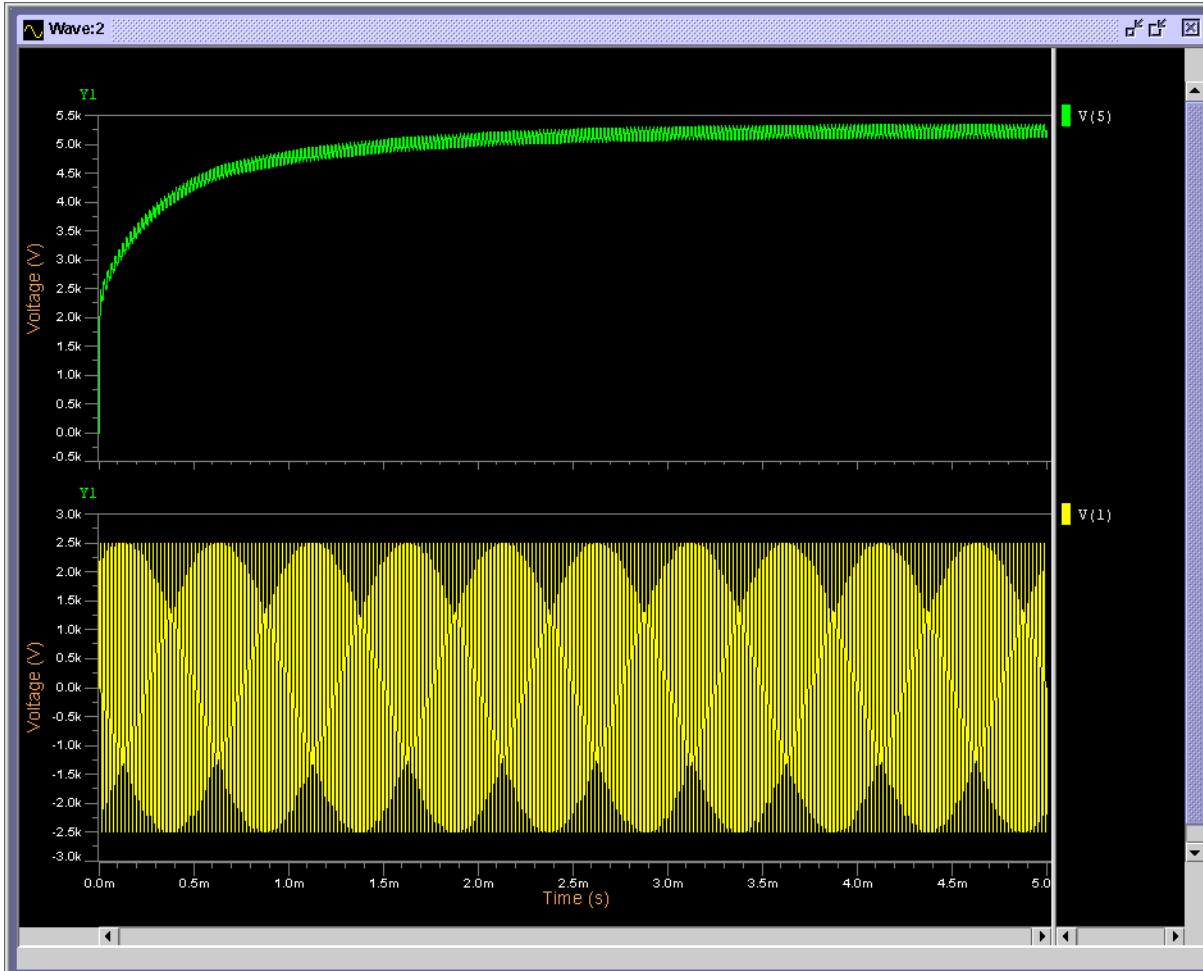
The above line specifies that a transient analysis should be performed on the circuit lasting 5ms with a plotting increment for the line printer of 0.5ms.

```
.plot tran v(5)
.plot tran v(1)
```

The above lines specify voltage/time plots to be performed on separate graphs of the voltages at nodes `5` and `1`.

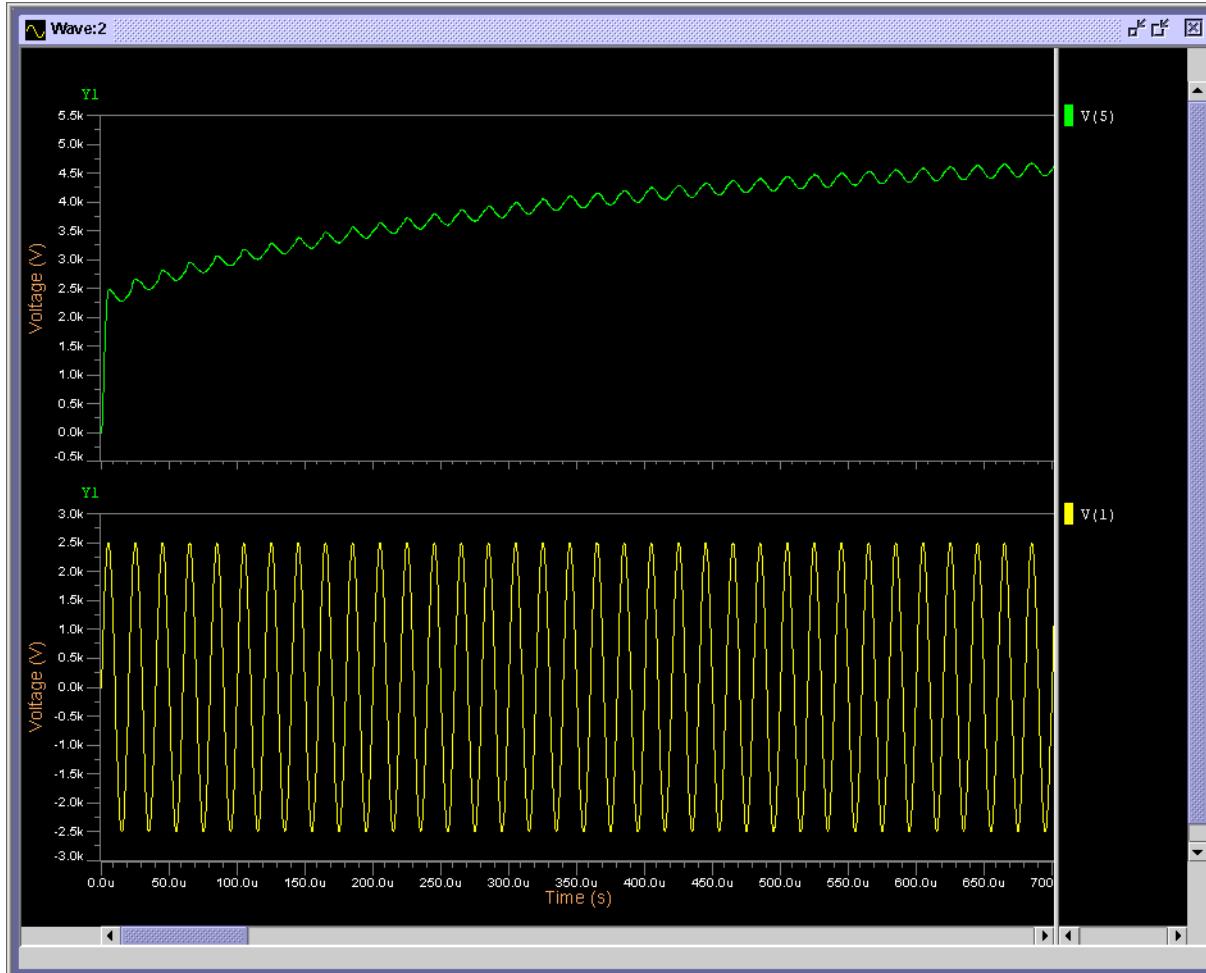
## Simulation Results—1

Figure 24-13. Tutorial #6—Simulation Results—1



## Simulation Results—2

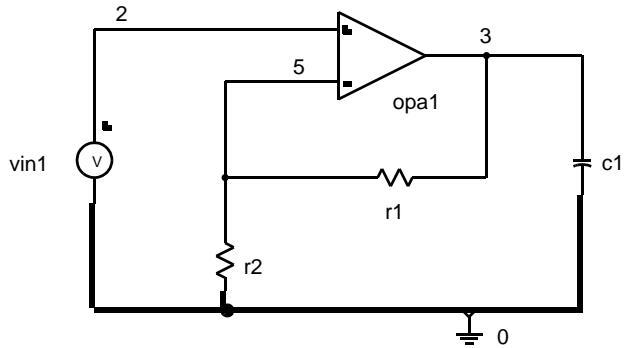
Figure 24-14. Tutorial #6—Simulation Results—2



## Tutorial #7—Non-inverting Amplifier

This tutorial deals with Monte Carlo analysis on a non-inverting amplifier circuit. A specified number of simulation runs are carried out to see the effect on the circuit of changing component values within a specified range during each simulation run. Upper and lower limits of the outputs are then displayed by Eldo at the end of the simulation. The complete netlist can be found in the file *noninvert\_amp.cir*.

**Figure 24-15. Non-inverting Amplifier**



Summary of Eldo Commands used in this Tutorial

- .AC—AC analysis
- .MC—Monte Carlo analysis
- .MODEL—Model definition
- .OPTION—Simulator configuration
- .PLOT—Plot simulator results

### Complete Netlist

```

Non-Inverting Amplifier
.model modres res lot=50% dev=60%
r1 5 3 modres 100k
r2 5 0 modres 1k
c1 3 0 1p
yopa opamp2 pin : 2 5 3 0 param: p1=1e3 p2=5e8 gain=5000
vin1 2 0 ac
.ac dec 30 1.0 100meg
.option eps=1.0e-6
.mc 7 vdb(3)
.print ac vdb(3)
.plot ac vdb(3)(-40,60)
.plot ac vp(3) (0,-90)
.end

```

## Netlist Explanation

```
Non-inverting amplifier
.model modres res lot=50% dev=60%
```

The above line specifies the Monte Carlo parameter limits for resistor model type `modres`. It indicates that for each Monte Carlo run the resistor values can change together by as much as 50% (`lot` tolerance). Additionally, the resistor values are allowed to change independently of each other by as much as 60% (`dev` tolerance). As can be seen below, these limits will have a more profound effect on the resistor `r2` than that of `r1`.

```
r1 5 3 modres 100k
r2 5 0 modres 1k
c1 3 0 1p
```

The above lines give the information of the resistors and capacitors that are present in the circuit to be simulated. Listed is the component name, the nodes between which the component is connected and the value of the component.

---

**Note**

---



The resistors are also defined to be of model type `modres`. This is present in order to specify Monte Carlo parameter limits for the resistors during the simulation runs.

---

```
yopa opamp2 pin: 2 5 3 0 param: p1=1e3 p2=5e8 gain=5000
vin1 2 0 ac
.ac dec 30 1.0 100meg
.option eps=1.0e-6
.mc 7 vdb(3)
```

The above line indicates that seven Monte Carlo simulation runs should be carried out on the voltage at node 3. The plotted output results contain nominal simulation results without any change in the circuit values, together with highest and lowest deviation results over the seven simulation runs.

```
.print ac vdb(3)
.plot ac vdb(3)(-40,60)
.plot ac vp(3) (0,-90)
.end
```



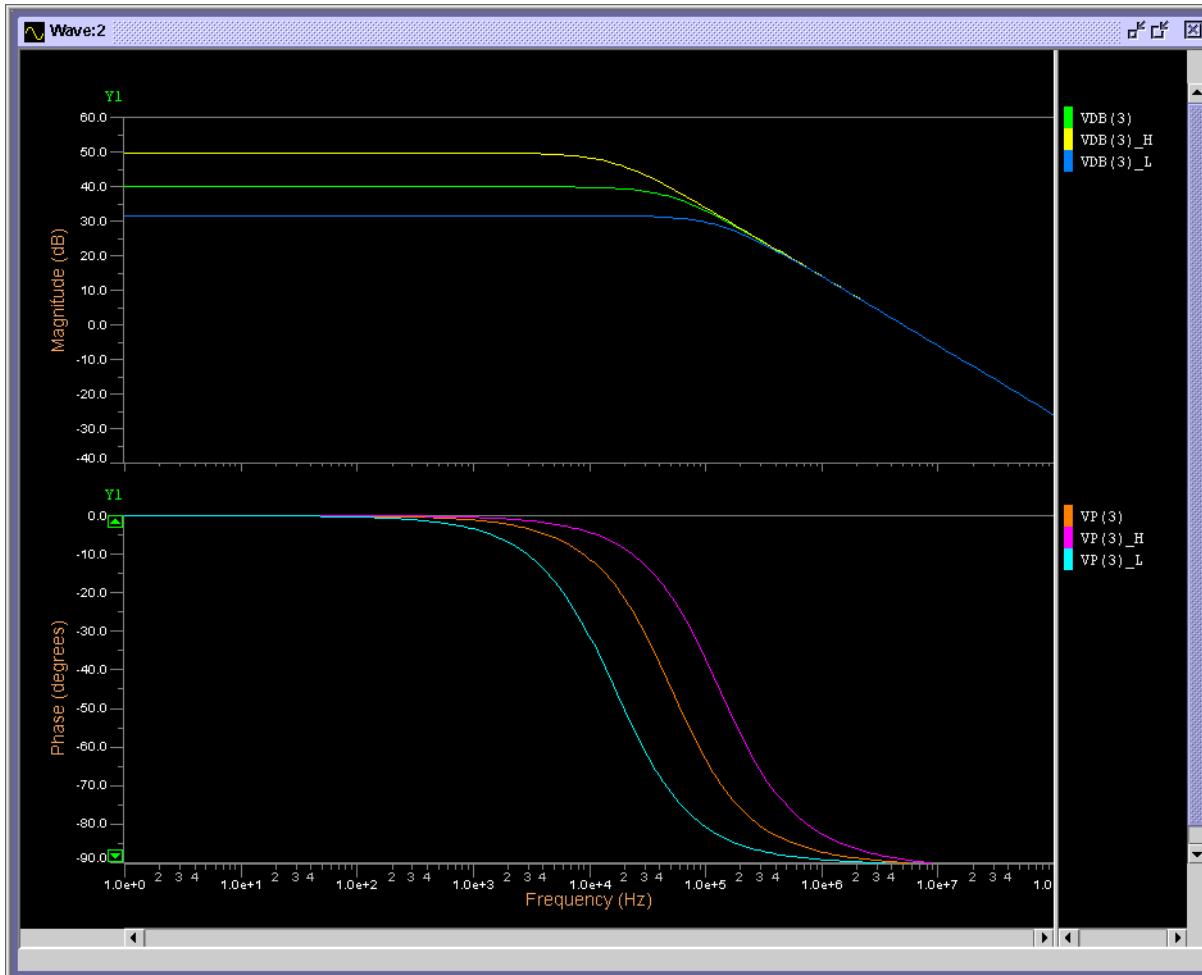

---

For more information on Monte Carlo analysis, refer to “[.MC](#)” on page 10-147.

---

## Simulation Results

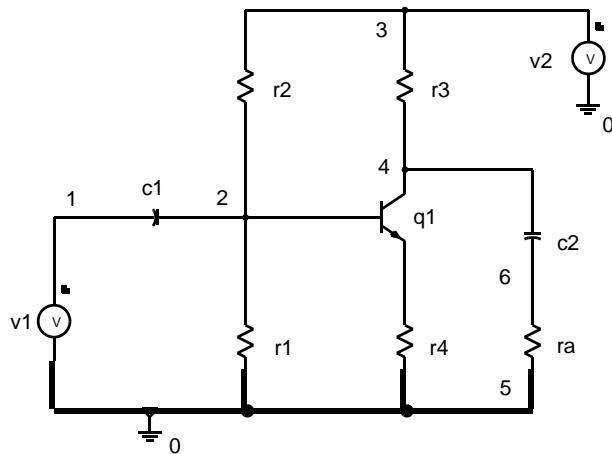
Figure 24-16. Tutorial #7—Simulation Results



## Tutorial #8—Bipolar Amplifier

This tutorial deals with a DC sensitivity analysis on the output of a bipolar amplifier circuit. The results show us which DC components have the most effect on the output of the circuit if they were to be changed. The complete netlist can be found in the file *bip\_amplifier.cir*.

**Figure 24-17. Bipolar Amplifier**



Summary of Eldo Commands used in this Tutorial

- .MODEL—Model definition
- .SENS—Sensitivity analysis

### Complete Netlist

```

bipolar amplifier
.model tun1 npn rb=524 irb=0.0 rbm=25 rc=150 re=1.0
+ is=121e-18 eg=1.206 xti=2 xtb=1.538 bf=137 ikf=6.9e-3
+ nf=1 vaf=159 ise=36e-16 ne=1.7 br=0.7 ikr=2.2e-3
+ nr=1 var=10.7 isc=0.0 nc=2 tf=0.6e-9 tr=54.e-9
+ cje=0.2e-12 vje=0.5 mje=0.24 cjc=1.8e-13 vjc=0.5
+ mjc=0.3 xcjc=0.3 cjs=1.3e-12 vjs=0.7 mjs=0.2 fc=0.9
+ itf=40.e-3 vtf=10 xtf=7
r1 2 0 6.8k
r2 3 2 100k
r3 3 4 1.8k
r4 5 0 100
ra 6 0 2.2k
c1 1 2 0.47u
c2 4 6 1u
q1 4 2 5 tun1
v1 1 0 0
v2 3 0 24
.sens v(4)
.end

```

## Netlist Explanation

```
bipolar amplifier
.model tunl npn rb=524 irb=0.0 rbm=25 rc=150 re=1.0
+ is=121e-18 eg=1.206 xti=2 xtb=1.538 bf=137 ikf=6.9e-3
+ nf=1 vaf=159 ise=36e-16 ne=1.7 br=0.7 ikr=2.2e-3
+ nr=1 var=10.7 isc=0.0 nc=2 tf=0.6e-9 tr=54e-9
+ cje=0.2e-12 vje=0.5 mje=0.24 cjc=1.8e-13 vjc=0.5
+ mjc=0.3 xcjc=0.3 cjs=1.3e-12 vjs=0.7 mjs=0.2 fc=0.9
+ itf=40.e-3 vtf=10 xtf=7
r1 2 0 6.8k
r2 3 2 100k
r3 3 4 1.8k
r4 5 0 100
ra 6 0 2.2k
c1 1 2 0.47u
c2 4 6 1u
q1 4 2 5 tunl
v1 1 0 0
v2 3 0 24
.sens v(4)
```

The above line indicates that a DC sensitivity analysis should be performed showing the relative sensitivities that the DC components have on the voltage on node 4, the output node of the amplifier. The results are listed in the *bip\_amplifier.chi* file.

```
.end
```

The results of the sensitivity analysis, found in the *bip\_amplifier.chi* file, are listed overleaf.

## Simulation Results

Figure 24-18. Tutorial #8—Simulation Results

| DC SENSITIVITIES OF OUTPUT V(4) |              |               |                                        |
|---------------------------------|--------------|---------------|----------------------------------------|
|                                 | ELEMENT NAME | ELEMENT VALUE | ELEMENT SENSITIVITY (VOLTS/UNIT)       |
|                                 |              |               | NORMALIZED SENSITIVITY (VOLTS/PERCENT) |
|                                 | R1           | 6.800E+03     | -1.39E-03                              |
|                                 | R2           | 1.000E+05     | 1.180E-04                              |
|                                 | R3           | 1.800E+03     | -3.90E-03                              |
|                                 | R4           | 1.000E+02     | 3.297E-02                              |
|                                 | RA           | 2.200E+03     | 0.000E+00                              |
|                                 | V1           | 0.000E+00     | 0.000E+00                              |
|                                 | V2           | 2.400E+01     | 4.585E-01                              |
| Q1                              | RB           | 3.489E+02     | 3.689E-04                              |
|                                 | RC           | 1.500E+02     | 9.155E-05                              |
|                                 | RE           | 1.000E+00     | 3.297E-02                              |
|                                 | BF           | 1.370E+02     | -1.82E-02                              |
|                                 | JLE/ISE      | 3.600E-15     | 6.409E+12                              |
|                                 | BR           | 7.000E-01     | 1.394E-11                              |
|                                 | JLC/ISC      | 0.000E+00     | 0.000E+00                              |
|                                 | JS/IS        | 1.210E-16     | -1.93E+15                              |
|                                 | NLE          | 1.700E+00     | -2.52E-01                              |
|                                 | NLC          | 2.000E+00     | 0.000E+00                              |
|                                 | JBF/IKF      | 6.900E-03     | -1.43E+02                              |
|                                 | JBR/IKR      | 2.200E-03     | 2.771E-11                              |
|                                 | VBF          | 1.590E+02     | 2.163E-03                              |
|                                 | VBR          | 1.070E+01     | -2.60E-02                              |

Referring to the above results, the *element sensitivity* is the change in the output of interest (in this case the voltage at node 4) due to a one unit change in the value of the element of interest (for example **r1**) and the *normalized sensitivity* is the change in the output of interest due to a one percent change in the value of the element of interest.

Looking at the sensitivity output in normalized sensitivity (volts/percent), it can be seen that the output at node 4 is most sensitive to the voltage source **v2** and also to the **bf** parameter in **q1**. As a result, variation in these values causes a significant effect on the output voltage.



Please refer to “**.SENS**” on page 10-278.

## Tutorial #9—SC Low Pass Filter

This example deals with the transient & small signal AC analysis of an SC low pass filter using the Z-domain switched capacitor models. The complete netlist can be found in the file *sc\_lowpass.cir*.

### Summary of Eldo Commands used in this Tutorial

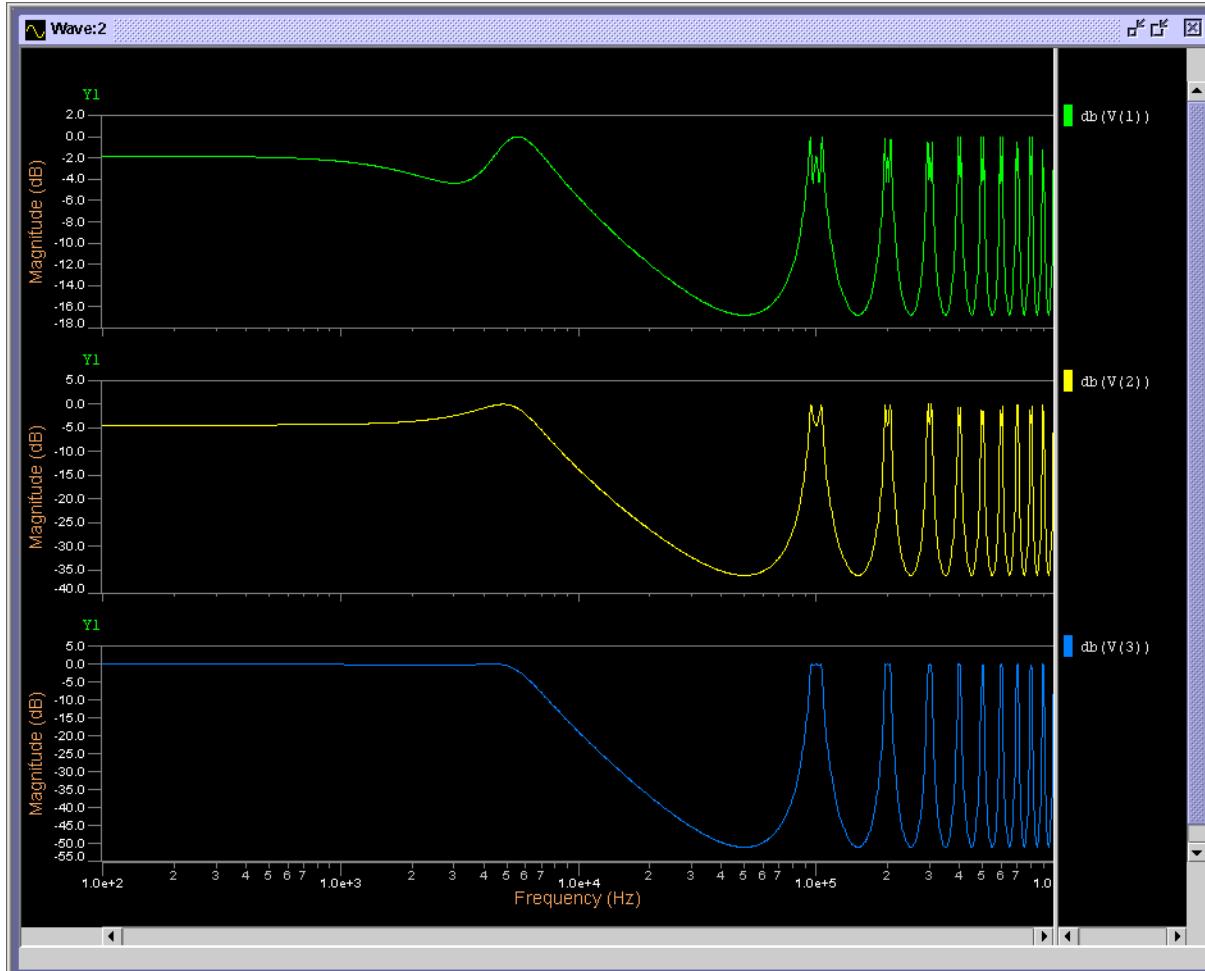
- **.AC**—AC analysis
- **PLOT**—Plot simulator results
- **.TRAN**—Transient analysis

### Complete Netlist

```
SC_lpfilt.cir
.param ts = 1.000000e-05
.subckt sc_int inp inm out
y1 sc_ideal inp inm out
y2 sc_u inm out param: c=c tp=tp
.ends sc_int
* INTEGRATOR 1
xi01 0 501 1 SC_INT c=5.173415p tp=ts
y002 sc_n pin: 2 0 501 0 param: c=1.347451p tp=ts
y003 sc_n pin: INPUT 0 501 0 param: c=1.623277p tp=ts
y004 sc_n pin: 1 0 501 0 param: c=1.000000p tp=ts
* INTEGRATOR 2
xi05 0 502 2 sc_int c=5.839726p tp=ts
y006 sc_i pin: 1 0 502 0 param: c=1.232076p tp=ts ldi=2
y007 sc_i pin: 3 0 502 0 param: c=1.000000p tp=ts ldi=2
* INTEGRATOR 3
xi08 0 503 3 sc_int c=4.117249p tp=ts
y009 sc_n pin: 2 0 503 0 param: c=1.660161p tp=ts
y010 sc_n pin: 3 0 503 0 param: c=1.000000p tp=ts
.tran lu 500u
.ac dec 500 100 1meg
.plot tran v(input)
.plot tran v(1)
.plot tran v(2)
.plot tran v(3)
.plot ac vdb(1)
.plot ac vdb(2)
.plot ac vdb(3)
vin input 0 dc 1.0 ac 1.0 pwl(0.0 0.0 10n 1.0)
.end
```

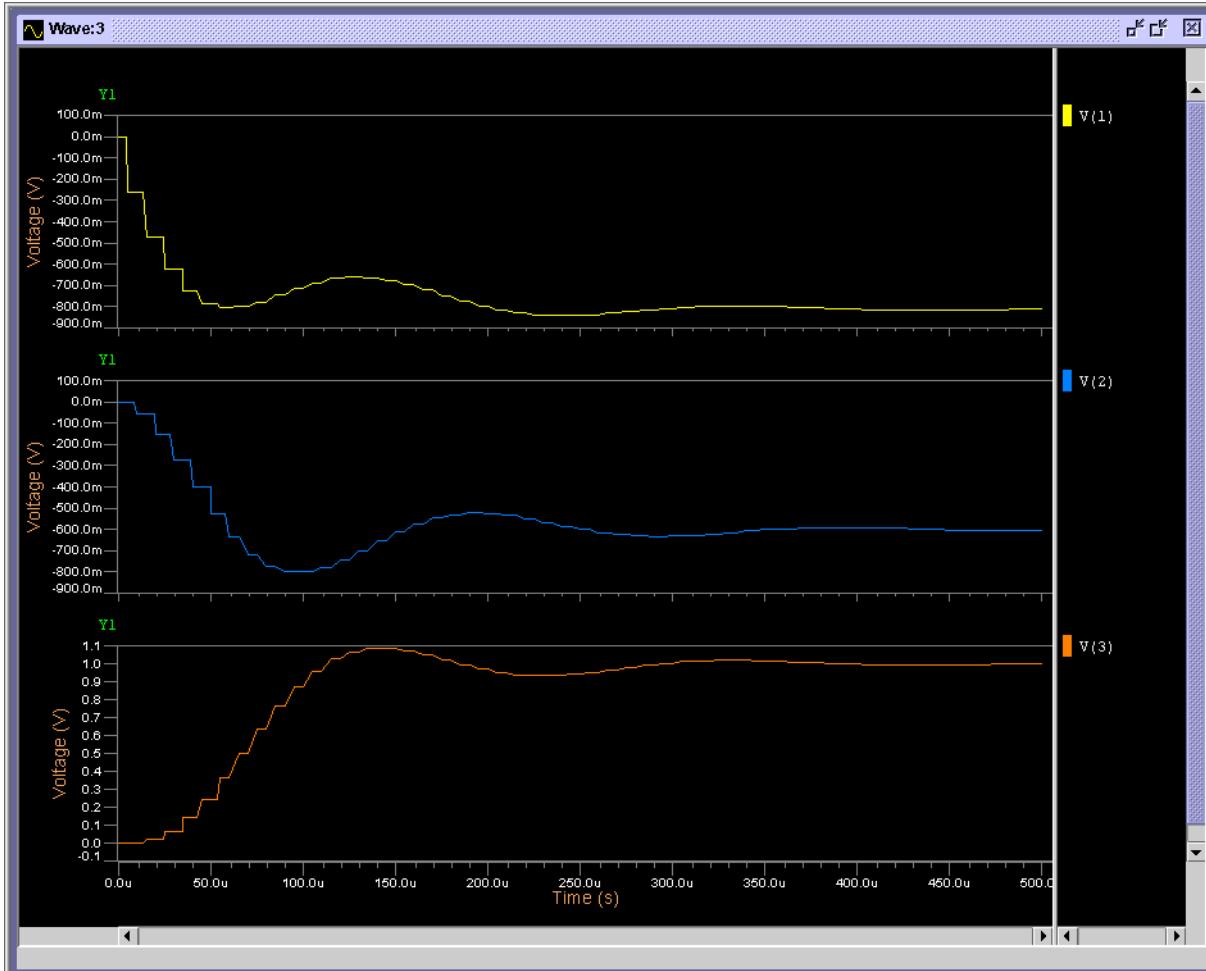
## Simulation Results—1

Figure 24-19. Tutorial #9—Simulation Results—1



## Simulation Results—2

Figure 24-20. Tutorial #9—Simulation Results—2





# Appendix A Error Messages

---

## Error Message Classification

When an error is detected during parsing of the simulation input file, a message is printed out specifying:

- The source line number.
- The text containing the error.

## Warnings

Warnings may be caused by improper use of commands or parameters which are then ignored by the analyzer. A warning does not prevent the continuation of analysis and simulation. By default, Eldo displays each type of node connection fault warnings (107, 108, 113 and 252) only three times; to override this default use the **MSGNODE** option.



For more information on this option, please refer to [page 11-42](#).

---

## Syntax Errors

Error messages usually result from commands or parameters which are not recognized by the analyzer. When a syntax error is detected the simulation does not continue. It is then necessary to edit *<circuit>.cir* and correct the syntax error. In the printout *<circuit>.chi*, the error location in the text is indicated by a “^” character, possibly accompanied by a message. The source line number and the source line may not be indicated if the syntax error is detected on a global basis without reference to a particular line.

## Effects

In the event of a warning, execution continues. If a syntax error is detected on the first analysis step (lexical and syntax analysis), circuit parsing continues until the end of this step, otherwise execution is aborted immediately. The numbers of errors and warnings found are displayed at the end of the first and second step.

**Note**

 After an error has been detected by the analyzer, subsequent error messages may be unfounded. It is therefore recommended to modify the source file to eliminate the first error before attempting to interpret the other messages.

---

## Error Messages

### Global Errors

**Table A-1. Global Errors**

| Error Number | Description                                                                                                                |
|--------------|----------------------------------------------------------------------------------------------------------------------------|
| Error 1      | Unable to open the file name                                                                                               |
| Error 2      | Unable to open the library file name                                                                                       |
| Error 5      | Unable to open the temporary file name                                                                                     |
| Error 6      | Nested #com statements are not allowed                                                                                     |
| Error 7      | Unable to delete the temporary file name                                                                                   |
| Error 9      | Internal error in AC analysis                                                                                              |
| Error 10     | Non invertible matrix                                                                                                      |
| Error 11     | Bad execution of name                                                                                                      |
| Error 12     | Check sum incorrect. Check your key                                                                                        |
| Error 13     | Error in reading key: (number). Check your key                                                                             |
| Error 14     | Error in reading ADC/DAC; file name not specified                                                                          |
| Error 15     | The environment variable USER is not set by the system. Set this variable in your .cshrc or .login script                  |
| Error 16     | Unable to allocate number bytes. Memory already allocated name                                                             |
| Error 17     | MEMALLOC: Unable to allocate number bytes. Memory already allocated name                                                   |
| Error 18     | MAMALLOC: Unable to allocate number bytes. Memory already allocated name                                                   |
| Error 19     | Unable to load file name. Analysis stopped                                                                                 |
| Error 20     | The number of plots of the current simulation does not match the number of plots of the previous one. File reading stopped |
| Error 21     | Storage capacity exceeded. Memory already allocated name, eldo_mem_used()                                                  |
| Error 22     | Circuit nesting error                                                                                                      |

**Table A-1. Global Errors**

| Error Number | Description                                                           |
|--------------|-----------------------------------------------------------------------|
| Error 23     | Unrecognized character or word                                        |
| Error 24     | Unexpected end of file                                                |
| Error 25     | Line not consistent with language syntax                              |
| Error 26     | No analysis specified                                                 |
| Error 27     | Voltage loop found                                                    |
| Error 28     | Unable to reallocate name                                             |
| Error 30     | Mismatch in model specification for device name                       |
| Error 31     | DATA or SWEEP specification: name                                     |
| Error 32     | Switch option cannot be used on device name                           |
| Error 34     | .OPTIMIZE MOD: no parameter specified                                 |
| Error 36     | .OPTION LVLTIM must be 1, 2, 3, or 4                                  |
| Error 38     | Compilation errors in file name                                       |
| Error 39     | Not enough memory to handle waves created through the DEFWAVE command |
| Error 40     | FML.exe not found                                                     |
| Error 41     | name: Real value expected                                             |
| Error 42     | name: Non-real expression expected                                    |
| Error 43     | Only DC followed by TRANSIENT analysis is allowed.                    |
| Error 44     | FasC model name already defined                                       |
| Error 45     | .OPTION DVDT must be -1 or 0.                                         |
| Error 46     | No plot to display for name analysis: Simulation stopped.             |
| Error 47     | Error in updating the library. Refer to file name                     |
| Error 48     | No voltage or current AC input source specified                       |
| Error 49     | Unable to spawn name: There might not be enough memory                |
| Error 50     | .OPTION NEWTON and OSR are not compatible                             |
| Error 51     | No plot matching analysis type: Analysis stopped.                     |
| Error 52     | No node "0" found in the circuit                                      |
| Error 54     | Syntax error parsing name                                             |
| Error 55     | The following line is too long                                        |
| Error 56     | No key to run name                                                    |

**Table A-1. Global Errors**

| Error Number | Description                                                                                                  |
|--------------|--------------------------------------------------------------------------------------------------------------|
| Error 57     | Nested DC Sweeps are not allowed within SimPilot                                                             |
| Error 58     | Error: Command name not allowed within SimPilot                                                              |
| Error 59     | No value given for parameter name                                                                            |
| Error 60     | Real value expected for name: End of line found                                                              |
| Error 61     | .OPTION IEM and OSR are not compatible                                                                       |
| Error 62     | .OPTION IEM and NEWTON are not compatible                                                                    |
| Error 63     | Unable to include file name                                                                                  |
| Error 64     | .MODDUP cannot be used in conjunction with LOT DEV                                                           |
| Error 65     | .MODDUP cannot be used in conjunction with .MC                                                               |
| Error 66     | .OPTION HMIN: Value must be strictly positive                                                                |
| Error 67     | .OPTION LVLCNV must be 0, 2, or 3                                                                            |
| Error 68     | COMMAND .INCLUDE/.LIB cannot find name                                                                       |
| Error 69     | Cannot find HDLA models                                                                                      |
| Error 70     | .OPTION name expects an integer value                                                                        |
| Error 71     | TUNING: Unexpected parameter name                                                                            |
| Error 72     | Probable syntax error in library name                                                                        |
| Error 73     | Fatal error in name model, please check output file for details. Eldo Kernel can't find the 'SBVAL.PAR' file |
| Error 74     | Cannot find #endcom statement                                                                                |
| Error 75     | ADVance MS command must be used in place of eldo in order to use VHDL AMS models                             |
| Error 76     | Internal error: Mismatch in parameter name                                                                   |
| Error 77     | Line not allowed inside command file                                                                         |
| Error 78     | Unknown command name                                                                                         |
| Error 79     | DEBUG command number name not found                                                                          |
| Error 80     | .RUN: Unknown argument name                                                                                  |
| Error 81     | Too many files opened                                                                                        |
| Error 82     | Parameter not known: name                                                                                    |
| Error 83     | Unable to alter name                                                                                         |
| Error 84     | Missing parameters                                                                                           |

**Table A-1. Global Errors**

| Error Number | Description                                                                                |
|--------------|--------------------------------------------------------------------------------------------|
| Error 85     | Syntax error or<br>Syntax error at or near name                                            |
| Error 86     | Error evaluating name                                                                      |
| Error 87     | Command cannot be issued at run time, or<br>Command name cannot be issued at run time      |
| Error 88     | name cannot change .MODEL card for that kind of element                                    |
| Error 89     | name disabled because AC analysis performed within transient analysis                      |
| Error 90     | SST analysis cannot be performed: Device name is not supported                             |
| Error 91     | SST analysis cannot be performed: Gudm model on name not supported                         |
| Error 92     | SST analysis cannot be performed: MOS model on name should be a charge-controlled<br>model |
| Error 93     | SST analysis cannot be performed: Non-Quasi Static effects on name is not supported        |
| Error 94     | Unknown directive: name                                                                    |
| Error 95     | No matching keyword name                                                                   |
| Error 96     | Parameter name cannot be evaluated                                                         |
| Error 97     | Mismatch between .iic and .cir files: Simulation stopped                                   |
| Error 98     | Having both .STEP and SWEEP on AC/TRAN/DC cards is not allowed                             |
| Error 99     | Unable to find a .DATA statement named name                                                |
| Error 100    | This version of Eldo does not include name                                                 |

## Errors Related to Nodes

The following error messages are preceded by NODE <name>:

**Table A-2. Errors Related to Nodes**

| Error Number | Description                                                                                                           |
|--------------|-----------------------------------------------------------------------------------------------------------------------|
| Error 101    | No source on this node                                                                                                |
| Error 102    | Multiple input signal applied                                                                                         |
| Error 103    | The DIGITAL to ANALOG Voltage Source Converter conflicts with another<br>Voltage Source already attached to this node |

**Table A-2. Errors Related to Nodes**

| Error Number | Description                                                                                                   |
|--------------|---------------------------------------------------------------------------------------------------------------|
| Error 104    | Multiple DTOA on this node and at least one of them is a Voltage Source Converter. This is not allowed        |
| Error 105    | Inconsistencies in the High Voltage specifications of the different RC DTOA Converters connected to this node |
| Error 106    | Inconsistencies in the Low Voltage specifications of the different RC DTOA Converters connected to this node  |
| Error 107    | Node name not known                                                                                           |
| Error 108    | Iout plot specification ignored on top-level nodes                                                            |
| Error 109    | This node is a floating gate                                                                                  |
| Error 111    | Less than two connections                                                                                     |
| Error 112    | No DC path to ground                                                                                          |
| Error 113    | A2D/D2A cannot be applied on non-existing nodes                                                               |
| Error 114    | Connectivity around that node makes Matrix singular                                                           |
| Error 115    | Related device not found                                                                                      |
| Error 116    | Only IN port seen, and no signal applied                                                                      |
| Error 117    | Missing A2D/D2A to connect to object                                                                          |

## Errors Related to Objects

The following error messages are preceded by OBJECT <name>:

**Table A-3. Errors Related to Objects**

| Error Number | Description                                                     |
|--------------|-----------------------------------------------------------------|
| Error 201    | Cannot be used in AC analysis, or<br>Cannot be used with CAPTAB |
| Error 202    | AC input has not been found                                     |
| Error 203    | Parameter sweep not available for this object                   |
| Error 204    | VTH1 and VTH2 are inverted                                      |
| Error 205    | Not found in file: name                                         |
| Error 206    | VHI and VLO are equal                                           |
| Error 207    | VHI and VLO are inverted                                        |
| Error 208    | Unrecognized character or word: name                            |

**Table A-3. Errors Related to Objects**

| Error Number | Description                                            |
|--------------|--------------------------------------------------------|
| Error 209    | Incorrect declaration of POLY names                    |
| Error 210    | Keyword FREQ expected                                  |
| Error 211    | Parameters are missing                                 |
| Error 212    | Unknown signal type: name                              |
| Error 213    | Incorrect geometrical dimension: name                  |
| Error 214    | No value specified for object: name                    |
| Error 215    | Macro already defined                                  |
| Error 216    | Too many controlling nodes (> 3)                       |
| Error 217    | Model not yet defined: model_name                      |
| Error 218    | Model not found: model_name                            |
| Error 219    | Short circuit element                                  |
| Error 220    | Voltage specifications are missing                     |
| Error 221    | Is a multidimensional element                          |
| Error 222    | SIN: Frequency below zero                              |
| Error 223    | SFFM: FC below zero                                    |
| Error 224    | SFFM: FS below zero                                    |
| Error 225    | EXP: TAU1 below zero                                   |
| Error 226    | EXP: TAU2 below zero                                   |
| Error 227    | Unknown signal applied                                 |
| Error 228    | Either TD or F must be specified                       |
| Error 229    | Multidimensional object. This variable is not declared |
| Error 230    | Output pin is already a voltage source                 |
| Error 231    | Output pin is already connected to a comparator output |
| Error 232    | Geometries are below zero: please check DW and DL      |
| Error 233    | Value becomes zero. Exited                             |
| Error 234    | Syntax error in the expression                         |
| Error 235    | Brackets are missing in the TABLE values               |
| Error 236    | Table input values must be properly ordered            |
| Error 238    | Bus specification ignored                              |

**Table A-3. Errors Related to Objects**

| <b>Error Number</b> | <b>Description</b>                                                         |
|---------------------|----------------------------------------------------------------------------|
| Error 240           | A time or a value specification is missing for this bus                    |
| Error 242           | This bus is already defined                                                |
| Error 244           | A signal has been already applied on this bus                              |
| Error 245           | More than one .CHECKBUS applied on this bus                                |
| Error 246           | No model specified in this functional call: name                           |
| Error 248           | Unable to plot the capacitances of this device. Only charges are available |
| Error 250           | Unable to plot the charges of this device. Only capacitances are available |
| Error 251           | Unexpected character found                                                 |
| Error 252           | Parameters or pins are missing for this device                             |
| Error 253           | Is not accessible                                                          |
| Error 254           | Unknown parameter: name                                                    |
| Error 255           | Unknown keyword: name                                                      |
| Error 256           | The sign : is missing after keyword                                        |
| Error 257           | '(' found when not expected                                                |
| Error 258           | Element not found: name                                                    |
| Error 259           | The following element is not a current source: name                        |
| Error 260           | The following element is not a voltage source: name                        |
| Error 261           | Pin not found: name                                                        |
| Error 262           | Number of pins: number                                                     |
| Error 263           | Number of controlling nodes: number                                        |
| Error 264           | Number of controlling zero voltage sources: number                         |
| Error 265           | Number of controlled voltage sources: number                               |
| Error 266           | Number of controlled current sources: number                               |
| Error 267           | Number of parameters: number                                               |
| Error 268           | Error when initializing SOLVE routines                                     |
| Error 269           | Error when initializing INTEGRAL routines                                  |
| Error 270           | Parameter not found                                                        |
| Error 271           | Duplicate definition of parameter: name                                    |
| Error 272           | Parameter already assigned to an equivalent: name                          |

**Table A-3. Errors Related to Objects**

| Error Number | Description                                                                          |
|--------------|--------------------------------------------------------------------------------------|
| Error 273    | Parameter index not found: name                                                      |
| Error 274    | Error in macro name                                                                  |
| Error 275    | VSTATUS already called with different pin order on: name                             |
| Error 276    | Unable to apply the function: name                                                   |
| Error 277    | Unable to find previous state: name                                                  |
| Error 278    | Error in function                                                                    |
| Error 279    | No model assigned to this element                                                    |
| Error 280    | Model attached to this device is also attached to another device of a different type |
| Error 281    | Its model is also attached to the following component: name                          |
| Error 282    | Access resistor becomes less than or equal to zero                                   |
| Error 283    | Error when computing R value. Division by 0                                          |
| Error 284    | Keyword PARAM expected instead of: name                                              |
| Error 285    | Equal sign '=' expected instead of: name                                             |
| Error 286    | Unknown Base specification                                                           |
| Error 287    | Cannot get the current through non-voltage source                                    |
| Error 288    | Unable to parse expression of                                                        |
| Error 289    | Radiation source not found                                                           |
| Error 290    | SOI back source not found                                                            |
| Error 291    | Radiation source specification error                                                 |
| Error 292    | SOI back source specification error                                                  |
| Error 293    | The current through cannot be used as argument                                       |
| Error 294    | Inconsistency in voltage specification                                               |
| Error 295    | Unexpected digit found in the bus: name                                              |
| Error 296    | Unable to code                                                                       |
| Error 297    | Unknown pin number                                                                   |
| Error 298    | R value is missing                                                                   |
| Error 299    | Mismatch in POLY specification                                                       |
| Error 801    | Parameterizable object not created                                                   |

**Table A-3. Errors Related to Objects**

| Error Number | Description                                                                                                 |
|--------------|-------------------------------------------------------------------------------------------------------------|
| Error 802    | This RC line refers to a model which is already attached to a standard resistance: name                     |
| Error 803    | This resistance refers to a model which is already attached to an RC line: name                             |
| Error 804    | Probably an RC line: please, specify level = 3 in the relevant .MODEL card                                  |
| Error 805    | Missing model specifications                                                                                |
| Error 806    | Delay cannot be 0.0                                                                                         |
| Error 807    | .PLOTLOG: Specified object is not 0xx instance                                                              |
| Error 808    | .PLOTLOG: Functional instance not defined                                                                   |
| Error 809    | BSIM1: Parameter K1 = K10 + K1L/Leff + K1W/Weff <= 0.0. Check your model card or the transistor dimensions. |
| Error 810    | .PBOUND: syntax is .PBOUND <pname> (mini,maxi)                                                              |
| Error 812    | Object not yet created                                                                                      |
| Error 813    | According to parameters specified, a model must be assigned to this device                                  |
| Error 814    | This device refers to an inconsistent model                                                                 |
| Error 815    | Inconsistency in thresholds/voltages declarations                                                           |
| Error 816    | The thresholds must be in the range [VLO + 1%, VHI - 1%]                                                    |
| Error 817    | The BS value for that core must be greater than the BS value                                                |
| Error 818    | The HC value for that core must be greater than zero                                                        |
| Error 819    | The temperature-adjusted BS value for that core must be greater than zero                                   |
| Error 820    | The temperature-adjusted BR value for that core must be greater than zero                                   |
| Error 821    | The temperature-adjusted HC value for that core must be greater than zero                                   |
| Error 822    | The temperature-adjusted BS value for that core must be greater than the temperature-adjusted BR.           |
| Error 823    | AREA must be greater than 0                                                                                 |
| Error 824    | FAS syntax expected on this element: HDL-A syntax found                                                     |
| Error 825    | HDL-A syntax expected on this element: FAS syntax found                                                     |
| Error 826    | LENGTH and AREA must be strictly positive                                                                   |
| Error 827    | Mismatch in port specification                                                                              |
| Error 828    | Equal sign '=' is missing in parameter list                                                                 |
| Error 829    | Unexpected equal sign '=' in parameter list                                                                 |

**Table A-3. Errors Related to Objects**

| Error Number | Description                                                                                                                                               |
|--------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------|
| Error 830    | Several ports with that index                                                                                                                             |
| Error 831    | No port index specified                                                                                                                                   |
| Error 832    | Missing R value                                                                                                                                           |
| Error 833    | device_name COUPLING: expecting keyword I or V: found name                                                                                                |
| Error 834    | Syntax error in PATTERN command                                                                                                                           |
| Error 835    | Object Error                                                                                                                                              |
| Error 836    | Mismatch in coupling specification                                                                                                                        |
| Error 838    | Already defined                                                                                                                                           |
| Error 839    | No model assigned to that device (MODDUP)                                                                                                                 |
| Error 840    | Short circuit on dipole device                                                                                                                            |
| Error 841    | This element cannot be converted into its switch equivalent                                                                                               |
| Error 842    | Odd number of pins                                                                                                                                        |
| Error 843    | Declaration of that object must be put at the very end of the netlist, since it is connected to hierarchical nodes, otherwise the node may not exist yet. |
| Error 844    | Pin specification is missing                                                                                                                              |
| Error 855    | TDEV should be specified on that device                                                                                                                   |
| Error 856    | Syntax HDLA_NAME = ELDO_NAME expected                                                                                                                     |
| Error 857    | Multiple SST specifications                                                                                                                               |
| Error 858    | Wrong object inside thermal network: only R/V/C device type can be specified                                                                              |
| Error 859    | AREA/PERI and W/L cannot be given together                                                                                                                |
| Error 860    | Multiple types of input applied                                                                                                                           |
| Error 861    | Not declared                                                                                                                                              |
| Error 862    | Unable to mix FPORT syntax with RPORT syntax. FPORT must be replaced by a voltage current source with R/C/L specifications.                               |
| Error 863    | RPORT can be applied on voltage source only                                                                                                               |
| Error 864    | Missing IPORT specification                                                                                                                               |
| Error 865    | AREA is negative or 0.0                                                                                                                                   |
| Error 866    | First pin of that element is undefined                                                                                                                    |
| Error 867    | Second pin of that element is undefined                                                                                                                   |
| Error 868    | .TRAN and .FOUR specification is not allowed on the same device                                                                                           |

**Table A-3. Errors Related to Objects**

| Error Number | Description                                                        |
|--------------|--------------------------------------------------------------------|
| Error 869    | Level not specified for geometric model                            |
| Error 870    | Generic/Parameter field appears twice                              |
| Error 871    | Only 'VALUE', 'TABLE' or 'AC' arguments expected                   |
| Error 872    | Device not found                                                   |
| Error 873    | Missing L or W specification to enable appropriate model selection |
| Error 874    | No character string allowed in Param/Generic list                  |
| Error 875    | DDT or IDT operator allowed on 'E' or 'G' sources only             |
| Error 876    | L and W must be specified before M(<param>)                        |
| Error 877    | Character not allowed on pattern                                   |
| Error 878    | Value not specified                                                |
| Error 879    | PORTS specification appears more than once                         |
| Error 880    | PINS specification appears more than once                          |
| Error 881    | Only PINS specification allowed on VHDL-AMS instances              |
| Error 882    | GENERIC specification appears more than once                       |
| Error 883    | Dimension must be 1 exclusively                                    |
| Error 884    | Second controlling node must be same as the second pin             |
| Error 885    | Second controlling node must be same as the first pin              |
| Error 886    | Parameters of denominator for S/Z transform are all zero.          |
| Error 887    | More than one value specified for resistor value                   |
| Error 888    | TSAMPLE not specified for PATTERN .CHECKBUS                        |
| Error 889    | Undeclared inductor                                                |
| Error 890    | Expects syntax PWL(1) N1 N2 MODEL value                            |
| Error 891    | Syntax error in IBIS model call                                    |
| Error 892    | Unknown IBIS device                                                |
| Error 893    | Component name is missing inside IBIS model call                   |
| Error 894    | filename is missing inside IBIS model call                         |
| Error 895    | MODEL and PIN can't be both specified                              |
| Error 896    | Cannot replace source, discontinuity found                         |
| Error 897    | Expects an even number of parameters                               |

**Table A-3. Errors Related to Objects**

| Error Number | Description                          |
|--------------|--------------------------------------|
| Error 898    | PZ or FNS allowed on E elements only |
| Error 899    | Neither MODEL nor PIN specified      |
| Error 900    | FNS denominator cannot be 0          |

## Errors Related to Commands

The following error messages are preceded by `COMMAND`

**Table A-4. Errors Related to Commands**

| Error Number | Description                                                           |
|--------------|-----------------------------------------------------------------------|
| Error 301    | .MC: Monte Carlo analysis is not compatible with .ALTER               |
| Error 302    | .MC: Monte Carlo analysis is not compatible with .STEP                |
| Error 303    | .ALTER: This command is not compatible with .STEP                     |
| Error 304    | .MC: Unable to reallocate .MODEL field                                |
| Error 305    | .MC: Internal memory error                                            |
| Error 307    | .CHRSIM: A previous transient analysis is required for the circuit    |
| Error 308    | .CHRSIM: A previous DC sweep analysis is required for the circuit     |
| Error 309    | .CHRSIM: No output found in previous circuit to be connected to input |
| Error 310    | .SENS: Unable to create sensitivity matrix                            |
| Error 311    | .SENS: Circuit size too large (>name) for sensitivity analysis        |
| Error 312    | .MC: Type not found                                                   |
| Error 313    | .OPTFOUR: Unknown output format                                       |
| Error 314    | .INIT: Time values are missing                                        |
| Error 315    | .IC: Time values are missing                                          |
| Error 316    | .USE name: Specification unknown.                                     |
| Error 317    | .TRAN: Parameters are missing                                         |
| Error 318    | .USE name: Specification unknown                                      |
| Error 319    | .PLOT name: This kind of analysis is not supported                    |
| Error 320    | .LOOP: Object type not allowed: name                                  |
| Error 321    | .PARAM: Attempt to divide by zero                                     |

**Table A-4. Errors Related to Commands**

| Error Number | Description                                                                                                   |
|--------------|---------------------------------------------------------------------------------------------------------------|
| Error 322    | .DC: W or L must be written in place of name                                                                  |
| Error 324    | .OPTIMIZE: Name or element name not yet defined                                                               |
| Error 325    | .DC: Too many values specified                                                                                |
| Error 326    | .RESTART: The file used to restart has the same name as the current circuit.<br>Rename the previous .cou file |
| Error 327    | .OPTIMIZE: name not yet created                                                                               |
| Error 328    | .OPTIMIZE: Model name not yet created                                                                         |
| Error 329    | .OPTIMIZE name: Incompatible operation                                                                        |
| Error 330    | .DC: Unknown object name                                                                                      |
| Error 331    | .OPTIMIZE: AC output node name not yet defined                                                                |
| Error 332    | .IC: Error on name                                                                                            |
| Error 333    | .SOLVE: Error in expression                                                                                   |
| Error 334    | .SOLVE: Element name not yet defined.                                                                         |
| Error 335    | .OPTIMIZE: Parameter name not known                                                                           |
| Error 336    | .PLOT: Specification name not allowed                                                                         |
| Error 337    | .DC: Sweep object not specified                                                                               |
| Error 338    | .OPTIMIZE: Use ACOUT=<pin_name> to select output                                                              |
| Error 339    | .NOISETRAN: Error with parameter                                                                              |
| Error 340    | The Thermal Noise Level (number) is not compatible with values extracted from MOS model                       |
| Error 341    | .NOISE: Element name not found                                                                                |
| Error 342    | .SOLVE: Element name not found                                                                                |
| Error 344    | .MFTA: Parameter name not found                                                                               |
| Error 345    | .MFTA: Missing argument                                                                                       |
| Error 346    | .SAVE: File name is already specified in a previous .SAVE command                                             |
| Error 348    | .INIT cannot be applied on node name: Conflict of signals                                                     |
| Error 350    | .WCASE: Unknown analysis specification name                                                                   |
| Error 352    | Expression name can accept only items V(node) or I(object)                                                    |
| Error 354    | Expression name can accept only items V(node), I(object) or W(w)                                              |
| Error 359    | .PARAM: “=” is missing in expression: name                                                                    |

**Table A-4. Errors Related to Commands**

| Error Number | Description                                                                                                            |
|--------------|------------------------------------------------------------------------------------------------------------------------|
| Error 360    | .OPTIMIZE: Unexpected expression: name                                                                                 |
| Error 361    | .EXTRACT: Unexpected expression: name                                                                                  |
| Error 362    | .STEP PARAM: Parameter name not specified                                                                              |
| Error 363    | .DC: The parameter name is not specified or is not a primitive.<br>Unable to perform this analysis                     |
| Error 364    | .STEP: Parameter name not defined at the top level                                                                     |
| Error 365    | .DC PARAM: Parameter name not defined at the top level                                                                 |
| Error 366    | .ALTER: Appears as the title                                                                                           |
| Error 367    | name: Unknown command                                                                                                  |
| Error 368    | .OPTIMIZE: sign = < > expected instead of name                                                                         |
| Error 369    | .DEFWAVE: Error name =                                                                                                 |
| Error 370    | .OPTIMIZE: name not allowed on parameter                                                                               |
| Error 371    | .PLOTLOG command: '-' sign is missing                                                                                  |
| Error 372    | .PLOT command: Found name while expecting a ( or )                                                                     |
| Error 373    | .SETSOA: Opening bracket expected after name                                                                           |
| Error 374    | .SETSOA: Closing bracket expected after name                                                                           |
| Error 375    | .SETSOA: Error in the expression name                                                                                  |
| Error 376    | .SETSOA: Expected .SETSOA EXPR <expression> = (MIN,MAX)                                                                |
| Error 377    | .SETSOA: Expected a * or a value for name                                                                              |
| Error 378    | .SETSOA: Model name not yet created                                                                                    |
| Error 379    | .WC cannot be used with command .MC                                                                                    |
| Error 380    | .CONNECT: Node name not yet created name The X instance which creates the node must be placed before the .CONNECT card |
| Error 381    | .SIGBUS: No parameters allowed (name)                                                                                  |
| Error 384    | .DISTRIB name: Opening bracket expected                                                                                |
| Error 385    | .DISTRIB name: Closing bracket expected                                                                                |
| Error 386    | .DISTRIB name is already defined                                                                                       |
| Error 387    | name: Specified terms must be properly ordered                                                                         |
| Error 388    | .DISTRIB name: <probability> must be in the range [0 to 1]                                                             |
| Error 389    | Distribution name referenced but not defined                                                                           |

**Table A-4. Errors Related to Commands**

| Error Number | Description                                                             |
|--------------|-------------------------------------------------------------------------|
| Error 390    | Error in LOT/DEV specification for expression name                      |
| Error 391    | .PARAM: Double specification for name                                   |
| Error 392    | .TRAN: No parameter allowed: (name found)                               |
| Error 393    | .DISTRIB name: <deviation> must be within the range [-1 to 1]           |
| Error 394    | Unable to measure ISUB (name): multiple connections                     |
| Error 395    | .OPTION name ignored                                                    |
| Error 396    | .STEP: DEC, LIN or OCT expected: name found                             |
| Error 397    | .STEP: No parameter allowed: name found                                 |
| Error 398    | .STEP: DEC/OCT values must be positive                                  |
| Error 399    | name: Negative X value found for DEC scale                              |
| Error 400    | .LMIN or .WMIN: Parameters are missing                                  |
| Error 401    | .OPTNOISE: Real value expected after keyword name                       |
| Error 402    | .OPTNOISE: Keywords D V TD TV expected: name found                      |
| Error 403    | .OPTNOISE: SV: Value must be less than 1 but greater than 0: name found |
| Error 404    | name: The sign ',' is not allowed                                       |
| Error 405    | .SOLVE: Cannot accept functions such as YVAL, TPD, MIN, MAX and INTEG   |
| Error 406    | .EXTRACT name: Cannot find a .EXTRACT with this label                   |
| Error 407    | .OPTFOUR: Unknown parameter: name                                       |
| Error 408    | .EXTRACT name: Cannot find a .FOUR with the label name                  |
| Error 409    | .SNF: Unknown parameter name                                            |
| Error 410    | .SNF: Double input specification for name field                         |
| Error 411    | Multiple .SNF cards not allowed                                         |
| Error 412    | .SNF card: name specification expected                                  |
| Error 413    | .FOUR card: Syntax is [LABEL=] <four_label> <wave>                      |
| Error 414    | .EXTRACT name: RMS accepts transient waves or AC noise waves only       |
| Error 415    | .OPTWIND or .OPTPWL: Parameter name cannot be changed                   |
| Error 416    | .OPTWIND or .OPTPWL: Missing time-value for name                        |
| Error 417    | .OPTWIND or .OPTPWL: Decreasing time-value for name                     |
| Error 418    | .AC: Syntax error card: name                                            |

**Table A-4. Errors Related to Commands**

| Error Number | Description                                                                                          |
|--------------|------------------------------------------------------------------------------------------------------|
| Error 419    | .OPTWIND or .OPTWPL: Missing value specification at or near name                                     |
| Error 420    | .OPTWIND or .OPTWPL: name                                                                            |
| Error 421    | .IFT: Unexpected argument name                                                                       |
| Error 422    | .OPTNOISE: NOISE and NONOISE cannot both be specified                                                |
| Error 423    | .OPTNOISE: ALL=OFF and NONOISE cannot both be specified                                              |
| Error 424    | .OPTNOISE: ALL=ON and NOISE cannot be both specified                                                 |
| Error 425    | .SNF: Element name not found                                                                         |
| Error 426    | .SNF: Element name appears as both input and output                                                  |
| Error 427    | .OPTFOUR: Parameter FS must be specified                                                             |
| Error 428    | .PZ: This analysis might give unexpected results because several AC sources are found in the circuit |
| Error 429    | .ALTER: Not allowed in Accusim netlist                                                               |
| Error 430    | .NOISETRAN: Too many runs specified                                                                  |
| Error 431    | .OPTFOUR: Bad value for parameters TSTART, TSTOP, NBPT and FS                                        |
| Error 432    | .STEP PARAM ( ): Syntax error                                                                        |
| Error 433    | .STEP PARAM ( ): Only LIST allowed for multi step                                                    |
| Error 434    | .STEP PARAM ( ): The number of values does not match the number of parameters                        |
| Error 435    | .STEP: TEMP keyword cannot appear more than once                                                     |
| Error 436    | .STEP: Unknown name                                                                                  |
| Error 437    | .CHECKSOA: Unexpected parameter                                                                      |
| Error 438    | .SETSOA: Expecting D, E or M: name found                                                             |
| Error 439    | .STEP: Missing increment parameter                                                                   |
| Error 440    | .STEP: Missing boundary specification                                                                |
| Error 441    | .WCASE: Missing analysis specification                                                               |
| Error 442    | .AC LIST: Frequencies are not properly ordered                                                       |
| Error 443    | .DC: Only a real value is expected                                                                   |
| Error 444    | .DEFWAVE: Duplicate definition of name                                                               |
| Error 446    | .OPTFOUR: Unexpected expression: name                                                                |
| Error 447    | .DEFMAC: Recursive definition of name                                                                |

**Table A-4. Errors Related to Commands**

| Error Number | Description                                                                                                 |
|--------------|-------------------------------------------------------------------------------------------------------------|
| Error 448    | .OPTFOUR: The expression for name could not be evaluated                                                    |
| Error 449    | .OPTFOUR: TSTART > TSTOP                                                                                    |
| Error 450    | .STEP LIST: name                                                                                            |
| Error 451    | .NOISETRAN: Cannot be used with .OPTION SAMPLE                                                              |
| Error 452    | .MEAS: Unknown analysis type name                                                                           |
| Error 453    | .MEAS: Unknown function name                                                                                |
| Error 454    | .MEAS: Unexpected wave name                                                                                 |
| Error 455    | .MEAS: Unexpected keyword: name                                                                             |
| Error 456    | .MEAS: Missing keyword: name                                                                                |
| Error 457    | .MEAS name: Cannot find a .MEAS of that name                                                                |
| Error 458    | .MEAS name: LAST not accepted                                                                               |
| Error 459    | .DC: name unknown                                                                                           |
| Error 460    | .STEP: Increment is 0.0                                                                                     |
| Error 461    | .SETSOA: FILE=name unexpected                                                                               |
| Error 462    | .EXTRACT: FILE=name must be place before the expression                                                     |
| Error 463    | .STEP: multiple declaration for name                                                                        |
| Error 464    | .MCDEV: syntax E(devname[,geo]) expected                                                                    |
| Error 465    | .TRAN: decreasing time not allowed: name                                                                    |
| Error 466    | .LOTGROUP: double specification for name                                                                    |
| Error 467    | .FFT: Syntax is .FFT <wave> <FFT parameters>                                                                |
| Error 468    | .POTBUS: specification error on name                                                                        |
| Error 469    | .MODEL name HOOK: syntax is .MODEL <name> HOOK<br>[<logical_lib_name>:]<entity_name>[(<architecture_name>)] |
| Error 470    | .MC specification on name: dead-zone too large compared to 'normal' zone                                    |
| Error 471    | Directive error name                                                                                        |
| Error 472    | .PLOT: SMITH chart available for AC only                                                                    |
| Error 473    | .SETBUS: node name does not exist                                                                           |
| Error 474    | .AC expecting DEC, LIN, OCT, LIST or DATA: name found                                                       |
| Error 475    | .PART: Unknown flag: name                                                                                   |

**Table A-4. Errors Related to Commands**

| Error Number | Description                                                                                   |
|--------------|-----------------------------------------------------------------------------------------------|
| Error 476    | .PART: Syntax error. Syntax is .PART <flag> SUBCKT INST (<list>)                              |
| Error 477    | .SETBUS: node name does not exist                                                             |
| Error 478    | .EXTRACT: such SST output .H extensions                                                       |
| Error 479    | .HOOK: expects 'MOD=' statement                                                               |
| Error 480    | .LOOP: unexpected string name                                                                 |
| Error 481    | .DEFWAVE name: expects .H extensions                                                          |
| Error 482    | name: expects an integer value                                                                |
| Error 483    | name: missing a '(' bracket                                                                   |
| Error 485    | .STEP: name                                                                                   |
| Error 486    | Two ports are required for getting RNEQ/NFMIN_MAG/GOPT/<br>GAMMA_OPT/GAMMA_OPT_MAG/PHI_OPT/NC |
| Error 487    | .SNF: error specifying SIDEband: name                                                         |
| Error 488    | .PARAMOPT name 3 or 4 values expected                                                         |
| Error 489    | .PLOT: (SMITH) or (SPECTRAL) incompatible with (VERSUS)                                       |
| Error 490    | .PLOT: (VERSUS) must be followed by only one wave.                                            |
| Error 491    | .PLOT: No wave found before (VERSUS)                                                          |
| Error 492    | .TVINCLUDE: syntax error in test vector name                                                  |
| Error 493    | .DATA: name                                                                                   |
| Error 494    | .EXTRACT: name: LABEL = <> expected                                                           |
| Error 495    | .EXTRACT: name: expects MEAS(label_name)                                                      |
| Error 496    | .SNF: name                                                                                    |
| Error 497    | name not allowed with Mach                                                                    |
| Error 498    | .PRINT/PLOT/PROBE/EXTRACT: bad number of tones specified                                      |
| Error 499    | .CORREL: parameter name cannot be found                                                       |
| Error 500    | .CORREL: Instance name cannot be found                                                        |

## Errors Related to Models

The following error messages are preceded by MODEL:

**Table A-5. Errors Related to Models**

| Error Number | Description                                                                                 |
|--------------|---------------------------------------------------------------------------------------------|
| Error 501    | Not found in library                                                                        |
| Error 502    | Unknown in the libraries                                                                    |
| Error 503    | Undeclared model reference                                                                  |
| Error 504    | Is defined twice                                                                            |
| Error 505    | Parameter unknown                                                                           |
| Error 506    | Model not yet defined                                                                       |
| Error 507    | Inconsistency in level specification. LEVEL: number                                         |
| Error 508    | Level not implemented                                                                       |
| Error 509    | R model not declared                                                                        |
| Error 510    | NSUB < NI. exited                                                                           |
| Error 511    | C model not declared                                                                        |
| Error 512    | G model not declared                                                                        |
| Error 514    | BIDIR type not allowed. Use ATOD or DTOA                                                    |
| Error 515    | Default BIDIR Converter not found                                                           |
| Error 516    | LEVEL already specified                                                                     |
| Error 517    | Parameter TOX must be greater than 0                                                        |
| Error 518    | Parameter STRUCT must be +1 or -1                                                           |
| Error 519    | Unacceptable parameter value                                                                |
| Error 520    | MJSW must be less than 1.0                                                                  |
| Error 521    | N must be positive                                                                          |
| Error 522    | FC must be less than 1.0                                                                    |
| Error 523    | The following parameter must only be used when eldomos is active or level 12 or 13 are used |
| Error 524    | Model Error                                                                                 |
| Error 526    | Unable to alter the parameters of that model                                                |
| Error 527    | This model is used by Eldo-XL and cannot be replaced                                        |

**Table A-5. Errors Related to Models**

| Error Number | Description                                                                                        |
|--------------|----------------------------------------------------------------------------------------------------|
| Error 528    | This model cannot be changed using .LIB in an interactive mode: Inconsistent level or architecture |
| Error 529    | The following parameter cannot be used with model MM9                                              |
| Error 530    | Unknown language type                                                                              |
| Error 531    | Double MC specification (via .MCMOD/.MODEL) for the parameter name                                 |
| Error 532    | Expects PWL(1) SYMMETRY NO SOURCE CARDS DATA                                                       |
| Error 533    | Parameter COX cannot be specified for that model                                                   |
| Error 534    | W model: missing declaration of parameter N                                                        |
| Error 535    | W model: incorrect number of parameters                                                            |
| Error 536    | parameter name already assigned                                                                    |
| Error 537    | vector size not consistent for parameter name                                                      |
| Error 538    | Missing inout values for parameter name                                                            |
| Error 539    | Size of vectors not consistent                                                                     |
| Error 540    | Missing call to MP_vect_array                                                                      |
| Error 541    | inconsistent dimension of the vector array                                                         |
| Error 542    | Position already allocated                                                                         |
| Error 543    | PASSFAIL/BISECTION expected after METHOD parameter                                                 |
| Error 544    | DEVX specification cannot be applied on .MODEL parameters                                          |
| Error 545    | This level is not supported yet                                                                    |

## Errors Related to AMODELS

The following error messages are preceded by `AMODEL <name>`:

**Table A-6. Errors Related to AMODELS**

| Error Number | Description                                                                  |
|--------------|------------------------------------------------------------------------------|
| Error 601    | Attempt to redefine an Amodel while the previous definition is not completed |
| Error 602    | Definition not completed                                                     |
| Error 603    | Macromodel not found                                                         |
| Error 604    | Pins must be specified before Ports                                          |

## Errors Related to Subcircuits

The following error messages are preceded by `SUBCKT <name>`:

**Table A-7. Errors Related to Subcircuits**

| Error Number | Description                                                              |
|--------------|--------------------------------------------------------------------------|
| Error 701    | Not found in library                                                     |
| Error 702    | Undeclared subcircuit reference                                          |
| Error 703    | is defined twice                                                         |
| Error 704    | Cannot be declared before the previous one has finished                  |
| Error 705    | Incorrect number of nodes in call at line                                |
| Error 708    | Recursive .SUBCKT calls are not allowed                                  |
| Error 709    | Unable to connect the output to a power supply or another digital output |
| Error 710    | Error in declaration                                                     |
| Error 711    | Already obtained from another LIB file                                   |
| Error 712    | Cannot define a parameter named 'M'                                      |
| Error 713    | An 'IF' statement is not properly closed                                 |

## Miscellaneous Errors

**Table A-8. Miscellaneous Errors**

| Error Number | Description                           |
|--------------|---------------------------------------|
| Error 901    | number: Unknown signal type at line   |
| Error 902    | name: Unknown parameter               |
| Error 903    | name: Specification ignored           |
| Error 904    | name: Unknown model type              |
| Error 905    | name: Parameter not found             |
| Error 906    | name: Unknown output format specified |
| Error 907    | name: Can not affect value            |
| Error 908    | name: Parameter not yet set           |
| Error 909    | name: Unknown specification           |

**Table A-8. Miscellaneous Errors**

| Error Number | Description                                                   |
|--------------|---------------------------------------------------------------|
| Error 910    | name: Model not yet defined                                   |
| Error 911    | Diagram not supported at line                                 |
| Error 912    | name: Unknown command or component at line                    |
| Error 913    | Decreasing time on input signal number name                   |
| Error 914    | Decreasing time on input signal assigned to object            |
| Error 915    | CFAS: Call to get_param_value() allowed in ALLOCATE MODE only |
| Error 916    | CFAS: Call to get_param_value_b4() not allowed                |
| Error 917    | Unable to read from file name                                 |
| Error 918    | Cannot reallocate component table                             |
| Error 919    | Cannot reallocate input signal                                |
| Error 920    | Unable to parse expression name                               |
| Error 921    | name is not specified in ATOD model                           |
| Error 922    | Description of DTOA name is missing                           |
| Error 923    | DTOA name is declared as ATOD                                 |
| Error 924    | Description of ATOD name is missing                           |
| Error 925    | ATOD name is declared as DTOA                                 |
| Error 926    | Error in expression name. I or V expected                     |
| Error 928    | BIDIR name is declared as ATOD or DTOA                        |
| Error 929    | MODPAR: Parameter name not known                              |
| Error 930    | MODPAR: Parameter name is missing                             |
| Error 931    | EVAL: Parameter name is missing                               |
| Error 932    | EVAL: Parameter name not known                                |
| Error 933    | Error in TRISE or TFALL or TCROSS specification               |
| Error 934    | .NOISE output must be a node voltage (V(xx))                  |
| Error 935    | .PBOUND: parameter name not found                             |
| Error 936    | name Inconsistent FAS instance                                |
| Error 937    | .LIB: keyword 'KEY =' expected: name found                    |
| Error 938    | .MCMOD: expecting LOT or DEV keyword: name                    |
| Error 939    | name Unknown .PLOT/.PRINT specification                       |

**Table A-8. Miscellaneous Errors**

| Error Number | Description                                                                  |
|--------------|------------------------------------------------------------------------------|
| Error 940    | .CHECKSOA: Not compatible with name                                          |
| Error 941    | name cannot be redefined in a .PARAM statement                               |
| Error 942    | .CHRSIM name: Unknown argument                                               |
| Error 950    | Missing FPORT number name                                                    |
| Error 951    | .FFILE: expecting unit (Hz, KHz, MHz, GHz): name found                       |
| Error 952    | .FFILE: expecting format MA or RI: name found                                |
| Error 954    | .PZ: expecting V(pin) or I(Vxx)                                              |
| Error 955    | Missing state name to be displayed for device name                           |
| Error 956    | .TEMP: real value expected, name found                                       |
| Error 957    | .FFILE: expecting S/Y/Z: name found                                          |
| Error 958    | .ALTER inside .INCLUDE is not allowed                                        |
| Error 959    | .PLOT (LOW, HIGH) values must be real: name found                            |
| Error 960    | .SETSOA: Unexpected keyword 'ELSE' found                                     |
| Error 961    | .SETSOA: source line name: Error in nesting IF-ENDIF statement               |
| Error 962    | PORT name has only one connection                                            |
| Error 963    | name: Invalid expression for .EXTRACT DC                                     |
| Error 964    | Port specification in PLOT/PRINT is larger than the number of declared PORTS |
| Error 965    | Inconsistent number of pins for devices name                                 |
| Error 966    | M factor is less than or equal to zero for device name                       |
| Error 967    | .LIB: .LIB KEY=name already exists                                           |
| Error 968    | Instance already defined: name                                               |
| Error 969    | Unknown MC specification name                                                |
| Error 970    | Missing MC specification in name                                             |
| Error 971    | name: Parameter name cannot start with a digit                               |
| Error 972    | FNS order must be an integer: name found                                     |
| Error 973    | .LIB: Unable to find library or variant                                      |
| Error 974    | LOTGROUP name not defined                                                    |
| Error 975    | FOUR output: .H extension expected on the wave name                          |

**Table A-8. Miscellaneous Errors**

| Error Number | Description                                                                                                                  |
|--------------|------------------------------------------------------------------------------------------------------------------------------|
| Error 976    | Too many simulation are required ( > ELDO_LONGVAL)                                                                           |
| Error 977    | .H extension on wave: only integer value expected: name found                                                                |
| Error 978    | name                                                                                                                         |
| Error 979    | IBIS: name                                                                                                                   |
| Error 980    | IBIS: bad value specified for parameter name                                                                                 |
| Error 981    | value cannot be negative                                                                                                     |
| Error 982    | Optimization cannot work when command name is active                                                                         |
| Error 983    | a non FFT output name appears in a FOUR or FOURMODSST extract                                                                |
| Error 984    | unmatched if/else/then statement at or near line name                                                                        |
| Error 985    | Unable to alter parameter name because it controls device instantiation                                                      |
| Error 986    | Entity name affected by both MC and a non-MC change                                                                          |
| Error 987    | .CHRISM: unexpected input format: name                                                                                       |
| Error 988    | .NET: name                                                                                                                   |
| Error 989    | At least two values are needed                                                                                               |
| Error 990    | .MPRUN: name                                                                                                                 |
| Error 991    | .EXTRACT mode                                                                                                                |
| Error 992    | Parameter name is not allowed                                                                                                |
| Error 993    | Unable to create/alter PORT structure on elements name which have not been instantiated with PORT information in the netlist |
| Error 994    | Parameter name should not contain boolean operator                                                                           |
| Error 995    | Parameter name should not contain any special characters                                                                     |
| Error 996    | .EXTRACT: wave name is incompatible with extract analysis                                                                    |
| Error 997    | .CORREL: need at least 2 name to generate a correlation                                                                      |
| Error 998    | .DSPMOD: name                                                                                                                |
| Error 999    | .DSP: name                                                                                                                   |
| Error 1000   | Using a parameter both in OPTIMIZATION and STEP is not allowed                                                               |
| Error 1501   | This circuit exhibits singularity, due to ...                                                                                |
| Error 1502   | .OPTION DSCGLOB: unexpected specification name                                                                               |
| Error 1503   | .OPTION parameter DSCGLOB disabled. Use now DSCGLOB = X or DSCGLOB = GLOB                                                    |

**Table A-8. Miscellaneous Errors**

| Error Number | Description                                                      |
|--------------|------------------------------------------------------------------|
| Error 1504   | .STEP LIB: name                                                  |
| Error 1505   | COMMAND name:                                                    |
| Error 1506   | Problem in using '\\\' in name                                   |
| Error 1507   | Object name: A periodic source must be associated to PHNOISE     |
| Error 3001   | COMMAND .MEAS: duplicate measurement name                        |
| Error 3002   | OBJECT "name": varying parameters can't be used with this object |
| Error 3003   | Unable to open format name                                       |
| Error 3004   | name should be an INTEGER                                        |
| Error 3005   | nesting error inside #MACHBB directive                           |
| Error 3006   | #MACHBB directive must be at top level                           |
| Error 3007   | Can't find subckt name for macro simulation                      |
| Error 3008   | Only commands are accepted here                                  |
| Error 3009   | COMMAND .CALL_TCL name                                           |
| Error 3010   | FBLOCK model is not available for HP 64bits version              |
| Error 3011   | COMMAND .PARAMOPT: no initial value given                        |

## Warning Messages

### Global Warnings

**Table A-9. Global Warnings**

| Warning Number | Description                                                                                                |
|----------------|------------------------------------------------------------------------------------------------------------|
| Warning 1      | Mismatch between .iic and .cir files. .RESTART ignored                                                     |
| Warning 2      | The restarting values for .RESTART DC are only usable when no DC analysis is required. DC analysis omitted |
| Warning 3      | Unusual simulation time                                                                                    |
| Warning 4      | Due to parameter PTF, the integration scheme has been changed from Trapeze to Backward Euler               |
| Warning 5      | unusual value for EPS name                                                                                 |

**Table A-9. Global Warnings**

| Warning Number | Description                                                                                  |
|----------------|----------------------------------------------------------------------------------------------|
| Warning 6      | VDS and VGS initializations are useless with Eldo                                            |
| Warning 7      | No output to display                                                                         |
| Warning 8      | Due to .OPTION SWITCH, the ANALOG specification is ignored                                   |
| Warning 10     | .RESTART: a catenated file                                                                   |
| Warning 12     | .OPTION EPSDIG ignored due to .OPTION ANALOG                                                 |
| Warning 13     | .AC: Values for lower and upper frequencies are inverted                                     |
| Warning 14     | Required 'raw' directory does not exist in the current directory; .OPTION sda=2 ignored      |
| Warning 15     | AC analysis: name                                                                            |
| Warning 16     | Digital description ignored in AC analysis                                                   |
| Warning 17     | .OPTION TIMEDIV ignored due to .OPTION SIMUDIV                                               |
| Warning 18     | .OPTION EPS set to name                                                                      |
| Warning 19     | Results from a TRAN analysis are being used for a DC analysis                                |
| Warning 20     | .OPTION GRAMP accepts only positive integer values below 20 .OPTION GRAMP ignored            |
| Warning 21     | Character 'O' found just after a '.' in line name                                            |
| Warning 22     | Non invertible matrix                                                                        |
| Warning 23     | There are BJT elements in the design, LVLTIM is set to 2                                     |
| Warning 24     | .TOPCELL should appear before any .SUBCKT definition                                         |
| Warning 25     | Rounding errors may occur between the Digital Simulator and Eldo                             |
| Warning 26     | NUMDGT value name not accepted: default value used                                           |
| Warning 27     | Probable syntax error in library name                                                        |
| Warning 28     | Negative or zero value for name ignored                                                      |
| Warning 29     | Such messages will not be displayed in future. Set .OPTION MSGNODE=0 to receive all warnings |
| Warning 30     | Unable to open file name                                                                     |
| Warning 31     | No transient analysis specified .RESTART ignored                                             |
| Warning 32     | No FOUR plot to display: FFT analysis removed                                                |
| Warning 33     | XL: VSW0 and VSW1 set to name                                                                |
| Warning 34     | UIC specification ignored on .TRAN card in ADMS                                              |

**Table A-9. Global Warnings**

| Warning Number | Description                                                                             |
|----------------|-----------------------------------------------------------------------------------------|
| Warning 35     | Multi-platform MC cannot be used                                                        |
| Warning 36     | Unable to open LIB file name                                                            |
| Warning 37     | Multiple definition of parameter name                                                   |
| Warning 38     | Command name is ignored                                                                 |
| Warning 39     | First line of the command file must be a comment                                        |
| Warning 40     | .OPTION parmhier: ‘local’ or ‘global’ expected                                          |
| Warning 41     | Signals will be displayed in the digital output file                                    |
| Warning 43     | overwriting MIN/MAX of parameter name                                                   |
| Warning 44     | .MP not implemented with SST analysis                                                   |
| Warning 45     | DATA name                                                                               |
| Warning 46     | Double definition for parameter name                                                    |
| Warning 47     | Unusual value given for name                                                            |
| Warning 48     | Multiple .TEMP not allowed in ADMS: only last TEMP value will be used                   |
| Warning 49     | This circuit exhibits singularity, due to name                                          |
| Warning 50     | name is ignored because of Mach                                                         |
| Warning 51     | Using .OPTION RMV0, name zero-voltage sources would have been removed from the database |
| Warning 52     | Optimizer not active with Eldo-demo                                                     |
| Warning 53     | Eldo Mach partition UI can’t be used in Eldo Mach Black-Box mode                        |
| Warning 54     | Eldo Mach Black-Box mode disabled because of Eldo Mach partition UI                     |
| Warning 55     | .OPTION name has been set to name                                                       |

## Warnings Related to Nodes

The following warning messages are preceded by NODE <name>:

**Table A-10. Warnings Related to Nodes**

| Warning Number | Description                                                                            |
|----------------|----------------------------------------------------------------------------------------|
| Warning 101    | Significant node voltage variation, but of a lesser magnitude than VTH at time: number |
| Warning 102    | Number of iterations at time: number                                                   |

**Table A-10. Warnings Related to Nodes**

| Warning Number | Description                                                                  |
|----------------|------------------------------------------------------------------------------|
| Warning 103    | Attempt to redefine .IC                                                      |
| Warning 104    | Attempt to redefine .NODESET                                                 |
| Warning 105    | Decreasing time on time-values associated to the .IC command                 |
| Warning 106    | Global node already defined                                                  |
| Warning 107    | Less than two connections                                                    |
| Warning 108    | This node is a floating gate                                                 |
| Warning 109    | This bulk node is not biased                                                 |
| Warning 110    | Connected to capacitors only                                                 |
| Warning 111    | Less than two connections. Line: number                                      |
| Warning 112    | Connected to grounded capacitors only. This node is removed from the netlist |
| Warning 113    | Not connected to any element. This node is removed from the netlist          |
| Warning 114    | Second input signal ignored on that node                                     |
| Warning 115    | This is an ATOD and DTOA node                                                |
| Warning 116    | STRENGTH's CONFLICT on this node. Return to DIGITAL                          |
| Warning 117    | Not connected to any power supply                                            |
| Warning 118    | No DC path to ground                                                         |
| Warning 119    | VTH1 > VTH2 on A2D convertor                                                 |
| Warning 120    | VLO > VHI on D2A convertor                                                   |
| Warning 121    | Signals will be displayed in the digital output file                         |
| Warning 122    | DC dangling node                                                             |
| Warning 123    | A capacitor of more than 1F is attached to that node                         |
| Warning 124    | Appears in .CONNECT only                                                     |
| Warning 125    | Previous IC value superseded                                                 |
| Warning 126    | Previous NODESET/GUESS value superseded                                      |
| Warning 127    | Possibly no DC path on that node                                             |
| Warning 128    | .CONNECT: node not found                                                     |
| Warning 129    | has been replaced by a DSPF network                                          |

## Warnings Related to Objects

The following warning messages are preceded by OBJECT <name>:

**Table A-11. Warnings Related to Objects**

| Warning Number | Description                                                                                                                                                                  |
|----------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Warning 201    | Cannot be parameterized                                                                                                                                                      |
| Warning 202    | Must be a differential amplifier. AC analysis omitted                                                                                                                        |
| Warning 203    | Object not in the linear domain. AC analysis omitted                                                                                                                         |
| Warning 204    | Already defined                                                                                                                                                              |
| Warning 205    | Unrecognized word: name                                                                                                                                                      |
| Warning 206    | No parameter or model specified                                                                                                                                              |
| Warning 207    | Second L value ignored                                                                                                                                                       |
| Warning 208    | Second W value ignored                                                                                                                                                       |
| Warning 209    | Parameter ignored: name                                                                                                                                                      |
| Warning 210    | Z0 set to 50 W                                                                                                                                                               |
| Warning 211    | RON set to 1 kW                                                                                                                                                              |
| Warning 212    | W is less than WMIN. W is set to WMIN                                                                                                                                        |
| Warning 213    | L is less than LMIN. L is set to LMIN                                                                                                                                        |
| Warning 214    | Value becomes negative                                                                                                                                                       |
| Warning 215    | W undefined. Default value is taken: number                                                                                                                                  |
| Warning 216    | L undefined. Default value is taken: number                                                                                                                                  |
| Warning 217    | RS is negative or zero: value reset to number                                                                                                                                |
| Warning 218    | Improper value. Default value is taken (1.0×100). Value: number                                                                                                              |
| Warning 219    | Undeclared inductor                                                                                                                                                          |
| Warning 220    | Default values used for time specification                                                                                                                                   |
| Warning 221    | Instability may occur because some of the root modules of the denominator are greater than 1, (outside the unit z circle). Poles are dumped in the standard output file      |
| Warning 222    | Instability may occur because the real parts of some of the denominator's root are greater than 1, (outside the unit z circle). Poles are dumped in the standard output file |
| Warning 223    | Last specification ignored                                                                                                                                                   |
| Warning 224    | Parameter already used as an equivalent: name                                                                                                                                |

**Table A-11. Warnings Related to Objects**

| Warning Number | Description                                                                              |
|----------------|------------------------------------------------------------------------------------------|
| Warning 225    | CPIN already called on pin: name                                                         |
| Warning 226    | VSTATUS already called on: name                                                          |
| Warning 227    | Resistor value is 0: object not called: name                                             |
| Warning 228    | This resistor is not a RC line. Parameter(s) ignored: name                               |
| Warning 229    | WEFF=W-2DW too small or negative. Default value RES used                                 |
| Warning 230    | LEFF=L-2DL £ 0.0. Default value RES is used                                              |
| Warning 231    | LEFF and WEFF £0.0. Default value CAP is used.                                           |
| Warning 232    | DTOA converter: Rhigh is not specified, the default value of 1 mW is taken               |
| Warning 233    | DTOA converter: Rlow is not specified, the default value of 1 mW is taken                |
| Warning 234    | This element cannot be converted into its switch equivalent                              |
| Warning 235    | Capacitance value is negative or zero                                                    |
| Warning 236    | Node name PARAM:xxxx found: This might be confused with PARAM:xxx                        |
| Warning 237    | The following dimension has an unusual value: name                                       |
| Warning 238    | .SIGBUS: VHI < VLO                                                                       |
| Warning 239    | VHI < VLO on .D2A                                                                        |
| Warning 240    | Object Warning                                                                           |
| Warning 241    | Unable to plot the capacitances of this device, only charges are available               |
| Warning 242    | Unable to plot the charges of this device, only capacitances are available               |
| Warning 243    | This object makes reference to the above node, which will be assumed to be ground node 0 |
| Warning 244    | Level cannot be given on the instance                                                    |
| Warning 245    | Self in short circuit: Object not created                                                |
| Warning 246    | Temperature given for TDEV device: TDEV ignored.                                         |
| Warning 247    | Capacitance value of more than 1F                                                        |
| Warning 248    | There are several ports with that index                                                  |
| Warning 249    | The first pin of that element is undefined: Object not created                           |
| Warning 250    | The second pin of that element is undefined: Object not created                          |
| Warning 251    | Old syntax: Please use Ixx or Vxx element with PORT specification                        |
| Warning 252    | Self-connected objected not created.                                                     |

**Table A-11. Warnings Related to Objects**

| Warning Number | Description                                                                                                                                         |
|----------------|-----------------------------------------------------------------------------------------------------------------------------------------------------|
| Warning 253    | Parameter PGATE is defined for JUNCAP model only                                                                                                    |
| Warning 254    | Differential lines are not supported, the reference plane will be replaced by the ground                                                            |
| Warning 255    | Size not consistent with LMIN/LMAX/WMIN/WMAX                                                                                                        |
| Warning 256    | Very small value specified.                                                                                                                         |
| Warning 257    | Values are out of range for                                                                                                                         |
| Warning 258    | May be diode instantiation was intended, but its name could not start by 'DEL', which is a prefix reserved for Eldo DELAY primitive.                |
| Warning 259    | May be CCCS instantiation was intended, but its name could not be started by '/FNZ', which is a prefix reserved for Eldo FNS/FNZ primitive.         |
| Warning 260    | May be SUBCKT instantiation was intended, but its name could not be started by 'XOR', which is a prefix reserved for Eldo XOR primitive.            |
| Warning 261    | May be Current Source instantiation was intended but its name could not be started by 'INV', which is a prefix reserved for Eldo INVERTOR primitive |
| Warning 262    | Incorrect value for keyword ABS (1 or 0 expected). Using 0 as default                                                                               |
| Warning 263    | Ambiguity on parameter name                                                                                                                         |
| Warning 264    | Object not yet created                                                                                                                              |
| Warning 265    | W or L not specified on that device. Default setting relevant to that model will be used                                                            |
| Warning 266    | The timestep chosen for the simulation exceeds the delay                                                                                            |
| Warning 267    | Very small resistors have been encountered. This may cause non-convergence problems                                                                 |

## Warnings Related to Commands

The following warning messages are preceded by `COMMAND <name>`:

**Table A-12. Warnings Related to Commands**

| Warning Number | Description              |
|----------------|--------------------------|
| Warning 301    | .AC: FREQ1 is set to 1Hz |
| Warning 302    | .AC: FREQ2 is set to 1Hz |
| Warning 303    | .IC name: Ignored        |
| Warning 304    | .NODESET name: Ignored   |

**Table A-12. Warnings Related to Commands**

| Warning Number | Description                                                                                                                              |
|----------------|------------------------------------------------------------------------------------------------------------------------------------------|
| Warning 305    | .SOLVE: R sweep must be between positive values                                                                                          |
| Warning 306    | .SOLVE: No solution has been found                                                                                                       |
| Warning 307    | .SOLVE: No solution has been found in the given interval                                                                                 |
| Warning 308    | .IC or .NODESET name: This value does not match the value specified in the input signal. Zero-time value is: number                      |
| Warning 309    | .CHRSIM: The input signal will force the simulation to end at V=number                                                                   |
| Warning 310    | .CHRSIM: The input signal will force the simulation to end at time = number                                                              |
| Warning 311    | .OPTION name: Ignored                                                                                                                    |
| Warning 312    | .OPTIMIZE name: Ignored                                                                                                                  |
| Warning 313    | .WIDTH name: Ignored                                                                                                                     |
| Warning 314    | .OPTFOUR: Obsolete statement ignored                                                                                                     |
| Warning 315    | .GLOBAL: Nodes specified in this card are considered as global nodes from the introduction of the corresponding .GLOBAL card, NOT BEFORE |
| Warning 316    | .ENDS: The name does not match the .SUBCKT name                                                                                          |
| Warning 317    | name: Option unknown                                                                                                                     |
| Warning 318    | .RAMP name: Specification unknown. .RAMP ignored                                                                                         |
| Warning 319    | .PZ: Previous specification on name ignored                                                                                              |
| Warning 320    | .STEP: Previous card superseded                                                                                                          |
| Warning 322    | .NOISE: Previous specification ignored                                                                                                   |
| Warning 323    | .DC: Previous values superseded                                                                                                          |
| Warning 324    | .RAMP: Previous values superseded                                                                                                        |
| Warning 325    | .TRAN: Previous values superseded                                                                                                        |
| Warning 326    | .OPTIMIZE/.EXTRACT: Node name is connected to ground                                                                                     |
| Warning 327    | .TRAN: The parameter TMAX is not used                                                                                                    |
| Warning 328    | .AC: Previous values superseded                                                                                                          |
| Warning 329    | .TRAN: Specification ignored                                                                                                             |
| Warning 330    | .TRAN: UIC specification accepted                                                                                                        |
| Warning 331    | .SENS: Element name not found                                                                                                            |
| Warning 332    | .SENS: Node name not found                                                                                                               |
| Warning 333    | .PZ: Node name not found                                                                                                                 |

**Table A-12. Warnings Related to Commands**

| <b>Warning Number</b> | <b>Description</b>                                                                          |
|-----------------------|---------------------------------------------------------------------------------------------|
| Warning 334           | .PZ: Element name not found                                                                 |
| Warning 335           | .TF: Node name not found: .TF ignored                                                       |
| Warning 336           | .TF: Element name not found: .TF ignored                                                    |
| Warning 337           | .FIX: Element name not found                                                                |
| Warning 338           | .UNFIX: Element name not found                                                              |
| Warning 339           | .NOISE: Element/object name not found                                                       |
| Warning 340           | .MC: Element name not found                                                                 |
| Warning 341           | .MC: Node name not found                                                                    |
| Warning 342           | .STEP: Parameter name not found                                                             |
| Warning 343           | .STEP: No MOS name found                                                                    |
| Warning 344           | .OPTFOUR: Voltage source name found                                                         |
| Warning 345           | .OPTFOUR: Node name found                                                                   |
| Warning 346           | .PRINT or .PLOT: Node name undeclared                                                       |
| Warning 347           | .PRINT or .PLOT: Element name undeclared                                                    |
| Warning 348           | .DC: Unknown sweep source name                                                              |
| Warning 349           | .AC: No output specified in AC analysis: use .PLOT or .PRINT statement                      |
| Warning 350           | .MC: Specification ignored                                                                  |
| Warning 351           | .OPTIMIZE: Ignored                                                                          |
| Warning 352           | .NOISE: Must be used in AC analysis only. NOISE analysis omitted                            |
| Warning 353           | .IC: Node name not found                                                                    |
| Warning 354           | .NODESET: Node name not found                                                               |
| Warning 355           | .GUESS: Node name not found                                                                 |
| Warning 356           | .CONSO: Voltage source name not found                                                       |
| Warning 357           | .PROBE: Too many nodes to display. This value can be overridden by .OPTION LIMPROBE = value |
| Warning 358           | .OPTIMIZE: New AC input name is ignored                                                     |
| Warning 359           | .PARAM: “=” is missing in expression name                                                   |
| Warning 360           | .OPTIMIZE: Unexpected expression: name                                                      |
| Warning 361           | .EXTRACT: Unexpected expression: name                                                       |

**Table A-12. Warnings Related to Commands**

| Warning Number | Description                                                                                                                                                                                              |
|----------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Warning 362    | .OPTIMIZE: Bounds for name must be MIN MAX                                                                                                                                                               |
| Warning 363    | .PLOT/.PRINT: Wave name will be plotted in .MEAS file                                                                                                                                                    |
| Warning 364    | .OPTIMIZE: GAIN target is unusually high: name                                                                                                                                                           |
| Warning 365    | .POTBUS: Bus name not defined                                                                                                                                                                            |
| Warning 366    | .ANALOG: name not found                                                                                                                                                                                  |
| Warning 367    | .PRINT or .PLOT or .PROBE: name not found                                                                                                                                                                |
| Warning 368    | .WCASE: name not known                                                                                                                                                                                   |
| Warning 369    | .MC ignored: Neither LOT or DEV specification found                                                                                                                                                      |
| Warning 370    | .SETSOA: Expecting D E or M: name found                                                                                                                                                                  |
| Warning 371    | .SETSOA: Specification name unknown                                                                                                                                                                      |
| Warning 372    | .SETSOA: Object name unknown                                                                                                                                                                             |
| Warning 373    | .SETSOA: Model name unknown                                                                                                                                                                              |
| Warning 374    | name analysis omitted because of the DCSWEEP analysis found                                                                                                                                              |
| Warning 375    | .SETSOA: Double specification on name                                                                                                                                                                    |
| Warning 376    | .LOOP: Object name undefined                                                                                                                                                                             |
| Warning 377    | .NOISETRAN is currently incompatible with name. Transient Noise Analysis is therefore omitted. For example, when a netlist contains both .MC and .NOISETRAN commands, the .NOISETRAN command is ignored. |
| Warning 378    | EPS is possibly too large (name): The usual EPS value for Transient Noise is 1.0e-6                                                                                                                      |
| Warning 379    | REL TOL is possibly too large (name): %e The usual REL TOL value for Transient Noise is                                                                                                                  |
| Warning 380    | VNTOL is possibly too large (name): %e The usual VNTOL value for Transient Noise is                                                                                                                      |
| Warning 381    | .PLOT/.PRINT: Wave name will not be plotted in .ASCII file                                                                                                                                               |
| Warning 382    | .OPTION name                                                                                                                                                                                             |
| Warning 392    | .STEP PARAM name: Sweep already defined in the .DC command                                                                                                                                               |
| Warning 393    | .PROBE ISUB (name): No wild character allowed                                                                                                                                                            |
| Warning 394    | .PZ ignored because of FNZ functions                                                                                                                                                                     |
| Warning 395    | .PROBE name: No nodes/objects found                                                                                                                                                                      |
| Warning 396    | ISUB(name) not accessible: name is a global node                                                                                                                                                         |

**Table A-12. Warnings Related to Commands**

| Warning Number | Description                                                                                                                                            |
|----------------|--------------------------------------------------------------------------------------------------------------------------------------------------------|
| Warning 397    | ISUB: Node name not found                                                                                                                              |
| Warning 398    | .PROBE: Missing keyword AC, DC or TRAN                                                                                                                 |
| Warning 399    | Unable to display I(M/B/Jxx): name assumed                                                                                                             |
| Warning 400    | .TRAN: TSTART > TSTOP: TSTART set to 0                                                                                                                 |
| Warning 401    | .STEP: INCR_SPEC > T2-T1                                                                                                                               |
| Warning 402    | .NOISE: zero or negative argument                                                                                                                      |
| Warning 403    | .AC: Frequency cannot be 0.0 or less: The value has been reset to 1.0                                                                                  |
| Warning 404    | .MODDUP: element name not found                                                                                                                        |
| Warning 405    | .MODDUP: cannot be used on element name                                                                                                                |
| Warning 406    | .LIB cannot find name                                                                                                                                  |
| Warning 407    | .SETSOA ignoring SOA on model name                                                                                                                     |
| Warning 408    | .SETSOA ignoring SOA on object name                                                                                                                    |
| Warning 409    | .TF analysis not performed: Bad operating point                                                                                                        |
| Warning 410    | The line is too long: It has been truncated                                                                                                            |
| Warning 411    | .OPTNOISE: previous command superseded                                                                                                                 |
| Warning 412    | .OPTNOISE: ON/OFF expected: name found                                                                                                                 |
| Warning 413    | .MC: Unknown specification name: LOT or DEV expected                                                                                                   |
| Warning 414    | .FOUR: Parameter TSTOP is ignored                                                                                                                      |
| Warning 415    | .TRAN: Parameter TPRINT is set to: name                                                                                                                |
| Warning 416    | .OPTFOUR: Previous values superseded                                                                                                                   |
| Warning 417    | .USE: DC value for source name has been changed                                                                                                        |
| Warning 418    | .GLOBAL: Some subcircuits are already defined, if these subcircuits make use of nodes which will appear in .GLOBAL statements, results can be in error |
| Warning 419    | .OPTFOUR: Using INTERPOLATE=0 may increase the number of points computed by the simulator                                                              |
| Warning 420    | Missing FPORT number name                                                                                                                              |
| Warning 421    | LOT/DEV specification on parameter name ignored                                                                                                        |
| Warning 422    | .PZ: Specified device is not a 'voltage-like' element                                                                                                  |
| Warning 423    | .OPTFOUR: TSTART > TSTOP: TSTART is ignored                                                                                                            |
| Warning 424    | .OPTFOUR: TSTART < 0: TSTART is ignored                                                                                                                |

**Table A-12. Warnings Related to Commands**

| Warning Number | Description                                                                                                                                                          |
|----------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Warning 425    | .OPTFOUR: TSTOP < 0: TSTOP is ignored                                                                                                                                |
| Warning 426    | .OPTFOUR: NBPT < 0: NBPT is ignored                                                                                                                                  |
| Warning 427    | .OPTFOUR: FS < 0: FS is ignored                                                                                                                                      |
| Warning 428    | .OPTFOUR: Fnormal < 0: Fnormal is ignored                                                                                                                            |
| Warning 429    | .OPTFOUR: Fmin < 0: Fmin is ignored                                                                                                                                  |
| Warning 430    | .OPTFOUR: Fmax < 0: Fmax is ignored                                                                                                                                  |
| Warning 431    | .OPTFOUR: Fund < 0: Fund is ignored                                                                                                                                  |
| Warning 432    | name: Only Magnitude or DB of NOISE outputs are available                                                                                                            |
| Warning 433    | LOT/DEV specification is ignored on the non-primitive parameter name                                                                                                 |
| Warning 434    | The given value for VSW0 is superior to VSW1: The values will be interchanged                                                                                        |
| Warning 435    | DEFAD/DEFAS/DEFPD/DEFPS overwritten by .OPTION XA for MOS models BERKELEY/BSIMxx/EKV and MM9                                                                         |
| Warning 436    | .EXTRACT: Unable to open file name                                                                                                                                   |
| Warning 437    | .PLOT/.PRINT: wave name will be plotted as                                                                                                                           |
| Warning 438    | .TRAN: last Tstep ignored.                                                                                                                                           |
| Warning 439    | .SETBUS: last Time point ignored                                                                                                                                     |
| Warning 440    | name ignored with Eldo Mach                                                                                                                                          |
| Warning 441    | .STEP: increment must be positive: name used                                                                                                                         |
| Warning 442    | .PARAM name                                                                                                                                                          |
| Warning 443    | .MEAS name                                                                                                                                                           |
| Warning 444    | .STEP: Compared to Eldo 5.4, the semantic of the .STEP ... LIN has changed: LIN stands for 'number of runs'<br>Rather use keyword INCR if 'increment value' is meant |
| Warning 445    | .TVINCLUDE: name                                                                                                                                                     |
| Warning 446    | .PLOT: name statement removed. It appears more than once.                                                                                                            |
| Warning 447    | .STEP LIB: name                                                                                                                                                      |
| Warning 448    | .MPRUN name Ignored: Cannot distribute jobs (no multi-run analysis or no hosts)                                                                                      |
| Warning 449    | .NET name                                                                                                                                                            |
| Warning 450    | EXTRACT MODE: name                                                                                                                                                   |
| Warning 451    | Can't update correlation coefficient depending of parameter name                                                                                                     |

**Table A-12. Warnings Related to Commands**

| Warning Number | Description                                                                           |
|----------------|---------------------------------------------------------------------------------------|
| Warning 452    | .CORREL ignored, no statistical analysis found.                                       |
| Warning 453    | .DSP name                                                                             |
| Warning 454    | .PLOT: unexpected output name                                                         |
| Warning 455    | .POTBUS: limited to 53 bits: name not plotted                                         |
| Warning 456    | .SIGBUS: Only first values specified for bus name will be taken for .SST analysis     |
| Warning 457    | .STEP: no SST analysis found, keyword (AUTOINCR) ignored                              |
| Warning 458    | .MC: vary was already specified...                                                    |
| Warning 459    | .EXTRACT DC: DCTRAN assumed since both transient and ac analysis found in the netlist |

## Warnings Related to Models

The following warning messages are preceded by MODEL <name>:

**Table A-13. Warnings Related to Models**

| Warning Number | Description                                                     |
|----------------|-----------------------------------------------------------------|
| Warning 501    | No parameter specified. Default values are used                 |
| Warning 502    | GOFF less than zero                                             |
| Warning 503    | Access resistors for this model are created as objects          |
| Warning 504    | Unable to match reverse and forward region. BV = number         |
| Warning 505    | Reset parameter                                                 |
| Warning 508    | LAMBDA value is high and may cause problems of convergence      |
| Warning 509    | KB1 is set to number                                            |
| Warning 510    | Parameter XQC ignored                                           |
| Warning 512    | IBV set to name                                                 |
| Warning 513    | Parameter CF1 set to the default value: number                  |
| Warning 514    | Parameter CF3 set to the default value: number                  |
| Warning 515    | Parameters specific to BSIM appear and LEVEL is neither 8 or 11 |
| Warning 516    | LOT and DEV cannot be applied on R L C parameters               |
| Warning 517    | Parameter MJSW must be less than 1.0. Default is used           |

**Table A-13. Warnings Related to Models**

| Warning Number | Description                                                                                              |
|----------------|----------------------------------------------------------------------------------------------------------|
| Warning 518    | Parameter MJ must be less than 1.0. Default is used                                                      |
| Warning 519    | LOT specification is less than or equal to zero                                                          |
| Warning 520    | DEV specification is less than or equal to zero                                                          |
| Warning 521    | The following parameter has an unusual value:                                                            |
| Warning 522    | DMUT parameter appears in a non-ST model                                                                 |
| Warning 523    | SUBSN=1 is not allowed on a lateral PNP                                                                  |
| Warning 524    | VTH1 > VTH2                                                                                              |
| Warning 525    | VLO > VHI                                                                                                |
| Warning 526    | Model Warning                                                                                            |
| Warning 527    | Parameter AF has an unusual value                                                                        |
| Warning 528    | Parameter KF has an unusual value                                                                        |
| Warning 529    | Only one threshold is allowed for BIT A2D models: The average value has been taken                       |
| Warning 530    | MCMOD cannot be applied on the model parameter: It can only be applied to a nominal value: MCMOD ignored |
| Warning 531    | Sinwave function: Theta factor is negative                                                               |
| Warning 532    | COX is ignored because TOX is given                                                                      |
| Warning 533    | The model parameters (DW/RSH) that are specific to RC WIRE have been ignored                             |
| Warning 534    | Double definition for parameter(s)                                                                       |
| Warning 535    | Level 21 is now replaced by Level name. For future versions make sure to use that level                  |
| Warning 536    | The model parameter RHO is needed for ST model                                                           |
| Warning 537    | name, parameter ignored                                                                                  |

## Warnings Related to Subcircuits

The following warning messages are preceded by SUBCKT <name>:

**Table A-14. Warnings Related to Subcircuits**

| Warning Number | Description            |
|----------------|------------------------|
| Warning 701    | Unused parameter: name |

**Table A-14. Warnings Related to Subcircuits**

| Warning Number | Description                                                     |
|----------------|-----------------------------------------------------------------|
| Warning 702    | name: Cannot be updated with .LIB in interactive mode           |
| Warning 703    | SUBCKT name cannot be that of a keyword                         |
| Warning 704    | This pin name appears more than once on definition              |
| Warning 705    | The header of this SUBCKT contains node names defined as global |

## Miscellaneous Warnings

**Table A-15. Miscellaneous Warnings**

| Warning Number | Description                                                                                                            |
|----------------|------------------------------------------------------------------------------------------------------------------------|
| Warning 901    | Last time ignored on input signal at line: number                                                                      |
| Warning 902    | name: Model parameter ignored                                                                                          |
| Warning 903    | Too many nodes to plot (>8)                                                                                            |
| Warning 904    | Previous value of name superseded                                                                                      |
| Warning 905    | VSAT is set to: number                                                                                                 |
| Warning 906    | VSATM is set to: number                                                                                                |
| Warning 907    | Parameter not used: name                                                                                               |
| Warning 908    | name: This parameter cannot be assigned                                                                                |
| Warning 909    | name: Unknown parameter                                                                                                |
| Warning 910    | Unable to measure ISUB(name): Multiple connections                                                                     |
| Warning 911    | PORT name has only one connection                                                                                      |
| Warning 912    | IBIS: name                                                                                                             |
| Warning 913    | IBIS: parameter name ignored                                                                                           |
| Warning 914    | IBIS: parameter name not in the interval [0,1] is ignored                                                              |
| Warning 915    | Probably missing a path to ground from object name                                                                     |
| Warning 916    | Unable to parse expression name                                                                                        |
| Warning 917    | SWEEP specification on TRAN/DC/AC ignored when .STEP are used: these two syntaxes are not compatible                   |
| Warning 918    | The check for instance duplicates has been disabled (netlist too large). Use .OPTION CHECKDUPL to override this limit. |

**Table A-15. Miscellaneous Warnings**

| Warning Number | Description                                        |
|----------------|----------------------------------------------------|
| Warning 919    | .OPTION ACCOUT=1 can not be applied to FOUR(name). |
| Warning 920    | Instances name, * not created because M is 0*      |
| Warning 921    | MC variation on name                               |

**Table A-16. Miscellaneous Warnings**

| Warning Number | Description                                                                                    |
|----------------|------------------------------------------------------------------------------------------------|
| Warning 1001   | .NOISETRAN ignored: Only single TRAN analysis is allowed                                       |
| Warning 1002   | .NOISE ignored because the output node is not defined                                          |
| Warning 1003   | .SWITCH: Element name not found                                                                |
| Warning 1004   | .MCMOD: Parameter name not known                                                               |
| Warning 1005   | .MCMOD: Element name not known                                                                 |
| Warning 1006   | COMMAND name removed from the include file                                                     |
| Warning 1007   | .SAVE and .RESTART used on the same file: .SAVE ignored                                        |
| Warning 1008   | COMMAND name found in include file                                                             |
| Warning 1009   | .OPTION: Incorrect range for name                                                              |
| Warning 1010   | .PRINT or .PLOT: AC-like output name for non-AC analysis                                       |
| Warning 1011   | .PRINT or .PLOT: Unexpected AC output name                                                     |
| Warning 1012   | .PLOT element name is not declared                                                             |
| Warning 1013   | .PROBE/.SAVE element name is not declared                                                      |
| Warning 1014   | .VIEW element name is not declared                                                             |
| Warning 1015   | Element name is not found                                                                      |
| Warning 1016   | .SOLVE: Previous specifications superseded                                                     |
| Warning 1017   | AC input sources used in connection with FPORT                                                 |
| Warning 1018   | Digital gates are not handled in DCSWEEP                                                       |
| Warning 1019   | .WC and .MC cannot be used together: .WC ignored                                               |
| Warning 1020   | .PZ analysis might give unexpected results because several AC sources are found in the circuit |
| Warning 1021   | .PRINT or .PLOT: Unexpected AC output name for SST                                             |
| Warning 1022   | .PRINT or .PLOT: Only V() or I() is allowed for SST type                                       |

**Table A-16. Miscellaneous Warnings**

| Warning Number | Description                                                      |
|----------------|------------------------------------------------------------------|
| Warning 1023   | .TDEV: Object name not found                                     |
| Warning 1024   | .PLOT element name not declared                                  |
| Warning 1025   | .PRINT or .PLOT: Unexpected FFT output name                      |
| Warning 1026   | .PRINT/.PLOT/.PROBE: Missing declaration .FOUR LABEL=name        |
| Warning 1027   | .PRINT or .PLOT SSTNOISE: Expected SSTNOISE output, and not name |
| Warning 1028   | .MEAS: Element name not declared                                 |
| Warning 1029   | .EXTRACT: Element name not declared                              |
| Warning 1030   | .PRINT: Smith chart expects non-AC wave names                    |
| Warning 1031   | .SSTPROBE name: second node assumed to be 0                      |
| Warning 1032   | .PRINT/PLOT/PROBE/EXTRACT: bad number of tones specified         |
| Warning 1033   | .PLOT: name. wave type is incompatible with extract type         |
| Warning 1034   | name: Group-Delay specifications ignored in SST analysis         |
| Warning 1035   | .PRINT or .PLOT: Only WOPT() is allowed for OPT type             |
| Warning 1036   | SUBCKT name will be simulated as top level                       |
| Warning 1037   | Standard Eldo Mach commands ignored in Eldo Mach Black-Box mode  |

# Appendix B

## Troubleshooting

---

### Introduction

Most users of any product will from time to time experience problems in its use. This appendix is intended to provide an easy to use quick reference from which such problems can be identified and corrected.

### Common Netlist Errors

Below is a list of the common errors to look out for when producing an input netlist:

1. First Line of a File Not the Title

The first line of a netlist *MUST* be the title of the circuit. If part of the circuit description is found on the first line, it will be ignored. If the first line is a blank, this will be taken as the title line.

2. Correct Usage of the Simulator Accuracy (eps, vntol, reltol)

The simulator accuracy parameters are the most important Eldo parameters—they must be correctly initialized.

---

 For more details refer to the [Speed and Accuracy](#) chapter.

---

3. Correct Units on Devices

It is very important to make sure that all the components in the netlist have correct units; one mistake can have a large effect. An example of this is specifying units of farads instead of picofarads.

4. Missing Model

Make sure that model definitions are available to the simulator, either directly in the netlist, or in a referenced library.

5. Missing Voltage Sources

Independent voltage sources defined to have a voltage of 0V may be removed by Eldo in order to simplify calculations. If you wish to use a voltage source as a current probe, avoid defining its voltage as 0V, but instead, define its voltage value to be very small. Moreover, currents through components may be measured directly, therefore, the use of voltage sources as current probes is not necessary.

## 6. Correct Model Name Syntax

The first character of a model name *CANNOT* be a number. This causes the compilation of the netlist to be interrupted, and an error message to be displayed.

## 7. Unknown Model Parameter

When defining model parameter values, care must be taken to ensure that the parameter is spelt correctly and that it is indeed a legal parameter of the device.

## 8. Ground (0) Node

A 0 node (i.e. ground) must always be present in the input netlist.

---

### Note

---

 Eldo does not recognize GND as GROUND!

---

## 9. Reserved Keywords

The following keywords are special in that they may appear in expressions. However, they may not be specified in a **.PARAM** command if an RF analysis is specified in the netlist.

**Table B-1. Reserved Keywords not available in .PARAM**

| AMNOISE  | BFACTOR           | BOPT      | FREQ     |
|----------|-------------------|-----------|----------|
| GA_mag   | GA_dB             | GAC       | GAM_mag  |
| GAM_dB   | GAMMA_OPT_MAG     | GASM_mag  | GASM_dB  |
| GAUM_mag | GAUM_dB           | GOPT      | GP_mag   |
| GP_dB    | GPC               | INOISE    | KFACTOR  |
| LSC      | MUFACTOR          | NFMIN_mag | NFMIN_dB |
| ONOISE   | PHI_OPT           | PHNOISE   | POWER    |
| RNEQ     | SCALE             | SNF_mag   | SNF_dB   |
| SSC      | TEMP              | TGP_mag   | TGP_dB   |
| TIME     | TNOM <sup>a</sup> | XAXIS     |          |

a. TNOM may be specified as a parameter in a **.PARAM** command when **.OPTION DEFPTNOM** is set. The temperature value used by the Eldo model evaluator is always that which is set with **.OPTION TNOM=val**.

If an RF analysis is specified in the netlist, and if any **.PARAM** is named with one of these keywords, it will be rejected. For example, the following statement will generate an error:

```
.PARAM SCALE=VAL
```

# Appendix C Examples

---

## Introduction

This chapter contains the set of examples that are distributed with the Eldo software package. Each example consists of the complete Eldo netlist of the circuit, together with output results obtained from EZwave, the Eldo waveform viewer. A summary of the examples is shown below. Listings for these examples may be found in the following subdirectories included with your software:

`$MGC_AMS_HOME/eldo/$eldover/examples/eldo/`

where `$MGC_AMS_HOME` is the directory where the software resides.

---

### Note

---



For more examples please refer to “[Examples for IEM](#)” on page 17-4 and the [Tutorials](#) chapter.

---

**Table C-1. Eldo Examples**

| Example No. | Circuit Name          | Eldo Description                                         |
|-------------|-----------------------|----------------------------------------------------------|
| 1           | <i>trigger.cir</i>    | <a href="#">SC Schmitt Trigger</a>                       |
| 2           | <i>ad4b.cir</i>       | <a href="#">4-bit Adder</a>                              |
| 3           | <i>aopalt.cir</i>     | <a href="#">CMOS Operational Amplifier (Open Loop)</a>   |
| 4           | <i>aopbou.cir</i>     | <a href="#">CMOS Operational Amplifier (Closed Loop)</a> |
| 5           | <i>ellipt5.cir</i>    | <a href="#">5th Order SC Low Pass Filter</a>             |
| 6           | <i>niv4_6.cir</i>     | <a href="#">Charge Control in MOS Models</a>             |
| 7           | <i>ua741.cir</i>      | <a href="#">Active RC Band Pass Filter</a>               |
| 8           | <i>integrator.cir</i> | <a href="#">Second Order Delta Sigma Modulator</a>       |
| 9           | <i>extract.cir</i>    | <a href="#">Operational Amplifier</a>                    |

## Example#1—SC Schmitt Trigger

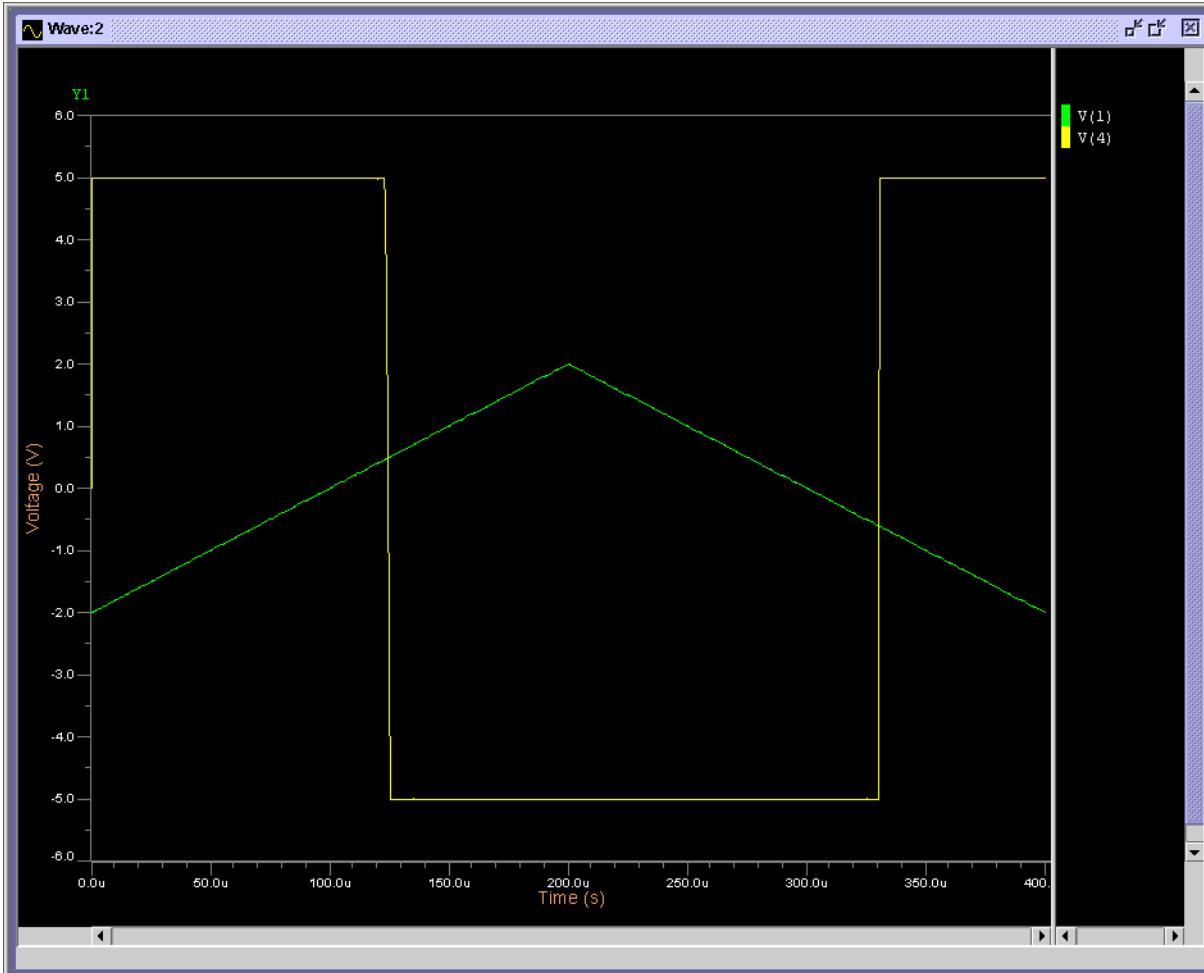
This example deals with a transient analysis of an SC Schmitt trigger circuit. The complete netlist can be found in the file *trigger.cir*.

### Complete Netlist

```
Schmitt trigger
.model ampop opa level=2 voff=0 sl=50e06
+ cin=0 rs=10 vsat=5 gain=5000 fc=5000
.subckt ampli 1 2 3
opa1 2 1 3 0 ampop
.ends ampli
s1 cl 1 2 1k
s2 cl 3 6 1k
s3 clb 6 8 1k
s4 clb 9 4 1k
s5 cl 9 0 1k
c1 2 0 0.01n
c2 3 0 0.01n
c3 6 8 0.01n
c4 8 9 0.001n
x1 2 3 4 ampli
x2 8 0 6 ampli
.chrent cl 0 -5 0.1u 5 4.9u 5 5u -5 10u -5 p
.chrent clb 0 -5 5u -5 5.1u 5 9.9u 5 10u -5 p
.chrent 1 0 -2 0.2m 2 0.4m -2 f
.tran 1u 0.4m uic
.plot tran v(1) v(4) (-6,6)
.print tran v(1) v(4)
.option eps=1e-4
.end
```

## Simulation Results

**Figure C-1. Example#1—Simulation Results**



## Example#2—4-bit Adder

This example deals with a transient analysis of a 4-bit adder. The complete netlist can be found in the file *ad4b.cir*.

### Complete Netlist

```
ad4b
.model mod1 nmos
+ niv=6 eox=25.0N muo=600 nb=2.0e+16 dphif=0.6 vl=2.0e5
+ kw=2.24U kl=2.24U gw=3.91U gl=0.7U dinf=0.1 ve=10.0e+4
+ ldif=10U cdifs0=0.0001 cdifp0=0 dw=0 dl=0.8u rec=0.15u
+ vt0=0.55 kb=0.1 tg=0.06
```

## Examples

### Example#2—4-bit Adder

---

```
.model mod2 pmos
+ niv=6 eox=25n muo=225 nb=2.0e+16 dphif=0.7 vl=1.9e+5
+ kw=0.52u kl=4u gw=0.453u gl=0.7u dinf=0.17 ve=7.35e+4
+ ldif=10u cdifs0=0.000316 cdifp0=0 dw=0 dl=1.5u
+ rec=0.5u vt0=0.55 kb=0.34 tg=0.14
.model mod3 nmos
+ niv=6 dw=0.5u dl=1.1u eox=60n dphif=0.7 vt0=-4.0 muo=446.0
+ kb=0.4 kw=0.0u kl=0.0u gw=2.4u gl=0.5u dinf=0.14 ve=12.8e+4
+ rec=0.4u tg=0.05 vl=1.0e+5 sh=0.1 nb=1e+15
.model mod4 nmos
+ niv=6 dw=0.5u dl=1.1u eox=60.0n dphif=0.7 vt0=0.7 muo=672.0
+ kb=0.234 kw=2.17u kl=0.49u gw=4.325u gl=0.872u dinf=0.105
+ ve=7.71e+4 rec=0.4u tg=0.05 vl=8.55e+4 sh=0.1 nb=1.0e+15
*subcircuit definition
.subckt flipflop vdd vss h d q qb
m1 n1 n2 0 vss mod4 w=15u l=3.5u
m2 n2 n1 0 vss mod4 w=15u l=3.5u
m3 n2 h 0 vss mod4 w=15u l=3.5u
m4 n4 n2 0 vss mod4 w=15u l=3.5u
m5 n1 n5 0 vss mod4 w=15u l=3.5u
m6 n4 h 0 vss mod4 w=15u l=3.5u
m7 n4 n5 0 vss mod4 w=15u l=3.5u
m8 n5 n4 0 vss mod4 w=15u l=3.5u
m9 n5 d 0 vss mod4 w=15u l=3.5u
m10 q n2 0 vss mod4 w=15u l=3.5u
m11 q qb 0 vss mod4 w=15u l=3.5u
m12 qb q 0 vss mod4 w=15u l=3.5u
m13 qb n4 0 vss mod4 w=15u l=3.5u
m14 vdd n1 n1 vss mod3 w=6.0u l=5.0u
m15 vdd n2 n2 vss mod3 w=6.0u l=5.0u
m16 vdd n4 n4 vss mod3 w=6.0u l=5.0u
m17 vdd n5 n5 vss mod3 w=6.0u l=5.0u
m18 vdd q q vss mod3 w=6.0u l=5.0u
m19 vdd qb vss mod3 w=6.0u l=5.0u
c1 n1 0 0.029p
c2 n2 0 0.048p
c3 n4 0 0.044p
c4 n5 0 0.046p
c5 q 0 0.066p
c6 qb 0 0.053p
.ends flipflop
*subcircuit definition
.subckt adder vdd vss a b er s sr
m1 n12 b a vss mod4 w=4.0u l=4.0u
m2 s n13 n14 vss mod4 w=4.0u l=4.0u
m3 s er n12 vss mod4 w=4.0u l=4.0u
m4 n12 n16 n11 vss mod4 w=4.0u l=4.0u
m5 n11 a 0 vss mod4 w=15.0u l=3.5u
m6 n16 b 0 vss mod4 w=15.0u l=3.5u
m7 n14 n12 0 vss mod4 w=15.0u l=3.5u
m8 n13 er 0 vss mod4 w=15.0u l=3.5u
m9 n17 b 0 vss mod4 w=15.0u l=3.5u
m10 n17 a 0 vss mod4 w=15.0u l=3.5u
```

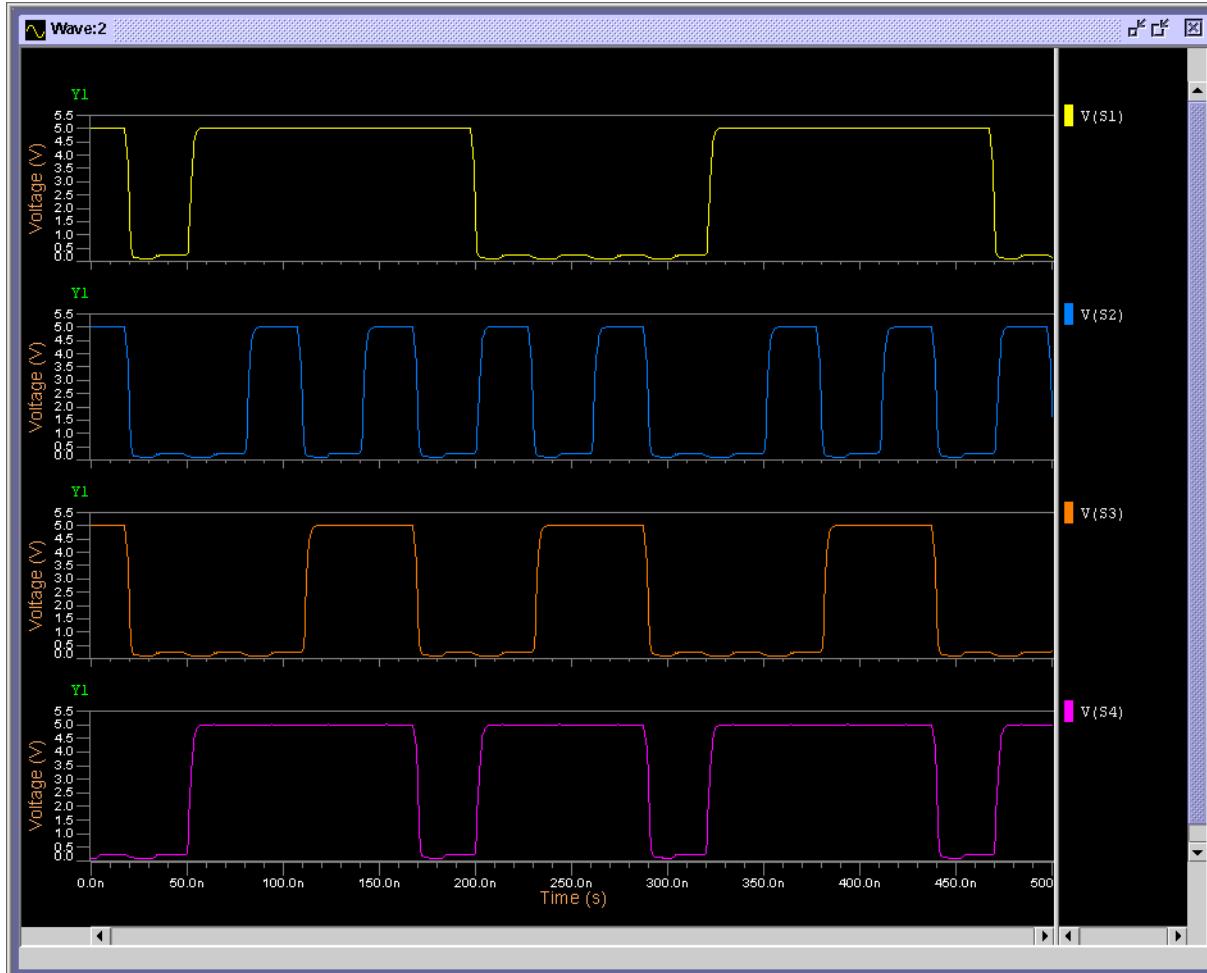
```

m11 n15 er 0 vss mod4 w=15.0u l=3.5u
m12 n15 n12 0 vss mod4 w=15.0u l=3.5u
m13 sr n15 0 vss mod4 w=15.0u l=3.5u
m14 sr n17 0 vss mod4 w=15.0u l=3.5u
m15 vdd n11 n11 vss mod3 w=6.0u l=5.0u
m16 vdd n16 n16 vss mod3 w=6.0u l=5.0u
m17 vdd n14 n14 vss mod3 w=6.0u l=5.0u
m18 vdd n13 n13 vss mod3 w=6.0u l=5.0u
m19 vdd n15 n15 vss mod3 w=6.0u l=5.0u
m20 vdd n17 n17 vss mod3 w=6.0u l=5.0u
m21 vdd Sr sr vss mod3 w=6.0u l=5.0u
c1 s 0 0.031p
c2 n11 0 0.31p
c3 n12 0 0.001p
c4 n13 0 0.018p
c5 n14 0 0.024p
c6 n15 0 0.033p
c7 n16 0 0.018p
c8 n17 0 0.036p
c9 sr 0 0.040p
.ends adder
*subcircuit calls
xa1 vdd vss n1 n4 0 n3 n5 adder
xa2 vdd vss a1 n7 n5 n6 n8 adder
xa3 vdd vss a2 n12 n8 n11 n13 adder
xb1 vdd vss h n3 n4 S1 flipflop
xb2 vdd vss h n6 n7 S2 flipflop
xb3 vdd vss h n11 n12 S3 flipflop
xb4 vdd vss h n13 n1 S4 flipflop
c1 n3 0 0.07p
c2 n4 0 0.04p
c3 n6 0 0.07p
c4 n7 0 0.04p
c5 n5 0 0.01p
c6 n8 0 0.01p
c7 n13 0 0.01p
c8 n1 0 0.04p
c9 n11 0 0.07p
c10 n12 0 0.04p
*voltage commands
.chrent a1 0 5 f
.chrent a2 0 5 f
.chrent h 0 5n 5 15n 5 20n 0 30n 0 p
.chrent vdd 0 5 f
.chrent vss 0 -2.5 f
*output commands
.plot tran v(h)
.plot tran v(s1)
.plot tran v(s2)
.plot tran v(s3)
.plot tran v(s4)
.plot tran v(n6)
.plot tran v(n11)
.plot tran v(n13)
.tran 1ns 500ns uic
.option eps=1.0e-2
.end

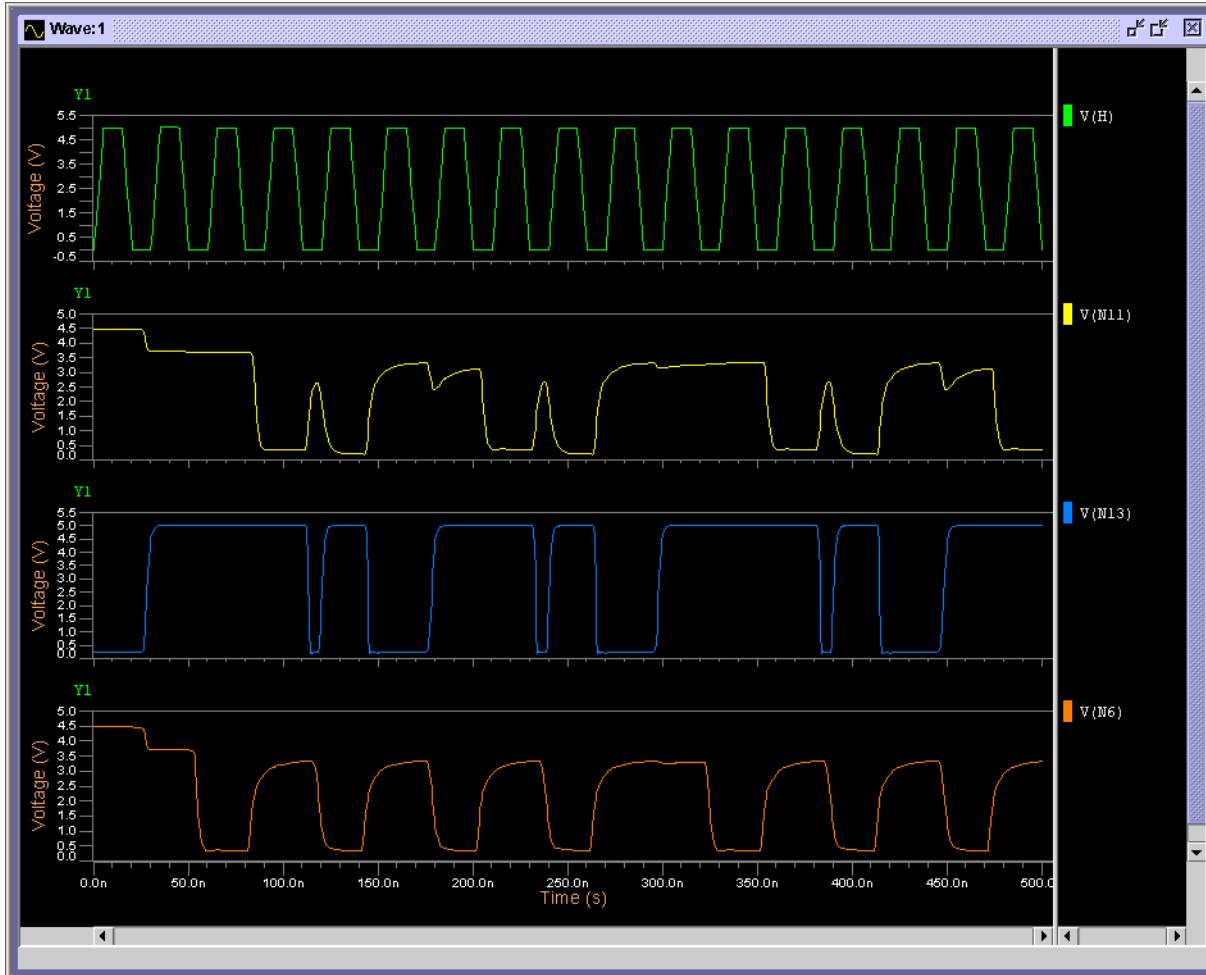
```

## Simulation Results

Figure C-2. Example#2—Simulation Results—1



**Figure C-3. Example#2—Simulation Results—2**



## Example#3—CMOS Op-amp (Open Loop)

This example deals with an AC analysis of an open loop CMOS operational amplifier circuit. The complete netlist can be found in the file *aopalt.cir*.

### Complete Netlist

```

aopalt
.model mod1 nmos level=merck2 eox=25.0n mu0=600 nb=2.0e+16
+ dphif=0.6 v1=2.0e+5 kw=2.24u kl=2.24u lot=0.5% gw=3.91u gl=0.7u
+ dev=0.07e-6 dinf=0.1
+ ve=10.0e+4 ldif=10u cdifs0=0.0001 cdifp0=0.0
+ dw=0.0 dl=0.8u rec=0.15u vt0=0.55 kb=0.1 tg=0.06

.model mod2 pmos level=merck2 eox=25n mu0=225 nb=2.0e+16
+ dphif=0.7 v1=1.9e+5 kw=0.52u kl=4u gw=0.453u gl=0.7u dinf=0.17
+ ve=7.35e+4 ldif=10u cdifs0=0.000316 cdifp0=0
+ dw=0 dl=1.5u rec=0.5u vt0=-0.55 kb=0.34 tg=0.14
*amplifier

```

## Examples

### Example#3—CMOS Op-amp (Open Loop)

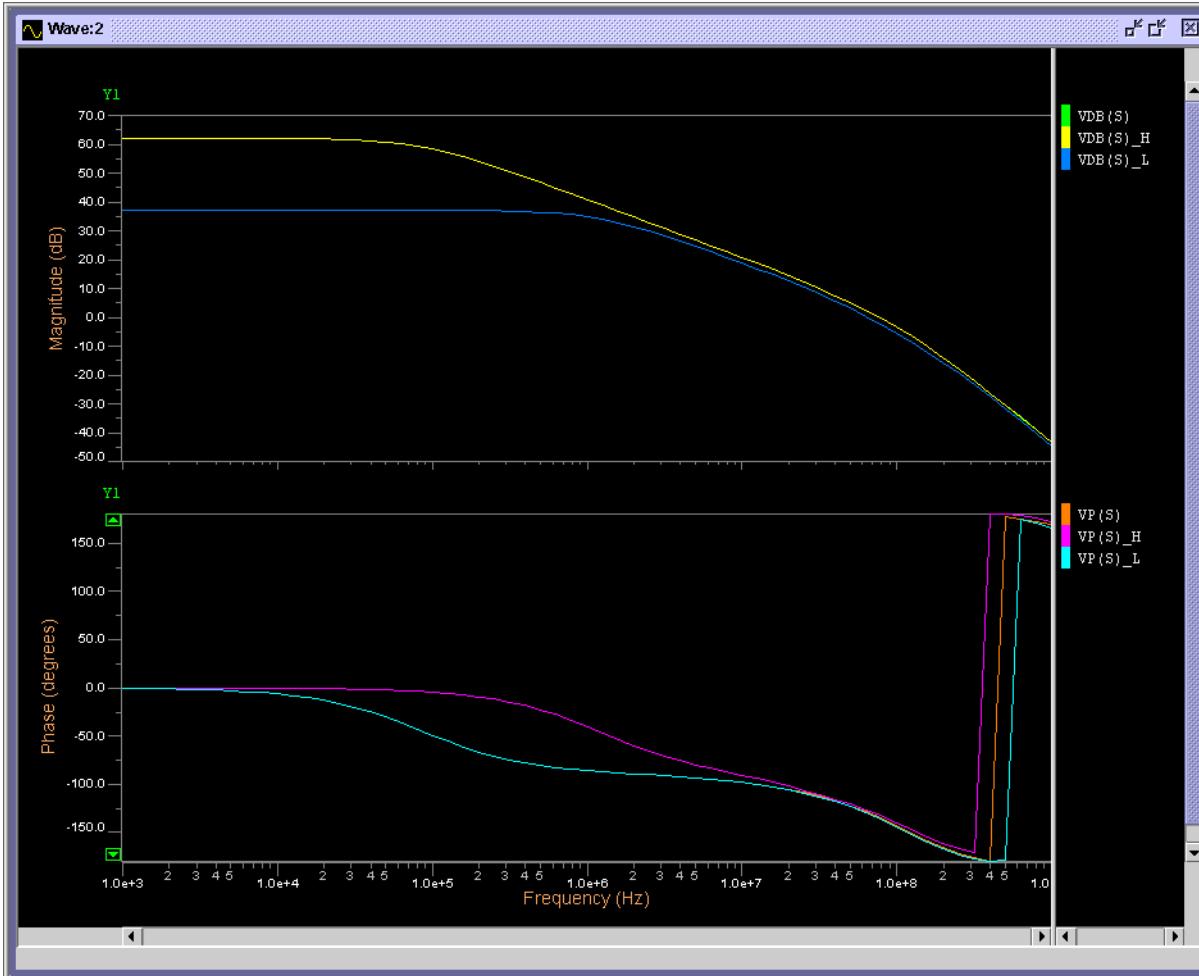
---

```
m1 a g vdd vdd mod2 w=120u l=5.5u
m2 b g vdd vdd mod2 w=120u l=5.5u
m3 d k a vdd mod2 w=116u l=3.5u
m4 s k b vdd mod2 w=116u l=3.5u
m5 c i vss vss mod1 w=63u l=6u
m6 a ep c vss mod1 w=130u l=4u
m7 b en c vss mod1 w=130u l=4u
m8 d d ff vss mod1 w=5.5u l=4.5u
m9 s d e vss mod1 w=5.5u l=4.5u
m10 ff e vss vss mod1 w=42u l=4u
m11 e e vss vss mod1 w=42u l=4u
m12 g g vdd vdd mod2 w=14.5u l=5.5u
m13 g g h vss mod1 w=9u l=5.5u
m14 i i h vdd mod2 w=19u l=4.5u
m15 i i vss vss mod1 w=6u l=6u
m16 j g vdd vdd mod2 w=20u l=5.5u
m17 j j k vss mod1 w=26u l=3.5u
m18 nl i k vdd mod2 w=3u l=3.5u
m19 nl nl vss vss mod1 w=4u l=3.5u
c1 s 0 1.5p
v1 vdd 0 3
v2 vss 0 -3
vinm en 0 0
vinp ep 0 ac

.mc 7 vdb(s)
.ac dec 10 1.0e3 1.0e9
.plot ac vdb(s)
.plot ac vp(s)
.option eps=1.0e-4
.end
```

## Simulation Results

**Figure C-4. Example#3—Simulation Results**



## Example#4—CMOS Op-amp (Closed Loop)

The circuit in this example is the same as in the last one with the exception being that the circuit is in a closed loop configuration. A transient analysis is performed and the complete netlist can be found in the file *aopbou.cir*.

### Complete Netlist

```

aopbau
.model mod1 nmos level=merck2 eox=25.0n mu0=600 nb=2.0e+16
+ dphif=0.6 v1=2.0e+5 kw=2.24u kl=2.24u gw=3.91u gl=0.7u dinf=0.1
+ ve=10.0e+4 ldif=10u cdifs0=0.0001 cdifp0=0.0
+ dw=0.0 dl=0.8u rec=0.15u vt0=0.55 kb=0.1 tg=0.06

.model mod2 pmos level=merck2 eox=25n mu0=225 nb=2.0e+16

```

## Examples

### Example#4—CMOS Op-amp (Closed Loop)

---

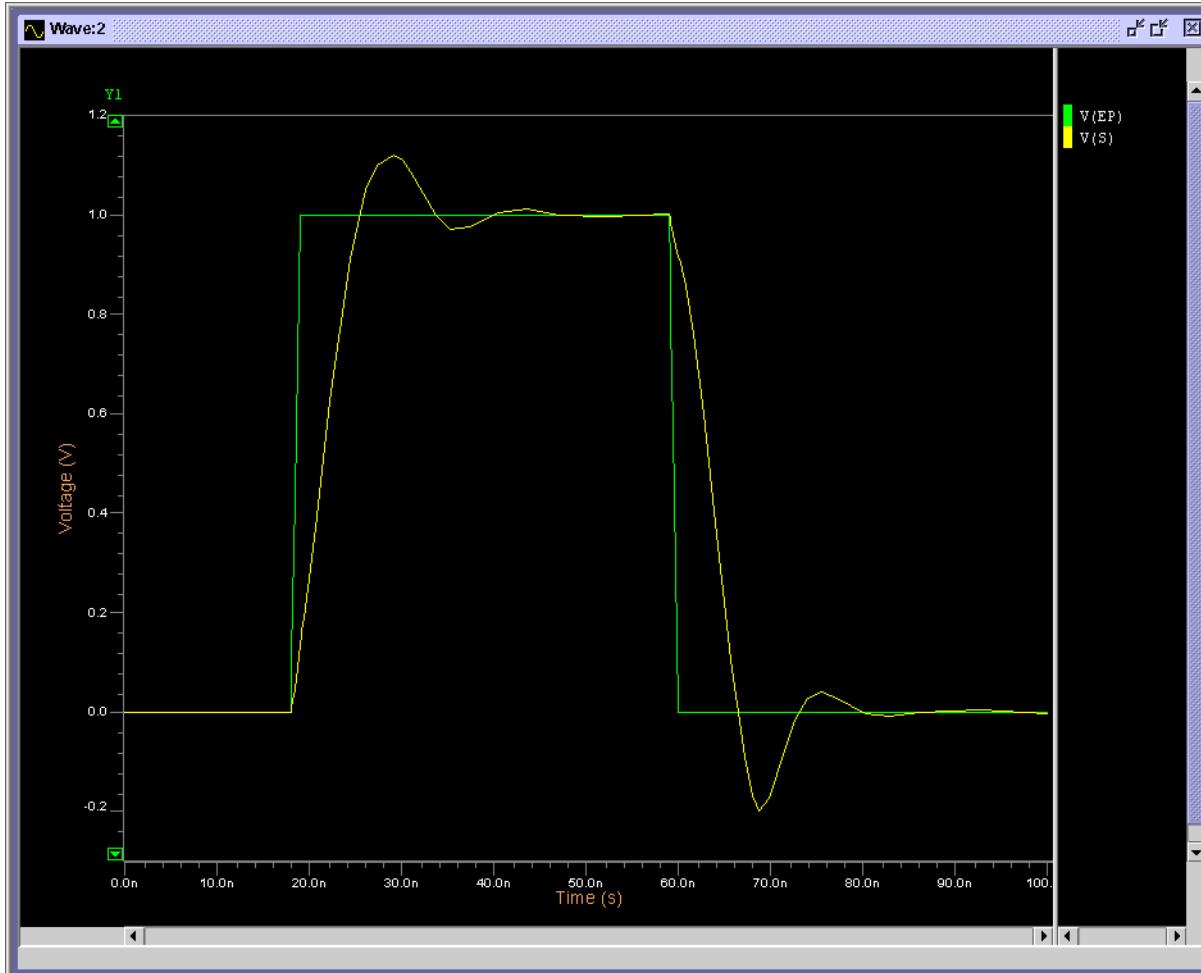
```
+ dphif=0.7 v1=1.9e+5 kw=0.52u kl=4u gw=0.453u gl=0.7u dinf=0.17
+ ve=7.35e+4 ldif=10u cdifs0=0.000316 cdifp0=0
+ dw=0 dl=1.5u rec=0.5u vt0=-0.55 kb=0.34 tg=0.14

*amplifier
m1 a g vdd vdd mod2 w=120u l=5.5u
m2 b g vdd vdd mod2 w=120u l=5.5u
m3 d k a vdd mod2 w=116u l=3.5u
m4 s k b vdd mod2 w=116u l=3.5u
m5 c i vss vss mod1 w=63u l=6u
m6 a ep c vss mod1 w=130u l=4u
m7 b s c vss mod1 w=130u l=4u
m8 d d ff vss mod1 w=5.5u l=4.5u
m9 s d e vss mod1 w=5.5u l=4.5u
m10 ff e vss vss mod1 w=42u l=4u
m11 e e vss vss mod1 w=42u l=4u
m12 g g vdd vdd mod2 w=14.5u l=5.5u
m13 g g h vss mod1 w=9u l=5.5u
m14 i i h vdd mod2 w=19u l=4.5u
m15 i i vss vss mod1 w=6u l=6u
m16 j g vdd vdd mod2 w=20u l=5.5u
m17 j j k vss mod1 w=26u l=3.5u
m18 nl i k vdd mod2 w=3u l=3.5u
m19 nl nl vss vss mod1 w=4u l=3.5u
c1 s 0 1.5p

v1 vdd 0 3
v2 vss 0 -3
v3 ep 0 pwl(0n 0 18n 0 19n 1 59n 1 60n 0)
.tran 1n 100n
.options eps=1.0e-6
.plot tran v(ep) v(s) (-0.3,1.2)
.end
```

## Simulation Results

**Figure C-5. Example#4—Simulation Results**



## Example#5—5th Order Elliptic SC Low Pass Filter

This example deals with a transient analysis on a 5th order SC low pass filter. This type of circuit is used in those applications requiring the analysis of sampled data in analog circuits, being used in certain ADC/DAC and switch capacitor filters designs. The complete netlist can be found in the file *ellipt5.cir*.

### Complete Netlist

```
ellipt5
.model ampop opa level=2 voff=0 sl=50e06 cin=0p
+ rs=1 vsat=5 gain=10000 fc=5e3 cmrr=0
```

## Examples

### Example#5—5th Order Elliptic SC Low Pass Filter

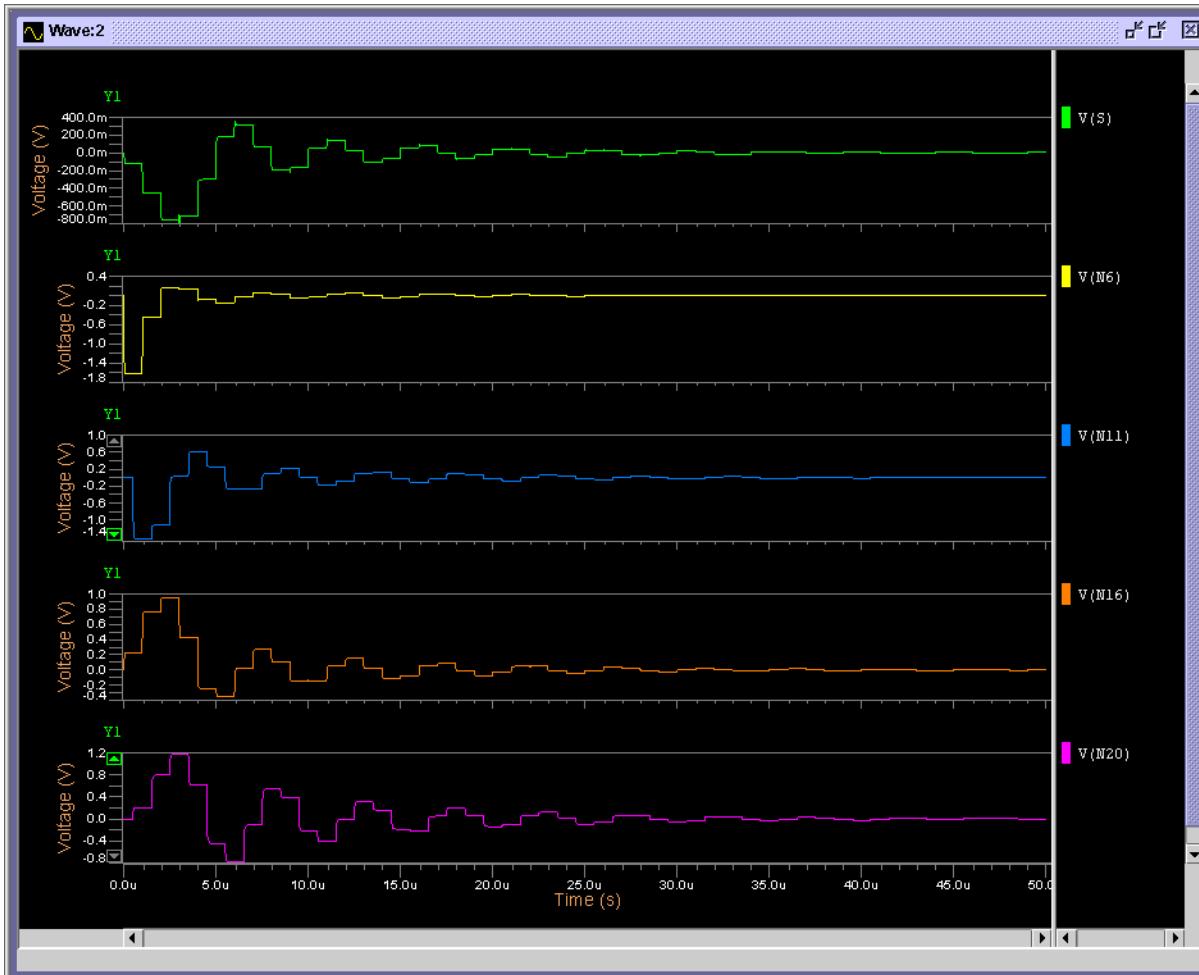
---

```
opa1    0    n5    n6    0    ampop
opa2    0    n10   n11   0    ampop
opa3    0    n15   n16   0    ampop
opa4    0    n21   n20   0    ampop
opa5    0    n25   s     0    ampop
sint1   ph1  e     n1    1.0k  0.0p
sint2   ph1  n11  n3    1.0k  0.0p
sint3   ph1  n4    n6    1.0k  0.0p
sint4   ph1  n2    n5    1.0k  0.0p
sint5   ph1  n6    n7    1.0k  0.0p
sint6   ph1  n16  n9    1.0k  0.0p
sint7   ph1  n8    0     1.0k  0.0p
sint8   ph1  n11  n12   1.0k  0.0p
sint9   ph1  n20  n13   1.0k  0.0p
sint10  ph1  n14  n15   1.0k  0.0p
sint11  ph1  n16  n17   1.0k  0.0p
sint12  ph1  s     n18   1.0k  0.0p
sint13  ph1  n19   0     1.0k  0.0p
sint14  ph1  n20  n22   1.0k  0.0p
sint15  ph1  S     n23   1.0k  0.0p
sint16  ph1  n24  n25   1.0k  0.0p
sint17  ph2  n1    0     1.0k  0.0p
sint18  ph2  n3    0     1.0k  0.0p
sint19  ph2  n4    0     1.0k  0.0p
sint20  ph2  n2    0     1.0k  0.0p
sint21  ph2  n7    0     1.0k  0.0p
sint22  ph2  n9    0     1.0k  0.0p
sint23  ph2  n8    n10   1.0k  0.0p
sint24  ph2  n12   0     1.0k  0.0p
sint25  ph2  n13   0     1.0k  0.0p
sint26  ph2  n14   0     1.0k  0.0p
sint27  ph2  n17   0     1.0k  0.0p
sint28  ph2  n18   0     1.0k  0.0p
sint29  ph2  n19  n21   1.0k  0.0p
sint30  ph2  n22   0     1.0k  0.0p
sint31  ph2  n23   0     1.0k  0.0p
sint32  ph2  n24   0     1.0k  0.0p
```

```
c1 n6 n5 1.50536963p
c2 n6 n15 0.2589059p
c3 n4 n2 1.0p
c4 n3 n2 1.0p
c5 n1 n2 1.0p
c6 n10 n11 0.96577222p
c7 n7 n8 1.0p
c8 n9 n8 1.0p
c9 n16 n15 2.3362151p
c10 n16 n5 0.2589059p
c11 n16 n25 0.94246071p
c12 n13 n14 1.0p
c13 n12 n14 1.0p
c14 n20 n21 0.51354696p
c15 n18 n19 1.0p
c16 n17 n19 1.0p
c17 s n25 0.75440758p
c18 s n15 0.94246071p
c19 n22 n24 1.0p
c20 n23 n24 1.0p
.chrent e 0n 4 700n 4 800n 0 f
.chrent phi 0.0n -5.0 10.0n 5.0 490.0n 5.0
+ 500.0n -5.0 1000.0n -5 p
.chrent ph2 0.0n -5.0 500.0n -5.0 510.0n 5.0
+ 990.0n 5.0 +1000.0n -5 p
.plot tran v(s) (-0.8, 0.4)
.plot tran v(n6) (-1.8, 0.4)
.plot tran v(n11) (-1.5, 1.0)
.plot tran v(n16) (-0.4, 1.0)
.plot tran v(n20) (-0.8, 1.2)
.tran 1u 50u uic
.option eps=le-4 hmin=5n
.end
```

## Simulation Results

**Figure C-6. Example#5—Simulation Results**



## Example#6—Charge Control in MOS 4 & 6

This example deals with the transient analysis of a MOS circuit containing both MOS level 4 and 6 models, illustrating the differences in model performance. The complete netlist can be found in the file *niv4\_6.cir*.

### Complete Netlist

```

niv4_6
.model m4 nmos level=4 vt0=1.2v eox=25n uo=600
+ nsub=2.0e16 phi=0.6 vmax=2.0e5 kw=2.24u kl=2.24u
+ gw=3.91u gl=0.7u dinf=0.1 kb=0.1 ve=1.0e4
+ ldif=10u cj=0.0001 cjsw=0 dw=0 dl=0.8u rec=0.15u
+ tg=0.06

```

```

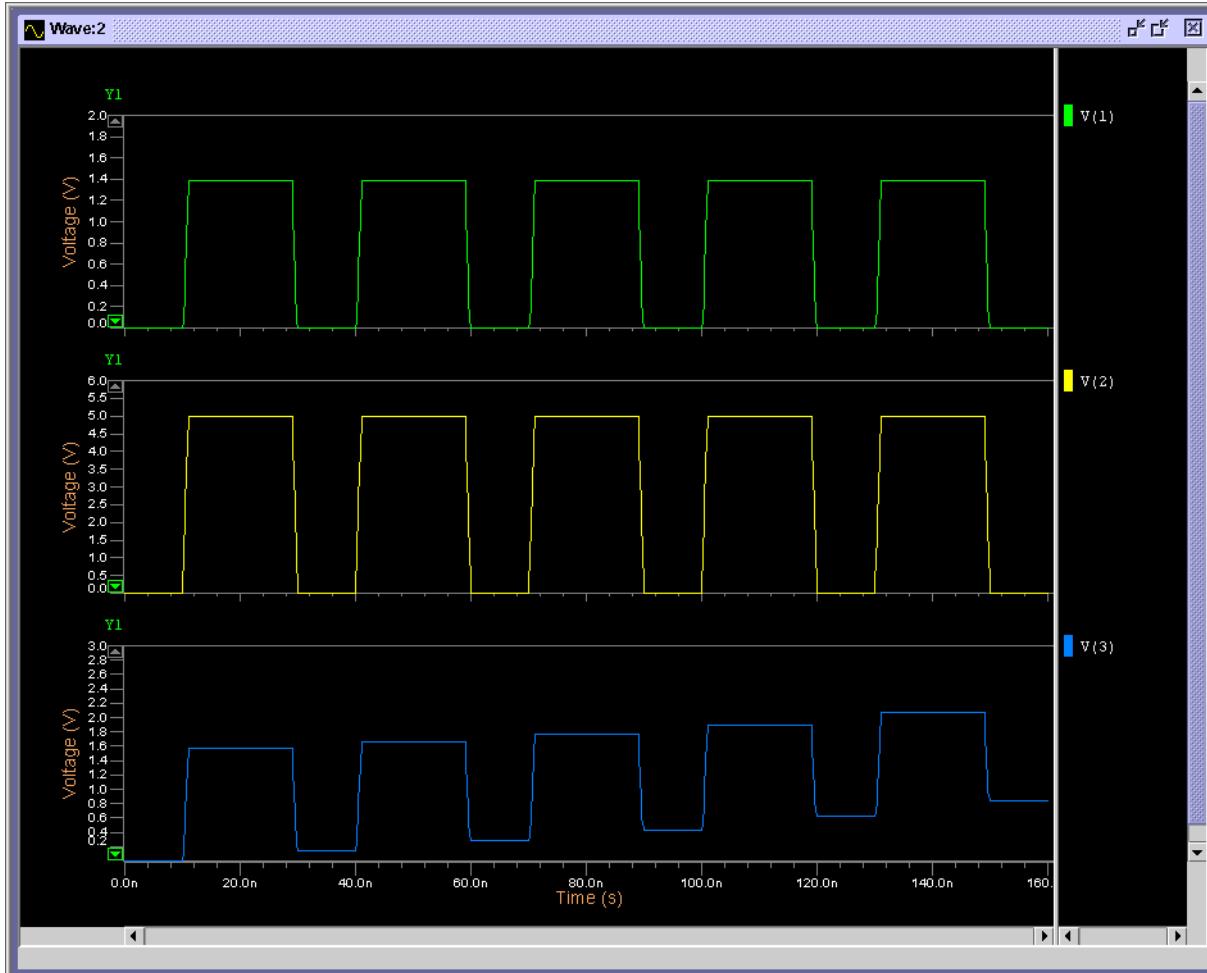
.model m2 nmos level=6 vt0=1.1v eox=25n uo=600
+ nsub=2.0e16 phi=0.6 vmax=2.0e5 kw=2.24u kl=2.24u
+ gw=3.91u gl=0.7u dinf=0.1 kb=0.1 ve=1.0e4
+ ldif=10u cj=0.0001 cjsw=0 dw=0 dl=0.8u rec=0.15u
+ tg=0.06
m1 1 2 1 bulk m4 w=10u l=3u
c1 1 0 0.05pf
m2 3 2 3 bulk m2 w=10u l=3u
c2 3 0 0.05pf
vb bulk 0 0
vin 2 0 pulse (0 5 10n 1n 1n 18n 30n)
.tran 1ns 160ns uic
.print tran v(1) v(2) v(3)
.plot tran v(1) (0, 2)
.plot tran v(2) (0, 6)
.plot tran v(3) (0, 3)
.option eps=1.0e-6
.end

```

In the simulation below, V(1) is the result of the charge control model and V(3) is the result of the capacitive model. For V(3) the error on the charge is accumulated and error is increasing at each period. However, for V(1) there is no error on the charges.

## Simulation Results

**Figure C-7. Example#6—Simulation Results**



## Example#7—Active RC Band Pass Filter

This example deals with the AC analysis of an active RC band pass filter circuit containing a UA741 operational amplifier. The complete netlist can be found in the file *ua741.cir*.

### Complete Netlist

```

ua741
* .MODEL definitions
.model npn npn bf=160 rb=100 cjs=2p tf=0.3n
+ tr=6n cje=3p cjc=2p vaf=100
.model npq npn bf=160 rb=100 cjs=2p tf=0.3n
+ tr=6n cje=3p cjc=2p vaf=100 is=2p
.model pnp pnp bf=20 rb=20 tf=1n tr=20n
+ cje=6p cjc=4p vaf=100

```

```
.model pnq pnp bf=20 rb=20 tf=ln tr=20n
+ cje=6p cjc=4p vaf=100 is=2p
*subcircuit ua741
.subckt ua741 2 3 6 4 7
r1 1 4 1k
r2 15 4 50k
r3 5 4 1k
r4 17 4 5k
r5 18 16 39k
r6 22 23 4.5k
r7 20 23 7.5k
r8 21 4 50k
r9 19 4 50
r10 24 6 25
r11 6 25 50
cx 22 14 30p
q1 9 3 10 npn
q2 12 13 10 pnp
q3 9 2 11 npn
q4 14 13 11 pnp
q5 12 15 1 npn
q6 14 15 5 npn
q7 7 12 15 npn
q8 9 9 7 pnp
q9 13 9 7 pnp
q10 13 16 17 npn
q11 16 16 4 npn
q12 18 18 7 pnp
q13 14 19 4 npn
q14 20 14 21 npn
q15 22 18 7 pnp
q16 22 23 20 npn
q17 20 21 19 npn
q18 22 24 6 npn
q19 7 22 24 npq
q20 4 20 25 pnq
q21 6 6 23 npn
.ends
r1 1 3 12.952k
r22 3 0 846.01
r2 4 5 322.2k
c1 3 5 100n
c2 3 4 100n
x1 4 0 5 40 70 ua741
r3 5 6 150k
r4 6 7 293.8k
r5 7 8 15k
r6 8 9 15k
r7 9 10 15k
r8 11 6 14.05k
c3 6 7 100n
c4 10 11 100n
```

## Examples

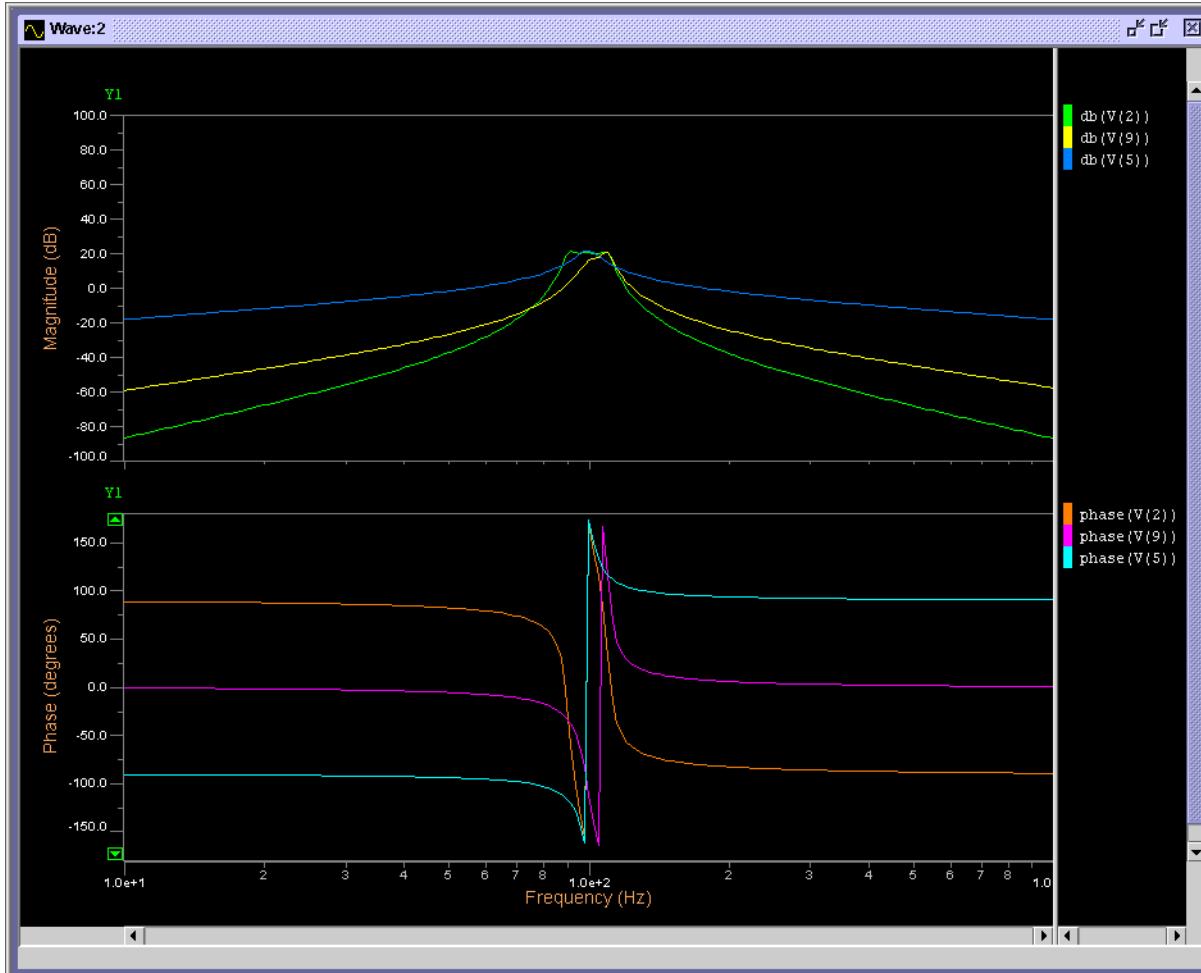
### Example#7—Active RC Band Pass Filter

---

```
x2 6 0 7 40 70 ua741
x3 8 0 9 40 70 ua741
x4 10 0 11 40 70 ua741
r9 9 12 47k
r10 12 13 365.6k
r11 13 14 15k
r12 14 2 15k
r13 2 15 15k
r14 16 12 20.71k
c5 12 13 100n
c6 15 16 100n
x5 12 0 13 40 70 ua741
x6 14 0 2 40 70 ua741
x7 15 0 16 40 70 ua741
vee 40 0 dc -15
vcc 70 0 dc 15
v0 1 0 ac 1
.ac dec 100 10 1000
.plot ac vdb(2) vdb(9) vdb(5) (-100, 100)
.plot ac vp(2) vp(9) vp(5) (180, -180)
.end
```

## Simulation Results

**Figure C-8. Example#7—Simulation Results**



## Example#8—2nd Order Delta Sigma Modulator

This example deals with the transient analysis of a second order delta sigma modulator circuit. The complete netlist can be found in the file *integrator.cir*.

### Complete Netlist

```

integrator
.model switch nsw CREC = 0.0 ron = 100
***** integrator 1 *****
s1 phi    inp 1   switch
s2 phib   1     0   switch
s1 1      2     2p
s3 phi    2     0   switch
s4 phib   2     ep  switch

```

## Examples

### Example#8—2nd Order Delta Sigma Modulator

---

```
s5 phi inm 3 switch
s6 phib 3 0 switch
c2 3 4 2p
s7 phi 4 0 switch
s8 phib 4 em switch
s10 phi ref 5 switch
s11 phib ref 6 switch
s12 phib 5 0 switch
s13 phi 6 0 switch
c3 5 7 2p
c4 6 8 2p
s14 phi 7 0 switch
s15 phi 8 0 switch
s16 phib 7 9 switch
s17 phib 8 10 switch
s18 comp 9 em switch
s19 compb 9 ep switch
s20 comp 10 ep switch
s21 compb 10 em switch
c5 ep sm 6p
c6 em sp 6p
eopal sm sp ep em -1meg
***** integrator 2 *****
s11 phi sm i1 switch
s12 phib i1 0 switch
c11 i1 i2 2p
s13 phib i2 0 switch
s14 phi i2 iep switch
s15 phi sp i3 switch
s16 phib i3 0 switch
c12 i3 i4 2p
s17 phib i4 0 switch
s18 phi i4 iem switch
s110 phi ref i5 switch
s111 phib ref i6 switch
s112 phib i5 0 switch
s113 phi i6 0 switch
c13 i5 i7 2p
c14 i6 i8 2p
s114 phib i7 0 switch
s115 phib i8 0 switch
s116 phi i7 i9 switch
s117 phi i8 i10 switch
s118 comp i9 iem switch
s119 compb i9 iep switch
s120 comp i10 iep switch
s121 compb i10 iem switch
c15 iep ism 6p
c16 iem isp 6p
eopa2 ism isp iep iem -1meg
vref ref 0 -1
***** comparator *****
compdiff ism isp voutp voutn vhi=2.5 vlo=-2.5 tcom=0.0n
+ tpd=0.0n
```

```
***** latch *****
s11 phi voutn memn switch
s12 phi voutp memp switch
nand1 memp qu qubar vhi=2.5 vlo=-2.5 tpd=0.0n
+ vthi=0.1 vtlo=-0.1
nand2 memn qubar qu vhi=2.5 vlo=-2.5 tpd=0.0n
+ vthi=0.1 vtlo=-0.1
***** delay *****
delayn qubar compb 81.3802n
delayp qu comp 81.3802n
***** voltage source input *****
vin1 inp 0 sin(0 0.5 12.288k 0 0)
vin2 inm 0 sin(0 -0.5 12.288k 0 0)
.chrent phib 0 -5 41.6901n -5 42.6901n 5 79.3802n 5
+ 80.3802n -5 81.3802n -5 P
.chrent phi 0 5 40.6901n 5 41.6901n -5
+ 80.3802n -5 81.3802n 5 P
.tran 81.3802n 81.3802us
.option eps=1.0e-9 step=5.0862625n be
.plot tran v(inp,inm) (-1,1)
.plot tran v(sm,sp) (-3.0,3.0)
.plot tran v(isp,ism) (-3.5,3.5)
.plot tran v(comp) (-3.0,3.0)
.end
```

## Simulation Results

**Figure C-9. Example#8—Simulation Results**



## Example#9—Operational Amplifier

This example deals with the transient analysis of an operational amplifier circuit. In addition, the slew rate and circuit settling time with a 5% error band are calculated and written to the ASCII output file. The complete netlist can be found in the file *extract.cir*.

### Complete Netlist

```

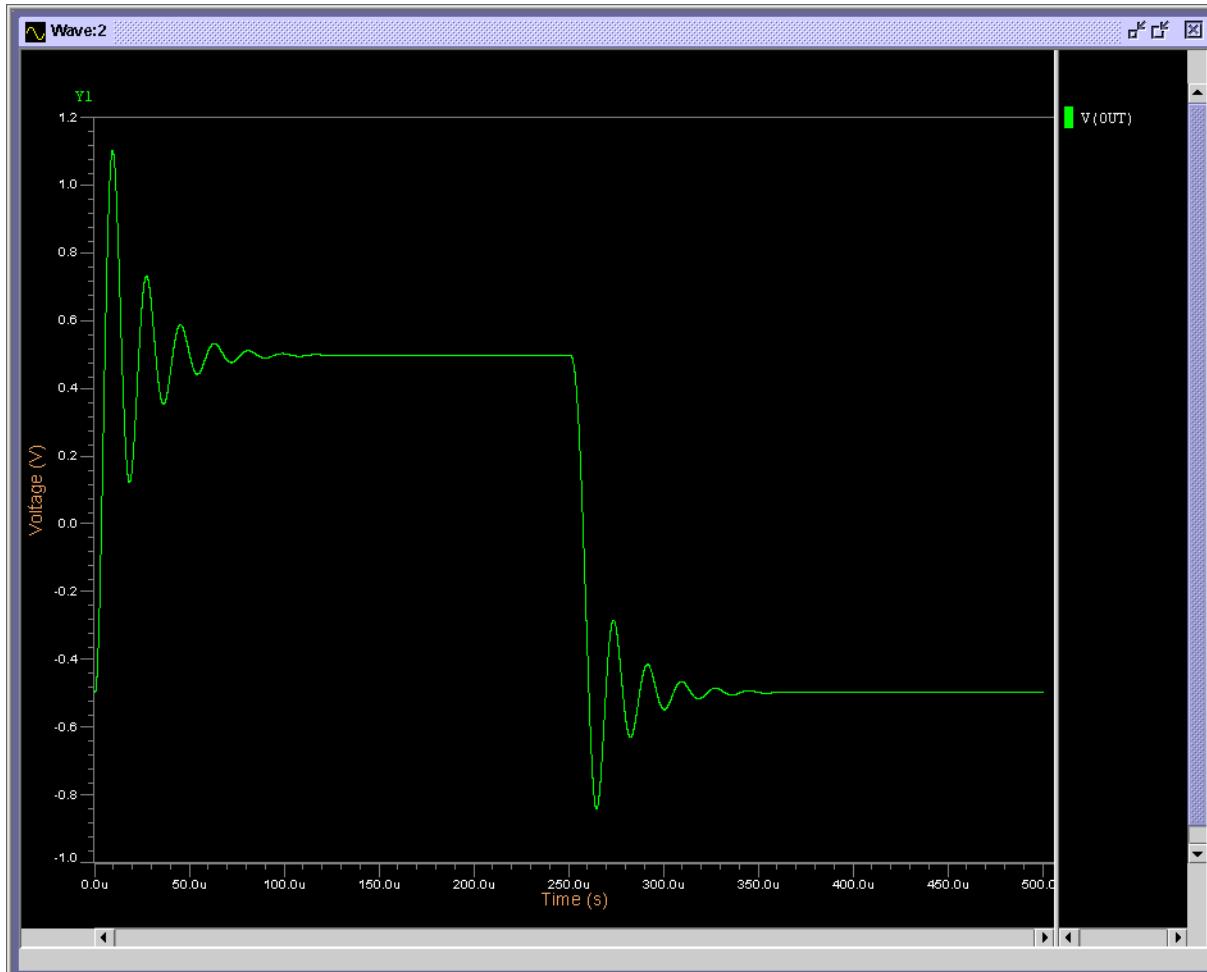
OPERATIONAL AMPLIFIER
.model bu opa level=2 sl=1e6
c1 out 0 10u
opal nonin in out 0 bu
rf in out 2.5k
ri 0 in 0.625k
v1 nonin 0
+ pwl (0.0 -0.1 0.1u 0.1 250u 0.1 260u -0.1 500u -0.1)

```

```
.tran 100u 500u
.option vntol=1e-8 reltol=1e-8 hmax=1u eps=1e-8
.plot tran v(out)
.defmac sett(x,y,tix,limit)=
+ (xycond(x,(y > (yval(y,tix)*(1.0+limit)))) ||
+ (y < (yval(y,tix)*(1.0-limit))),tix,0.0))
.defmac slewrate(a)=
+ ((slope(a,yval(a,0.0)+(max(a)-yval(a,0.0))/3.0) +
+ slope(a,yval(a,0.0)+(max(a)-yval(a,0.0))*2.0/3.0))/2.0)
.extract $sett(xaxis,v(out),200e-6,0.05)
.extract $slewrate(v(out))
.end
```

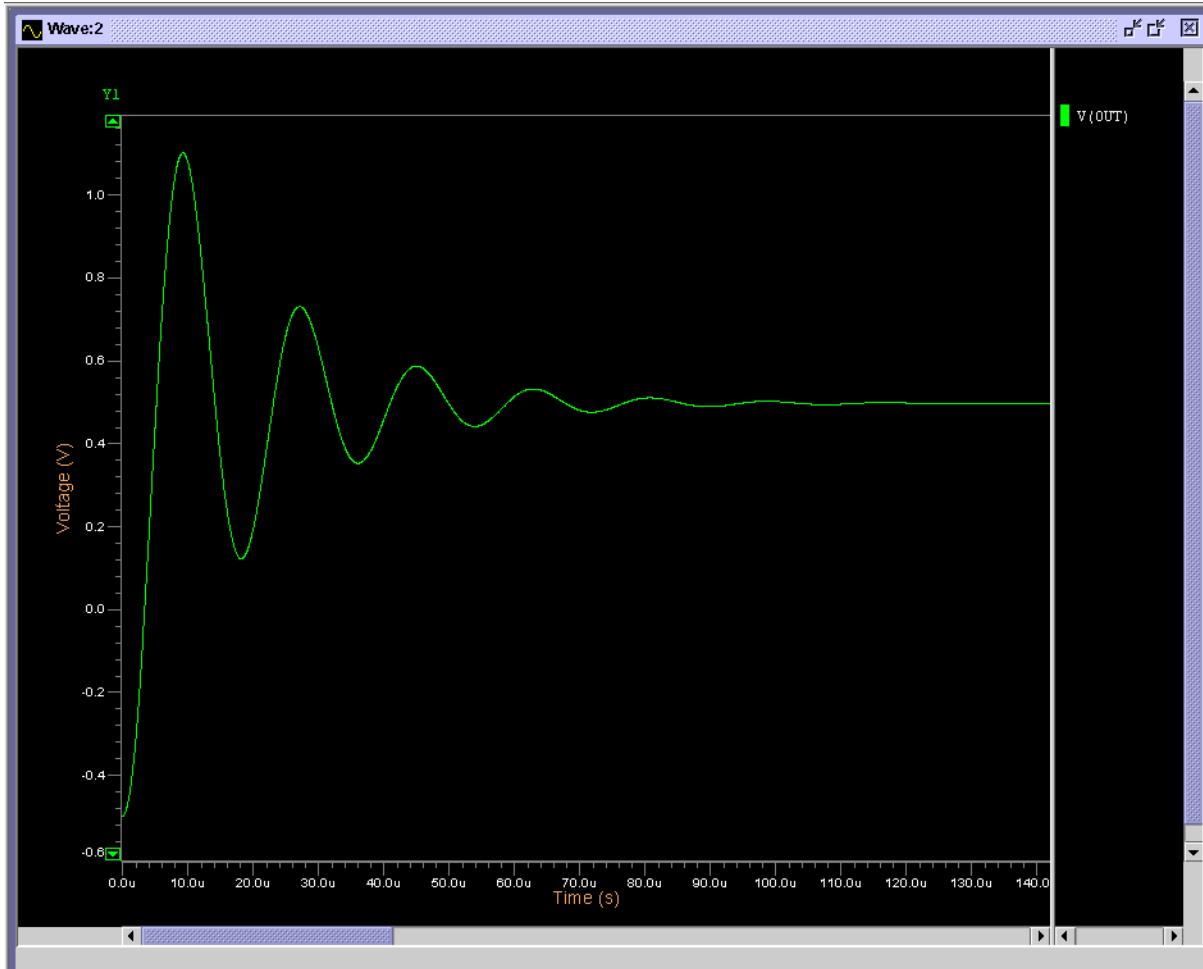
## Simulation Results—1

Figure C-10. Example#9—Simulation Results—1



## Simulation Results—2 (Zoom)

Figure C-11. Example#9—Simulation Results—2 (Zoom)



# Appendix D

## Eldo Interactive Mode

---

### Introduction

Eldo Interactive is a way of invoking Eldo and sending commands to it interactively instead of sending the commands in the netlist. SimPilot makes use of Eldo in the interactive mode, although this is all transparent to the SimPilot user.

To invoke Eldo in the interactive mode, type:

```
eldo cir_file_name -inter
```

where `cir_file_name` is the name of the `.cir` control file to be simulated. Default extension is `.cir`.

When working in Eldo interactive mode, a prompt will appear:

```
eldo>
```

There is some help information available: you can type `help` at the `eldo>` prompt and you will see the list of available commands; type `help <command_name>` for more information about a particular command, or type `help all` to obtain the complete help listing.

When working in Eldo interactive mode, you can type your command after the `eldo>` prompt. For continuation lines, you must type the backslash character (\) at the end of a line.

Commands can be read from a file specified via either of the following:

- `.eil file_name` in the input file
- `-eil file_name` at invocation of the executable

The `eldo>` prompt appears before the first simulation, unless:

- option `DOSIMCIR` is specified
- `-DOSIMCIR` used at invocation of the executable

To control simulation, use the following commands:

```
QUIT
LOAD
RUN
GO
```

**CONT**  
**NEXT**

To check netlist elements, use the following commands:

**DISPLAY**  
**LIST**

To handle simulation information, use the following commands:

**STATUS**  
**PRINT**  
**DELETE**  
**RESET**  
**OPTRESET**

To handle breakpoints, use the following commands:

**STOP IF** (*expression*)  
**STATUS BREAK** *n*  
**DELETE** <*index*>|**ALL**

To alter simulation conditions, use the following Eldo/Spice commands:

**.TRAN**  
**.DC**  
**.AC**  
**.OP**  
**.STEP**  
**.TEMP**  
**.ALTER**  
**.NOISETRAN**

To change stimuli, re-instantiate the device with new values:

**Vxx** *n1 n2 <new\_stimuli>*

To change element/model parameters, or parameters, use:

**SET**

Use the command, **eilout** *file\_name*, to redirect the outputs generated by specific Eldo interactive commands to the specified file. Errors are still displayed on the terminal window.

Use the command, **eilout stdout**, to switch back to the default mode, where information is sent to the terminal window.

## Shared Commands

This is a generic section listing commands available in both interactive and normal Eldo mode.

**SETBUS** see “[.SETBUS](#)” on page 10-282.

**SIGBUS** see “[.SIGBUS](#)” on page 10-291.

## To Read Information

|                       |                                                    |
|-----------------------|----------------------------------------------------|
| <b>STATUS [BREAK]</b> | shows breakpoints                                  |
| <b>STATUS ANAL</b>    | shows status of the current simulation             |
| <b>STATUS AC</b>      | shows the <b>AC</b> simulation command and stimuli |
| <b>STATUS TRAN</b>    | shows the <b>TRAN</b> simulation command           |
| <b>STATUS DC</b>      | shows the <b>DC</b> simulation command             |
| <b>STATUS MC</b>      | shows the <b>MC</b> simulation command             |
| <b>STATUS PZ</b>      | shows the <b>PZ</b> simulation command             |
| <b>STATUS EXTRACT</b> | shows the expressions to extract                   |
| <b>STATUS PLOT</b>    | shows the <b>.PLOT</b> commands typed              |
| <b>STATUS PROBE</b>   | shows the <b>.PROBE</b> commands typed             |
| <b>STATUS PRINT</b>   | shows the <b>.PRINT</b> commands typed             |
| <b>STATUS PARAM</b>   | shows the parameters ( <b>.PARAM</b> )             |
| <b>STATUS INPUT</b>   | shows the input stimuli                            |
| <b>STATUS SAVE</b>    | shows the <b>.SAVE</b> commands issued             |
| <b>STATUS RESTART</b> | shows the <b>.RESTART</b> commands issued          |
| <b>STATUS MCMOD</b>   | shows the MC specifications on models              |
| <b>STATUS MCPARAM</b> | shows the MC specifications on parameters          |
| <b>STATUS STAT</b>    | shows the statistics at runtime                    |

## PRINT <expression>

The **PRINT** command can be issued to display:

1. Voltage or current (like plot command)
2. Values from the extract-command language
3. **EXTRACT** index  
where index is the key returned by **STATUS EXTRACT**
4. Values of Device/Models; syntax is:

```
PRINT E (<device_name>[ ,W/L/AD/AS/PD/PS/NRD/NRS/AREA] )
PRINT M (<model_name>,<parameter_name> )
PRINT P (<param_name> )
```

## 5. OPTION <name>

### Examples

```
PRINT V(S)
PRINT V(S) + V(SS)
PRINT TPD(E,S)
```

Please note that the extract information issued through a **PRINT** command work only if the corresponding **.EXTRACT** command had been specified before running the simulation, or if Eldo is able to extract the information from the binary output file (**.PLOT** available, and general-purpose extraction language used).

## DISPLAY E string[\*]

|                         |                                                                           |
|-------------------------|---------------------------------------------------------------------------|
| <b>LIST string*</b>     | List all elements whose name begins with <i>string</i>                    |
| <b>LSMOD string*</b>    | List all models whose name begins with <i>string</i>                      |
| <b>LSMODEV string*</b>  | List all elements which have <i>string*</i> as model                      |
| <b>LSSUB string*</b>    | List all subcircuits whose name begins with <i>string</i>                 |
| <b>LSSUBDEV string*</b> | List all instances of the subcircuit whose name begins with <i>string</i> |

## To Reset Several Features

**RESET [PRINT | PLOT | PROBE | IC | NODESET | GUESS | EXTRACT]**  
Used to remove a set of commands.

**RESET FILES**  
Rewind output files *.cou* and *.chi*.

## DC Control Options

```
GMIN
NMAXSIZE
ITL1
GRAMP
NETSIZE
VMIN
VMAX
```

## Accuracy Control Options

```
ITOL
EPS
VNTOL
RELTOL
```

RELERR  
PIVREL  
PIVTOL  
ABSTOL  
FLXTOL  
MAXORD

## Time-step Control Options

ZOOMTIME  
STEP  
STARTSMP  
FREQSMP  
OUT\_RESOL  
TRTOL  
HMIN  
ITL3  
ITL4  
FT  
DCLOG  
LVLTIM  
LVLCNV  
DVDT  
RELVAR  
ABSVAR  
SAMPLE  
HMAX  
SPICDC  
NOSPICDC

Options which can be set, but not reset:

SPIOUT  
NEWTON  
OSR  
TRAP  
GEAR  
BE  
PROBEOP  
NOLAT  
NWLAT  
ANALOG  
BBDEBUG  
NOSIZECHK  
QTRUNC  
UNBOUND  
LCAPOP

## Change Features

### SET

The `SET` command is used to change device geometries or models parameters.

## Syntax

```
SET E (<element_name> [,W/L/AD/AS/PD/PS/NRD/NRS/]) [=] <value>
SET M (<model_name>,<parameter_name>) [=] <value>
SET P (<parameter_name>) [=] <value>
```

Some Eldo commands are available from interactive mode; they are:

```
.TRAN .AC .DC .STEP .TEMP
.PLOT .PRINT .PROBE .EXTRACT .OPTION
.NODESET .IC .GUESS
.SAVE .RESTART .USE .LIB
```

It is possible to change the stimuli in interactive mode. The complete line must be re-entered with new stimuli.

## Example

```
V1 1 2 PWL (0 0 20n 0 30n 5)
```

The program checks the component name, and applies the new stimuli.

## DELETE

```
DELETE BREAK index
DELETE BREAK ALL
DELETE <cmd> index
```

**DELETE** is used to remove breakpoints used to stop Eldo at run time. The third syntax is used to remove the command corresponding to the index returned by the command **STATUS <cmd>**.

## DISABLE

```
DISABLE BREAK index
DISABLE BREAK ALL
```

**DISABLE** is used to remove breakpoints used to stop Eldo at run time. Breakpoints can be enabled back using command **ENABLE**.

## ENABLE

```
ENABLE BREAK index
ENABLE BREAK ALL
```

**ENABLE** is used to re-activate breakpoints used to stop Eldo at run time, whenever these breakpoints have been disabled by **DISABLE**.

## CHMOD

```
CHMOD <model1> <model2>
```

Replace the model `<model1>` by the model `<model2>` for all the devices using `<model1>` (can be used for M, Q, D, J models).

## FORCE

```
FORCE <node_name>=<value>
```

Force the node `node_name` to `value` after **RISE\_TIME** or **FALL\_TIME** depending on current value (see further information below). Used to impose a voltage on a node. If multiple `FORCE` are applied on the same node, the last command is used. If an input signal was applied on the node it will be ignored. The voltage imposed by `FORCE` can be removed using `RELEASE`.

## HIGH

```
HIGH <node_name>
```

Force the node `node_name` to **HIGHVOLTAGE** after **RISE\_TIME** (see further information below).

## LOW

```
LOW <node_name>
```

Force the node `node_name` to **LOWVOLTAGE** after **FALL\_TIME** (see further information below).



The previous three commands contain the parameters, **HIGHVOLTAGE**, **LOWVOLTAGE**, **RISE\_TIME** and **FALL\_TIME**. These parameters can be set using the `.OPTION` command. For more information see [page 11-24](#).

---

## PWL

```
PWL <node_name> t1 v1 [t2 v2] ...
```

Force a PWL on the specified node `<node_name>` during transient simulation. `tn` are times relative to the current time. `vn` are the source values at `tn`.

## RELEASE

```
RELEASE <node_name> ...
```

Release force on node. If a signal is present on node <node\_name>, and if no **FORCE** command had been applied on the node, then the signal is disabled (node will be computed by Eldo as any other node).

## TRANSITION

```
TRANSITION <node_name> TT VALUE [DELAY]
```

**TT** Transition Time

**VALUE** Value of the signal after the transition

**DELAY** Time delay before the transition starts

Overwrite the signal which was on node <node\_name> by a PWL:

```
Vxx <node_name> 0 PWL (<current_time> <current_value> <current_time +  
DELAY> <current_value> <TT + DELAY> <VALUE>)
```

## SETBUS

```
SETBUS <hierarchical_name>[[:<msb>:<lsb>]] <type> <value>
```

Set a value on a bus. The **SETBUS** command sets a value on the specified bus. Values may be specified as binary, octal, hexadecimal, or decimal. The bus bit accepts the standard digital values 1, 0, X and Z.

The command automatically sets all signals in a bus if the subscripted values are omitted. The default bit order is the European style D<sub>0</sub>...D<sub>N-1</sub>. Therefore, for a hierarchical base name that defines an 8-bit bus, the command:

```
setbus Xcircuit.port[0:7] b00001111
```

is equivalent to:

```
setbus Xcircuit.port b00001111
```

To use the American style D<sub>N-1</sub>...D<sub>0</sub> bit order, you must specify the range of bits explicitly:

```
setbus Xcircuit.port[7:0] b00001111
```

The **SETBUS** command ignores the most significant portion of values that are wider than the specified bus. Therefore, if you set a value of hexadecimal 8F to a 12 bit bus, the bus receives the value 0F.

---

### Note



This **SETBUS** syntax is different to the Eldo **.SETBUS** syntax used in the netlist *.cir* file. However, if you want to use the netlist syntax in interactive mode, specify the Eldo interactive command:

```
eldo>LSIM OFF
```

---

## BUS

```
BUS <name> <type> <list_of_signals>
```

Define a bus. The **BUS** command defines a bus with several different nodes by grouping them together.

**<name>** Name of the new bus

**<type>** Ignored (kept for Lsim compatibility)

**<list\_of\_signals>**

List of nodes composing the bus (msb...lsb).

### Example

```
bus foo x A[0] A[1] A[2] A[3] clk reset
bus foo x A[0:3] clk reset
bus ADD  x A0 A1 A2 A3 A4 A5
```

## To Control Execution

### LOAD <filename>

This command stops the current simulation, and causes Eldo to load the file *<filename>*. Extension *.cir* is assumed.

### SAVESIM <filename>

Eldo creates three files:

1. *filename.eil*

This file contains all the commands typed since the loading of the circuit file; it is possible to re-execute this set of functions by:

```
<eldo> -i circuit -eil filename.eil
```

2. *filename.chi*

This file contains the ascii output

3. *filename.cou*

This file is the binary output file readable by graphic-postprocessors

---

**Note**

 A **.pz** file might be created if a **.pz** card exists.

---

The files *circuit.cou* and *circuit.chi* are then reset.

## STOP IF <condition>

Set breakpoints in interactive mode. **<condition>** can refer to the **SWEEP** value, and/or any valid plot command.

## Examples

```
STOP IF (SWEEP > 1n)
STOP IF ((SWEEP > 10n) && (V(S) > 2.5))
```

## STOP SIMU

Stop the current simulation; Eldo is then ready to run a new simulation.

## RUN

Restart the simulation.

**RUN FOR <value>** Run until the sweep value has changed of **<value>**

**RUN UNTIL <value>** Run until the sweep value reaches **<value>**

---

**Note**

 **<value>** can be a parameter, this must be specified on the **.PARAM** command in the netlist.

---

## NEXT [SIMU]

**NEXT**

Eldo will stop at the next sweep value.

**NEXT SIMU**

Eldo will stop after the next simulation if **.TEMP** or **.STEP** command are found.

## CONT

**CONT**

Forces Eldo to continue the simulation(s) until breakpoints are encountered, or the requested number of simulation is completed.

## QUIT

**QUIT**

Quit the application.

**QUIT SIMU**

Stop current analysis.

## CONNECT XELGA

Used to connect Eldo to Xelga in Eldo standalone mode.

## VIEW plot\_name

## UNVIEW plot\_name

Used for adding or removing signals whenever Xelga is connected to Eldo, running in interactive mode.



# Appendix E Eldo Utilities

---

## Utility to Convert .chi to .cir

A utility **chi2cir** converts an Eldo ASCII output (*.chi*) file into a netlist (*.cir*) file. The resulting netlist file is independent of any libraries (no **.LIB/.INCLUDE** required), and all information required for another simulation is inside the netlist file (providing that there were no commands such as **.NOTRC** preventing the writing of the circuit description in the *.chi* file).

This could be useful to exchange testcases between user's avoiding library dependence and would also simplify the creation of different tests on the same circuit.

Once the circuit has been run once (to generate the *.chi* file), run this **chi2cir** utility to generate a testcase from this *.chi* output in order to modify some parameters/options to re-run other simulations.

### Usage

```
chi2cir [chifile cirfile]  
  
chifile      Input filename, the Eldo ASCII output (.chi) file.  
cirfile      Output filename for the resulting netlist (.cir) file.
```

If both options are not specified, the utility will prompt you for filenames.

### Example

```
chi2cir trigger.chi my_trigger1.cir
```

Once the original *trigger.cir* circuit had been run once (to generate the *.chi* file), running the **chi2cir** utility on the *trigger.chi* ASCII output file generates a testcase file *my\_trigger1.cir*. Some parameters/options can then be modified in this testcase to re-run other simulations.

### Notes

No provision is made for **.ALTER** statements for multiple simulation runs, only the first simulation is taken into account.

## Eldo Encryption

`encrypt_eldo` is the tool to encrypt the libraries containing **.SUBCKT** definitions, **.MODEL/ .PARAM** cards, and **.PROTECT/.UNPROTECT** blocks. It uses DES encryption with an internal 56-bit key. The file will be automatically decrypted by Eldo at run time, but none of the encrypted information will be displayed in the ASCII output file (*.chi*). This guarantees the confidentiality of the data.

### **Caution**



Only **.MODEL**, **.PARAM**, **.SUBCKT** or **.PROTECT** cards will be encrypted. In the case of a **.MODEL** card concluding the file, the model must be followed by an additional line (empty or comment line).

---

## Usage

```
encrypt_eldo -i input_file [-o output_file]
+ [-lic "eldo" | "stm" | "moto"] [-compat]
```

**-i** Input filename. File can contain model cards, parameter cards and subcircuit definitions.

**-o** Output filename. If this option is not specified, the name will be *input\_file.crypt*.

**-lic** Identifier of the license controlling the execution. Identifier can be:

"**eldo**"  
(default) for control by the analogmodelspi license

"**stm**"  
for control by the analogmodelst license (STMicroelectronics)

"**moto**"  
for control by the analogmodelmot license (Motorola)

**-compat** Simulator compatibility argument. Only **.PROTECT/.UNPROTECT** blocks will be encrypted.

## Example

1. Unencrypted netlist (*diode.lib*):

```
.model DNPPSJU
+ d level=8 diolev=9 tr=27
+ vr=0
+ cjgr=3.946e-10
+ jsdgr=6.9543e-14
+ jsggr=6.4326e-10
+ vdgr=0.79381
+ pg=0.43889
tnom=27
cjsr=8.649e-12
jsdsr=3.4372e-12
jsgsr=1.0506e-09
vdsr=0.49482
ps=0.99
nsj=1.2
```

```

+ ngj=1

.subckt diode A B

.model my_diode D diolev=9 tr=27          tnom=27
+ vr=0           cjbbr=0.00072727    cjsr=8.649e-12
+ cjgr=3.946e-10      jsdbr=3.5987e-07  jsdsr=3.4372e-12

R1 A A1 1k
D1 A1 B1 my_diode
R2 B1 B 1k
.ends

```

## 2. Command for encrypting this netlist with the Eldo license:

```
encrypt_eldo -i diode.lib -o diode_crypt.lib
```

## 3. Output file generated (*diode\_crypt.lib*):

```

% .model DNPPSJU
% 6A41CAC8CE3D50CB0B85F0126F6D1CCF23E5CA9C3BD0E7F21716251CB19DA7
% 36FBBF69EF23E826A0478CA39376DCC8C4B7DBCD4A5A61CD26999C7F74FA2C
% 75D9133AD4A91885751DDCE235FB1D944B96A4491CE3EDC524758A7697FA1
% F912ED3488C832AB598321B58D3F25D176D794F6376924596B7948A4068996
% 35363770AA370F3C10331A8FC0B5822FFDA963774FFFC74F1FFA3021EA693F
% D88A1A3E8DDBC62012F317FE3BF379A999B6638BFE8A7A97E6B46C6E1E2B60
% A2A9362162265083A1A898E5CF8EF5A5F6FF420A52C2E23FB9364E2BC13F29
% 6F79757FF86BB080DCBAA2B4FC7EBF863B3B0C7C896BAF6525DF00901CD9E1
% 6FC7E1BD4E691E6B006451AAE3302D64AD912114D5AC4F3D436BB1AAE5582F
% 1A

.subckt diode A B
% A40ACD98849922AC04809D607E2424D4255F6765FF779A2EF49C24DC0
% 483152A2024CA2B0C2E975EE8E41C1DCD2098C152082EA626C990AE64
% 7B7F38B8FFE9868E28164C55E19139F8378A9FF6BB9946CC7448D744
% 127ADEA26B511101C631E3D455880E6E56EFBFEB481964A7300C9FBF
% 5B57245A6CC15D2A72BC833D8437DCF53D85F1B017366485676443756
% 2BFFAC6DFEB58B5A43ACD64C30ECA1F7491380ED4F8777FBDF2BF6DA1
% 287BD1F4C86D96B25ABA5164851F196324189F42BFC67116235368287
% 5
.ends

```

## 4. Input file (*test.cir*)

```

* example for encryption netlist
.lib diode_crypt.lib
v1 1 0 1
d1 1 2 DNPPSJU
x1 2 0 diode
.op
.end

```

## 5. Invoking Eldo.

```
eldo test.cir
```

## Notes

The netlist is not displayed inside the Eldo output (*.chi*) file, as shown below:

```
.MODEL DNPPSJU      ----> the lines 4 to 13 are not displayed here.  
.SUBCKT DIODE A B  -> the lines 16 to 23 are not here.  
Warning 902: "DIOLEV in model DIODE.MY_DIODE": Model parameter ignored.  
Warning 902: "TR in model DIODE.MY_DIODE": Model parameter ignored.  
Warning 902: "VR in model DIODE.MY_DIODE": Model parameter ignored.  
Warning 902: "CJBR in model DIODE.MY_DIODE": Model parameter ignored.  
Warning 902: "CJSR in model DIODE.MY_DIODE": Model parameter ignored.  
Warning 902: "CJGR in model DIODE.MY_DIODE": Model parameter ignored.  
Warning 902: "JSDBR in model DIODE.MY_DIODE": Model parameter ignored.  
Warning 902: "JSDSR in model DIODE.MY_DIODE": Model parameter ignored.  
.ENDS  
  
V1 1 0 1  
D1 1 2DNPPSJU  
X1 2 1 DIODE
```

The encrypted model parameters are not displayed inside the Eldo output *.chi* file.

# Appendix F

## Spectre to Eldo Converter

---

The Spectre to Eldo converter is a tool to convert libraries and netlists from Spectre format (Spectre language syntax) into Eldo format (Eldo syntax). The script is named *spect2el*.

## Prerequisites

- A valid “Eldo kernel” license key.
- Eldo version v5.8\_1.1 or later. Earlier versions are not guaranteed to achieve Spectre compatibility on the converted libraries. Some Error/Warning messages may be issued.
- Korn shell must be installed under the */bin* directory. The tool will not be able to run if */bin/ksh* cannot be invoked.
- The “gawk” utility. This is provided along with the tool.

## Supported Features

The tool can convert Spectre syntax to guarantee compatible results for:

- Basic component instantiations
- Device models and instances
- Subcircuit definitions and instances
- Controlled sources
- Functions, and simple “if” statements
- Statistics
- nport instances

The tool can handle libraries with nested includes for any number of levels and for any directory structure.

The following Spectre device models are supported:

- BSIM3v3 MOS model
- VBIC bipolar model
- HICUM bipolar model

- BJT level one model (Gummel Poon)
- Diode level one model
- JFET level one model
- MOS1 model
- Polynomial models for R, L, and C
- Geometric Resistor model
- Physical Resistor model (phyres)
- MOS0 model. (However, since this model has no equivalent in Eldo, it is mapped to the MOS level=1 model of Eldo. Full compatibility is not guaranteed.)

---

**Note**



Other device models are converted from the syntax point of view only. The list includes:  
BSIM4, DP500, MOS2, MOS3, BSIM1, BSIM2, BSIM3v2, EKV, BSIM3SOI PD,  
GAAS, TOM2, BJT504, JFET level 2, JFET level 4.

---

**Note**



For proper conversion of netlists or libraries, all the included files should exist. If any instance in the input library or netlist refers to a model, a subcircuit or a Verilog-A module, which is not defined in the input library/netlist or one of its existing included files, this instance can not and will not be recognized or converted.

---

## Netlist Conversion

The netlist conversion features come with the *spect2el* tool. This feature automates the netlists conversion process from Spectre syntax to Eldo syntax.

## Supported Features

### Analyses

- AC analysis
- DC analysis
- Transient analysis
- S parameters analysis
- Monte Carlo analysis
- Sweep analysis

- Noise analysis
- Sensitivity analysis
- Transfer function analysis

## Control statements

- IC and nodeset statements
- Option statements:

|        |         |         |      |
|--------|---------|---------|------|
| Reltol | Vabstol | Iabstol | Temp |
| Tnom   | Scalem  | Scale   | Gmin |
| Digits | Pivrel  | Pivab   |      |

- Paramset statement
- Save statement
- Simple combinations of alter, set, and altergroup statements

## Sources

- Independent Sources:
  - Isource
  - Vsource
- Polynomial Controlled Sources:
  - Pcccs
  - Pccvs
  - Pvccs
  - Pvcvss

## Components

- Current probe (iprobe)
- Independent resistive source (port)

## Limitations

The following features cannot be converted with the tool. They require some manual modifications.

- Libraries and netlists must be case insensitive. Case sensitive libraries and netlists cannot be handled.
- Libraries and netlists must use Spectre's syntax (simulator lang=spectre). Any section written in spice syntax (simulator lang=spice) cannot be converted correctly.
- Nested sweeps that sweeps more than one paramset.

Example:

```
sweep1 sweep paramset=paramset1 {  
    sweep2 sweep paramset=paramset2 {  
        ....  
    }  
}
```

Complicated combinations of the alter, set and altergroup statements.

---

**Note**



AHDL (Verilog-A) includes and instants could be also converted, however this feature is still unqualified and is considered a beta feature. By default, this feature is not active. To activate it the option “-do\_va” should be used. For more details please see the “Usage” section below.

---

## Usage

The syntax for the tool is as follows:

```
spect2el  
-in_dir <dir1>  
[-file_list <list>]  
[-out_dir <dir2>]  
[-eldo <eldo version>]  
[-remove_param 1/0]  
    [-remove_param_warn 1/0]  
    [-do_va 1/0]  
    [-inline_warnings 1/0]  
    [-netlist_converter 1/0]  
    [-database_dir <db_dir>]
```

All the above arguments are optional except for `-in_dir`.

|                              |                                                                                                                                                                            |
|------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>-in_dir dir1</code>    | Input directory where the library/netlist to be converted reside. This could be a full path or a relative path, relative to where the tool is initiated. <b>Required</b> . |
| <code>-file_list list</code> | File list to be converted. It defaults to all files in the given directory if not specified. Default ‘*’.                                                                  |

**Note**



The file list should only include the library main file(s), which is to be directly included in the netlist in case of library conversion, or only the netlist file in case of netlist conversion. Other files, which are included from within those main file(s) are handled automatically. In other words, each file in the file list should not be included in any other file in the file list.

---

**Note**



Each file in the file list should be a stand alone file. It should not depend on any other file in the file list. i.e. it should not use any model, parameter, subcircuit, which is not defined in it or in any of the files it includes.

---

|                                   |                                                                                                                                                                                                                                                                                                                                                                                                    |
|-----------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>-out_dir dir2</code>        | Output directory where the converted files are to be written. This could be a directory name or a path. This directory will be created by the tool. If this directory already exists a warning message is issued. If not specified, it defaults to the same name as the input directory name but with a suffix <code>_converted.</code> Default <code>./{in_dir}_converted</code> for output name. |
| <code>-eldo eldo_version</code>   | Specifies the version of Eldo to be used. Default <code>\$eldover</code> .                                                                                                                                                                                                                                                                                                                         |
| <code>-remove_param</code>        | If this is set, it will remove some model card parameters which are not supported by Eldo. Default 1 (set). If set, parameters removed are:<br>“dope” for Diode level 1.<br>“n”, “meto”, “wnoi” and “vbox” for BSIM3v3.                                                                                                                                                                            |
| <code>-remove_param_warn</code>   | If this is set, a warning will be issued each time one of the above parameters is removed. (This works only if <code>remove_param</code> is set.) Default 0 (not set).                                                                                                                                                                                                                             |
| <code>-do_va</code>               | This attempts to convert Verilog-A instances used inside Spice netlists. (Note, this feature is not 100% guaranteed.) Default 0 (not set).                                                                                                                                                                                                                                                         |
| <code>-inline_warnings</code>     | Useful converter warnings will be printed inline as comments in the converted netlist to ease any required manual conversion work. Default 0 (not set).                                                                                                                                                                                                                                            |
| <code>-netlist_converter</code>   | Enable the netlist conversion feature. Default 0 (not set).                                                                                                                                                                                                                                                                                                                                        |
| <code>-database_dir db_dir</code> | Allows you to make use of a library that has been converted before and is included in your netlist instead of having to convert it again for each design and for each time you modify your design. This is achieved by saving a database for the library when it is first converted, and then reading this database later to convert the netlists using this library. This                         |

database is written to or read from the path specified by *DB\_dir*. You should have write access to this path. The converter will check for every file if it has an entry in this database or not. If a file has an entry it will not be converted but will be retrieved from the database instead. Otherwise, if the file does not have an entry in the database (or is more recent than the entry), this file will be converted and saved to the database. Note that *DB\_dir* should be independent from the *out\_dir*. *DB\_dir* cannot be set to the same directory as *out\_dir* or any of its subdirectories.

**-h**

Display the tool usage.

# Appendix G

## EZwave and Xelga Differences

---

### Introduction

This appendix describes the main differences between EZwave and Xelga, especially concerning the overall user model.

### Xelga

Xelga is a selection based graphical user interface dedicated to display and to analyze simulation results.

The main user model is to select “objects” prior to acting upon them. The selection paradigm is:

- Left mouse button click for single selection
- Middle mouse button click for multiple selection
- Right mouse button click for deselection.

The objects that can be selected are:

- Waveforms
- Waveform Names
- Graphs
- Cursors
- Labels

It is possible to select different objects at the same time, but the actions on them are separate.

The graphical interface contains only “pull-down” menus to access the different features. However, almost all menu items can be accessed through keyboard shortcuts, for example, Alt-Z enables mouse zooming.

There are two different kinds of window:

- Analog window  
The *analog* window can display both analog and digital waveforms (in separate graphs) but the organization within the window is more “analog-dedicated”. An analog window contains multiple graphs with multiple X-axes.

- Digital window

The *digital* window can display both analog and digital waveforms but the organization within the window is more “digital-dedicated”. A digital window contains a unique X-axis.

The user can open multiple analog and digital windows.

## Post processing: waveform processor

Xelga contains a set of built-in functions that can be applied to number, matrix or waveform, it is displayed in a list-box. The basic expression syntax is:

<result\_name> = <expression>

Where expression is a “C-like” syntax coming from an internal language. It is possible to write/save/reuse “User Define Functions” based on the expression language. The analog waveforms resulting from post processing can be saved to a set of different “.cou” files with default names:

- Wave\_Processor.cou (=> results coming from the wave processor)
- DSP.cou (=> results coming from the DSP functions)
- D2A.cou (=> results coming from Digital To Analog conversion)
- Phase\_Noise.cou (=> results coming from Phase Noise calculation)

The post-processing functions in the wave processor do not handle digital waveforms.

## Multiple-run handling

It is possible for the user to choose “Multi-Simulation Mode” to handle waveforms corresponding to different runs of the same element as grouped waveforms. It is not possible to apply calculations on a grouped waveform.

## Configuration files

A set of attributes can be saved in a configuration file after the user manually edit and save the configuration file.

## EZwave

EZwave provides an advanced graphical user interface that displays and analyzes analog, digital, and mixed-signal waveform databases saved in JWDB (Joint Waveform Database) format.

It is possible to select some “objects” prior to acting upon them, or to retrieve them from proposed lists.

The selection paradigm is:

- Left mouse button click on an object: single selection
- <ctrl key> + Left mouse button click: multiple <on/off> selection
- <shift key> + Left mouse button click: adjacent multiple selection
- Left mouse button click on empty area: deselect all

The objects that can be selected are:

- Waveforms
- Waveform names
- Text Annotations
- Measurement results markers

In addition of selection based actions, there is a set of other actions than can be accessed through contextual popup menus:

- Cursor (and data values)
- Row
- X axis
- Y axes
- Window
- Workspace
- ...

Whenever applicable, it is possible to combine “multiple-selection” action on a “contextual” popup menu. For example: to delete multiple waveforms from graphical window, select multiple waveforms using <ctrl key> + Left Mouse Button, right click on a selected waveform, choose “Delete” item in the list of proposed actions.

The graphical user interface contains all the modern methods to access features in addition of the “pull down” menu bar. For example, it contains a “toolbar” to quickly access a subset of features, and many different contextual “popup menus” allowing access to a subset of features dedicated to the selected object.

Mouse dragging in the graphical window default enables zooming in one or two directions.

EZwave also supports “drag and drop” in order to easily and quickly move objects from a location to another.

As part of modern graphical user interface look and feel, EZwave also support “ToolTips” (or help balloons) in order to provide fast information on the object that is pointed by the mouse.

EZwave is based on multiple graphical windows contained in a unique session window. There is a unique type of graphical window containing a unique X-axis per window, but allow displaying both analog and digital waveforms in separate rows. (Note: It is possible to overlap a digital waveform on top of an analog one)

## Post processing: waveform calculator

EZwave contains a “waveform calculator” based on a calculator graphical user interface, containing buttons in order to allow quick access to a subset of built-in functions and operators. The calculations can be applied to either numbers or waveforms. Both analog and digital waveforms are supported. Some operations can also be applied to vectors. The result waveforms are automatically updated during a simulation whenever it is applicable.

The waveform calculator is organized through a set of panels organized by theme:

- Complex
- Logic
- RF
- Signal Processing
- Statistical
- Trigonometric

It also contains a list of other built-in functions not contained in the buttons. The basic expression syntax is:

<expression> or <result\_name> = <expression>

Where expression is a “C-like” syntax following the “Python” syntax.

User Defined Functions can be written, using a set of Tcl commands, using the same <expression> syntax for calculator commands, for example, wfc{<expression>}.

In addition to the waveform calculator, EZwave also supports a “Measurement Tool”. This is very useful to annotate results directly on the waveforms or create new waveforms containing measurement results (including as a function of a simulation parameter).

The analog and digital waveforms created from post processing are stored in the database with default filename to: calc.wdb

When waveforms resulting from post processing are displayed in a graphical window, it is possible to get calculation re-executed with new simulation data using the “File->Reload” menu. Another way is to save window contains in a “.swd” file.

## Multiple-run handling

The results inside a database are organized by grouped waveforms called “compound waveforms” in the case of multiple-run analyses. The “compound waveforms” can be used as any other single waveform, including for post processing. The individual elements of compound waveforms can be used if necessary.

## Configuration files

A set of attributes is automatically saved into configuration file. Some of these attributes can be changed in the graphical window opened through “Edit->Options ...” in the session window or in the waveform calculator.

EZwave also supports new functionality such as “event search” available on analog or digital waveforms or complicated expressions. It is also possible to “export” window contents to JPEG format.

The EZwave online help allows a keyword search in order to efficiently retrieve help on the required topic.

## Xelga-EZwave cross references

**Table G-1. Xelga-EZwave cross references**

| <b>Xelga</b>                                                                                        | <b>EZwave</b>                                                                                    |
|-----------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------|
| Selection:<br>Left mouse button<br>Middle mouse button<br>Right mouse button                        | Selection:<br>Left Mouse button<br><Ctrl> + Left mouse button<br>Left mouse button on empty area |
| Graph                                                                                               | Row                                                                                              |
| Analog Window: multiple X-axes                                                                      | Graphical window: Unique X-axis                                                                  |
| Digital Window: single X-axis                                                                       | Graphical window: Unique X-axis                                                                  |
| Waveform Processor                                                                                  | Waveform Calculator                                                                              |
| Save Post processing:<br>Go to “Control->Edit Files”, then choose the desired databases and save it | Save Post processing:<br>Right click on “calc” database and choose “Save as ...”                 |
| Move a waveform:<br>Select waveform,<br>menu “wave->move”,<br>click on the new location             | Move a waveform:<br>“drag and drop” waveform to new location                                     |
| Mouse Zooming:<br>Go to “View->Zoom->Mouse”                                                         | Mouse Zooming:<br>“Drag Mouse” in a row or on Axes                                               |

**Table G-1. Xelga-EZwave cross references**

|                                                                                    |                                                                                       |
|------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------|
| Multiple Run handling:<br>Go to “Control->Multi-simulation mode”                   | Multiple Run handling:<br>“compound waveforms” stored by simulators                   |
| Configuration Files:<br>Edit and save a set of attributes into a .xelga_setup file | Configuration Files:<br>Automatic save.<br>Go to “Edit->Options” to modify attributes |
| Analog database: Cou Files                                                         | JWDB                                                                                  |
| Digital database: Dou Files                                                        | JWDB                                                                                  |
| Cursors are based on row                                                           | Cursors are based on window                                                           |

# Appendix H

## STMicroelectronics Models

### Introduction

This section describes how to use the ST models inside Eldo, which in this case is called Eldo-ST.

### How to Invoke Eldo-ST

Eldo-ST is invoked as follows:

```
eldo -stver ... cir_file_name
```

or by adding *before* the **.MODEL** cards the following:

```
.OPTION STVER
```

### What Does it Change?

When Eldo-ST is invoked, model levels are automatically changed to use Eldo-ST in exactly the same way as ST-SPICE:

Table H-1. Eldo/Eldo-ST Model Levels

| Component | Eldo                                       | Eldo-ST                   |
|-----------|--------------------------------------------|---------------------------|
| MOS       | <b>.MODEL</b> ... LEVEL=18 or LEVEL=STMOS1 | <b>.MODEL</b> ... LEVEL=1 |
|           | <b>.MODEL</b> ... LEVEL=19 or LEVEL=STMOS3 | <b>.MODEL</b> ... LEVEL=3 |
| Bipolar   | <b>.MODEL</b> ... LEVEL=2                  | <b>.MODEL</b> ... LEVEL=1 |
| Diode     | <b>.MODEL</b> ... LEVEL=4                  | <b>.MODEL</b> ... LEVEL=1 |
|           | <b>.MODEL</b> ... LEVEL=5                  | <b>.MODEL</b> ... LEVEL=2 |
| JFET      | <b>.MODEL</b> ... LEVEL=6                  | <b>.MODEL</b> ... LEVEL=3 |
|           | <b>.MODEL</b> ... LEVEL=4                  | <b>.MODEL</b> ... LEVEL=1 |
| Resistor  | <b>.MODEL</b> ... LEVEL=5                  | <b>.MODEL</b> ... LEVEL=2 |
|           | <b>.MODEL</b> ... LEVEL=2                  | <b>.MODEL</b> ... LEVEL=1 |

All the standard Eldo levels may be used in Eldo-ST by adding the following in the **.MODEL** card:

MODTYPE=ELDO

## Licensing

To use the MOS, Bipolar, Diode, JFET and PROM STMicroelectronics models, a special license is required (FEATURE 100104). For resistor and capacitor models, no license is necessary.

## STMicroelectronics Version of Eldo

Specially tuned device models have been developed for use when simulating STMicroelectronics designs. These STMicroelectronics models are available within the standard version of Eldo, for example as can be seen from the list of diode models available:

**Table H-2. STMicroelectronics Model Selection**

| LEVEL Value | Model Name                        |
|-------------|-----------------------------------|
| 1           | Berkeley Level 1                  |
| 2           | Modified Berkeley Level 1         |
| 3           | Fowler-Nordheim                   |
| 4           | <b>STMicroelectronics Level 1</b> |
| 5           | <b>STMicroelectronics Level 2</b> |
| 6           | <b>STMicroelectronics Level 3</b> |

A special version of Eldo is also available which has been developed for use within the STMicroelectronics design kit. This version is identical to the standard version in every respect except that the STMicroelectronics device models occupy different levels within the lists of available device models. The different device model levels used within the STMicroelectronics version of Eldo are listed below:

## Junction Diode Models

**Table H-3. STMicroelectronics Junction Diode Model Selection**

| LEVEL Value | Model Name                        |
|-------------|-----------------------------------|
| 1           | <b>STMicroelectronics Level 1</b> |
| 2           | <b>STMicroelectronics Level 2</b> |
| 3           | <b>STMicroelectronics Level 3</b> |

## MOSFET Models

**Table H-4. STMicroelectronics MOSFET Model Selection**

| LEVEL Value | Model Name                 |
|-------------|----------------------------|
| 1           | STMicroelectronics Level 1 |
| 3           | STMicroelectronics Level 3 |

## Bipolar Junction Transistor Model

**Table H-5. STMicroelectronics BJT Model Selection**

| LEVEL Value | Model Name                 |
|-------------|----------------------------|
| 1           | STMicroelectronics Level 1 |



# Index

## — Symbols —

! comment delimiter, 3-2

\* comment delimiter, 3-2

.A2D, 3-29, 10-4

.AC, 3-29, 10-12

.NOISE, 10-180

.PLOT, 10-221, 10-256

.PRINT, 10-254 *to* 10-255

.PROBE, 10-259

Examples, 24-2, 24-5, 24-9, 24-13, 24-23,  
C-7, C-16

.ADDLIB, 3-29, 10-18

.ALTER, 10-23

.MODEL, 10-160

.SUBCKT, 10-306

.AGE, 3-30, 10-20, 21-2

.AGEMODEL, 3-30, 10-21, 21-5

.ALTER, 3-30, 10-22

-compat flag, 10-25

.BIND, 3-30, 10-26

.CALL\_TCL, 3-30, 10-28, 22-3

.CHECKBUS, 3-30, 10-30

.CHECKSOA, 3-30, 10-33

.SETSOA, 10-284

.CHRENT, 3-31, 10-35

.CHRSIM, 3-31, 10-37

.cir File Structure Overview, 3-1

.COMCHAR, 3-31, 10-39

.CONNECT, 3-31, 10-40

.ALTER, 10-22

.CONSO, 3-31, 10-41

.ALTER, 10-22

.CORREL, 3-31, 10-42

.D2A, 3-31, 10-44

.DATA, 3-31, 10-51

.DC, 3-32, 10-54

.PARAM, 10-205

.PLOT, 10-221, 10-256

.PRINT, 10-254

.PROBE, 10-259

.DCMISMATCH, 3-32, 10-61

.DEFAULT, 10-65

.DEFMAC, 3-32, 10-66

.EXTRACT, 10-99

.PARAM, 10-205

.DEFMOD, 3-32

.DEFPLOTDIG, 3-32, 10-69, 10-246

.DEFWAVE, 3-32, 10-70

.PARAM, 10-205

.PLOT, 10-222

.PRINT, 10-233 *to* 10-255

.DEL, 3-32, 10-74

.DISCARD, 3-33, 10-75

.DISFLAT, 3-33, 10-76

.DISTRIB, 3-33, 10-77

.DSP, 3-33, 10-78

.DSPF\_INCLUDE, 3-33, 10-79

.DSPMOD, 3-33, 10-85

.END, 3-33, 10-89

.ALTER, 10-22

.INCLUDE, 10-128

.ENDL, 3-33, 10-90

.ENDS, 3-33, 10-91

.SUBCKT, 10-306

.EQUIV, 3-34, 10-92

.EXTMOD, 10-94

.EXTRACT, 3-34, 10-95

.ALTER, 10-22

.DEFMAC, 10-66

.DEFWAVE, 10-71

.PARAM, 10-205

.FFILE, 3-34, 10-115

.FORCE, 3-34, 10-118

.FOUR, 3-34, 10-119

.FUNC, 10-120

.GLOBAL, 3-34, 10-121

.ALTER, 10-22

.SUBCKT, 10-306

.GUESS, 3-34, 10-122

.LOAD, 10-139

- .NODESET, 10-179
- .SAVE, 10-274
- .USE, 10-328
- .HIER, 3-34, 10-123
- .IC, 3-34, 10-125
  - .LOAD, 10-139
  - .SAVE, 10-274
  - .USE, 10-328
- .IGNORE\_DSPF\_ON\_NODE, 10-127
- .INCLUDE, 3-35, 10-128
  - .ALTER, 10-22
- .INIT, 3-35, 10-130
- .IPROBE, 10-131
- .LIB, 3-35, 10-133
  - .MODEL, 10-160
  - .SUBCKT, 10-306
- .LOAD, 3-35, 10-139
- .LOOP, 3-35, 10-140
- .LOTGROUP, 3-35, 10-142
- .LSTB, 3-35, 10-144
- .MAP\_DSPF\_NODE\_NAME, 10-146
- .MC, 3-35, 10-147
  - .ALTER, 10-23
  - .SETSOA, 10-284
  - Examples, 24-23
- .MCMOD, 3-35, 10-153
- .MEAS, 3-35, 10-155
- .MODDUP, 3-36, 10-159
- .MODEL, 3-36, 10-160
  - Examples, 24-9, 24-13, 24-16, 24-19, 24-23, 24-26
- .MODEL ... LOGIC, 3-26, 7-3
- .MODLOGIC, 3-36, 10-164
- .MONITOR, 10-165
- .MPRUN, 10-166
- .MSELECT, 10-173
- .NET, 3-36, 10-176
- .NEWPAGE, 3-37, 10-177
- .NOCOM, 3-37, 10-178
- .NODESET, 3-37, 10-179
  - .GUESS, 10-122
  - .LOAD, 10-139
  - .SAVE, 10-274
  - .USE, 10-328
- .NOISE, 3-37, 10-180
- .PLOT, 10-221, 10-256
- .PRINT, 10-254 *to* 10-255
- .PROBE, 10-259
  - Examples, 14-14, 24-9
- .NOISETRAN, 3-37, 10-182, 14-3
  - Examples, 14-9, 14-18
- .NOTRC, 3-37, 10-185
- .NWBLOCK, 3-37, 10-186
- .OP, 3-37, 10-187
  - .ALTER, 10-22
- .OPTFOUR, 3-37, 10-192
- .OPTIMIZE, 3-37, 10-197
- .OPTION, 3-38
  - .ALTER, 10-22
  - .PROBE, 10-258
  - .TRAN, 10-319
  - Examples, 24-2, 24-16, 24-19, 24-23
  - see also Options
- .OPTNOISE, 3-38, 10-199
- .OPTPWL, 3-38, 10-203
- .OPTWIND, 3-38, 10-204
- .PARAM, 3-38, 10-205
  - .ALTER, 10-22
  - .MODEL, 10-161
  - .SUBCKT, 10-306
  - compat flag, 10-213
  - Examples, 24-19
  - Multiple Affectation of Parameters, 10-213
- .PART, 10-214
- .PLOT, 3-38, 10-216
  - .AC, 10-13
  - .ALTER, 10-22
  - .CHRSIM, 10-37
  - .MC, 10-147
  - .NOISE, 10-180
  - compat flag, 10-225
  - Examples, 24-2, 24-5, 24-9, 24-13, 24-16, 24-19, 24-23
- .PLOTBUS, 3-38, 10-250
- .PRINT, 3-38, 10-254
  - .AC, 10-13
  - .ALTER, 10-22
  - .MC, 10-147
  - .NOISE, 10-180
  - .PROBE, 10-254, 10-261

- .STEP, 10-301
- .PRINTBUS, 3-39, 10-252
- .PRINTFILE, 10-256
- .PROBE, 3-39, 10-258
- .PROTECT, 3-39, 10-267
- .PZ, 3-39, 10-268
- .RAMP, 3-39, 10-269
  - DC Operating Point Calculation, 16-20
- .RESTART, 3-39, 10-271
  - .SAVE, 10-274
- .SAVE, 3-39, 10-274
  - .LOAD, 10-139
  - .USE, 10-328
    - DC Operating Point Calculation, 16-22
    - End of Simulation Results, 16-23
- .SCALE, 10-277
- .SENS, 3-39, 10-278
  - .ALTER, 10-22
  - Examples, 24-26
- .SENPARAM, 3-40, 10-280
- .SETBUS, 3-40, 10-282
  - .PLOTBUS, 10-250, 10-252
  - .SIGBUS, 10-291
- .SETKEY, 3-40, 10-283, 21-6
- .SETSOA, 3-40, 10-284
  - .CHECKSOA, 10-33
- .SIGBUS, 10-291
  - .PLOTBUS, 10-250, 10-252
  - .SETBUS, 10-282
- .SINUS, 3-40, 10-294
- .SNF, 3-40, 10-295
- .SOLVE, 3-41, 10-296
- .STEP, 3-41, 10-298
  - .ALTER, 10-23
  - .PARAM, 10-205
  - .SAVE, 10-275
  - .SETSOA, 10-284
- .SUBCKT, 3-41, 10-306
  - .PARAM, 10-206, 10-211
  - .PRINT, 10-233 to 10-241
    - compat flag, 10-311
  - Examples, 24-13
  - Subcircuit Instance, 4-171
- .SUBDUP, 3-41, 10-312
- .TABLE, 3-41, 10-313
- .TEMP, 3-41, 10-314
- .SETSOA, 10-284
  - Resistor Model, 4-9, 5-6, 5-13
- .TF, 3-41, 10-315
- .TITLE, 3-41, 10-316
- .TOPCELL, 3-41, 10-317
- .TRAN, 3-42, 10-318
  - .PLOT, 10-221, 10-256
  - .PRINT, 10-254
  - .PROBE, 10-259
    - compat flag, 10-319
  - Examples, 24-5, 24-16, 24-19, C-2, C-3, C-9, C-11, C-14, C-19, C-22
  - UIC Parameter, 4-15, 4-21, 10-118, 10-125, 16-19
- .TSAVE, 3-42, 10-323
- .TVINCLUDE, 3-42, 10-325
- .UNPROTECT, 3-42, 10-327
- .USE, 3-42, 10-328
  - .SAVE, 10-274
    - DC Operating Point Calculation, 16-22
- .USE\_TCL, 3-42, 10-331, 22-1
- .USEKEY, 3-42, 10-330, 21-7
- .WCASE, 3-42, 10-332
  - .SETSOA, 10-284
- .WIDTH, 3-42, 10-335

## — Numerics —

- 2-Input Digital Gates, 3-27
- 2-port Parameters
  - see Two-port Parameters
- 3-Input Digital Gates, 3-27, 7-11
- 4-bit Adder Example, C-3
- 4th Order Butterworth Filter Tutorial, 24-5
- 5th Order Elliptic SC Low Pass Filter Example, C-11

## — A —

- ABS (Absolute value), 3-8
- ABSTOL option, 11-13
- ABSVAR option, 11-13
- AC Analysis (.AC), 10-12
  - .LOOP, 10-141
  - .MC, 10-147
  - .NOISE, 10-180
  - .PRINT, 10-228, 10-232, 10-234

- .PZ, 10-268  
Examples, 24-2, 24-5, 24-9, 24-13, C-7, C-16  
AC Noise Analysis (.OPTNOISE), 10-199  
Accuracy  
    by Time Window (.OPTPWL), 10-203  
    by Time Window (.OPWIND), 10-204  
EPS Parameter, 4-172, 9-4, 10-307, B-1  
VNTOL parameter, B-1  
ACM option, 11-30  
ACOS (Arc cosine of value), 3-7  
ACOUT option, 11-44, 12-5  
Active RC Band Pass Filter Example, C-16  
A-D Converter Macromodel, 3-27, 7-14  
Adder Macromodel, 3-26, 6-58  
    Parameters, 6-58  
Adder, Subtractor, Multiplier, Divider Macromodels  
    Parameters, 6-58  
ADJSTEPTRAN option, 11-13  
Admittance parameters, 15-1  
AEX option, 11-52  
Age Analysis (Reliability), 10-20, 21-2  
ALIGNEXT option, 11-52  
ALT  
    .SAVE, 10-275  
ALTER\_SUFFIX option, 11-44  
ALTINC option, 11-7  
AMMETER option, 11-24  
Amode Error Messages, A-21  
Amplitude Modulation Function, 5-17  
Amplitude Modulator Macromodel, 3-25, 6-32  
    Characteristics, 6-33  
    Parameters, 6-32  
Analog Macromodels, 3-24 to 3-26, 6-1  
ANALOG option, 11-56  
Analysis  
    AC (.AC), 10-12  
    DC (.DC), 10-54  
    Loop Stability (.LSTB), 10-144  
    Monte Carlo (.MC), 10-147  
    Noise (.NOISE), 10-180  
    Noise (.OPTNOISE), 10-199  
    Pole-Zero (.PZ), 10-268  
Sensitivity (.SENS), 10-278  
Sensitivity (.SENSPARAM), 10-280  
Transient (.TRAN), 10-318  
Worst Case (.WCASE)  
    .MC, 10-148  
AND Gate  
    see Double, Triple and Multiple Input Digital Gates  
Anti-logarithmic Amplifier Macromodel, 3-26, 6-52  
Parameters, 6-52  
Applications  
    of Switched Capacitor Macromodels, 9-30  
        Euler Forward Integrator, 9-33  
        LDI Phase Control Euler Backward Integrator, 9-36  
        LDI Phase Control Euler Forward Integrator, 9-34  
        LDI Phase Control Non-inverting Integrator, 9-32  
        Non-inverting Integrator, 9-31  
Arithmetic Functions & Operators  
    ABS(VAL), 3-8  
    ACOS(VAL), 3-7  
    ASIN(VAL), 3-7  
    ATAN(VAL), 3-7  
    BITOF(a, b), 3-8  
    COMPLEX(), 3-8  
    CONJ(), 3-8  
    COS(VAL), 3-7  
    COSH(VAL), 3-7  
    DB(VAL), 3-7  
    DDT(VAL), 3-8  
    DERIV(VAL), 3-8  
    DMAX(VAL1, ..., VALn), 3-8  
    DMIN(VAL1, ..., VALn), 3-8  
    EXP(VAL), 3-7  
    IDT(VAL), 3-8  
    IMAG(), 3-8  
    INT(VAL), 3-8  
    LIMIT(a, b, c), 3-8  
    LOG(VAL), 3-7  
    LOG10(VAL), 3-7  
    MAGNITUDE(), 3-8  
    MAX(VAL1, ..., VALn), 3-8

- MIN(VAL1, ..., VALn), [3-8](#)  
POW(VAL1, VAL2), [3-8](#)  
PWR(VAL1, VAL2), [3-8](#)  
REAL(), [3-8](#)  
ROUND(VAL), [3-8](#)  
SGN(VAL), [3-7](#)  
SIGN(VAL), [3-7](#)  
SIGN(VAL1, VAL2), [3-7](#)  
SIN(VAL), [3-7](#)  
SINH(VAL), [3-7](#)  
SQRT(VAL), [3-7](#)  
STOSMITH(), [3-8](#)  
TAN(VAL), [3-7](#)  
TANH(VAL), [3-7](#)  
TRUNC(VAL), [3-8](#)  
YTOSMITH(), [3-8](#)  
ZTOSMITH(), [3-8](#)
- Arithmetic Functions & Operators in Eldo, [3-7](#)  
    Arithmetic operators, [3-11](#)  
    Bitwise operators, [3-11](#)  
    Boolean operators, [3-11](#)  
    -compat flag, [3-9](#), [12-8](#)  
    Expressions, [3-12](#)
- ASCII option, [11-44](#), [11-52](#)  
ASIN (Arc sine of value), [3-7](#)  
ASPEC option, [11-31](#)  
ATAN (Arc tangent of value), [3-7](#)  
A-to-D Converter  
    .A2D, [10-4](#)  
Automatic Ramping (.RAMP), [10-269](#)  
AUTOSTOP option, [11-24](#)  
AUTOSTOPMODULO option, [11-24](#)  
AVERAGE (extract function), [10-101](#)
- B —
- Band Pass Filter Tutorial, [24-9](#)  
BE option, [11-55](#)  
Berkeley BSIM3SOI (Level 55) MOSFET Model, [4-151](#)  
Berkeley BSIM3SOI (Level 56) MOSFET Model, [4-154](#)  
Berkeley BSIM3v2 (Level 47) MOSFET Model, [4-146](#)  
Berkeley BSIM3v3 (Level 53) MOSFET Model, [4-147](#)
- Berkeley BSIM4 (Level 60) MOSFET Model, [4-158](#)  
Berkeley BSIM5 (Level 68) MOSFET Model, [4-164](#)  
Berkeley Level 1 Diode Model, [4-106](#)  
    Parameters, [4-106](#)  
Berkeley SPICE BSIM1 MOSFET Model, [4-142](#)  
Berkeley SPICE BSIM2 MOSFET Model, [4-143](#)  
Berkeley SPICE MOSFET Models (Levels 1-3), [4-139](#)  
Bi-linear Switched Capacitor Macromodel, [3-29](#), [9-26](#)  
    Equations, [9-27](#)  
    Parameters, [9-26](#)  
Binning parameters, [4-131](#), [11-35](#)  
Bipolar Amplifier Tutorial, [24-26](#), [24-29](#)  
Bipolar Junction Transistor (BJT) Models, [3-19](#), [4-111](#)  
.SENS, [10-278](#)  
HICUM Model (Eldo Level 9), [4-116](#), [4-119](#)  
Mextram 503.2 Model, [4-114](#)  
Mextram 504, [4-117](#)  
Mextram 504 Model, [4-117](#)  
Modella Model, [4-117](#)  
Modified Gummel-Poon Parameters, [4-114](#)  
STMicroelectronics, [H-3](#)  
VBIC v1.1.5, [4-116](#)  
    Parameters, [4-116](#)  
VBIC v1.2, [4-115](#)  
    Parameters, [4-115](#)
- BITOF, [3-8](#)  
Bitwise operators, [3-11](#)  
BLK\_SIZE option, [11-44](#)  
BLOCKS=IEM option, [11-56](#)  
BLOCKS=NEWTON option, [11-56](#)  
Boolean operators, [3-11](#)  
BSIM3VER option, [11-31](#)  
BSLASHCONT option, [11-7](#), [12-6](#)  
BTA HVMOS Model, [4-166](#)  
Bus Creation, [10-282](#)  
Bus Signals

Checking, 3-30, 10-30  
Plotting of, 10-250, 10-252  
Setting of, 3-40, 10-291  
voltage threshold, 10-250, 10-252

— C —

CADENCE Compatibility  
see Options  
Calculation of Transfer Function (.TF), 10-315  
Call Tcl Function, 3-30  
.CALL\_TCL, 10-28, 22-3  
Capacitor Model, 3-17, 4-13, 4-16  
.LOOP, 10-140  
Parameters, 4-16  
CAPANW option, 11-13  
CAPTAB option, 11-45  
CARLO\_GAUSS option, 11-24  
Cascaded Inverter Circuit, 1-4  
CEIL (Value rounded up to integer), 3-8  
Change comment character (.COMCHAR),  
10-39  
Charge Control in MOS Models 4 and 6  
Example, C-14  
Check Bus Signal (.CHECKBUS), 10-30  
Check Safe Operating Area Limits  
(.CHECKSOA), 10-33  
CHECKDUPL option, 11-8  
CHGTOL option, 11-13  
Chi file output, 3-13  
Convergence Information, 3-14  
Grounded Capacitors Information, 3-14  
Matrix Information, 3-14  
Newton Block Information, 3-14  
Node & Element Information, 3-13  
chi2cir utility, E-1  
Circuit Partitioning (.PART), 10-214  
Circuit Temperature  
Setting of (.TEMP), 10-314  
Classification of Error Messages, A-1  
CMOS Operational Amplifier (Closed Loop)  
Example, C-9  
CMOS Operational Amplifier (Open Loop)  
Example, C-7  
CNTTHREAD option, 11-8  
Colpitts Oscillator Tutorial, 24-16  
COMJ (Conjugate of complex number), 3-8

Command  
Error Messages, A-13  
Warning Messages, A-32  
Command Description  
.A2D, 3-29, 10-4  
.AC, 3-29, 10-12  
.ADDLIB, 3-29, 10-18  
.AGE, 3-30  
.AGEMODEL, 3-30, 10-21  
.ALTER, 3-30, 10-22  
.BIND, 10-26  
.CALL\_TCL, 3-30, 10-28  
.CHECKBUS, 3-30, 10-30  
.CHECKSOA, 3-30, 10-33  
.CHRENT, 3-31, 10-35  
.CHRSIM, 3-31, 10-37  
.COMCHAR, 3-31, 10-39  
.CONNECT, 3-31, 10-40  
.CONSO, 3-31, 10-41  
.CORREL, 3-31, 10-42  
.D2A, 3-31, 10-44  
.DATA, 3-31, 10-51  
.DC, 3-31, 10-54  
.DCMISMATCH, 3-32, 10-61  
.DEFAULT, 3-32, 10-65  
.DEFMAC, 3-32, 10-66  
.DEFMOD, 3-32, 10-68  
.DEFPLOTDIG, 3-32, 10-69  
.DEFWAVE, 3-32, 10-70  
.DEL, 3-32, 10-74  
.DISCARD, 3-33, 10-75  
.DISFLAT, 3-33, 10-76  
.DISTRIB, 3-33, 10-77  
.DSP, 3-33, 10-78  
.DSPF\_INCLUDE, 3-33, 10-79  
.DSPMOD, 3-33, 10-85  
.END, 3-33, 10-89  
.ENDL, 3-33, 10-90  
.ENDS, 3-33, 10-91  
.EQUIV, 3-34, 10-92  
.EXTMOD, 3-34, 10-94  
.EXTRACT, 3-34, 10-95  
.FFILE, 3-34, 10-115  
.FORCE, 3-34, 10-118  
.FOUR, 3-34, 10-119

- .FUNC, 10-120
- .GLOBAL, 3-34, 10-121
- .GUESS, 3-34, 10-122
- .HIER, 3-34, 10-123
- .IC, 3-34, 10-125
- .IGNORE\_DSPF\_ON\_NODE, 3-34, 10-127
- .INCLUDE, 3-35, 10-128
- .INIT, 3-35, 10-130
- .IPROBE, 10-131
- .LIB, 3-35, 10-133
- .LOAD, 3-35, 10-139
- .LOOP, 3-35, 10-140
- .LOTGROUP, 3-35, 10-142
- .LSTB, 3-35, 10-144
- .MALIAS, 10-145
- .MAP\_DSPF\_NODE\_NAME, 10-146
- .MAP\_DSPFNODE\_NAME, 3-35
- .MC, 3-35, 10-147
- .MCMOD, 3-35, 10-153
- .MEAS, 3-35, 10-155
- .MODDUP, 3-36, 10-159
- .MODEL, 3-36, 10-160
- .MODLOGIC, 3-36, 10-164
- .MONITOR, 10-165
- .MPRUN, 3-36, 10-166
- .MSELECT, 10-173
- .NET, 3-36, 10-176
- .NEWPAGE, 3-37, 10-177
- .NOCOM, 3-37, 10-178
- .NODESET, 3-37, 10-179
- .NOISE, 3-37, 10-180
- .NOISETRAN, 3-37, 10-182, 14-3
- .NOTRC, 3-37, 10-185
- .NWBLOCK, 3-37, 10-186
- .OP, 3-37, 10-187
- .OPTFOUR, 3-37, 10-192
- .OPTIMIZE, 3-37, 10-197
- .OPTION, 3-38, 10-198, 11-1 to 11-60
- .OPTNOISE, 3-38, 10-199
- .OPTPWL, 3-38, 10-203
- .OPTWIND, 3-38, 10-204
- .PARAM, 3-38, 10-205
- .PART, 10-214
- .PLOT, 3-38, 10-216
- .PLOTBUS, 3-38, 10-250
- .PRINT, 3-38, 10-254
- .PRINTBUS, 3-39, 10-252
- .PRINTFILER, 3-39, 10-256
- .PROBE, 3-39, 10-258
- .PROTECT, 3-39, 10-267
- .PZ, 3-39, 10-268
- .RAMP, 3-39, 10-269
- .RESTART, 3-39, 10-271
- .SAVE, 3-39, 10-274
- .SCALE, 10-277
- .SENS, 3-39, 10-278
- .SENSPARAM, 3-40, 10-280
- .SETBUS, 3-40, 10-282
- .SETKEY, 3-40, 10-283
- .SETSOA, 3-40, 10-284
- .SIGBUS, 3-40, 10-291
- .SINUS, 3-40, 10-294
- .SNF, 3-40, 10-295
- .SOLVE, 3-41, 10-296, 10-297
- .STEP, 3-41, 10-298
- .SUBCKT, 3-41, 10-306
- .SUBDUP, 3-41, 10-312
- .TABLE, 3-41, 10-313
- .TEMP, 3-15, 3-41, 10-314
- .TF, 3-41, 10-315
- .TITLE, 3-41, 10-316
- .TOPCELL, 3-41, 10-317
- .TRAN, 10-318
- .TSAVE, 3-42, 10-323
- .TVINCLUDE, 3-42, 10-325
- .UNPROTECT, 3-42, 10-327
- .USE, 3-42, 10-328
- .USE\_TCL, 3-42, 10-331
- .USEKEY, 3-42, 10-330
- .WCASE, 3-42, 10-332
- .WIDTH, 3-42, 10-335
- Command Line operation
  - compat flag, 2-7
  - cou47, 2-3
- Commands, 10-1
- Comment Lines in Eldo, 3-2
- Common Netlist Errors, B-1
- Comparator Macromodel, 3-24, 6-3
- compat flag

- .ALTER, 10-25  
.PARAM, 10-213  
.PLOT, 10-225  
.SUBCKT, 10-311  
.TRAN, 10-319  
Arithmetic Functions & Operators in Eldo, 3-9, 12-8  
Running from the Command Line, 2-7  
COMPAT option, 11-6  
Compatibility  
    TI-Spice, 13-1  
COMPEXUP option, 11-8  
COMPLEX (Complex number function), 3-8  
COMPMOD option, 11-7  
COMPNET option, 11-7  
Component Names in Eldo, 3-3  
COMPRESS (extract function), 10-101  
Conditions of DC Analysis (.NODESET), 10-179  
Configuration of Simulator (.OPTION), 10-198, 11-1  
Configure Spice Descriptions  
    .BIND, 10-26  
Connect Two Nodes (.CONNECT), 10-40  
Constant Gain Circles  
    see Two-port Constant Gain Circles  
Continuation Lines in Eldo, 3-2  
CONTINUE\_INCLUDE option, 11-8  
Control Language, 3-1  
Control options (.OPTION), 10-198, 11-1  
Control page layout (.NEWPAGE), 10-177  
Convergence Information in Eldo, 3-14  
Correlation Coefficient, 10-42  
COS (Cosine of value), 3-7  
COSH (Hyperbolic cosine of value), 3-7  
COU, 2-3  
COU option, 11-52  
-cou47  
    Running from the Command Line, 2-3  
Coupled Inductor Model, 3-17, 4-27  
CPTIME option, 11-25  
Create Bus (.SETBUS), 10-282  
CSDF, 2-4  
CSDF option, 11-52  
CSHUNT option, 11-57  
CTEPREC option, 11-59  
Current Controlled Current Source, 5-46  
Current Controlled Switch Macromodel, 3-25, 6-21  
Current Controlled Voltage Source, 5-57  
Current Used by a Circuit (.CONSO), 10-41  
**— D —**  
D\_WA (extract function), 10-102  
D2A  
    Logical states, 10-48  
D2DMVL9BIT option, 11-57  
D-A Converter  
    .D2A, 10-44  
D-A Converter Macromodel, 3-27, 7-16  
Databases  
    Loading, 2-3  
DB (Value in decibels), 3-7  
DC Analysis (.DC), 10-54  
    .IC, 10-125  
    .MC, 10-147  
    .NODESET, 10-179  
    .OP, 10-187  
    .PRINT, 10-226  
    .SOLVE, 10-296  
        .RELTOL, 10-296  
    .STEP, 10-301  
        Switch Macromodel, 9-8  
DC Analysis Conditions (.NODESET), 10-179  
DC Mismatch Analysis (.DCMISMATCH), 10-61  
DC Operating Point, 10-12, 10-54, 10-122, 10-179, 10-269, 11-17, 11-23, 16-19  
    .RAMP, 10-269  
DC Operating Point Calculation (.OP), 10-187  
    .SAVE and .USE, 16-22  
DC Sweep, 10-56  
    .CHRSIM, 10-37  
DCLOG option, 11-59  
DCM (extract function), 10-102  
DCPART option, 11-57  
DDT (Derivative of value), 3-8  
DEFA2D option, 11-58  
DEFAD option, 11-31  
DEFAS option, 11-31  
DEFCONVMSG option, 11-58

- DEFD2A option, [11-58](#)
- DEFL option, [4-127](#), [11-31](#)
- DEFNRD option, [11-31](#)
- DEFNRS option, [11-31](#)
- DEFPD option, [11-31](#)
- DEFPS option, [11-31](#)
- DEFPTNOM option, [11-25](#)
- DEFW option, [4-127](#), [11-31](#)
- Delay Macromodel, [3-24](#), [3-26](#), [6-14](#), [7-6](#)
- DERIV (Derivative of value), [3-8](#)
- Device Description
  - Berkeley BSIM3SOI (Level 55) Model, [4-151](#)
  - Berkeley BSIM3SOI (Level 56) Model, [4-154](#)
  - Berkeley BSIM3v2 (Level 47) Model, [4-146](#)
  - Berkeley BSIM3v3 (Level 53) Model, [4-147](#)
  - Berkeley BSIM4 (Level 60) Model, [4-158](#)
  - Berkeley BSIM5 (Level 68) Model, [4-164](#)
  - Berkeley SPICE BSIM2 MOSFET Model, [4-143](#)
  - Bipolar Junction Transistor (BJT), [4-109](#)
  - Capacitor, [4-13](#)
  - Coupled Inductor, [3-17](#), [4-27](#)
  - Diode, [4-102](#)
  - Enhanced Berkeley SPICE Level 2 (Level 17) Model, [4-145](#)
  - Inductor, [4-20](#)
  - Junction Diode, [4-102](#)
  - Junction Field Effect Transistor (JFET), [4-120](#)
  - Lossy Transmission Line, [3-18](#), [4-43](#)
    - U Model, [4-70](#)
    - U model, [3-18](#)
    - W Model, [4-59](#)
    - W model, [3-18](#)
  - Metal Field Effect Transistor (MESFET), [3-19](#), [4-123](#)
  - Metal Oxide Field Effect Transistor (MOSFET), [3-19](#)
  - MOSFET Models, [4-124](#)
    - Berkeley SPICE BSIM2, [4-143](#)
    - BTA HVMOS (Level =101), [4-166](#)
- EKV MOS (Level =EKV or 44), [4-145](#)
- Modified Berkeley SPICE Level 2 (Level 12), [4-144](#)
- Modified Berkeley SPICE Level 3 (Level 13), [4-144](#)
- Modified Lattin-Jenkins Grove Model (Level 16), [4-144](#)
- Motorola SSIM (Level 54 or SSIM) Model, [4-151](#)
- Philips MOS 9 (Level 59 or MOSP9) Model, [4-157](#)
- Philips PSP (Level 70) Model, [4-165](#)
- RC Wire, [4-28](#)
- Resistor, [4-3](#)
- S-Domain Filter, [3-19](#), [4-167](#)
- Semiconductor Resistor, [3-17](#), [4-34](#)
- Subcircuit Instance, [3-20](#), [4-171](#)
- TFT Amorphous-Si (Level 64) Model, [4-161](#)
- TFT Polysilicon (Level 62) Model, [4-159](#)
- Transmission Line, [3-17](#), [4-41](#)
- Z-Domain Filter, [3-19](#), [4-169](#)
- Device Model Description (.MODEL), [10-160](#)
- Device Models, [3-17](#) to [3-20](#), [4-1](#)
- Dialogue
  - of Pole-Zero Post-processor, [18-2](#)
- Differential Comparator Macromodel, [3-24](#), [6-3](#)
- Differential Operational Amplifier Macromodel
  - Linear, [6-5](#)
  - Linear 1-pole, [6-7](#)
  - Linear 2-pole, [6-11](#)
- Differential Output Level Detector Macromodel, [3-26](#), [6-48](#)
- Differentiated Accuracy System, [4-171](#), [4-172](#), [10-307](#)
- Differentiator Macromodel, [3-26](#), [6-54](#)
  - Parameters, [6-54](#)
- Digital Circuit Conditions
  - Initialization of (.INIT), [10-130](#)
- Digital Gate with Double Input Macromodel
- Digital Gate with Multiple Input Macromodel, [7-12](#)

- Digital Gate with Triple Input Macromodel, [7-11](#)  
Digital Macromodels, [3-26](#), [7-1](#)  
Digital Model Definition  
  .MODEL ... LOGIC, [7-3](#)  
  .MODLOGIC, [10-164](#)  
DIGITAL option, [11-56](#)  
Diode Models, [3-19](#), [4-102](#), [4-105](#)  
  .SENS, [10-278](#)  
  Berkeley Level 1, [4-106](#)  
  Fowler-Nordheim Model (Eldo Level 3),  
    [4-106](#)  
  Modified Berkeley Level 1 (Eldo Level 2),  
    [4-106](#)  
Disable Flat Netlist Model (.DISFLAT), [10-76](#)  
DISPLAY\_CARLO option, [11-45](#)  
DISTO (extract function), [10-102](#)  
Distribution Sharing  
  (.LOTGROUP), [10-142](#)  
Divider Macromodel, [3-26](#), [6-58](#)  
  Parameters, [6-58](#)  
DMAX (Maximum value), [3-8](#)  
DMIN (Minimum value), [3-8](#)  
Double Input AND Gate Macromodel, [3-27](#),  
  [7-9](#), [7-11](#), [7-12](#)  
Double Input Digital Gates  
Double Input NAND Gate Macromodel, [3-27](#),  
  [7-9](#), [7-11](#), [7-12](#)  
Double Input NOR Gate Macromodel, [3-27](#),  
  [7-9](#), [7-11](#), [7-12](#)  
Double Input OR Gate Macromodel, [3-27](#), [7-9](#),  
  [7-11](#), [7-12](#)  
Double Input XOR Gate Macromodel, [3-27](#),  
  [7-9](#), [7-11](#), [7-12](#)  
DPTRAN option, [11-57](#)  
DSCGLOB option, [11-25](#)  
DSP computation (.DSP), [10-78](#)  
DSPF files, [10-79](#)  
DSPF\_LEVEL option, [11-25](#)  
DTC (extract function), [10-102](#)  
DVDT option, [11-14](#)
- **E** —
- Effects of Error Messages, [A-1](#)  
Efficient Usage of Eldo, [4-172](#), [10-271](#),  
  [10-274](#), [10-307](#), [10-328](#), [11-13](#)  
EKV MOS Model (Level =EKV or 44), [4-145](#)  
Eldo  
  An Introduction, [1-1](#)  
  chi2cir utility, [E-1](#)  
  Control Language, [3-1](#)  
  convert .chi to .cir, [E-1](#)  
  Efficient Usage, [16-1](#)  
  Efficient Usage of, [4-172](#), [10-271](#), [10-274](#),  
    [10-307](#), [10-328](#), [11-13](#)  
  Encryption, [E-2](#)  
  Getting Started, [1-1](#)  
  Input and Output Files, [1-2](#)  
  Running from the Command Line, [2-1](#)  
  Running of, [1-4](#)  
  Speed and Accuracy, [16-1](#)  
  Syntax, [3-2](#)  
  Temperature Handling, [3-15](#)  
  Utilities, [E-1](#)  
eldo.ini, [2-11](#)  
ELDOMOS option, [11-31](#)  
Eldo-XL, [4-171](#)  
Encryption, [E-2](#)  
End Eldo Library Variant Description  
  (.ENDL), [10-90](#)  
End Eldo Netlist (.END), [10-89](#)  
End Eldo Subcircuit Description (.ENDS),  
  [10-91](#)  
ENGNOT option, [11-43](#)  
Enhanced Berkeley SPICE Level 2 (Level 17)  
  MOSFET Model, [4-145](#)  
EPS option, [4-172](#), [9-4](#), [10-307](#), [11-14](#), [B-1](#)  
EPSO option, [11-59](#)  
Error Messages, [A-1](#), [A-2](#)  
  Classification of, [A-1](#)  
  Effects, [A-1](#)  
  Global, [A-2](#)  
  Miscellaneous, [A-22](#)  
  Related to Amodels, [A-21](#)  
  Related to Commands, [A-13](#)  
  Related to Models, [A-20](#)  
  Related to Nodes, [A-5](#)  
  Related to Objects, [A-6](#)  
  Related to Subcircuits, [A-22](#)  
Euler Backward Integrator, [9-35](#)  
Euler Forward Integrator, [9-33](#)

- EVAL (extract function), [10-103](#)  
Examples, [C-1](#)  
    4-bit Adder, [C-3](#)  
    5th Order Elliptic SC Low Pass Filter, [C-11](#)  
    Active RC Band Pass Filter, [C-16](#)  
    Cascade of Inverters, [1-4](#)  
    Charge Control in MOS Models 4 and 6, [C-14](#)  
    CMOS Operational Amplifier (Closed Loop), [C-9](#)  
    CMOS Operational Amplifier (Open Loop), [C-7](#)  
    Post-Processing Library, [22-29](#)  
    SC—Schmitt Trigger, [C-2](#)  
    Second Order Delta Sigma Modulator, [C-19](#)  
Exclusive-OR Gate, [7-8](#)  
EXP (Exponent of value), [3-7](#)  
Exponential Function, [3-21](#), [5-19](#)  
Exponential Pulse With Bit Pattern Function (EBIT), [3-22](#)  
Expressions in Eldo, [3-12](#)  
EXTCGS option, [11-45](#)  
EXTFILE option, [11-45](#)  
EXTMKSA option, [11-45](#)  
Extract Mode (.EXTMOD), [10-94](#)  
Extract Waveform Characteristics (.EXTRACT), [10-95](#)  
Extract Waveform Characteristics (.MEAS), [10-155](#)  
EZwave, [2-4](#)
- F —**
- FALL\_TIME option, [11-26](#)  
FAS  
    Macromodel Usage, [6-2](#)  
FASTRLC option, [11-14](#)  
Feedback Loop  
    Insertion of (.LOOP), [10-140](#)  
FFT  
    post-processor options (.OPTFOUR), [10-192](#)  
    select waveform (.FOUR), [10-119](#)  
5th Order Elliptic SC Low Pass Filter Example, [C-11](#)  
First Line in Eldo, [3-2](#)
- FLICKER\_NOISE option, [11-41](#)  
FLOOR (Value rounded down to integer), [3-8](#)  
FLUXTOL option, [11-14](#)  
FNLEV option, [11-32](#)  
FNS Model, [18-4](#)  
4-bit Adder Example, [C-3](#)  
Fowler-Nordheim Diode Model (Eldo Level 3), [4-106](#)  
    Parameters, [4-107](#)  
FREQSMP option, [11-14](#)  
Frequency Limit  
    for Pole-Zero Post-processing, [18-3](#)  
FT option, [11-15](#)  
Functions & Operators  
    see Arithmetic Functions & Operators
- G —**
- GA, [10-244](#)  
GAC, [10-243](#)  
Gain Extract  
    see Two-port Gain Extract  
GAM, [10-244](#)  
GASM, [10-244](#)  
GAUM, [10-245](#)  
GEAR option, [11-55](#)  
Generalized Re-run Facility, [10-22](#)  
GENK option, [11-33](#), [12-6](#)  
Getting Started  
    with Eldo, [1-1](#)  
Global  
    Declarations (.PARAM), [10-205](#)  
    Error Messages, [A-2](#)  
    Node Allocation (.GLOBAL), [10-121](#)  
    Warning Messages, [A-26](#)  
GMIN option, [11-32](#)  
GMIN\_BJT\_SPICE option, [11-32](#)  
GMINDC option, [11-32](#)  
GNODE option, [11-55](#)  
GP, [10-245](#)  
GPC, [10-244](#)  
GRAMP option, [11-33](#)  
Grounded Capacitors Information, [3-14](#)  
GSHUNT option, [11-57](#)
- H —**
- Hand Selection

in Pole-Zero Post-processing, 18-3  
HICUM Model  
Parameters, 4-116, 4-117, 4-119  
Hierarchical Nodes  
Monitoring of, 10-241  
Hierarchy Separator  
changing (.HIER), 10-123  
High Voltage Cascade Tutorial, 24-19  
High-rate Particle Detector Circuit, 14-4  
HIGHVOLTAGE option, 11-27  
HIGHVTH option, 11-27  
HiSIM (Eldo Level 66) MOSFET Model, 4-162  
HISTLIM option, 11-46  
Histogram computation (.DSPMOD), 10-85  
HMAX option, 11-15  
HMIN option, 11-15  
How to Run Eldo, 1-4  
HRISEFALL option, 11-15

— | —

IBIS  
Detailed Syntax, 23-6  
General Syntax, 23-1  
Input Buffers, 23-9  
input\_ecl, 23-10  
IO buffer, 23-10  
IO ECL buffer, 23-10  
Output Buffer, 23-6  
Output ECL buffer, 23-8  
tristate buffer, 23-10  
tristate ECL buffer, 23-11  
IBIS\_SEARCH\_PATH option, 11-33  
ICDC option, 11-28  
ICDEV option, 11-28  
Ideal Operational Amplifier Macromodel, 3-28, 9-12  
IDT (Integral of value), 3-8  
IEM, 17-1 to 17-20  
Accuracy  
RELTOL, 11-20  
IEM option, 11-55  
IKF2 option, 11-41  
IMAG (Imaginary part of complex number), 3-8  
Impedance parameters, 15-1

INCLIB option, 11-15  
Include a File in an Input Netlist (.INCLUDE), 10-128  
Independent Current Source, 5-11  
.LOOP, 10-140  
.SENS, 10-278  
Independent Sources, 5-1  
Independent Voltage Source, 5-4  
.LOOP, 10-140  
.SENS, 10-278  
Multi-tone, 5-5, 5-12  
Inductor Model, 3-17, 4-20, 4-25  
Parameters, 4-25  
INFODEV option, 11-53  
INFOMOD option, 11-52  
INGOLD option, 11-43  
Initial DC Analysis Conditions (.GUESS), 10-122  
Initial Digital Circuit Conditions (.INIT), 10-130  
Initial Transient Analysis Conditions (.FORCE), 10-118  
Initial Transient Analysis Conditions (.IC), 10-125  
Initialization file, 2-11  
INOISE, 10-224  
Input and Output Files for Eldo, 1-2  
Input from a Prior Simulation (.CHRSIM), 10-37  
INPUT option, 11-46  
Insert  
a Feedback Loop (.LOOP), 10-140  
a Model or Subcircuit File, 10-18  
Circuit Information from a Library File  
see Library Files—Insertion of (.LIB)  
INT (Integer of value), 3-8  
INTEG (extract function), 10-103  
Integral Equation Method  
see IEM  
Integrator Macromodel, 3-26, 6-56  
Parameters, 6-56  
Interactive Mode, D-1, G-1  
INTERP option, 11-27  
Interruption of a Simulation, 16-24  
Inverter Macromodel, 3-27, 7-7

Inverting Switched Capacitor Macromodel,  
3-28, 9-15

Equations, 9-16

Parameters, 9-15

ISDB, 2-4

ISDB option, 11-53

Iteration Techniques

Newton Raphson, 4-171, 10-307, 11-56

One Step Relaxation, 11-56

ITL1 option, 11-15

ITL3 option, 11-15

ITL4 option, 11-15

ITL6 option, 11-15

ITL7 option, 11-15

ITL8 option, 11-16

ITOL option, 11-16

— J —

JFET Model, 4-122

Parameters, 4-122

JTHNOISE option, 11-41

JUNCAP Level 8 Diode Model

Parameters, 4-107

Junction Diode Models, 4-102, 4-105

Berkeley Level 1, 4-106

STMicroelectronics, H-2

Junction Field Effect Transistor (JFET) Model,  
3-19, 4-120

JWDB, 1-2, 2-3

JWDB option, 11-53

— K —

Keywords, reserved, 3-3

KFACTOR, 10-242

KFACTOR (extract function), 10-103

KLIM option, 11-33

KWSCALE option, 11-38, 12-5

— L —

LCAPOP option, 11-46

LDI Definition, 9-11

LDI Phase Control Euler Backward Integrator,  
9-36

LDI Phase Control Euler Forward Integrator,  
9-34

LDI Phase Control Non-inverting Integrator,  
9-32

LDTL

see Lossy Transmission Line

Level Detector Macromodel, 6-48

Differential Output Level Detector, 6-48

Parameters, 6-48

Single Output Level Detector, 6-48

LIBINC option, 11-16, 12-6

Library Files

Insertion of (.LIB), 10-133

Library variant Description Termination  
(.ENDL), 10-90

LICN option, 11-28

LIMIT, 3-8

LIMNWRMOS option, 11-16

LIMPROBE option, 10-258, 11-46

Linear Dependent Sources, 5-1

Linear Magnetic Core Macromodel, 3-28, 8-9

LIST option, 11-46

Load DSPF File (.DSPF\_INCLUDE), 10-79

Loading Large Databases, 2-3

LOG (Neperian log of value), 3-7

LOG10 (Decimal log of value), 3-7

Logarithmic Amplifier Macromodel, 3-26,  
6-50

Parameters, 6-50

Loop Stability Analysis (.LSTB), 10-144

Lossy Transmission Line Model, 4-43

LDTL, 4-43

Level 1, 4-43

Level 2, 4-45

Level 3, 4-46

Level 4, 4-48

U Model, 3-18, 4-70

W Model, 3-18, 4-59

LOT & DEV Variation Specification on Model  
Parameters (Monte Carlo) (.MCMOD),  
10-153

Low Pass Filter Tutorial, 24-13

LOWVOLTAGE option, 11-27

LOWVTH option, 11-27

LSC

see Two-port Stability Circles

LSF, 10-170

LVLTIM option, 11-16

— M —

M53 option, 11-28

Macro Definition (.DEFMAC), 10-66

Macromodel Description

  2-Input Digital Gates, 3-27

  3-Input Digital Gates, 3-27, 7-11

  A-D Converter, 3-27, 7-14

  Adder, 3-26, 6-58

  Adder, Subtractor, Multiplier, Divider,  
    3-26, 6-58

  Amplitude Modulator, 3-25, 6-32

  Anti-logarithmic Amplifier, 3-26, 6-52

  Bi-linear Switched Capacitor, 3-29, 9-26

  Comparator, 3-24, 6-3

  Current Controlled Switch, 3-25, 6-21

  D-A Converter, 3-27, 7-16

  Delay, 3-24, 3-26, 6-14, 7-6

  Differential Comparator, 3-24, 6-3

  Differential Operational Amplifier

    Linear, 3-24, 6-5

    Linear 1-pole, 3-24, 6-7

    Linear 2-pole, 3-24, 6-11

  Differential Output Level Detector, 3-26,  
    6-48

  Differentiator, 3-26, 6-54

  Divider, 3-26, 6-58

  Double Input Digital Gates, 3-27

  Exclusive-OR Gate, 3-27, 7-8

  Ideal Operational Amplifier, 3-28, 9-12

  Ideal Transformer, 3-28, 8-13

  Integrator, 3-26, 6-56

  Inverter, 3-27, 7-7

  Inverting Switched Capacitor, 3-28, 9-15

  Level Detector, 3-26, 6-48

    Differential Output, 3-26, 6-48

    Single Output, 3-26, 6-48

  Linear Magnetic Core, 3-28, 8-9

  Logarithmic Amplifier, 3-26, 6-50

  Magnetic Air Gap, 3-28, 8-10

  Mixed Signal Macromodels, 7-13

  Multiple Input Digital Gates, 3-27, 7-12

  Multiplier, 3-26, 6-58

  Non-inverting Switched Capacitor, 3-28,  
    9-17

  Non-linear Magnetic Core 1, 3-28, 8-3

  Non-linear Magnetic Core 2, 3-28, 8-6

  Operational Amplifier

    Linear, 3-24, 6-5

    Linear 1-pole, 3-24, 6-7

    Linear 2-pole, 3-24, 6-11

    Switched Capacitor, 9-3

  Operational Amplifier (SC), 3-28

  Parallel Switched Capacitor, 3-29, 9-19

  Peak Detector, 3-26, 6-45

  Pulse Amplitude Modulator, 3-25, 6-34

  Pulse Width Modulator, 3-25, 6-40

  Sample and Hold, 3-25, 6-36

  Saturating Resistor, 3-24, 6-15

  Sawtooth Waveform Generator, 3-25, 6-28

  Serial Switched Capacitor, 3-29, 9-21

  Serial-parallel Switched Capacitor, 3-29,  
    9-23

  Single Output Level Detector, 3-26, 6-48

  Staircase Waveform Generator, 3-25, 6-26

  Subtractor, 3-26, 6-58

  Switch (SC), 3-28, 9-8

  Track and Hold, 3-25, 6-38

  Transformer Winding, 3-27, 8-2

  Transformer with Variable Number of  
    Windings, 3-28, 8-11

  Triangle Waveform Generator, 3-25, 6-30

  Triangular to Sine Wave Converter, 3-25,  
    6-24

  Triple Input Digital Gates, 3-27, 7-11

  Unswitched Capacitor, 3-29, 9-28

  Voltage Controlled Oscillator (VCO), 3-26,  
    6-43

  Voltage Controlled Switch, 3-25, 6-19

  Voltage Limiter, 3-25, 6-17

Macromodels, 6-1 to 6-59, 9-1 to 9-29

  Analog, 3-24 to 3-26, 6-1

  Digital, 3-26, 7-1

  General Notes on the Use of, 9-11

  Magnetic, 3-27, 8-1

  Mixed, 7-13

  Mixed Signal, 3-27

  Switched Capacitor, 3-28, 9-1

  Magnetic Air Gap Macromodel, 3-28, 8-10

  Magnetic Macromodels, 3-27, 8-1

- MAGNITUDE (Magnitude of complex number), [3-8](#)
- Mapping Model Names (.DEFMOD), [10-68](#)
- Mapping Model Names (.MALIAS), [10-145](#)
- Matrix Information in Eldo, [3-14](#)
- MAX (extract function), [10-103](#)
- MAX (Maximum value), [3-8](#)
- MAX\_DSPF\_PLOT option, [11-46](#)
- MAXADS option, [11-33](#)
- MAXL option, [11-33](#)
- MAXNODEORD option, [11-59](#)
- MAXNODES option, [11-17](#)
- MAXORD option, [11-55](#)
- MAXTRAN option, [11-17](#)
- MAXV option, [11-17](#)
- MAXW option, [11-34](#)
- MERCKEL MOSFET Models (Levels 4-6), [4-140](#)
- Parameters, [4-141](#)
- MESFET Model, [3-19](#), [4-123](#)
- Parameters, [4-123](#)
- Metal Field Effect Transistor (MESFET) Model, [3-19](#), [4-123](#)
- Metal Oxide Field Effect Transistor (MOSFET) Model, [3-19](#)
- METHOD=GEAR option, [11-55](#)
- Mextram 503.2 Model, [4-114](#)
- Mextram 504 Model, [4-117](#)
- Microstrip Models, [4-76](#) to [4-101](#)
- 90-degree Microstrip Bend, [4-83](#), [4-85](#), [4-88](#)
- Cylindrical Via Hole in Microstrip, [4-92](#)
- MBEND, [4-80](#)
- MBEND2, [4-83](#)
- MBEND3, [4-85](#)
- MCORN, [4-88](#)
- Microstrip Bend, [4-80](#)
- Microstrip Step in Width (MSTEP), [4-90](#)
- Microstrip T Junction, [4-77](#)
- MTEE, [4-77](#)
- Stripline Step in Width (SSTEP), [4-99](#)
- Stripline T Junction (STEE), [4-97](#)
- Unmitered Stripline Bend (SBEND), [4-95](#)
- VIA2, [4-92](#)
- MIN (extract function), [10-103](#)
- MIN (Minimum value), [3-8](#)
- MINADS option, [11-34](#)
- MINL option, [11-34](#)
- MINPDS option, [11-34](#)
- MINRACC option, [11-34](#)
- MINRESISTANCE option, [11-34](#)
- MINRVAL option, [11-34](#)
- MINW option, [11-34](#)
- Miscellaneous
- Error Messages, [A-22](#)
- Warning Messages, [A-40](#)
- Mixed Signal Macromodels, [3-27](#), [7-13](#)
- MIXEDSTEP option, [11-57](#)
- MNUMER option, [11-35](#)
- MOD4PINS option, [11-35](#)
- Model
- Error Messages, [A-20](#)
- File Insertion (.ADDLIB), [10-18](#)
- Names in Eldo, [3-5](#)
- Warning Messages, [A-38](#)
- Modella Model, [4-117](#)
- Models
- Berkeley BSIM3SOI (Level 55), [4-151](#)
- Berkeley BSIM3SOI (Level 56), [4-154](#)
- Berkeley BSIM3v2 (Level 47), [4-146](#)
- Berkeley BSIM3v3 (Level 53), [4-147](#)
- Berkeley BSIM4 (Level 60), [4-158](#)
- Berkeley BSIM5 (Level 68), [4-164](#)
- Berkeley SPICE BSIM1, [4-142](#)
- Berkeley SPICE BSIM2, [4-143](#)
- Bipolar Junction Transistor (BJT), [4-111](#), [4-114](#)
- BJT, [3-19](#)
- BTA HVMOS (Level =101), [4-166](#)
- Capacitor, [3-17](#), [4-16](#)
- Comparator, [6-4](#)
- Coupled Inductor, [3-17](#), [4-27](#)
- Diode, [4-105](#)
- EKV MOS (Level=EKV or 44), [4-145](#)
- Enhanced Berkeley SPICE Level 2 (Level 17), [4-145](#)
- HiSIM MOSFET (Eldo Level 66), [4-162](#)
- Inductor, [3-17](#), [4-25](#)
- JFET, [3-19](#)
- Junction Diode, [3-19](#), [4-105](#)

- Junction Field Effect Transistor (JFET), [4-122](#)  
Lossy Transmission Line, [3-18](#), [4-43](#)  
    Level 1, [4-43](#)  
    Level 2, [4-45](#)  
    Level 3, [4-46](#)  
    Level 4, [4-48](#)  
    U Model, [3-18](#), [4-70](#)  
    W Model, [3-18](#), [4-59](#)  
MESFET, [3-19](#), [4-123](#)  
Microstrip, [4-76](#) to [4-101](#)  
Microstrip Bend, [3-18](#)  
    Milttered, [3-18](#)  
    Optimally milttered, [3-18](#)  
    Unmilttered, [3-18](#)  
Microstrip T junction, [3-18](#)  
Modified Berkeley Level 2 (Level 12), [4-144](#)  
Modified Berkeley Level 3 (Level 13), [4-144](#)  
Modified Lattin-Jenkins Grove (Level 16), [4-144](#)  
MOSFET, [3-19](#), [4-124](#)  
Motorola SSIM (Level 54 or SSIM), [4-151](#)  
Philips MOS 11 Level 1100 (Eldo Level 65 or MOSP11), [4-162](#)  
Philips MOS 11 Level 1100 (Eldo Level 69), [4-164](#)  
Philips MOS 11 Level 1101 (Eldo Level 63 or MOSP11), [4-160](#)  
Philips MOS 9 (Level 59 or MOSP9), [4-157](#)  
Philips PSP (Level 70), [4-165](#)  
RC Wire, [3-17](#), [4-30](#)  
Resistor, [3-17](#), [4-8](#)  
S-Domain Filter, [3-19](#), [4-167](#)  
Semiconductor Resistor, [3-17](#), [4-34](#)  
SP MOSFET (Eldo Level 67), [4-163](#)  
STMicroelectronics, [H-1](#)  
Subcircuit Instance, [3-20](#), [4-171](#)  
TFT Amorphous-Si (Level 64), [4-161](#)  
TFT Polysilicon (Level 62), [4-159](#)  
Transmission Line, [3-17](#), [4-41](#)  
Z-Domain Filter, [3-19](#), [4-169](#)
- Modified Berkeley Level 1 (Eldo Level 2)  
    Diode Model, [4-106](#)  
    Parameters, [4-106](#)  
Modified Berkeley SPICE Level 2 (Level 12)  
    MOSFET Model, [4-144](#)  
    Parameters, [4-144](#)  
Modified Berkeley SPICE Level 3 (Level 13)  
    MOSFET Model, [4-144](#)  
    Parameters, [4-144](#)  
Modified Gummel-Poon Parameters, [4-114](#)  
MODPAR (extract function), [10-104](#)  
MODWL option, [11-35](#)  
MODWLDT option, [11-36](#)  
Monitor Simulation Steps (.MONITOR), [10-165](#)  
Monitoring of Hierarchical Nodes, [10-241](#)  
Monte Carlo Analysis (.MC), [10-147](#)  
    Capacitor Model, [4-16](#)  
    Examples, [24-23](#)  
    Inductor Model, [4-25](#)  
    Multiple Runs, [10-151](#)  
MOSFET Models, [4-124](#)  
    Berkeley BSIM3SOI (Level 55), [4-151](#)  
    Berkeley BSIM3SOI (Level 56), [4-154](#)  
    Berkeley BSIM3v2 (Level 47), [4-146](#)  
    Berkeley BSIM3v3 (Level 53), [4-147](#)  
    Berkeley BSIM4 (Level 60), [4-158](#)  
    Berkeley BSIM5 (Level 68), [4-164](#)  
    Berkeley SPICE BSIM1, [4-142](#)  
    Berkeley SPICE BSIM2, [4-143](#)  
    Berkeley SPICE Levels 1-3, [4-139](#)  
    BTA HVMOS (Level =101), [4-166](#)  
    EKV MOS (Level =EKV or 44), [4-145](#)  
    Enhanced Berkeley SPICE Level 2 (Level 17), [4-145](#)  
    HiSIM (Eldo Level 66), [4-162](#)  
    MERCKEL MOS Levels 4-6, [4-140](#)  
    Modified Berkeley SPICE Level 2 (Level 12), [4-144](#)  
    Modified Berkeley SPICE Level 3 (Level 13), [4-144](#)  
    Modified Lattin-Jenkins Grove (Level 16), [4-144](#)  
    Motorola SSIM (Level 54 or SSIM), [4-151](#)

Philips MOS 11 Level 1100 (Eldo Level 65 or MOSP11), [4-162](#)  
Philips MOS 11 Level 1100 (Eldo Level 69), [4-164](#)  
Philips MOS 11 Level 1101 (Eldo Level 63 or MOSP11), [4-160](#)  
Philips MOS 9 (Level 59 or MOSP9), [4-157](#)  
Philips PSP (Level 70), [4-165](#)  
SP (Eldo Level 67), [4-163](#)  
STMicroelectronics, [H-3](#)  
TFT Amorphous-Si (Level 64), [4-161](#)  
TFT Polysilicon (Level 62), [4-159](#)  
UDMP, [4-130](#)  
**MOTOROLA**  
option, [11-7](#)  
SSIM (Level 54 or SSIM) MOSFET Model, [4-151](#)  
version of Eldo, [2-10](#)  
MSGBIAS option, [11-42](#)  
MSGNODE option, [11-42](#)  
MTHREAD option, [11-8](#)  
Multiple Affectation of Parameters  
.PARAM  
-compat flag, [10-213](#)  
Multiple Input AND Gate Macromodel, [3-27](#)  
Multiple Input Digital Gates, [3-27](#), [7-12](#)  
Multiple Input NAND Gate Macromodel, [3-27](#)  
Multiple Input NOR Gate Macromodel, [3-27](#)  
Multiple Input OR Gate Macromodel, [3-27](#)  
Multiplier Macromodel, [3-26](#), [6-58](#)  
Parameters, [6-58](#)  
Multi-tone, [5-5](#), [5-12](#)

— **N** —

NAND Gate  
see Double, Triple and Multiple Input Digital Gates

Nested Output Requests, [10-241](#)  
Netlist Protection  
.PROTECT, [10-267](#)  
.UNPROTECT, [10-327](#)  
Netlist Termination (.END), [10-89](#)  
NETSIZE option, [11-17](#)  
Network Analysis  
(.NET), [3-36](#), [10-176](#)

NEWACCT option, [11-47](#)  
Newton Block Information in Eldo, [3-14](#)  
NEWTON option, [11-55](#)  
Newton Raphson Accuracy  
RELTOL, [11-20](#)  
NGATEDDEF option, [11-36](#)  
NGTOL option, [11-17](#)  
NMAXSIZE option, [11-18](#)  
NOAEX option, [11-52](#)  
NOASCII option, [11-44](#)  
NOAUTOCTYPE option, [11-36](#)  
NOBOUND\_PHASE option, [11-47](#)  
NOBSLASHCONT option, [11-8](#)  
NOCKRSTS option, [11-53](#)  
NOCMPUNIX option, [11-8](#), [11-9](#)  
NOCONVASSIST option, [11-18](#)  
NOCOU option, [11-53](#)  
Nodal  
Error Messages, [A-5](#)  
Warning Messages, [A-28](#)  
NODCINFOTAB option, [11-47](#)  
NODCPART option, [11-57](#)  
Node and Element Information in Eldo, [3-13](#)  
Node Names  
inside Subcircuits, [10-310](#)  
NODE option, [11-47](#)  
NODEFNEWTON option, [11-55](#)  
Nodes  
Global (.GLOBAL), [10-121](#)  
Node Names in Eldo, [3-4](#)  
Node Names inside Subcircuits, [3-4](#)  
NOELDOSWITCH option, [11-59](#), [12-6](#)  
NOEXTRACTCOMPLEX option, [11-48](#)  
NOFNSIEM option, [11-60](#)  
NOIICXNAME option, [11-53](#)  
NOINIT option, [11-60](#)  
Noise  
Function, [3-22](#), [5-21](#)  
Performance Analysis, [14-5](#)  
Noise Analysis  
.NOISE, [10-180](#)  
.PRINT, [10-229](#), [10-236](#)  
AC (.OPTNOISE), [10-199](#)

- NONOISE, 4-5, 4-103, 4-110, 4-121, 4-126, 4-172, 5-8, 5-15  
Transient, 14-1  
Noise Circles  
  see Two-port Noise Circles  
Noise Function, 3-22  
Noise Parameters  
  see Two-port Noise Parameters  
NOJWDB option, 11-53  
NOKEYWPARAMSST option, 11-9  
NOKWSCALE option, 11-38, 12-5  
NOLAT option, 11-18  
NOMEMSTP option, 11-29  
NOMOD option, 11-48  
Non-inverting Amplifier Tutorial, 24-23  
Non-inverting Integrator, 9-31  
Non-inverting Switched Capacitor  
  Macromodel, 3-28, 9-17  
  Equations, 9-18  
  Parameters, 9-17  
Non-linear Dependent Sources, 5-2  
Non-linear Magnetic Core 1 Macromodel, 3-28, 8-3  
Non-linear Magnetic Core 2 Macromodel, 3-28, 8-6  
NONOISE option, 4-5, 4-103, 4-110, 4-121, 4-126, 4-172, 5-8, 5-15, 11-42  
NONWRMOS option, 11-18  
NOOP option, 11-48  
NOPAGE option, 11-48  
NOPROBEOP option, 11-53  
NOQTRUNC option, 11-18  
NOR Gate  
  see Double, Triple and Multiple Input Digital Gates  
NORMOS option, 11-56  
NOSIZECHK option, 11-48  
NOSSTKEYWORD option, 11-9  
NOSTATP option, 11-50  
NOSWITCH option, 11-18  
NOTRC option, 11-48  
NOTRCLIB option, 11-43  
NOWARN option, 11-42  
NOWAVECOMPLEX option, 11-48  
NOXTABNOISE option, 11-48  
  
NUMDGT option, 11-43  
NWRMOS option, 11-36  
  
— O —  
Object  
  Error Messages, A-6  
  Warning Messages, A-30  
ONOISE, 10-224  
Operating Point  
  Calculation of, 16-19  
  DC, 11-17, 11-23  
Operational Amplifier (SC) Macromodel, 3-28, 9-3  
  Equations (Single Stage), 9-5  
  Equations (Two Stage), 9-5  
  Equivalent Circuit, 9-5  
  Parameters, 9-6  
Operational Amplifier Macromodel  
  Linear, 3-24, 6-5  
    Characteristics, 6-6  
    Parameters, 6-5  
  Linear 1-pole, 3-24, 6-7  
    Application Area, 6-9  
    Characteristics, 6-8  
    Parameters, 6-7  
  Linear 2-pole, 3-24, 6-11  
    Characteristics, 6-12  
    Parameters, 6-11  
Operator Precedence, 3-10  
Operators, 3-10  
Operators in Eldo, 3-10  
OPMODE (extract function), 10-104  
OPSELD0\_ABSTRACT option, 11-51, 20-45  
OPSELD0\_DETAIL option, 11-51, 20-45  
OPSELD0\_DISPLAY\_GAOFITTING option, 11-51, 20-45  
OPSELD0\_FORCE\_GAOFITTING option, 11-51, 20-45  
OPSELD0\_NETLIST option, 11-51, 20-45  
OPSELD0\_NOGAOFITTING option, 11-51, 20-46  
OPSELD0\_OUTER option, 11-51, 20-46  
OPSELD0\_OUTPUT option, 11-51, 20-46  
Optimization  
  .OPTIMIZE, 10-197  
Optimizer, 20-1

- .OPTIMIZE, [10-197](#)
- Overview, [20-1](#)
- Options
  - CADENCE Compatibility
    - SDA, [11-6](#)
    - WSF, [11-6](#)
    - WSFASCII, [11-6](#)
  - File Generation
    - AEX, [11-52](#)
    - ALIGNEXT, [11-52](#)
    - ASCII, [11-52](#)
    - COU, [11-52](#)
    - CSDF, [11-52](#)
    - INFODEV, [11-53](#)
    - INFOMOD, [11-52](#)
    - ISDB, [11-53](#)
    - JWDB, [11-53](#)
    - NOAEX, [11-52](#)
    - NOCKRSTSAVE, [11-53](#)
    - NOCOU, [11-53](#)
    - NOIICXNAME, [11-53](#)
    - NOJWDB, [11-53](#)
    - NOPROBEOP, [11-53](#)
    - OUT\_ABSTOL, [11-54](#)
    - OUT\_REDUCE, [11-53](#)
    - OUT\_RELTOL, [11-54](#)
    - PROBE, [11-54](#), [12-5](#)
    - PROBEOP, [11-54](#)
    - PROBEOP2, [11-54](#)
    - PSF, [11-54](#)
    - PSFASCII, [11-54](#)
    - SAVETIME, [11-54](#)
  - Mathematical Algorithm
    - ANALOG, [11-56](#)
    - BE, [11-55](#)
    - BLOCKS, [11-56](#)
    - CSHUNT, [11-57](#)
    - DCPART, [11-57](#)
    - DIGITAL, [11-56](#)
    - GEAR, [11-55](#)
    - GNODE, [11-55](#)
    - GSHUNT, [11-57](#)
    - IEM, [11-55](#)
    - MAXORD, [11-55](#)
    - METHOD, [11-55](#)
  - Newton Method
    - NEWTON, [11-55](#)
    - NODCPART, [11-57](#)
    - NODEFNEWTON, [11-55](#)
    - NORMOS, [11-56](#)
    - OPTRAN, [11-57](#)
    - OSR, [11-56](#)
    - PSTRAN, [11-56](#)
    - SMOOTH, [11-55](#)
    - TRAP, [11-55](#)
  - Miscellaneous Simulation Control
    - AMMETER, [11-24](#)
    - AUTOSTOP, [11-24](#)
    - AUTOSTOPMODULO, [11-24](#)
    - CARLO\_GAUSS, [11-24](#)
    - CPTIME, [11-25](#)
    - DEFPTNOM, [11-25](#)
    - DSCGLOB, [11-25](#)
    - DSPF\_LEVEL, [11-25](#)
    - FALL\_TIME, [11-26](#)
    - HIGHVOLTAGE, [11-27](#)
    - HIGHVTH, [11-27](#)
    - ICDC, [11-28](#)
    - ICDEV, [11-28](#)
    - INTERP, [11-27](#)
    - LICN, [11-28](#)
    - LOWVOLTAGE, [11-27](#)
    - LOWVTH, [11-27](#)
    - M53, [11-28](#)
    - NOMEMSTP, [11-29](#)
    - OPSELD\_O\_OUTER, [20-46](#)
    - PARAMOPT\_NOINITIAL, [11-29](#)
    - RANDMC, [11-29](#)
    - RGND, [11-29](#)
    - RGNDI, [11-29](#)
    - RISE\_TIME, [11-26](#)
    - SIGTAIL, [11-30](#)
    - TNOM, [11-30](#)
    - TPIEEE, [11-30](#)
    - ULOGIC, [11-30](#)
    - ZOOMTIME, [11-30](#)
  - Mixed-Mode
    - D2DMVL9BIT, [11-57](#)
    - DEFA2D, [11-58](#)
    - DEFCONVMSG, [11-58](#)
    - DEFD2A, [11-58](#)

- MIXEDSTEP, 11-57  
Model Control  
  ACM, 11-30  
  ASPEC, 11-31  
  BSIM3VER, 11-31  
  DEFAD, 11-31  
  DEFAS, 11-31  
  DEFL, 11-31  
  DEFNRD, 11-31  
  DEFNRS, 11-31  
  DEFPD, 11-31  
  DEFPS, 11-31  
  DEFW, 11-31  
  ELDOMOS, 11-31  
  FNLEV, 11-32  
  GENK, 11-33, 12-6  
  GMIN, 11-32  
  GMIN\_BJT\_SPICE, 11-32  
  GMINDC, 11-32  
  GRAMP, 11-33  
  IBIS\_SEARCH\_PATH, 11-33  
  KLIM, 11-33  
  KWSCALE, 11-38, 12-5  
  MAXADS, 11-33  
  MAXL, 11-33  
  MAXPDS, 11-33  
  MAXW, 11-34  
  MINADS, 11-34  
  MINL, 11-34  
  MINPDS, 11-34  
  MINRACC, 11-34  
  MINRESISTANCE, 11-34  
  MINRVAL, 11-34  
  MINW, 11-34  
  MNUMER, 11-35  
  MOD4PINS, 11-35  
  MODWL, 11-35  
  MODWLDOT, 11-36  
  NGATEDEF, 11-36  
  NOAUTOCODE, 11-36  
  NOKWSCALE, 11-38, 12-5  
  NWRMOS, 11-36  
  PGATEDEF, 11-36  
  RAILRESISTANCE, 11-36  
  REDUCE, 11-36  
  RESNW, 11-37  
  RMMINRVAL, 11-37  
  RMOS, 11-37  
  SCALE, 11-37  
  SCALEBSIM, 11-38  
  SCALM, 11-38  
  SOIBACK, 11-38  
  SPMODLEV, 11-38  
  TMAX, 11-39  
  TMIN, 11-39  
  USEDEFAP, 11-39  
  VBICLEV, 11-39  
  WARNMAXV, 11-39  
  WL, 11-39  
  YMFAC, 11-39  
Netlist Parser Control  
  ALTINC, 11-7  
  BSLASHCONT, 11-7, 12-6  
  CHECKDUPL, 11-8  
  CNTTHREAD, 11-8  
  COMPEXUP, 11-8  
  CONTINUE\_INCLUDE, 11-8  
  MTHREAD, 11-8  
  NOBSLASHCONT, 11-8  
  NOCMPUNIX, 11-8  
  NOKEYPARAMSST, 11-9  
  NOSSTKEYWORD, 11-9  
  NOZSINXX, 11-9  
  PARHIER, 11-9, 12-6  
  PSTRICT, 11-11  
  STOPONFIRSTERROR, 11-12  
  SUBFLAGPAR, 11-12  
  USETHREAD, 11-13  
Noise Analysis  
  FLICKER\_NOISE, 11-41  
  IKF2, 11-41  
  NONOISE, 11-42  
  THERMAL\_NOISE, 11-41  
  UTHNOISE, 11-41  
Optimizer Output Control  
  OPSEUDO\_ABSTRACT, 11-51  
  OPSEUDO\_DETAIL, 11-51  
  OPSEUDO\_DISPLAY\_GOALFITTING  
    G, 11-51

|                                     |                            |
|-------------------------------------|----------------------------|
| OPSELDO_FORCE_GOALFITTING,<br>11-51 | LIMNWRMOS, 11-16           |
| OPSELDO_NETLIST, 11-51              | LVLTIM, 11-16              |
| OPSELDO_NOGOALFITTING,<br>11-51     | MAXNODES, 11-17            |
| OPSELDO_OUTER, 11-51                | MAXTRAN, 11-17             |
| OPSELDO_OUTPUT, 11-51               | MAXV, 11-17                |
| RESET_MULTIPLE_RUN, 11-52           | NETSIZE, 11-17             |
| Other                               | NGTOL, 11-17               |
| CTEPREC, 11-59                      | NMAXSIZE, 11-18            |
| DCLOG, 11-59                        | NOCONVASSIST, 11-18        |
| EPSO, 11-59                         | NOLAT, 11-18               |
| MAXNODEORD, 11-59                   | NONWRMOS, 11-18            |
| NOFNSIEM, 11-60                     | NOQTRUNC, 11-18            |
| NOINIT, 11-60                       | NOSWITCH, 11-18            |
| NOOP, 11-48                         | PCS, 11-19                 |
| SEARCH, 11-60                       | PCSPERIOD, 11-19           |
| VAMAXEXP, 11-60                     | PCSSIZE, 11-19             |
| ZCHAR, 11-60                        | PIVREL, 11-19              |
| PRECISE Compatibility               | PIVTOL, 11-20              |
| PRECISE, 11-6                       | PSOSC, 11-20               |
| Simulation Accuracy & Efficiency    | QTRUNC, 11-20              |
| ABSTOL, 11-13                       | RATPRINT, 11-20            |
| ABSVAR, 11-13                       | RELTOL, 11-20              |
| ADJSTEPTRAN, 11-13                  | RELTRUNC, 11-21            |
| CAPANW, 11-13                       | RELVAR, 11-21              |
| CHGTOL, 11-13                       | SAMPLE, 11-21              |
| DVDT, 11-14                         | SPLITC, 11-21              |
| EPS, 11-14                          | STARTSMP, 11-21            |
| FASTRLC, 11-14                      | STEP, 11-21                |
| FLUXTOL, 11-14                      | TIMESMP, 11-22             |
| FREQSMP, 11-14                      | TRTOL, 11-22               |
| FT, 11-15                           | TUNING, 11-22              |
| HMAX, 11-15                         | UNBOUND, 11-23             |
| HMIN, 11-15                         | VMAX, 11-23                |
| HRISEFALL, 11-15                    | VMIN, 11-23                |
| INCLIB, 11-15                       | VNTOL, 11-23               |
| ITL1, 11-15, 11-18                  | XA, 11-24                  |
| ITL3, 11-15                         | Simulation Display Control |
| ITL4, 11-15                         | ENGNOT, 11-43              |
| ITL6, 11-15                         | INGOLD, 11-43              |
| ITL7, 11-15                         | MSGBIAS, 11-42             |
| ITL8, 11-16                         | MSGNODE, 11-42             |
| ITOL, 11-16                         | NOTRCLIB, 11-43            |
| LIBINC, 11-16, 12-6                 | NOWARN, 11-42              |
|                                     | NUMDGT, 11-43              |
|                                     | PRINTLG, 11-43             |

- STAT, [11-50](#)  
VERBOSE, [11-43](#)  
WBULK, [11-42](#)  
Simulation Output Control  
  ACOUT, [11-44](#), [12-5](#)  
  ALTER\_SUFFIX, [11-44](#)  
  ASCII, [11-44](#)  
  BLK\_SIZE, [11-44](#)  
  CAPTAB, [11-45](#)  
  DISPLAY\_CARLO, [11-45](#)  
  EXTCGS, [11-45](#)  
  EXTFILE, [11-45](#)  
  EXTMKSA, [11-45](#)  
  HISTLIM, [11-46](#)  
  INPUT, [11-46](#)  
  LCAPOP, [11-46](#)  
  LIMPROBE, [11-46](#)  
  LIST, [11-46](#)  
  MAX\_CHECKBUS, [11-46](#)  
  MAX\_DSPF\_PLOT, [11-46](#)  
  NEWACCT, [11-47](#)  
  NOASCII, [11-44](#)  
  NOBOUND\_PHASE, [11-47](#)  
  NODCINFOTAB, [11-47](#)  
  NODE, [11-47](#)  
  NOEXTRACTCOMPLEX, [11-48](#)  
  NOMOD, [11-48](#)  
  NOPAGE, [11-48](#)  
  NOSIZECHK, [11-48](#)  
  NOSTATP, [11-50](#)  
  NOTRC, [11-48](#)  
  NOWAVECOMPLEX, [11-48](#)  
  NOXTABNOISE, [11-48](#)  
  OPSEUDO\_ABSTRACT, [20-45](#)  
  OPSEUDO\_DETAIL, [20-45](#)  
  OPSEUDO\_DISPLAY\_GOALFITTING  
    G, [20-45](#)  
  OPSEUDO\_FORCE\_GOALFITTING,  
    [20-45](#)  
  OPSEUDO\_NETLIST, [20-45](#)  
  OPSEUDO\_NOGOALFITTING,  
    [20-46](#)  
  OPSEUDO\_OUTPUT, [20-46](#)  
  OPTYP, [11-48](#)  
  OUT\_RESOL, [11-49](#)  
OUT\_SMP, [11-49](#)  
OUT\_STEP, [11-49](#)  
POST, [11-49](#)  
SIMUDIV, [11-49](#)  
STAT, [11-50](#)  
TEMPCOUK, [11-50](#)  
TIMEDIV, [11-50](#)  
VBCSAT, [11-50](#)  
VXPROBE, [11-50](#)  
Simulator Compatibility  
  COMPAT, [11-6](#)  
  COMPMOD, [11-7](#)  
  COMPNET, [11-7](#)  
  MOTOROLA, [11-7](#)  
  NOELDOSWITCH, [11-59](#), [12-6](#)  
SPICE Compatibility  
  SPI3ASC, [11-6](#)  
  SPI3BIN, [11-6](#)  
  SPICEDC, [11-6](#)  
  SPIOUT, [11-6](#)  
OPTYP option, [11-48](#)  
OR Gate  
  see Double, Triple and Multiple Input  
    Digital Gates  
OSR option, [11-56](#)  
OUT\_ABSTOL option, [11-54](#)  
OUT\_REDUCE option, [11-53](#)  
OUT\_RELTOL option, [11-54](#)  
OUT\_RESOL option, [11-49](#)  
OUT\_SMP option, [11-49](#)  
OUT\_STEP option, [11-49](#)  
Output Shortform (.PROBE), [10-258](#)  
Overview  
  of the .cir File Structure, [3-1](#)  
**— P —**  
Parallel LCR Circuit Tutorial, [24-2](#)  
Parallel Switched Capacitor Macromodel,  
  [3-29](#), [9-19](#)  
  Equations, [9-20](#)  
  Parameters, [9-19](#)  
PARAM  
  .STEP, [10-301](#)  
Parameter Correlation, [10-42](#)  
Parameter declaration, [3-38](#)  
Parameter Extraction, [5-3](#), [5-61](#), [15-1](#) to [15-19](#)

- Parameter Names in Eldo, [3-3](#)  
Parameter Sweep  
  .DATA, [10-51](#)  
  .STEP, [10-298](#)  
PARAMOPT\_NOINITIAL option, [11-29](#)  
Parasitics  
  effects, [10-79](#)  
  elements, [10-79](#)  
PARHIER option, [11-9](#), [12-6](#)  
Partition Netlist into Newton Blocks  
  (.NWBLOCK), [10-186](#)  
Partitioning netlists  
  Newton Block accuracy  
    RELTOL, [10-186](#)  
  voltage accuracy  
    VNTOL, [10-186](#)  
Pattern Function, [3-22](#), [5-23](#)  
PCS option, [11-19](#)  
PCSPERIOD option, [11-19](#)  
PCSSIZE option, [11-19](#)  
Peak Detector Macromodel, [3-26](#), [6-45](#)  
  Parameters, [6-45](#)  
PGATEDEF option, [11-36](#)  
Philips MOS 11 Level 1100 (Eldo Level 65 or  
  MOSP11) MOSFET Model, [4-162](#)  
Philips MOS 11 Level 1100 (Eldo Level 69)  
  MOSFET Model, [4-164](#)  
Philips MOS 11 Level 1101 (Eldo Level 63 or  
  MOSP11) MOSFET Model, [4-160](#)  
Philips MOS 9 (Level 59 or MOSP9) MOSFET  
  Model, [4-157](#)  
Philips PSP (Level 70) MOSFET Model, [4-165](#)  
Piece-Wise-Linear Function, [3-22](#), [5-27](#), [10-35](#)  
PIVREL option, [11-19](#)  
PIVTOL option, [11-20](#)  
Plotting  
  of Bus Signals (.PLOTBUS), [10-250](#)  
  of Simulation Results (.PLOT), [10-216](#)  
  of Subcircuit Nodes, [4-173](#), [10-233](#), [10-268](#)  
  Output, [10-216](#)  
Pole-Zero  
  Analysis (.PZ), [10-268](#)  
  Cancellation by Threshold, [18-3](#)  
  Post-processor, [1-3](#), [10-268](#), [18-1 to 18-5](#)  
  Dialogue of, [18-2](#)  
Polynomial, non-linear  
  Capacitor model, [4-13](#)  
  Inductor model, [4-20](#)  
  Resistor model, [4-3](#)  
POST option, [11-49](#)  
Post-Processing Library (PPL), [10-28](#), [10-331](#),  
  [22-1 to 22-30](#)  
  Examples, [22-29](#)  
Post-processors  
  Pole-Zero, [1-3](#), [10-268](#), [18-1 to 18-5](#)  
POW, [3-8](#)  
POW (extract function), [10-104](#)  
POWER (extract function), [10-104](#)  
PPL (Post-Processing Library), [22-1](#)  
PRECISE option, [11-6](#)  
Previously Simulated Results  
  Usage of (.LOAD), [10-139](#)  
  Usage of (.USE), [10-328](#)  
Print Tabular Output File (.PRINTFILE),  
  [10-256](#)  
Printer Paper  
  Setting Width of (.WIDTH), [10-335](#)  
Printing  
  of Bus Signals (.PRINTBUS), [10-252](#)  
Printing of Results (.PRINT), [10-254](#)  
PRINTLG option, [11-43](#)  
Prior Simulation Input (.CHRSIM), [10-37](#)  
Probe Current Between Pins (.IPROBE),  
  [10-131](#)  
PROBE option, [11-54](#), [12-5](#)  
PROBEOP option, [11-54](#)  
PROBEOP2 option, [11-54](#)  
PSD computation (.DSPMOD), [10-85](#)  
PSF, [2-4](#)  
PSF option, [11-54](#)  
PSFASCII option, [11-54](#)  
PSOSC option, [11-20](#)  
PSTRAN option, [11-56](#)  
PSTRICT option, [11-11](#)  
Pulse Amplitude Modulator Macromodel,  
  [3-25](#), [6-34](#)  
  Characteristics, [6-35](#)  
  Parameters, [6-34](#)  
Pulse Function, [3-22](#), [5-25](#)

Pulse Width Modulator Macromodel, 3-25,  
6-40

Characteristics, 6-41

Parameters, 6-40

PVAL (extract function), 10-105

PWL, 3-9

PWR, 3-8

— Q —

QTRUNC option, 11-20

— R —

RAILRESISTANCE option, 11-36

Ramping, 16-20

.RAMP, 10-269

DC, 10-270

Transient, 10-270

RANDMC option, 11-29

RATPRINT option, 11-20

RC Wire Model, 3-17, 4-29

REAL (Real part of complex number), 3-8

REDUCE option, 11-36

Reliability Model Parameter Declaration,  
10-21, 21-5

Reliability Simulation, 21-1

RELTOL option, 10-186, 10-296, 11-20, 17-3

RELTRUNC option, 11-21

RELVAR option, 11-21

Remove library name (.DEL), 10-74

REPEAT

.SAVE, 10-274

Replace Node Name (.EQUIV), 10-92

reserved keywords, 10-206

RESET\_MULTIPLE\_RUN option, 11-52

Resistor Model, 3-17, 4-3, 4-8

.LOOP, 10-140

.SENS, 10-278

Parameters, 4-8

RESNW option, 11-37

Restart Simulation, 16-20

.RESTART, 10-271

RF keywords, 10-206

RGND option, 11-29

RGNDI option, 11-29

RISE\_TIME option, 11-26

RMMINRVAL option, 11-37

RMOS option, 11-37

RMS (extract function), 10-105

ROUND (Value rounded to integer), 3-8

Running

Eldo from the Command Line, 2-1

the Motorola Version of Eldo, 2-10

the STMicroelectronics Version of Eldo,  
2-10

— S —

S, Y, Z Parameter Extraction, 3-24, 5-3, 5-61,  
15-1 to 15-19

output file specification (.FFILE), 10-115,  
15-5

Source Syntax, 5-61, 15-1, 15-2

Safe Operating Area

Setting, 10-284

Sample and Hold Macromodel, 3-25, 6-36

Parameters, 6-36

SAMPLE option, 11-21

Saturating Resistor Macromodel, 3-24, 6-15

Characteristics, 6-16

Parameters, 6-15

Save Simulation Run, 16-20

.SAVE, 10-274

Save Simulation Run at Multiple Time Points  
.TSAVE, 10-323

SAVETIME option, 11-54

Sawtooth Waveform Generator Macromodel,  
3-25

Parameters, 6-28

Scale Factors in Eldo, 3-5

SCALE option, 11-37

SCALEBSIM option, 11-38

SCALM option, 11-38

Scattering parameters, 15-1

Schichman & Hodges, 4-122

SC-Integrators & LDI's, 9-13

SC—Schmitt Trigger Example, C-2

SDA option, 11-6

S-Domain Filter Model, 3-19, 4-167

Transfer Function, 4-167

SEARCH option, 11-60

Second Order Delta Sigma Modulator  
Example, C-19

Select Index

- in Pole-Zero Post-processing, 18-4  
Semiconductor Resistor Model, 3-17, 4-34  
Parameters, 4-34  
Sensitivity Analysis (.SENS), 10-278  
Examples, 24-26  
Sensitivity Analysis (.SENSPARAM), 10-280  
SEQ  
.SAVE, 10-275  
Serial Switched Capacitor Macromodel, 3-29, 9-21  
Equations, 9-22  
Parameters, 9-21  
Serial-parallel Switched Capacitor Macromodel, 3-29, 9-23  
Equations, 9-24  
Parameters, 9-24  
Set  
Bus Signal (.SIGBUS), 10-291  
Circuit Temperature (.TEMP), 3-15, 10-314  
Printer Paper Width (.WIDTH), 10-335  
Safe Operating Area (.SETSOA), 10-284  
Set Reliability Model Key (Password), 10-283, 21-6  
Set title (.TITLE), 10-316  
SGN, 3-7  
Shortform of Output (.PROBE), 10-258  
SIGN, 3-7  
Signal Monitoring Specifications  
Used with .PROBE Only, 10-264  
SIGTAIL option, 11-30  
SIMUDIV option, 11-49  
Simulation  
Accuracy  
RELTOL, 17-3  
VNTOL, 11-23, 17-3  
ANALOG parameter, 10-214  
Counters, 3-13  
Convergence Information, 3-14  
Grounded Capacitors Information, 3-14  
Matrix Information, 3-14  
Newton Block Information, 3-14  
Node & Element Information, 3-13  
Current Accuracy  
ITOL, 11-16  
ELDO parameter, 10-214  
Interruption, 16-24  
MACH parameter, 10-214  
Output  
Plotting of, 10-216  
Printing of, 10-254  
Re-run Facility (.ALTER), 10-22  
Restart, 16-20  
Restart (.RESTART), 10-271  
Running, 1-6  
Save, 16-20  
Saving of (.SAVE), 10-274  
Simulator  
Commands, 10-1  
Configuration (.OPTION), 10-198, 11-2  
SIN (Sine of value), 3-7, 3-22, 5-33  
Sine Function, 3-22, 5-33  
Single Frequency FM Function, 3-22, 5-32  
Single Output Level Detector Macromodel, 3-26, 6-48  
SINH (Hyperbolic sine of value), 3-7  
Sinusoidal Voltage Source (.SINUS), 10-294  
Sizing Facility (.SOLVE), 10-296  
SLEWRATE (extract function), 10-105  
SLOPE (extract function), 10-105  
S-Model, 15-8  
Applications, 15-12  
FBLOCK, 15-14  
Functionality, 15-13  
Implementation, 15-11  
Syntax, 15-14  
SMOOTH option, 11-55  
SOIBACK option, 11-38  
Source Description  
Amplitude Modulation Function, 5-17  
Current Controlled Current Source, 3-23, 5-46  
Current Controlled Voltage Source, 3-23, 5-57  
Exponential Function, 3-21, 5-19  
Exponential Pulse With Bit Pattern Function, 3-22, 5-37  
Independent Current Source, 3-21, 5-11  
Independent Voltage Source, 3-20, 5-4  
Noise Function, 3-22, 5-21

- Pattern Function, 3-22, 5-23  
Piece-Wise-Linear Function, 3-22, 5-27  
Piece-Wise-Linear Source (.CHRENT), 10-35  
Pulse Function, 3-22, 5-25  
S, Y, Z Parameter Extraction, 5-61  
Sine Function, 3-22, 5-33  
Single Frequency FM Function, 3-22, 5-32  
Trapezoidal Pulse With Bit Pattern Function, 3-22, 5-35  
Voltage Controlled Voltage Source, 3-22  
Voltage/Expression Controlled Current Source, 3-23, 5-50  
Voltage/Expression Controlled Voltage Source, 5-39  
Sources, 3-20 to 3-24, 5-1  
SP (Eldo Level 67) MOSFET Model, 4-163  
Speed and Accuracy, 16-1  
SPI3ASC option, 11-6  
SPI3BIN option, 11-6  
SPICE, 10-221, 10-254, 10-256, 11-2  
Spice, 10-221, 10-254, 10-256  
SPICEDC option, 11-6  
SPIOUT option, 11-6  
SPLITC option, 11-21  
SPMODLEV option, 11-38  
Spot Noise Figure (.SNF), 10-295  
SQRT (Square root of value), 3-7  
SSC  
    see Two-port Stability Circles  
Stability Circles  
    see Two-port Stability Circles  
Staircase Waveform Generator Macromodel, 3-25, 6-26  
    Parameters, 6-26  
Standard Deviation, 10-147  
STARTSMP option, 11-21  
STAT option, 11-50  
STEP option, 11-21  
STMicroelectronics Models, H-1  
    Version of Eldo, 2-10, H-2  
STOPONFIRSTERROR option, 11-12  
STOSMITH (Smith function), 3-8  
String parameters in Eldo, 3-3  
Stripline Models  
    see Microstrip Models  
Subcircuit  
    Definition (.SUBCKT), 10-306  
    Duplicate parameters (.SUBDUP), 10-312  
    Error Messages, A-22  
    File Insertion (.ADDLIB), 10-18  
    Instance Model, 3-20, 4-171  
    Nodes  
        Plotting of, 4-173, 10-233, 10-268  
    Warning Messages, A-39  
Subcircuit Description Termination (.ENDS), 10-91  
Subcircuits  
    Access of Nodes, 10-310  
    Access of Nodes from within, 3-4  
SUBFLAGPAR option, 11-12  
Subtractor Macromodel, 3-26, 6-58  
    Parameters, 6-58  
Suppress  
    Comment Lines from an Output File (.NOCOM), 10-178  
    Netlist from an Output File (.NOTRC), 10-185  
Sweeping of Parameters (.STEP), 10-298  
Switch  
    .SENS, 10-278  
    Current Controlled Macromodel, 3-25, 6-21  
        Characteristics, 6-22  
        Parameters, 6-21  
    Noise Model, 14-14  
    SC Macromodel, 3-28, 9-8  
        Equivalent Circuit, 9-9  
        Parameters, 9-10  
    Voltage Controlled Macromodel, 3-25, 6-19  
        Characteristics, 6-20  
        Parameters, 6-19  
Switched Capacitor Macromodels, 3-28, 9-1  
    Applications, 9-30  
Syntax  
    Errors, A-1  
    General aspects  
        Comment lines, 3-2

- Component names, 3-3  
Continuation lines, 3-2  
Directives, 3-5  
First line, 3-2  
Model names, 3-5  
Node names, 3-4  
Parameter names, 3-3  
Reserved keywords, 3-3  
Scale factors, 3-5  
String parameters, 3-3  
Values, 3-4  
of Analog Macromodels, 3-24 to 3-26, 6-1 to 6-59  
of Commands, 3-29 to 3-42, 10-1 to 10-335  
of Devices, 3-17 to 3-20, 4-1 to 4-173  
of Digital Macromodels, 3-26, 7-1 to 7-17  
of Eldo, 3-2 to 3-42  
of Macromodels, 3-24 to 3-29, 6-1 to 6-59, 9-1 to 9-29  
of Magnetic Macromodels, 3-27, 8-1 to 8-13  
of Mixed Signal Macromodels, 3-27  
of Sources, 3-20 to 3-24, 5-1 to 5-61  
of Switched Capacitor Macromodels, 3-28, 9-1 to 9-36
- T —
- T Parameter, 3-15  
TAN (Tangent of value), 3-7  
TANH (Hyperbolic tangent of value), 3-7  
TCROSS (extract function), 10-105  
TEMP Variable, 3-16  
.STEP, 10-301  
TEMPCOUK option, 11-50  
TEMPER Variable, 3-16  
Temperature  
.TEMP, 3-15  
Handling in Eldo, 3-15  
T Parameter, 3-15  
TEMP Parameter, 3-16  
TEMPER Parameter, 3-16  
TMAX option, 11-39  
TMIN option, 11-39  
TMOD Parameter, 3-15  
TNOM Parameter, 3-15  
Test Vector Files (.TVINCLUDE), 10-325
- TFALL (extract function), 10-107  
TFT Amorphous-Si (Level 64) MOSFET Model, 4-161  
TFT Polysilicon (Level 62) MOSFET Model, 4-159  
TGP, 10-245  
THERMAL\_NOISE option, 11-41  
Three Input Digital Gates  
see Triple Input Digital Gates  
TIME  
.SAVE, 10-274  
TIMEDIV option, 11-50  
TIMESMP option, 11-22  
Time-step  
Fixed, 11-21  
TINTEG (extract function), 10-106  
TI-Spice Compatibility, 13-1  
TMAX option, 11-39  
TMIN option, 11-39  
TMOD Parameter, 3-15  
TNOM option, 3-15, 4-9, 5-6, 5-13, 10-314, 11-30  
TOPCELL (.TOPCELL), 10-317  
Touchstone Data Format, 15-18  
TPD (extract function), 10-106  
TPDDD (extract function), 10-107  
TPDDU (extract function), 10-107  
TPDUD (extract function), 10-107  
TPDUU (extract function), 10-107  
TPIEEE option, 11-30  
Track and Hold Macromodel, 3-25, 6-38  
Parameters, 6-38  
Transfer Function (.TF), 10-315  
Transformer (Ideal) Macromodel, 8-13  
Transformer Winding Macromodel, 3-27, 8-2  
Transformer with Variable Number of Windings Macromodel, 3-28, 8-11  
Transient Analysis (.TRAN), 10-318  
.CHRSIM, 10-37  
.CONSO, 10-41  
.IC, 10-125  
.LOOP, 10-140  
.MC, 10-147  
.PRINT, 10-226

- Examples, 24-5, 24-16, 24-19, C-2, C-3, C-9, C-11, C-14, C-19, C-22  
Switch Macromodel, 9-8  
Transient Noise Analysis, 14-1  
.NOISETRAN, 10-182  
Transmission Line Model, 3-17, 4-41  
TRAP option, 11-55  
Trapezoidal Pulse With Bit Pattern Function (PBIT), 3-22  
Triangle Waveform Generator Macromodel, 3-25, 6-30  
Parameters, 6-30  
Triangular to Sine Wave Converter  
Macromodel, 3-25, 6-24  
Characteristics, 6-25  
Parameters, 6-24  
Triple Input AND Gate Macromodel, 3-27  
Triple Input Digital Gates, 3-27, 7-11  
Triple Input NAND Gate Macromodel, 3-27  
Triple Input NOR Gate Macromodel, 3-27  
Triple Input OR Gate Macromodel, 3-27  
TRISE (extract function), 10-107  
Trouble Shooting, B-1  
TRTOL option, 11-22  
TRUNC (Truncated value), 3-8  
TUNING option, 11-22  
Tutorials, 24-1  
1—Parallel LCR Circuit, 24-2  
2—4th Order Butterworth Filter, 24-5  
3—Band Pass Filter, 24-9  
4—Low Pass Filter, 24-13  
5—Colpitts Oscillator, 24-16  
6—High Voltage Cascade, 24-19  
7—Non-inverting Amplifier, 24-23  
8—Bipolar Amplifier, 24-26, 24-29  
Two Input Digital Gates  
see Double Input Gates  
Two-port Constant Gain Circles  
GAC, 10-243  
GPC, 10-244  
Two-port Gain Extract, 10-244  
Available Power Gain  
GA, 10-244  
GAM, 10-244  
GASM, 10-244  
GAUM, 10-245  
GP, 10-245  
TGP, 10-245  
Two-port Noise Circles  
NC, 10-243  
Two-port Noise Parameters, 10-242  
BOPT, 10-242  
GOPT, 10-242  
NFMIN, 10-242  
RNEQ, 10-242  
Two-port Stability Circles, 10-245  
LSC, 10-246  
SSC, 10-245  
Two-port Stability Factors  
BFACTOR, 10-242  
KFACTOR, 10-242  
MUFATOR, 10-242  
**— U —**  
UDMP, 4-130  
UIC Parameter, 4-15, 10-118, 10-125, 16-19  
ULOGIC option, 11-30  
UNBOUND option, 11-23  
Unswitched Capacitor Macromodel, 3-29, 9-28  
Equations, 9-29  
Parameters, 9-28  
Usage of FAS Macromodels, 6-2  
Use Previously Simulated Results  
.LOAD, 10-139  
.USE, 10-328  
Use Reliability Model Key (Password), 10-330, 21-7  
Use Tcl File  
.USE\_TCL, 10-331, 22-1  
USEDEFAP option, 11-39  
User Defined Distributions (Monte Carlo)  
(.DISTRIB), 10-77  
User Defined Function (.FUNC), 10-120  
USETHREAD option, 11-13  
Utilities, E-1  
**— V —**  
VALAT (extract function), 10-107  
Value Tables (.TABLE), 10-313  
Values in Eldo, 3-4  
VAMAXEXP option, 11-60

VBCSAT option, 11-50  
VBIC v1.1.5 Model, 4-116  
    Parameters, 4-116  
VBIC v1.2 Model, 4-115  
    Parameters, 4-115  
VBICLEV option, 11-39  
VERBOSE option, 11-43  
VMAX option, 11-23  
VMIN option, 11-23  
VNTOL, 10-186, 17-3  
    Correct usage, B-1  
VNTOL option, 11-23  
Voltage Controlled Oscillator (VCO)  
    Macromodel, 3-26, 6-43  
    Parameters, 6-43  
Voltage Controlled Switch Macromodel, 3-25, 6-19  
    Parameters, 6-19  
Voltage Controlled Voltage Source, 3-22  
Voltage Limiter Macromodel, 3-25, 6-17  
    Parameters, 6-17  
Voltage Source  
    Sinusoidal (.SINUS), 10-294  
Voltage/Expression Controlled Current Source, 5-50  
Voltage/Expression Controlled Voltage Source, 5-39  
VXPROBE option, 11-50

**— W —**

Warning Messages, A-1, A-26  
    Global, A-26  
    Miscellaneous, A-40  
    Related to Commands, A-32  
    Related to Models, A-38  
    Related to Nodes, A-28  
    Related to Objects, A-30  
    Related to Subcircuits, A-39  
Warnings  
    .SETSOA, 10-284 to 10-288  
    Messages, A-1  
WARNMAXV option, 11-39  
Waveform Definition (.DEFWAVE), 10-70  
WBULK option, 11-42  
WFREQ (extract function), 10-108  
WINTEG (extract function), 10-108

WL option, 11-39  
Worst Case Analysis (.WCASE), 10-332  
    .MC, 10-148  
WSF option, 11-6  
WSFASCII option, 11-6

**— X —**

XA option, 11-24  
XCOMPRESS (extract function), 10-108  
XDOWN (extract function), 10-109  
XMAX (extract function), 10-109  
XMIN (extract function), 10-109  
XTHRES (extract function), 10-109  
XUP (extract function), 10-109  
XYCOND (extract function), 10-109

**— Y —**

YMFAC option, 11-39  
YTOSMITH (Smith function), 3-8  
YVAL (extract function), 10-110

**— Z —**

ZCHAR option, 11-60  
Z-Domain Filter Model, 3-19, 4-169  
    Transfer Function, 4-169  
Z-Domain Representation of SC Macromodels, 9-11  
ZOOMTIME option, 11-30  
ZTOSMITH (Smith function), 3-8

# End-User License Agreement

**IMPORTANT - USE OF THIS SOFTWARE IS SUBJECT TO LICENSE RESTRICTIONS.  
CAREFULLY READ THIS LICENSE AGREEMENT BEFORE USING THE SOFTWARE.**

This license is a legal “Agreement” concerning the use of Software between you, the end user, either individually or as an authorized representative of the company acquiring the license, and Mentor Graphics Corporation and Mentor Graphics (Ireland) Limited, acting directly or through their subsidiaries or authorized distributors (collectively “Mentor Graphics”). **USE OF SOFTWARE INDICATES YOUR COMPLETE AND UNCONDITIONAL ACCEPTANCE OF THE TERMS AND CONDITIONS SET FORTH IN THIS AGREEMENT.** If you do not agree to these terms and conditions, promptly return, or, if received electronically, certify destruction of, Software and all accompanying items within five days after receipt of Software and receive a full refund of any license fee paid.

## END-USER LICENSE AGREEMENT

1. **GRANT OF LICENSE.** The software programs you are installing, downloading, or have acquired with this Agreement, including any updates, modifications, revisions, copies, documentation and design data (“Software”) are copyrighted, trade secret and confidential information of Mentor Graphics or its licensors who maintain exclusive title to all Software and retain all rights not expressly granted by this Agreement. Mentor Graphics grants to you, subject to payment of appropriate license fees, a nontransferable, nonexclusive license to use Software solely: (a) in machine-readable, object-code form; (b) for your internal business purposes; and (c) on the computer hardware or at the site for which an applicable license fee is paid, or as authorized by Mentor Graphics. A site is restricted to a one-half mile (800 meter) radius. Mentor Graphics' standard policies and programs, which vary depending on Software, license fees paid or service plan purchased, apply to the following and are subject to change: (a) relocation of Software; (b) use of Software, which may be limited, for example, to execution of a single session by a single user on the authorized hardware or for a restricted period of time (such limitations may be communicated and technically implemented through the use of authorization codes or similar devices); (c) support services provided, including eligibility to receive telephone support, updates, modifications and revisions. Current standard policies and programs are available upon request.
2. **ESD SOFTWARE.** If you purchased a license to use embedded software development (“ESD”) Software, Mentor Graphics grants to you a nontransferable, nonexclusive license to reproduce and distribute executable files created using ESD compilers, including the ESD run-time libraries distributed with ESD C and C++ compiler Software that are linked into a composite program as an integral part of your compiled computer program, provided that you distribute these files only in conjunction with your compiled computer program. Mentor Graphics does NOT grant you any right to duplicate or incorporate copies of Mentor Graphics' real-time operating systems or other ESD Software, except those explicitly granted in this section, into your products without first signing a separate agreement with Mentor Graphics for such purpose.
3. **BETA CODE.** Portions or all of certain Software may contain code for experimental testing and evaluation (“Beta Code”), which may not be used without Mentor Graphics' explicit authorization. Upon Mentor Graphics' authorization, Mentor Graphics grants to you a temporary, nontransferable, nonexclusive license for experimental use to test and evaluate the Beta Code without charge for a limited period of time specified by Mentor Graphics. This grant and your use of the Beta Code shall not be construed as marketing or offering to sell a license to the Beta Code, which Mentor Graphics may choose not to release commercially in any form. If Mentor Graphics authorizes you to use the Beta Code, you agree to evaluate and test the Beta Code under normal conditions as directed by Mentor Graphics. You will contact Mentor Graphics periodically during your use of the Beta Code to discuss any malfunctions or suggested improvements. Upon completion of your evaluation and testing, you will send to Mentor Graphics a written evaluation of the Beta Code, including its strengths, weaknesses and recommended improvements. You agree that any written evaluations and all inventions, product improvements, modifications or developments that Mentor Graphics conceives or makes during or subsequent to this Agreement, including those based partly or wholly on your feedback, will be the exclusive property of Mentor Graphics. Mentor Graphics will have exclusive rights, title and interest in all such property. The provisions of this subsection shall survive termination or expiration of this Agreement.
4. **RESTRICTIONS ON USE.** You may copy Software only as reasonably necessary to support the authorized use. Each copy must include all notices and legends embedded in Software and affixed to its medium and container as received from Mentor Graphics. All copies shall remain the property of Mentor Graphics or its licensors. You shall maintain a record of the number and primary location of all copies of Software, including copies merged with other software, and shall make those records available to Mentor Graphics upon request. You shall not make Software available in any form to any

person other than employees and contractors, excluding Mentor Graphics' competitors, whose job performance requires access. You shall take appropriate action to protect the confidentiality of Software and ensure that any person permitted access to Software does not disclose it or use it except as permitted by this Agreement. Except as otherwise permitted for purposes of interoperability as specified by applicable and mandatory local law, you shall not reverse-assemble, reverse-compile, reverse-engineer or in any way derive from Software any source code. You may not sublicense, assign or otherwise transfer Software, this Agreement or the rights under it, whether by operation of law or otherwise ("attempted transfer") without Mentor Graphics' prior written consent and payment of Mentor Graphics then-current applicable transfer charges. Any attempted transfer without Mentor Graphics' prior written consent shall be a material breach of this Agreement and may, at Mentor Graphics' option, result in the immediate termination of the Agreement and licenses granted under this Agreement. The provisions of this section 4 shall survive the termination or expiration of this Agreement.

## 5. LIMITED WARRANTY.

5.1. Mentor Graphics warrants that during the warranty period, Software, when properly installed, will substantially conform to the functional specifications set forth in the applicable user manual. Mentor Graphics does not warrant that Software will meet your requirements or that operation of Software will be uninterrupted or error free. The warranty period is 90 days starting on the 15th day after delivery or upon installation, whichever first occurs. You must notify Mentor Graphics in writing of any nonconformity within the warranty period. This warranty shall not be valid if Software has been subject to misuse, unauthorized modification or installation. MENTOR GRAPHICS' ENTIRE LIABILITY AND YOUR EXCLUSIVE REMEDY SHALL BE, AT MENTOR GRAPHICS' OPTION, EITHER (A) REFUND OF THE PRICE PAID UPON RETURN OF SOFTWARE TO MENTOR GRAPHICS OR (B) MODIFICATION OR REPLACEMENT OF SOFTWARE THAT DOES NOT MEET THIS LIMITED WARRANTY, PROVIDED YOU HAVE OTHERWISE COMPLIED WITH THIS AGREEMENT. MENTOR GRAPHICS MAKES NO WARRANTIES WITH RESPECT TO: (A) SERVICES; (B) SOFTWARE WHICH IS LICENSED TO YOU FOR A LIMITED TERM OR LICENSED AT NO COST; OR (C) EXPERIMENTAL BETA CODE; ALL OF WHICH ARE PROVIDED "AS IS."

5.2. THE WARRANTIES SET FORTH IN THIS SECTION 5 ARE EXCLUSIVE. NEITHER MENTOR GRAPHICS NOR ITS LICENSORS MAKE ANY OTHER WARRANTIES, EXPRESS, IMPLIED, OR STATUTORY, WITH RESPECT TO SOFTWARE OR OTHER MATERIAL PROVIDED UNDER THIS AGREEMENT. MENTOR GRAPHICS AND ITS LICENSORS SPECIFICALLY DISCLAIM ALL IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE AND NON-INFRINGEMENT OF INTELLECTUAL PROPERTY.

6. **LIMITATION OF LIABILITY.** EXCEPT WHERE THIS EXCLUSION OR RESTRICTION OF LIABILITY WOULD BE VOID OR INEFFECTIVE UNDER APPLICABLE LAW, IN NO EVENT SHALL MENTOR GRAPHICS OR ITS LICENSORS BE LIABLE FOR INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES (INCLUDING LOST PROFITS OR SAVINGS) WHETHER BASED ON CONTRACT, TORT OR ANY OTHER LEGAL THEORY, EVEN IF MENTOR GRAPHICS OR ITS LICENSORS HAVE BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. IN NO EVENT SHALL MENTOR GRAPHICS' OR ITS LICENSORS' LIABILITY UNDER THIS AGREEMENT EXCEED THE AMOUNT PAID BY YOU FOR THE SOFTWARE OR SERVICE GIVING RISE TO THE CLAIM. IN THE CASE WHERE NO AMOUNT WAS PAID, MENTOR GRAPHICS AND ITS LICENSORS SHALL HAVE NO LIABILITY FOR ANY DAMAGES WHATSOEVER.

7. **LIFE ENDANGERING ACTIVITIES.** NEITHER MENTOR GRAPHICS NOR ITS LICENSORS SHALL BE LIABLE FOR ANY DAMAGES RESULTING FROM OR IN CONNECTION WITH THE USE OF SOFTWARE IN ANY APPLICATION WHERE THE FAILURE OR INACCURACY OF THE SOFTWARE MIGHT RESULT IN DEATH OR PERSONAL INJURY.

8. **INDEMNIFICATION.** YOU AGREE TO INDEMNIFY AND HOLD HARMLESS MENTOR GRAPHICS AND ITS LICENSORS FROM ANY CLAIMS, LOSS, COST, DAMAGE, EXPENSE, OR LIABILITY, INCLUDING ATTORNEYS' FEES, ARISING OUT OF OR IN CONNECTION WITH YOUR USE OF SOFTWARE AS DESCRIBED IN SECTION 7.

## 9. INFRINGEMENT.

9.1. Mentor Graphics will defend or settle, at its option and expense, any action brought against you alleging that Software infringes a patent or copyright or misappropriates a trade secret in the United States, Canada, Japan, or member state of the European Patent Office. Mentor Graphics will pay any costs and damages finally awarded against you that are attributable to the infringement action. You understand and agree that as conditions to Mentor Graphics' obligations under this section you must: (a) notify Mentor Graphics promptly in writing of the action; (b)

provide Mentor Graphics all reasonable information and assistance to defend or settle the action; and (c) grant Mentor Graphics sole authority and control of the defense or settlement of the action.

- 9.2. If an infringement claim is made, Mentor Graphics may, at its option and expense: (a) replace or modify Software so that it becomes noninfringing; (b) procure for you the right to continue using Software; or (c) require the return of Software and refund to you any license fee paid, less a reasonable allowance for use.
- 9.3. Mentor Graphics has no liability to you if infringement is based upon: (a) the combination of Software with any product not furnished by Mentor Graphics; (b) the modification of Software other than by Mentor Graphics; (c) the use of other than a current unaltered release of Software; (d) the use of Software as part of an infringing process; (e) a product that you make, use or sell; (f) any Beta Code contained in Software; (g) any Software provided by Mentor Graphics' licensors who do not provide such indemnification to Mentor Graphics' customers; or (h) infringement by you that is deemed willful. In the case of (h) you shall reimburse Mentor Graphics for its attorney fees and other costs related to the action upon a final judgment.
- 9.4. **THIS SECTION 9 STATES THE ENTIRE LIABILITY OF MENTOR GRAPHICS AND ITS LICENSORS AND YOUR SOLE AND EXCLUSIVE REMEDY WITH RESPECT TO ANY ALLEGED PATENT OR COPYRIGHT INFRINGEMENT OR TRADE SECRET MISAPPROPRIATION BY ANY SOFTWARE LICENSED UNDER THIS AGREEMENT.**
10. **TERM.** This Agreement remains effective until expiration or termination. This Agreement will automatically terminate if you fail to comply with any term or condition of this Agreement or if you fail to pay for the license when due and such failure to pay continues for a period of 30 days after written notice from Mentor Graphics. If Software was provided for limited term use, this Agreement will automatically expire at the end of the authorized term. Upon any termination or expiration, you agree to cease all use of Software and return it to Mentor Graphics or certify deletion and destruction of Software, including all copies, to Mentor Graphics' reasonable satisfaction.
11. **EXPORT.** Software is subject to regulation by local laws and United States government agencies, which prohibit export or diversion of certain products, information about the products, and direct products of the products to certain countries and certain persons. You agree that you will not export any Software or direct product of Software in any manner without first obtaining all necessary approval from appropriate local and United States government agencies.
12. **RESTRICTED RIGHTS NOTICE.** Software was developed entirely at private expense and is commercial computer software provided with RESTRICTED RIGHTS. Use, duplication or disclosure by the U.S. Government or a U.S. Government subcontractor is subject to the restrictions set forth in the license agreement under which Software was obtained pursuant to DFARS 227.7202-3(a) or as set forth in subparagraphs (c)(1) and (2) of the Commercial Computer Software - Restricted Rights clause at FAR 52.227-19, as applicable. Contractor/manufacturer is Mentor Graphics Corporation, 8005 SW Boeckman Road, Wilsonville, Oregon 97070-7777 USA.
13. **THIRD PARTY BENEFICIARY.** For any Software under this Agreement licensed by Mentor Graphics from Microsoft or other licensors, Microsoft or the applicable licensor is a third party beneficiary of this Agreement with the right to enforce the obligations set forth in this Agreement.
14. **AUDIT RIGHTS.** With reasonable prior notice, Mentor Graphics shall have the right to audit during your normal business hours all records and accounts as may contain information regarding your compliance with the terms of this Agreement. Mentor Graphics shall keep in confidence all information gained as a result of any audit. Mentor Graphics shall only use or disclose such information as necessary to enforce its rights under this Agreement.
15. **CONTROLLING LAW AND JURISDICTION.** THIS AGREEMENT SHALL BE GOVERNED BY AND CONSTRUED UNDER THE LAWS OF OREGON, USA, IF YOU ARE LOCATED IN NORTH OR SOUTH AMERICA, AND THE LAWS OF IRELAND IF YOU ARE LOCATED OUTSIDE OF NORTH AND SOUTH AMERICA. All disputes arising out of or in relation to this Agreement shall be submitted to the exclusive jurisdiction of Dublin, Ireland when the laws of Ireland apply, or Wilsonville, Oregon when the laws of Oregon apply. This section shall not restrict Mentor Graphics' right to bring an action against you in the jurisdiction where your place of business is located.
16. **SEVERABILITY.** If any provision of this Agreement is held by a court of competent jurisdiction to be void, invalid, unenforceable or illegal, such provision shall be severed from this Agreement and the remaining provisions will remain in full force and effect.
17. **MISCELLANEOUS.** This Agreement contains the parties' entire understanding relating to its subject matter and supersedes all prior or contemporaneous agreements, including but not limited to any purchase order terms and

conditions, except valid license agreements related to the subject matter of this Agreement (which are physically signed by you and an authorized agent of Mentor Graphics) either referenced in the purchase order or otherwise governing this subject matter. This Agreement may only be modified in writing by authorized representatives of the parties. Waiver of terms or excuse of breach must be in writing and shall not constitute subsequent consent, waiver or excuse. The prevailing party in any legal action regarding the subject matter of this Agreement shall be entitled to recover, in addition to other relief, reasonable attorneys' fees and expenses.

Rev. 020826, Part Number 214231