LTspice® XVII

www.linear.com

www.analog.com

In memory of
Peanut, Spider and Toad.

Introduction

I've been writing physical simulators since 1975[1]. Speaking from benefit of the perspective of having written physical simulators for some decades, I see the best simulators developed by the concerns that actually need them and not software companies. I wrote LTspice explicitly to outperform analogous tools for sale from software companies in the interest of having the highest performance SPICE program available from anyone at any price. LTspice is used in-house for IC design and has become part of Linear Technology Corporation's strategy for developing high performance analog products.

What is unusual about LTspice is that is it also freely distributed. Linear Technology Corporation does not freely distribute LTspice out of a charitable interest in the wellbeing of mankind. We are a for-profit organization freely distributing LTspice in the interest of helping customers simulate LTC products with a better simulator than is otherwise available. Further, the freely distributed version of LTspice is exactly the same as us used in-house for IC design. No version of LTspice is crippled in anyway to artificially limit its capability.

This is a unique situation for a SPICE simulator and has made LTspice's popularity no less than fantastic. LTspice is overwhelmingly the most widely distributed and used SPICE program in the industry to date. It has become the de facto standard SPICE program.

LTspice XVII is a partial rewrite of LTspice IV with a modern graphics library for native multi-monitor support. XVII also introduces

- 64-bit support
- UNICODE(use any character of any living language in schematics, netlists, or plot)
- New device equations(e.g. IGBT, diode soft recovery, and an arbitrary state machine)
- Improved GUI, e.g., editors for most SPICE commands
- Extensions to Microsoft Windows for schematic thumbnails and schematic preview.

Version XVII of LTspice comes in LTspice's 17th year of general release. LTspice XVII was released via an international seminar series during 2016. The first public announcement of LTspice XVII was on May 9th, 2016 in Hanoi, Vietnam.

--Mike Engelhardt / 2016

1] For inelastic nuclear relativistic kinematics used at a cyclotron lab when I was a teenager.

Software Installation

LTspice XVII runs on 32- or 64-bit editions of Windows 7, 8, or 10. Windows XP is not supported. Windows XP users can run LTspice IV[1], which is still be available in observance of Linear Technology Corporation's zero obsolescence tradition.

LTspice XVII can be downloaded from http://www.linear.com. A direct link to the distribution file is

http://LTspice.linear.com/software/LTspiceXVII.exe

The file LTspiceXVII.exe is a self-extracting gzipped file that installs LTspice XVII.

LTspice XVII is updated often. After LTspice XVII is initially installed, you can use a built-in update menu command that will bring your installation to the current revision level if you have access to the web. The update process will first download a master index file from Linear's website that has the size and checksum of each file in the distribution. If a file is missing, is of a different size, or differs in checksum, then that file will be updated. Component databases are merged in the update process so if you've added devices to your installation, those additions won't be lost when you run the automatic update utility.

1] LTspice IV is no longer updated.

LTspice -- License Agreement/Disclaimer

LTspice is Linear Technology Corporation's analog circuit simulation software.

This software is copyrighted. You are granted a non-exclusive, non-transferable, non-sublicenseable, royalty-free right to evaluate LTC products and also to perform general circuit simulation. Linear Technology Corporation owns the software. You may not modify, adapt, translate, reverse engineer, decompile, or disassemble the software executable(s) or models of LTC products provided. We take no responsibility for the accuracy of third party models used in the simulator whether provided by LTC or the user.

While we have made every effort to ensure that LTspice operates in the manner described, we do not guarantee operation to be error free. Upgrades, modifications, or repairs to this program will be strictly at the discretion of LTC. If you encounter problems installing or operating LTspice for the purpose of selecting and evaluating LTC products, you may obtain technical assistance by calling our Applications Department at (408) 432-1900, between 8:00 am and 5:00 pm Pacific time, Monday through Friday. We do not provide such technical support for general circuit simulations that are not for the evaluation of LTC products. Because of the great variety of PC-compatible computer systems, operating system versions, and peripherals currently in use, we do not guarantee that you will be able to use LTspice successfully on all such systems. If you are unable to use LTspice, LTC does provide design support for LTC switching regulator ICs by whatever means necessary.

The software and related documentation are provided "AS IS" and without warranty of any kind and Linear Technology Corporation expressly disclaims all other warranties, express or implied, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. Under no circumstances will LTC be liable for damages, either direct or consequential, arising from the use of this product or from the inability to use this product, even if we have been informed in advance of the possibility of such damages.

Redistribution of this software is permitted as long as it is distributed in its entirety, with all documentation, example files, symbols, and models without modification or additions.

This program is specifically not licensed for use by semiconductor manufacturers in the design, promotion, demonstration or sale of their products. Specific permission must be obtained from Linear Technology for the use of LTspice for these applications.

The Export Control Classification Number(ECCN) of LTspice is EAR99.

Linear Technology Corporation is now part of Analog Devices.

Modes of Operation

LTspice XVII is intended to be used as a general purpose schematic capture program with an integrated SPICE simulator. The idea is you draw a circuit(or start with an example circuit that's already drafted) and observe its operation in the simulator. The design process involves iterating the circuit until the desired circuit behavior is achieved in simulation.

The schematic is ultimately converted to a textual SPICE netlist that is passed to the simulator. While the netlist is usually extracted from a graphical schematic drafted in LTspice, an imported netlist can be run directly without having a schematic. This second mode of operation has some uses: (i) Linear Technology's filter synthesis program, FilterCAD, can synthesize a netlist for LTspice to simulate the time domain or frequency response of a filter. (ii) it simplifies benchmarking LTspice against other SPICE programs (iii) professionals historically experienced with SPICE circuit simulators are familiar with working directly with the textual netlists because schematic capture was not integrated with SPICE simulators in very old simulators.

Example Circuits

General Purpose Schematic Driven SPICE

Externally Generated Netlists

Efficiency Report

Command Line Switches

Example Circuits

There are several source of example circuits for LTspice XVII. There is the directory

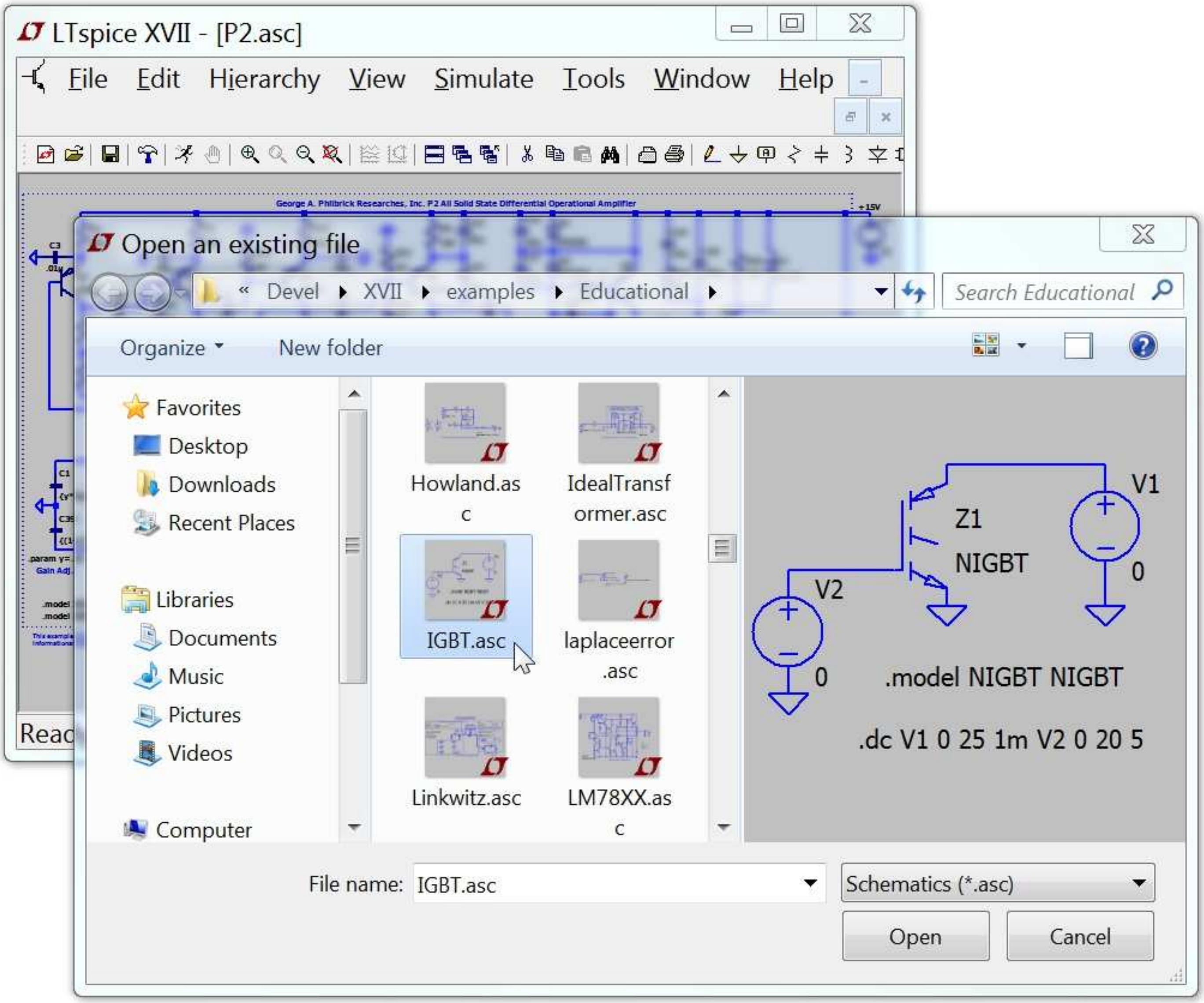
```
   %HOMEPATH%\Documents\LTspiceXVII\examples\Educational
```

that gives numbers non-commercial examples of SPICE simulations that illustrate different analysis types, methods or program features.

In the directory

```
   %HOMEPATH%\Documents\LTspiceXVII\examples\jigs
```

there is an example simulation for every Linear Technology device with a macromodel in LTspice XVII. Note that these jig circuits are often only test jigs for the macromodel, not necessarily recommended reference designs. You will need to audit the value of the capacitors used in the SMPS circuits. Your local Linear Technology office should be able to give you design support specific for your application needs.



```
.model NIGBT NIGBT
.dc V1 0 25 1m V2 0 20 5
```

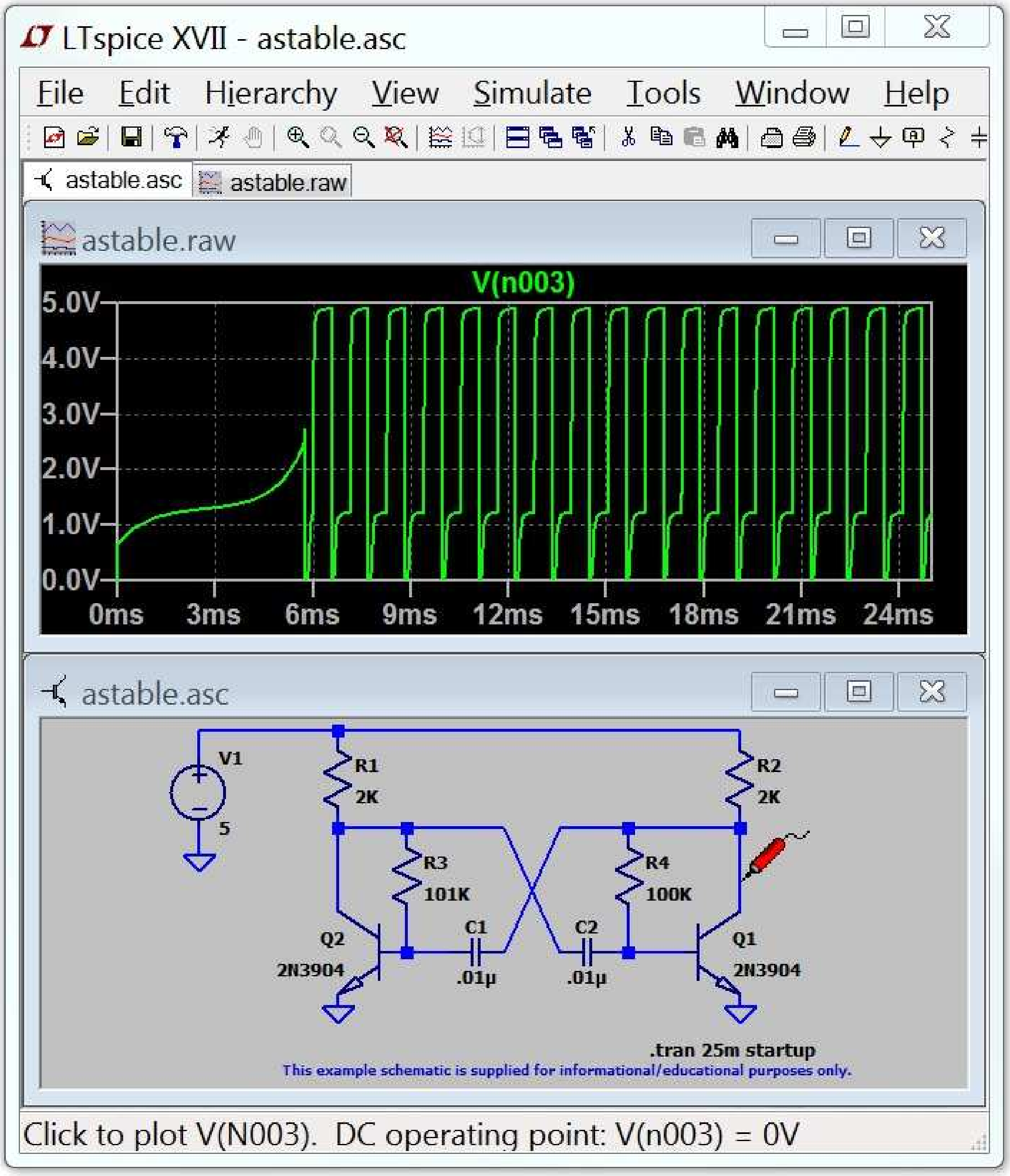
File name: IGBT.asc

Schematics (*.asc)

Notice that LTspice XVII includes handlers to give a preview of the schematics in Windows Explorer and file utilities.

General Purpose Schematic Driven SPICE

This is the main use of the LTspice XVII simulator. Within the restrictions found in the license, you are free to use LTspice XVII as your general-purpose schematic capture/SPICE program even for circuits which do not use Linear Technology products. Many companies standardize on LTspice as their EDA tool.



```
.tran 25m startup
```

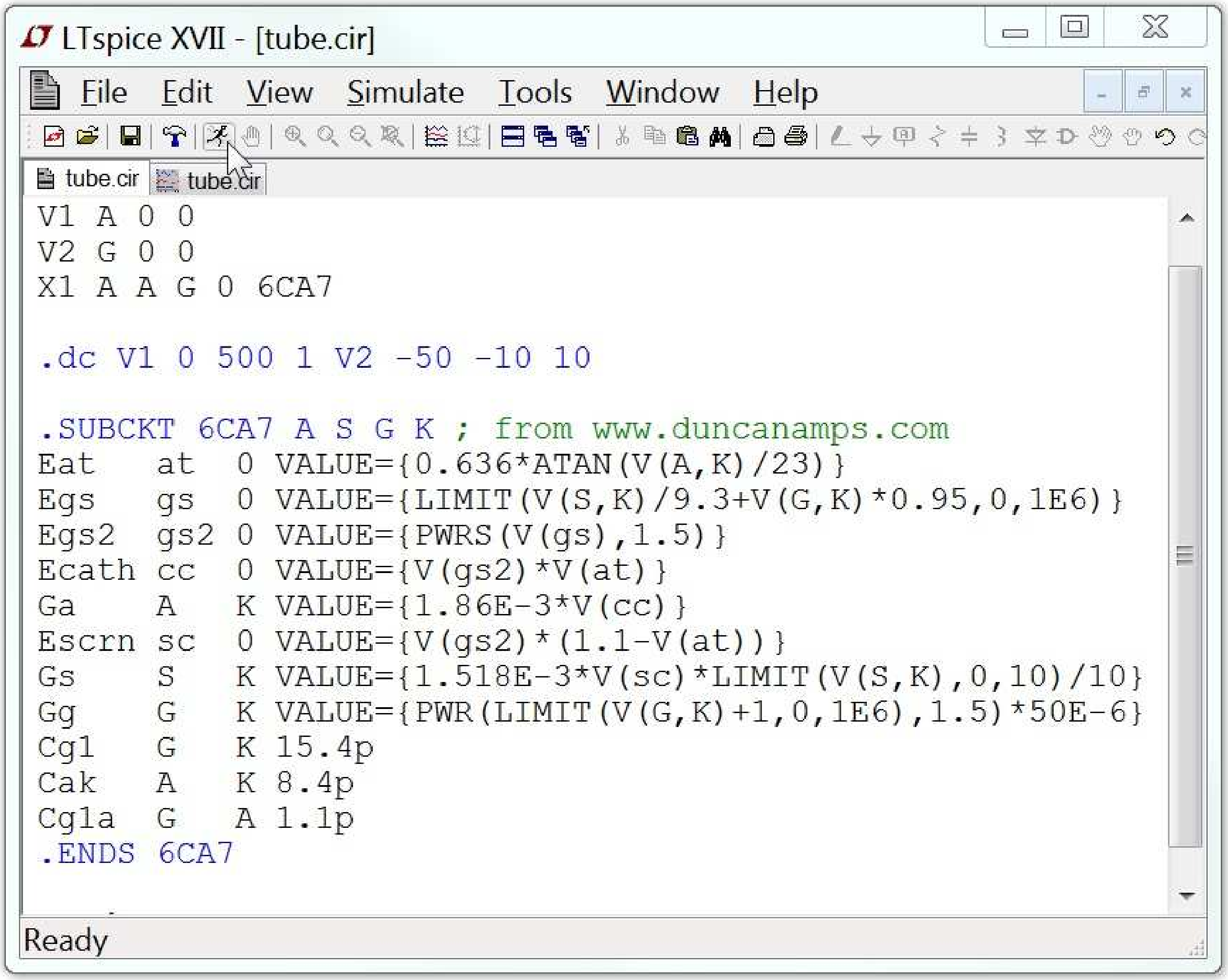
This example schematic is supplied for informational/educational purposes only.

Click to plot V(N003). DC operating point: V(n003) = 0V

LTspice XVII allows you to draft and simulation circuits of unlimited size and content. Marching waveforms, cross probing, reverse cross probing, and unlimited hierarchy are supported.

Externally Generated Netlists

You can open netlists generated either by hand or from other schematic capture programs. These files usually have a filename extension of ".cir", but ".net" and ".sp" are understood. The text editor used for netlist files colorizes SPICE syntax for readability. The menu command Tools=>Color Preferences can be used to adjust the colors used in the editor. If the context of the netlist is ASCII, then the file is stored as ASCII. Otherwise the file format is UNICODE which includes every character of every living language. The LTspice simulator reads ASCII and UNICODE with equal ease.


LTspice XVII - [tube.cir]
File Edit View Simulate Tools Window Help
tube.cir
tube.cir
Ready

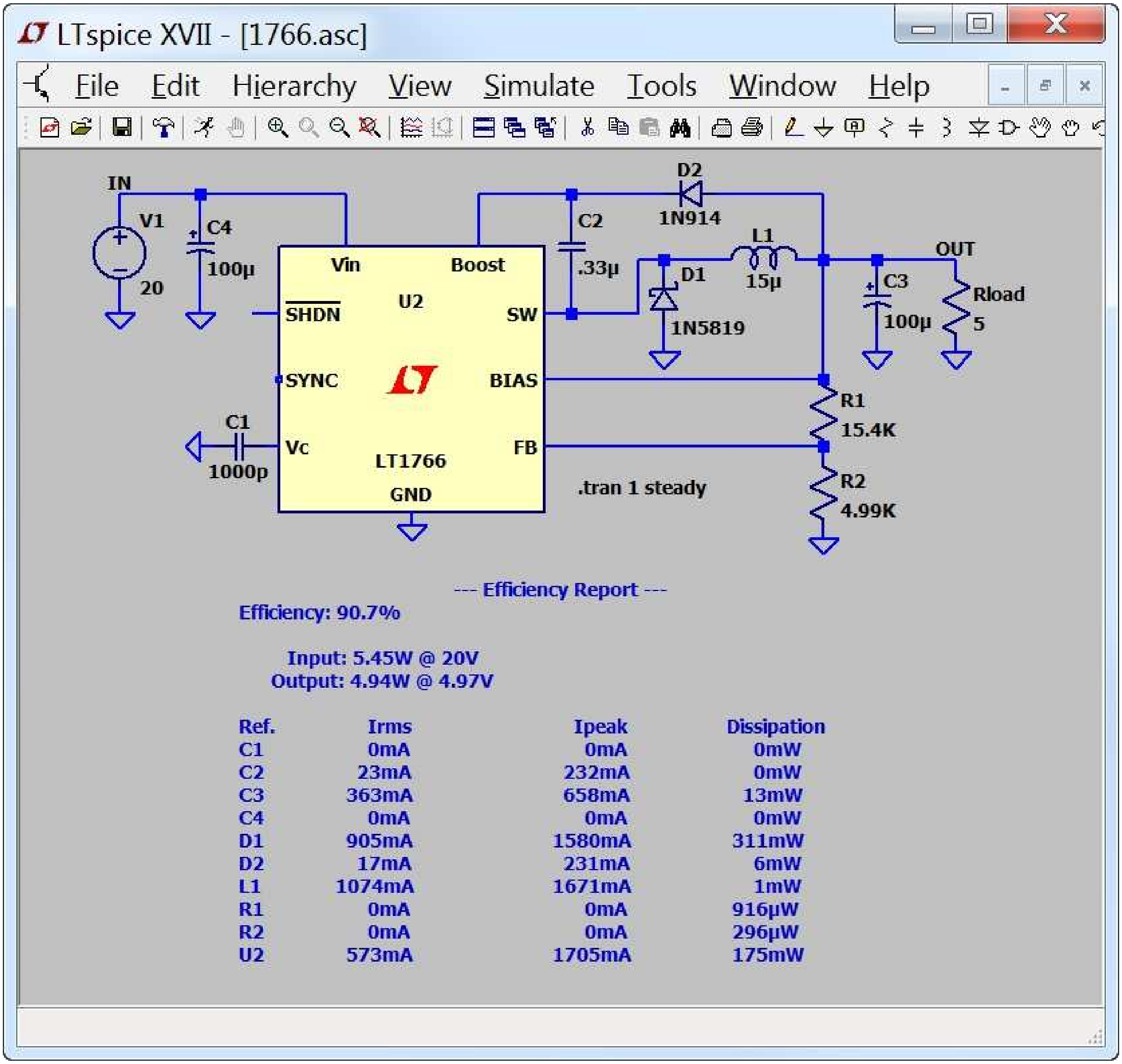
```
V1 A 0 0
V2 G 0 0
X1 A A G 0 6CA7

.dc V1 0 500 1 V2 -50 -10 10

.SUBCKT 6CA7 A S G K ; from www.duncanamps.com
Eat   at  0 VALUE={0.636*ATAN(V(A,K)/23)}
Egs   gs  0 VALUE={LIMIT(V(S,K)/9.3+V(G,K)*0.95,0,1E6)}
Egs2  gs2 0 VALUE={PWRS(V(gs),1.5)}
Ecath cc  0 VALUE={V(gs2)*V(at)}
Ga    A   K VALUE={1.86E-3*V(cc)}
Escrn sc  0 VALUE={V(gs2)*(1.1-V(at))}
Gs    S   K VALUE={1.518E-3*V(sc)*LIMIT(V(S,K),0,10)/10}
Gg    G   K VALUE={PWR(LIMIT(V(G,K)+1,0,1E6),1.5)*50E-6}
Cg1   G   K 15.4p
Cak   A   K 8.4p
Cg1a  G   A 1.1p
.ENDS 6CA7
```

Efficiency Report

It is possible to obtain an efficiency report from a DC-DC converter from a time domain .tran analysis that contains the keyword "steady". After a steady state simulation, an efficiency report can be made visible on the schematic as a block of comment text:



.tran 1 steady

```
                --- Efficiency Report ---
Efficiency: 90.7%

      Input: 5.45W @ 20V
    Output: 4.94W @ 4.97V

Ref.         Irms              Ipeak         Dissipation
C1            0mA                0mA             0mW
C2           23mA              232mA             0mW
C3          363mA              658mA            13mW
C4            0mA                0mA             0mW
D1          905mA             1580mA           311mW
D2           17mA              231mA             6mW
L1         1074mA             1671mA             1mW
R1            0mA                0mA           916µW
R2            0mA                0mA           296µW
U2          573mA             1705mA           175mW
```

The efficiency of the DC-DC converter is derived in the following manner. In order to identify the input and output, there should be exactly one voltage source and one current source. The voltage source is assumed to be the input while the current source is assumed to be the output. The output can also be identified if the load is a resistor with an instance name of Rload. The circuit is run until steady state is sensed by the simulator. This requires the SMPS macromodels to be written with information on how to detect steady state. Usually this is detected by noting when the error amp current, averaged over a clock cycle, diminishes to a small value for several cycles. Then at a clock edge, the energy stored in each reactance is noted and the simulation is run for another ten clock cycles but now integrating the dissipation in every device. At the clock edge of the last cycle, the energy stored in every reactance is noted again and the simulation is stopped. The efficiency is reported as the ratio of output power delivered to the load by the input power sourced by the input voltage after making an adjustment for the change in energy stored in the reactances. Since the dissipation of each device was also noted, it is possible to look how close the energy checksum is to zero.

You can usually compute efficiency of SMPS circuits you draft yourself by using checking the "Stop simulating if steady state is detected" on the Edit Simulation Command editor. After the simulation, use the menu command View=>Efficiency Report.

Automatic detection of steady state doesn't always work. Sometimes the criteria for steady state detection is too strict and sometimes too lenient. You then either adjust the option parameter sstol or simply interactively set the limits for the efficiency integration.

Command Line Switches

The following table summarizes the command line switches understood by the LTspice executable, XVIIx64.exe(XVIIx86.exe on 32-bit systems):

Flag	Description
-alt	Set solver to Alternate.
-ascii	Use ASCII .raw files. Seriously degrades program performance.
-b	Run in batch mode. E.g. "XVIIx64.exe -b deck.cir" will leave the data in file deck.raw
-big	Start as a maximized window.
-encrypt	Encrypt a model library. For 3rd parties wishing to allow people to use libraries without revealing implementation details. Not used by Linear Technology Corporation models.
-FastAccess	Batch conversion of a binary .raw file to Fast Access format.
-ini <path>	Specify an .ini file to use other than %APPDATA%\LTspiceXVII.ini
-max	Synonym for -big
-netlist	Batch conversion of a schematic to a netlist.
-norm	Set solver to Normal.
-PCBnetlist	Batch conversion of a schematic to a PCB format netlist.
-registry	Force LTspice to store user preferences, MRU, etc. in the registry instead of the file %APPDATA%\LTspiceXVII.ini
-Run	Start simulating the schematic opened on the command line without pressing the Run button.
-SOI	Allow MOSFET's to have up to 7 nodes even in subcircuit expansion.
-sync	Sync release
-uninstall	Executes one step of the uninstallation process.

In addition to the above command line switches, LTspice accepts two environmental variables. If PASTE_OMEGA is set, the UNICODE capital omega symbol, Š, is pasted on the clipboard instead of "Ohm." If the environmental variable CAPITAL_KILO is set, LTspice understands you prefer an upper case metric multiplier for all multipliers greater than 1, not just those greater than 1000.

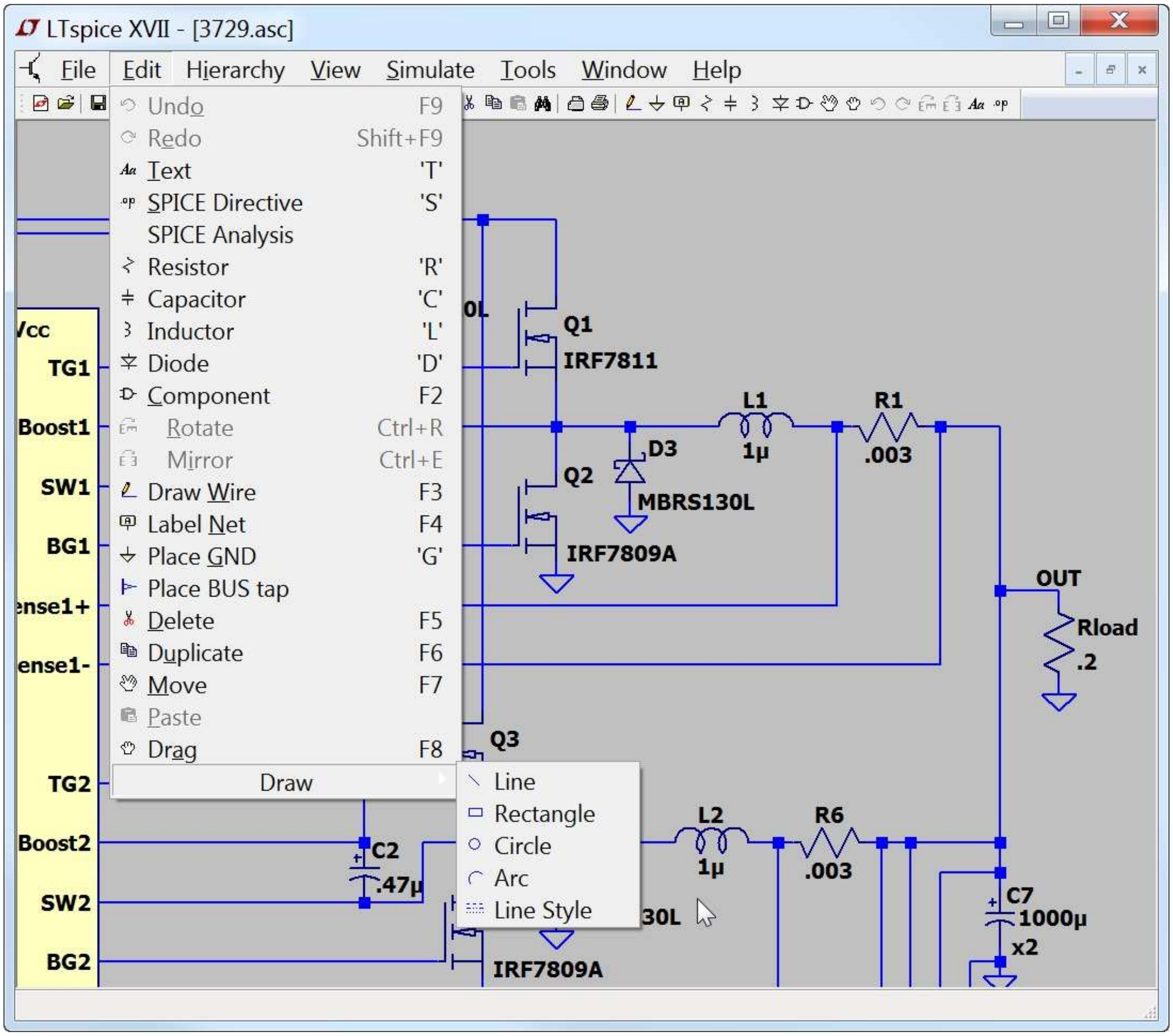
Schematic Capture

LTspice XVII includes a general purpose schematic capture program. It allows you to draft schematics, create symbols, generate netlists, and cross probe simulation data. Unlimited schematic size and hierarchy are supported.

Basic Schematic Editing

The schematic capture program is used to create new schematics or modify the example circuits provided. The circuit size and depth of hierarchy is limited only by computer resources.

The program ships with over 2,000 symbols. These symbols cover most of LTC's power ICs, opamps, comparators, and many general-purpose devices for circuit design. You can also draw your own symbols for devices you wish to import into the program.



LTspice schematic editing is a "verb-noun" interface instead of "noun-verb." That means that instead of first selecting objects(the nouns) to move, drag, copy, or delete(the verbs); you first select the action and then the object. Hence, when you wish to move, mirror, rotate, drag or delete objects, first select the move, drag or delete command. Then you can select an object by clicking on it. You can select multiple objects by dragging a box about them. The program will stay in the move, drag, or delete mode until the right mouse button is clicked or the Esc key is pressed. The verb-noun approach reduces the amount of mouse syntax required to edit schematics. Once mastered, it allows the user to draft faster in LTspice than is possible with pencil and paper.

Undo: Undo the last command.

Redo: Redo the last Undo command.

Text: Place text on the schematic. This merely annotates the schematic with information. This text has no electrical impact on the circuit.

SPICE Directive: Place text on the schematic that will be included in the netlist. This lets you mix schematic capture with a SPICE netlist. It lets you set simulation options, include files that contain models, define new models, or use any other valid SPICE commands. You can even use it to run a subcircuit that you don't have a symbol for by stating an instance of the model(a SPICE command that begins with and 'X') on the schematic and including the definition.

SPICE Analysis: Enter/edit the simulation command.

Resistor: Place a new resistor on the schematic.

Capacitor: Place a new capacitor on the schematic.

Inductor: Place a new inductor on the schematic.

Diode: Place a new diode on the schematic.

Component: Place a new component on the schematic. The command brings up a dialog that lets you browse and preview the symbol database. This is a more general form of the Resistor, Capacitor, Inductor, and Diode commands.

Rotate: Rotate the sprited objects. Note this is grayed out when there are no objected sprited.

Mirror: Mirror the sprited objects. Note this is grayed out when there are no objected sprited

Draw Wire: Click the left mouse button to start a wire. Each mouse click will define a new wire segment. Click on an existing wire segment to join the new wire with an existing one. Right click once to cancel the current wire. Right click again to quit this command. You can draw wires through components such as resistors. The wire will automatically be cut such that the resistor is now in series with the wire.

Label Net: Specify the name of a node so an arbitrary one isn't generated by the netlister for this node.

Place GND: Place a GROUND symbol. This is node "0", the global circuit common.

Delete: Delete objects by clicking on them or dragging a box around them.

Duplicate: Duplicate objects by clicking on them or dragging a box around them. You can copy from one schematic to another if they are both opened in the same invocation of LTspice XVII. Start the Duplicate command in the window of the first schematic. Then make the second schematic the active window and type Ctrl-V.

Move: Click on or drag a box around the objects you wish to move. Then you can move those objects to a new location.

Paste: It is enabled in a new schematic window when objects were already selected with the 'Duplicate' command.

Drag: Click on or drag a box around the objects you wish to drag. Then you can move those objects to a new location and the attached the wires are rubber-band with the new location.

Draw=>Line: Draw a line on the schematic. Such lines have no electrical impact on the circuit, but can be useful for annotating the circuit with notes.

Draw=>Rectangle: Draw a rectangle on the schematic. This rectangle has no electrical impact on the circuit, but can be useful for annotating the circuit with notes.

Draw=>Circle: Draw a circle on the schematic. This circle has no electrical impact on the circuit, but can be useful for annotating the circuit with notes.

Draw=>Arc: Draw an arc on the schematic. This arc has no electrical impact on the circuit, but can be useful for annotating the circuit with notes.

NOTE: The graphical annotations to the schematic; lines, rectangles, circles, and arcs; snap by default to the same grid as the used for electrical contacts of wires and pins. Hold down the control key while positioning these to defeat this snap.

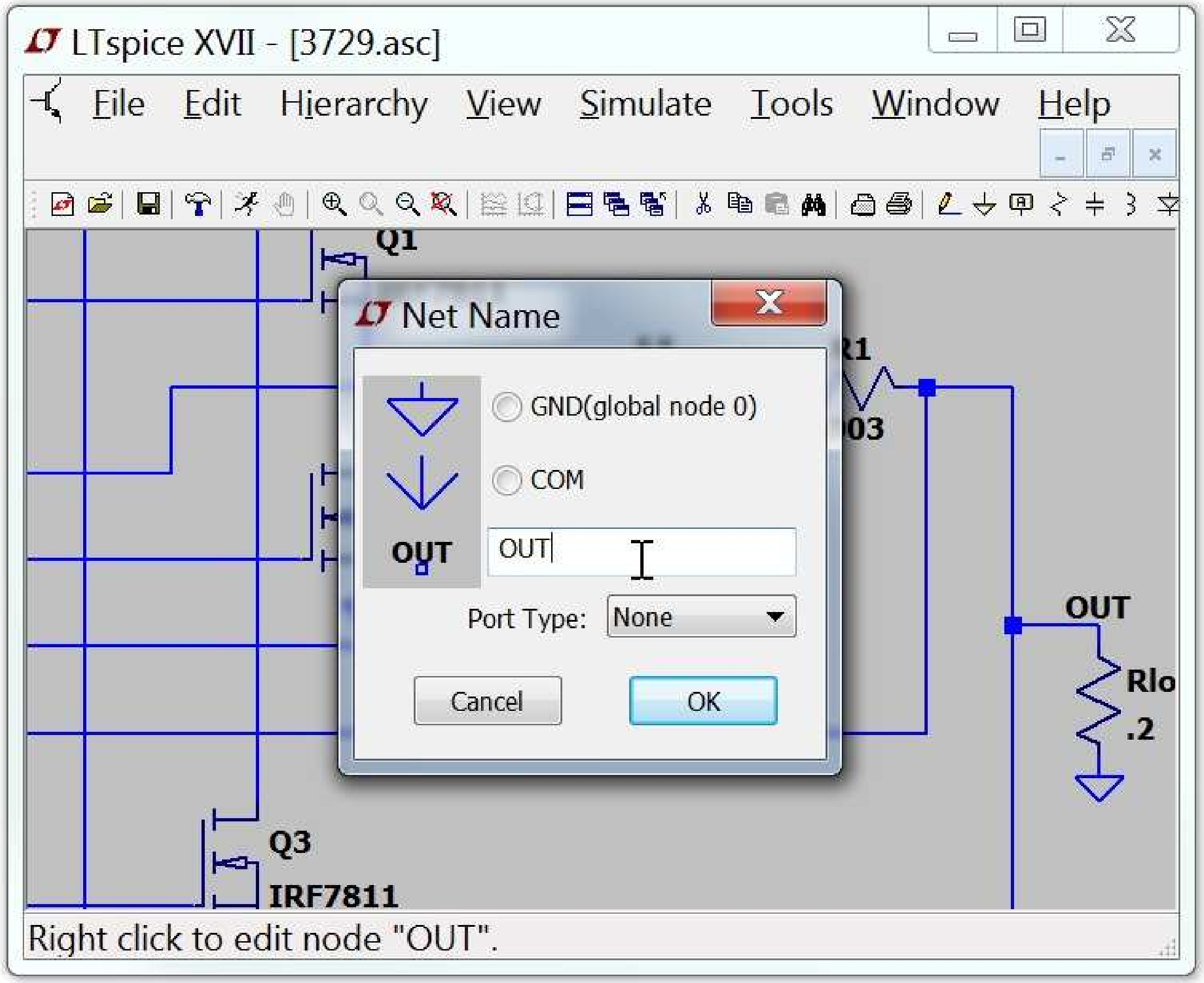
Label a Node Name

Each node in the circuit requires a unique name. You can specify the name of a node so the netlister doesn't make one up for you. Note that as you edit the schematic, automatically generated node names will of course change as the topology of the circuit changes.

Some designers prefer to label many nodes. My own preferred style, and what I recommend to others, is to not label any nodes and only plot data only via cross-probing. This is analogous to the way I scope out a circuit -- I don't label the node before attaching the scope probe. Node "0" is the circuit global ground and is drawn with a special graphical symbol instead of the name "0".

There is also a graphical symbol defined for node "COM", but this node has no special significance. That is, it's not the SPICE global common and it's not even a global node. It's just sometimes convenient to have a graphical symbol associated with a node distinct from ground.

If you give a node a name starting with the characters "$G_"; as in for example, "$G_VDD"; then that node is global no matter where the name occurs in the circuit hierarchy.


LTspice XVII - [3729.asc]
File Edit Hierarchy View Simulate Tools Window Help
Q1
R1
03
OUT
Rlo
.2
Q3
IRF7811
Net Name
OUT
Cancel
OK

○ GND(global node 0)

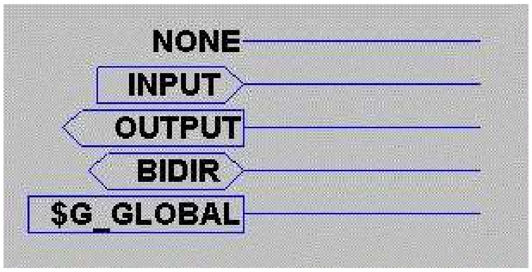
○ COM

OUT

Port Type: None

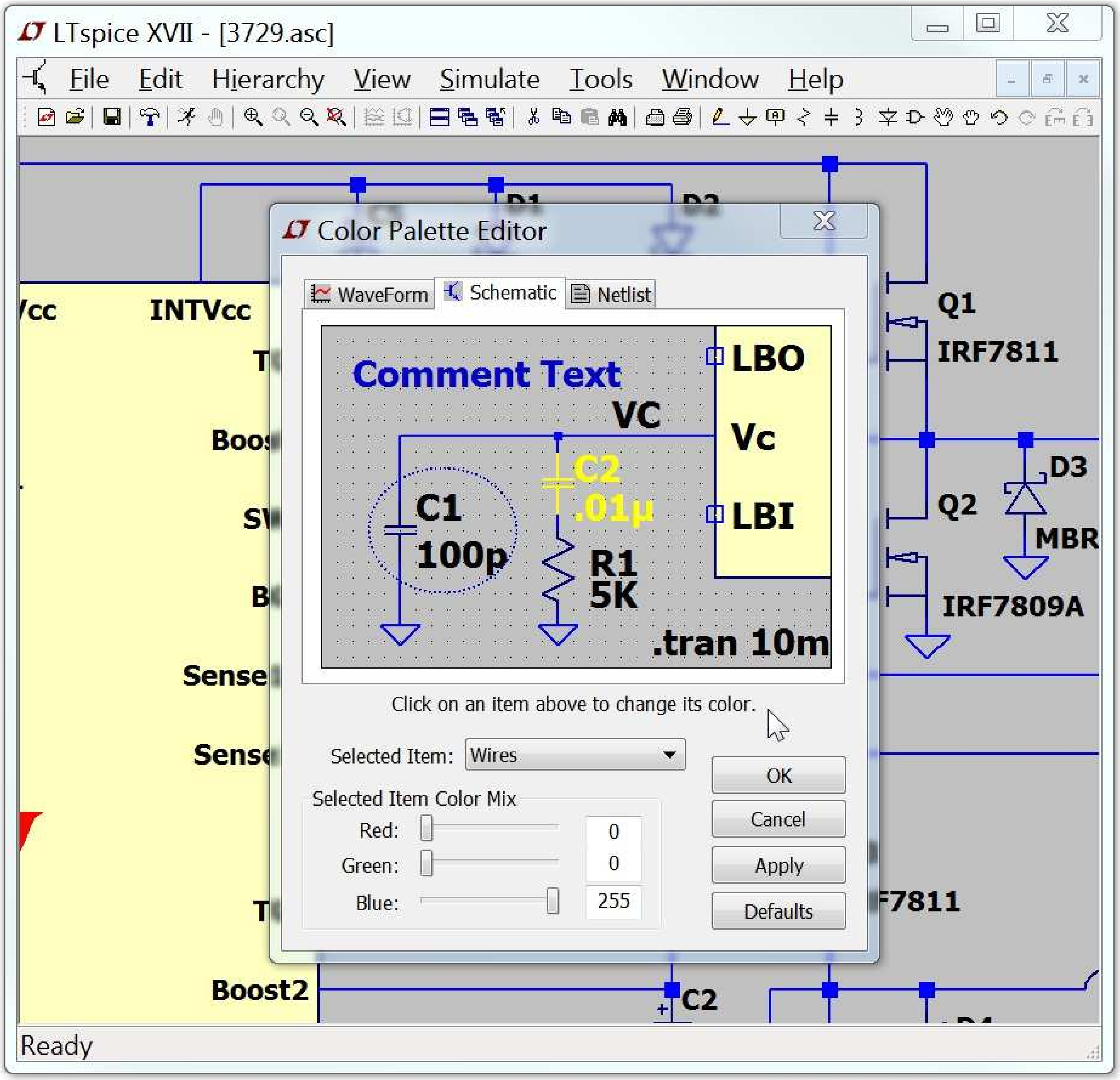
Right click to edit node "OUT".

It is possible to indicate that a node is a port of type input, output, or bi-directional. These port types will be drawn differently but have no significance to the netlister. Indicating a port type can make circuit more readable. Global nodes are also drawn differently in that a box is drawn around the name.



Schematic Colors

Menu command Tools=>Color Preferences colors allows you to set the colors used in displaying the schematics. You click on an object in the sample schematic and use the red, green and blue sliders to adjust the colors to your preferences.



Click on an item above to change its color.

Selected Item: Wires

Selected Item Color Mix

Red: 0

Green: 0

Blue: 255

Placing New Components

Certain frequently used components; such as resistors, capacitors, and inductors; can be selected for placing on the schematic with a toolbar button.

For most symbols, use menu command Edit=>Component to start a dialog to browse for the device you wish.

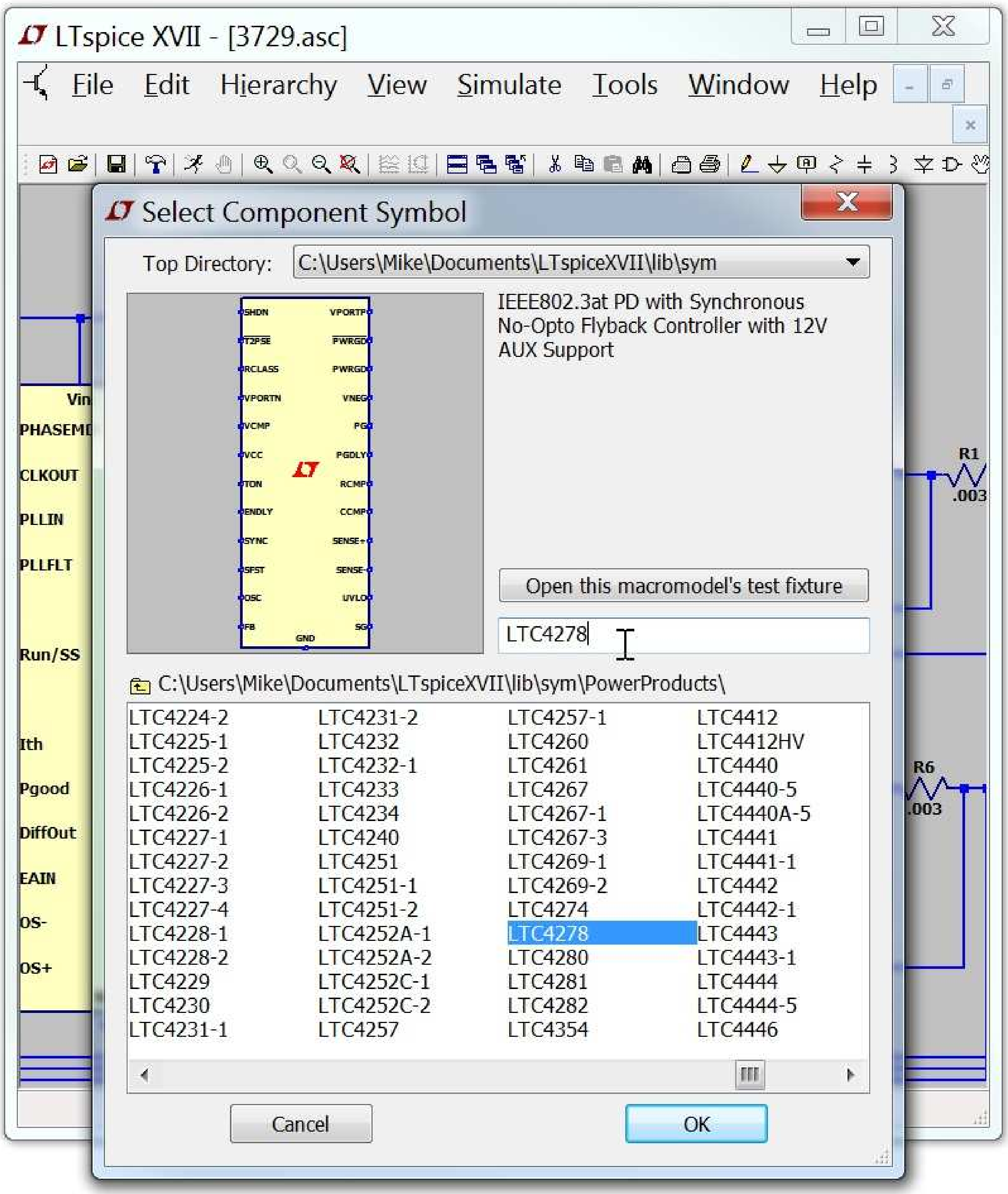
The symbol browser includes some important features:

1. You can type in the first few letters of the symbol name and the browser will jump to that symbol.

2. There is a button to open the test fixture for the Linear Technology models.

3. The symbol and its description is displayed.

4. Any of the symbol search paths set up in the control panel can be selected.



Top Directory: C:\Users\Mike\Documents\LTspiceXVII\lib\sym

IEEE802.3at PD with Synchronous No-Opto Flyback Controller with 12V AUX Support

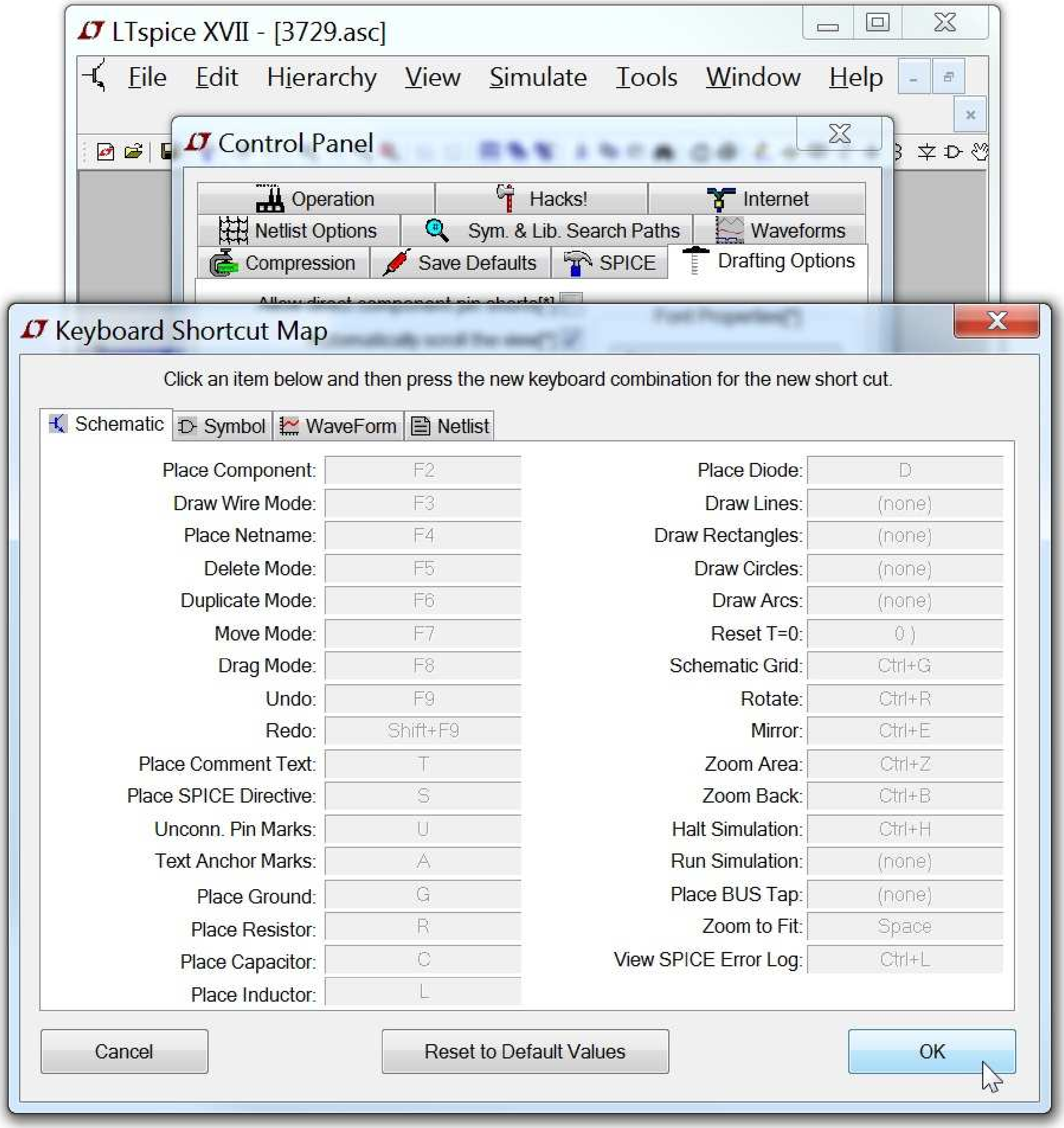
LTC4278

C:\Users\Mike\Documents\LTspiceXVII\lib\sym\PowerProducts\

LTC4224-2
LTC4225-1
LTC4225-2
LTC4226-1
LTC4226-2
LTC4227-1
LTC4227-2
LTC4227-3
LTC4227-4
LTC4228-1
LTC4228-2
LTC4229
LTC4230
LTC4231-1
LTC4231-2
LTC4232
LTC4232-1
LTC4233
LTC4234
LTC4240
LTC4251
LTC4251-1
LTC4251-2
LTC4252A-1
LTC4252A-2
LTC4252C-1
LTC4252C-2
LTC4257
LTC4257-1
LTC4260
LTC4261
LTC4267
LTC4267-1
LTC4267-3
LTC4269-1
LTC4269-2
LTC4274
LTC4278
LTC4280
LTC4281
LTC4282
LTC4354
LTC4412
LTC4412HV
LTC4440
LTC4440-5
LTC4440A-5
LTC4441
LTC4441-1
LTC4442
LTC4442-1
LTC4443
LTC4443-1
LTC4444
LTC4444-5
LTC4446

Programming Keyboard Shortcuts

The menu command Tools=>Control Panel=>Drafting Options=>Hot Keys allows you to program the keyboard short cuts for most commands. Simply mouse click on a command and then press the key or key combination you would like to code for the command. To remove a shortcut, click on the command and press the "Delete" key.



Click an item below and then press the new keyboard combination for the new short cut.

Place Component:	F2	Place Diode:	D
Draw Wire Mode:	F3	Draw Lines:	(none)
Place Netname:	F4	Draw Rectangles:	(none)
Delete Mode:	F5	Draw Circles:	(none)
Duplicate Mode:	F6	Draw Arcs:	(none)
Move Mode:	F7	Reset T=0:	0)
Drag Mode:	F8	Schematic Grid:	Ctrl+G
Undo:	F9	Rotate:	Ctrl+R
Redo:	Shift+F9	Mirror:	Ctrl+E
Place Comment Text:	T	Zoom Area:	Ctrl+Z
Place SPICE Directive:	S	Zoom Back:	Ctrl+B
Unconn. Pin Marks:	U	Halt Simulation:	Ctrl+H
Text Anchor Marks:	A	Run Simulation:	(none)
Place Ground:	G	Place BUS Tap:	(none)
Place Resistor:	R	Zoom to Fit:	Space
Place Capacitor:	C	View SPICE Error Log:	Ctrl+L
Place Inductor:	L		

PCB Netlist Extraction

The schematic menu command Tools=>Export Netlist allows you to generate the ASCII netlist for PCB layout. Note that you would have to make a set of symbols that have the same pin order as the LTspice symbol. Some PCB tools don't even define their diode to netlist against standard SPICE pin order. Also, LTspice symbol pin numbers are often different than the Linear Technology product pin numbers, especially when a product is available in more than one package.

The following formats are available: Accel, Algorex, Allegro, Applicon Bravo, Applicon Leap, Cadnetix, Calay, Calay90, CBDS, Computervision, EE Designer, ExpressPCB, Intergraph, Mentor, Multiwire, PADS, Scicards, Tango, Telesis, Vectron, and Wire List.

Editing Components

There are three ways to edit components:

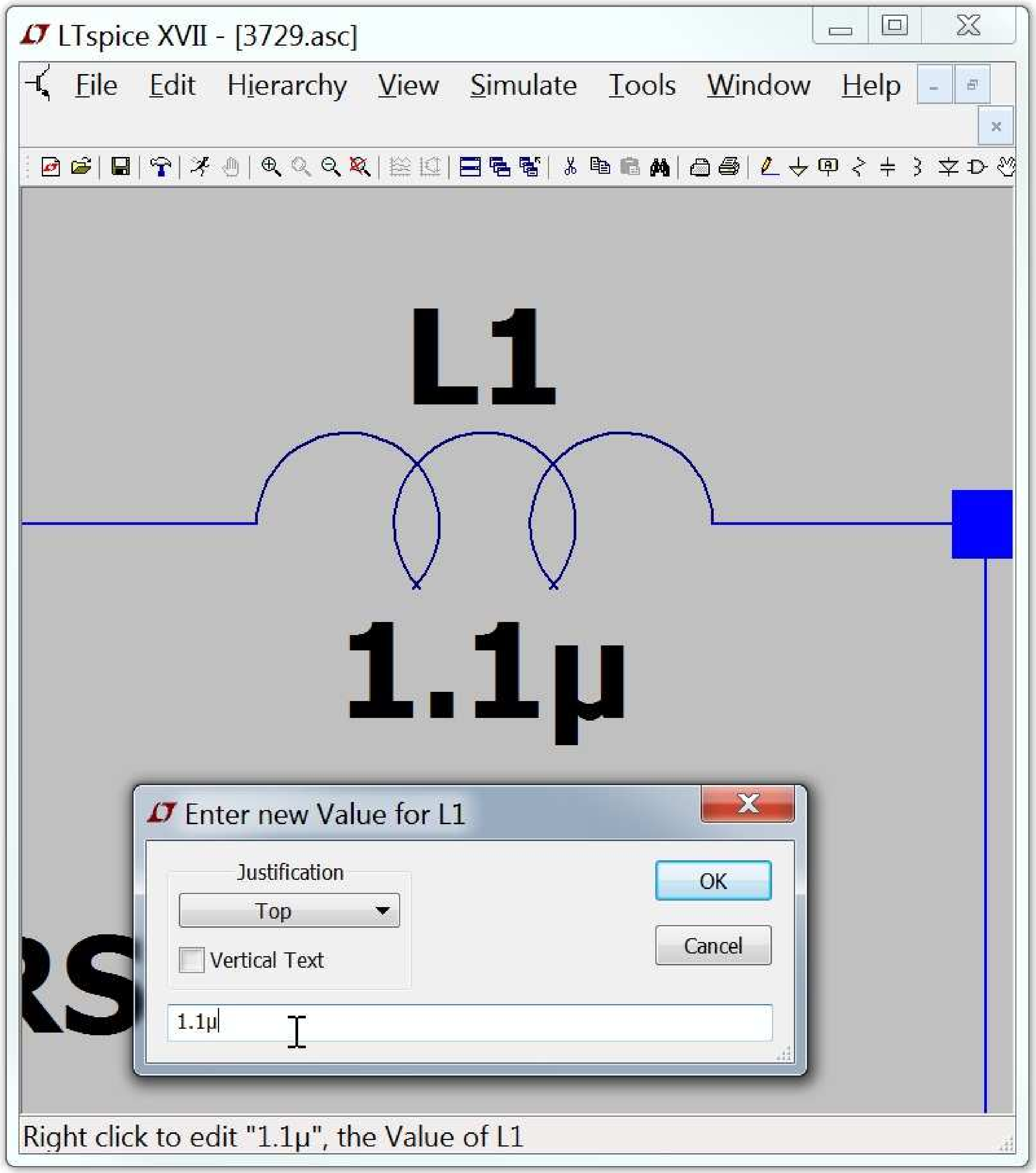
1. **Expert Mode**: This is the mode you use most of the time. Simply point at the text you want to edit, like a component value, right click, and type in the text you want. When you point at the text, the mouse cursor will turn into a text caret if you can edit it.

2. **Assisted Mode**: To enter Assisted Mode, right click on the body of the component. That will invoke a GUI that will help you edit that type of component. It's useful when you are unsure of the SPICE syntax to what you are trying to accomplish, such as entering a piecewise linear data for a voltage source or when you want to pick a semiconductor from a list of modeled devices.

3. **Super Expert Mode**: To enter Super Expert Mode, point at the body of a symbol, hold down the control key, and right mouse click. You can then have complete control of every attribute. You can make any attribute and string you wish and you can choose whether or not the attribute is visible from the schematic.

Edit a Visible Attribute

Most visible component attribute fields can be edited by pointing at it with the mouse and then right clicking. The mouse cursor will turn into a text caret when it's pointing at the text. This is a convenient way of changing the value of a component.



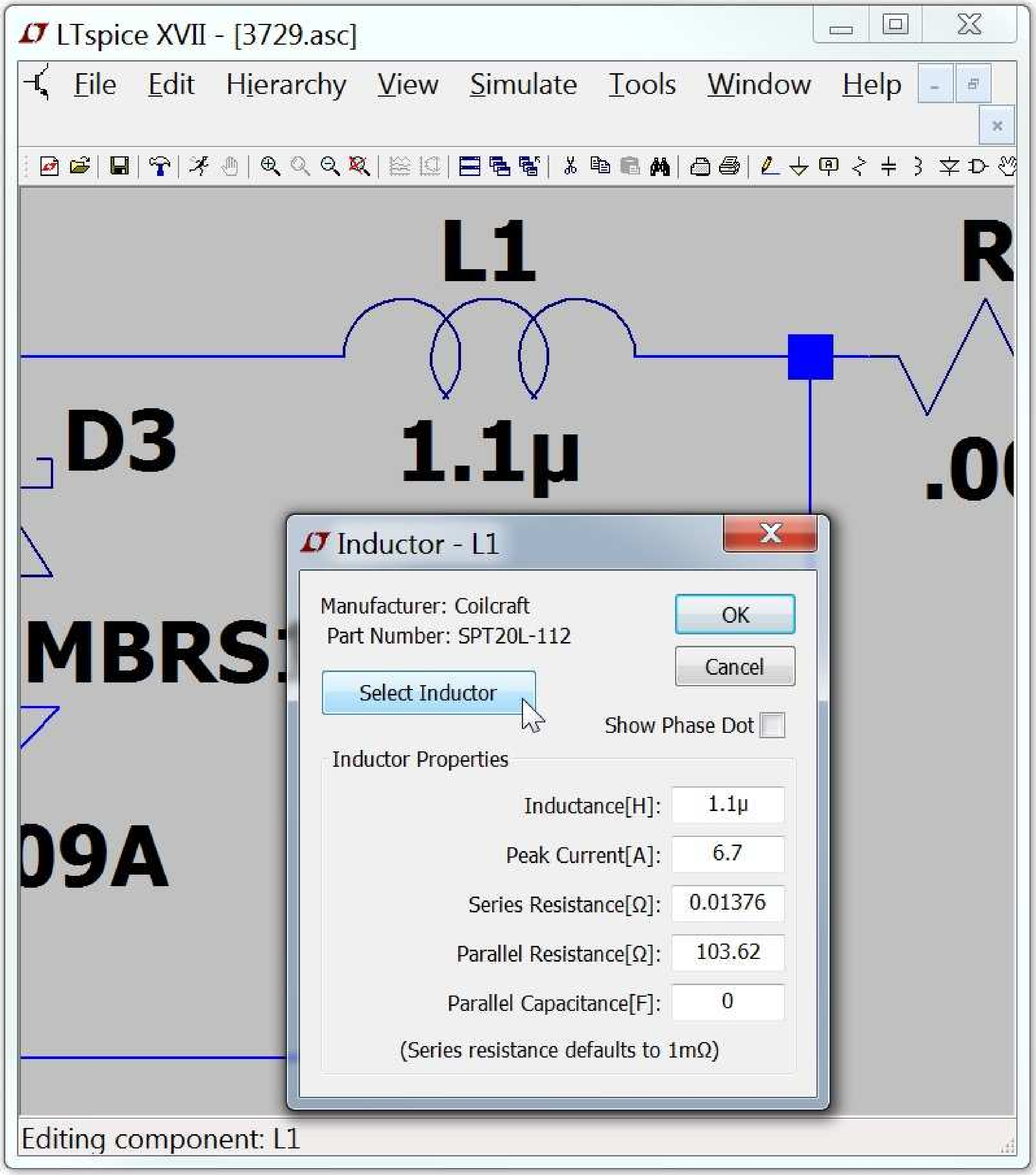
□ Vertical Text

1.1µ

Right click to edit "1.1µ", the Value of L1

Specialized Component Editors

Many component types, such are resistors, capacitors, inductors, diodes, bipolar transistors, MOSFET transistors, JFET transistors, independent voltage sources, independent current sources, and hierarchical circuit blocks have special editors. These editors can access the appropriate database of related components. To use these editors, right mouse click on the body of the component.



Manufacturer: Coilcraft
Part Number: SPT20L-112

Show Phase Dot ☐

Inductor Properties

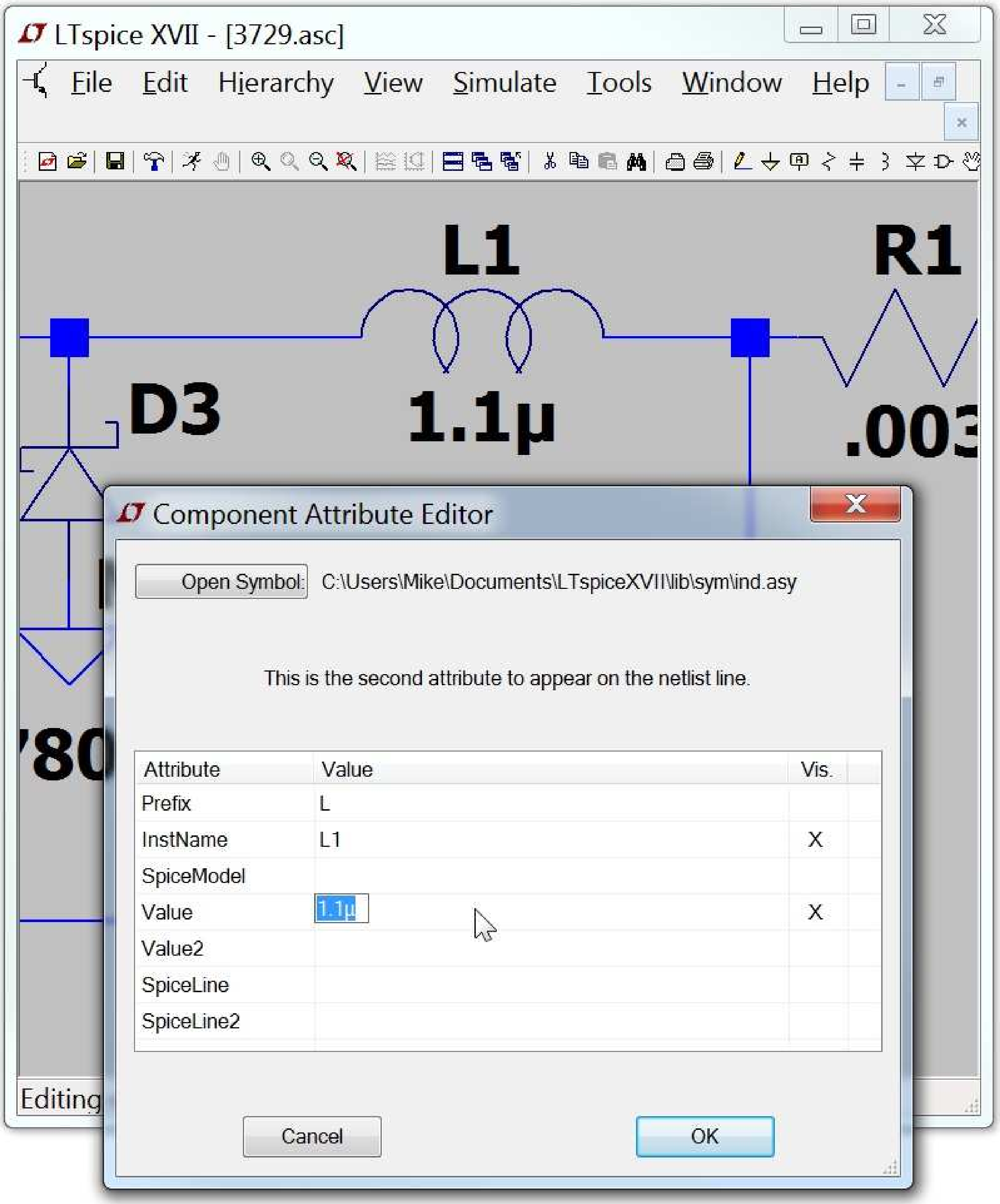
Inductance[H]:	1.1µ
Peak Current[A]:	6.7
Series Resistance[Ω]:	0.01376
Parallel Resistance[Ω]:	103.62
Parallel Capacitance[F]:	0

(Series resistance defaults to 1mΩ)

Editing component: L1

General Attribute Editor

Sometimes it is desired to get direct access to every available component attribute to edit their contents and visibility. An editor that allows you to do this can be reached by placing the mouse over the body of a symbol, holding down the control key, and clicking the right mouse button. A dialog box will appear that displays all available symbol attributes. Next to each field is a check box to indicate if the field should be visible on the schematic.



Component Attribute Editor

Open Symbol: C:\Users\Mike\Documents\LTspiceXVII\lib\sym\ind.asy

This is the second attribute to appear on the netlist line.

Attribute	Value	Vis.
Prefix	L	
InstName	L1	X
SpiceModel		
Value	1.1µ	X
Value2		
SpiceLine		
SpiceLine2		

The attributes SpiceModel, Value, Value2, SpiceLine, and SpiceLine2 are all part of the overall value of the component. In terms of the way the component is netlisted for SPICE, the component will generate a line of SPICE that looks like this:

```
  <name> node1 node2 [...] <SpiceModel>
  + <Value> <Value2> <SpiceLine> <SpiceLine2>
```

The prefix attribute character is prefixed to the reference designator if different than the first character of the reference designator. The Prefix character and InstName will be separated with a '§' character in this case. For example, if you have a Prefix attribute of "M" and an InstName attribute of "Q1", the name in the netlist will be M§Q1. This allows you use reference designators with a leading character different than SPICE uses to identify the type of device.

There are three exceptions to the above rule. There is one special symbol, jumper, that does not translate into a circuit element, but is a directive to the netlist generator that there are two different names for the same electrically identical node. Another exception is a symbol defined to have a prefix of 'X' and both a Value and Value2 attributes defined. Such a component netlists as two lines of SPICE:

```
  .lib <SpiceModel>
  <name> node1 node2 [...] <Value2>
```

This allows symbols to be defined that automatically include the library that contains the definition of the subcircuit called by the component. The netlist compiler removes duplicate .lib statements. Note that such components are not editable on the schematic. The third exception is a symbol that has other exception is a symbol defined to have a prefix of 'X' and a ModelFile attribute defined. Such a component also netlists as two lines of SPICE:

```
  .lib <ModelFile>
  <name> node1 node2 [...] <SpiceModel> <Value> <Value2>
  <SpiceLine> <SpiceLine2>
```

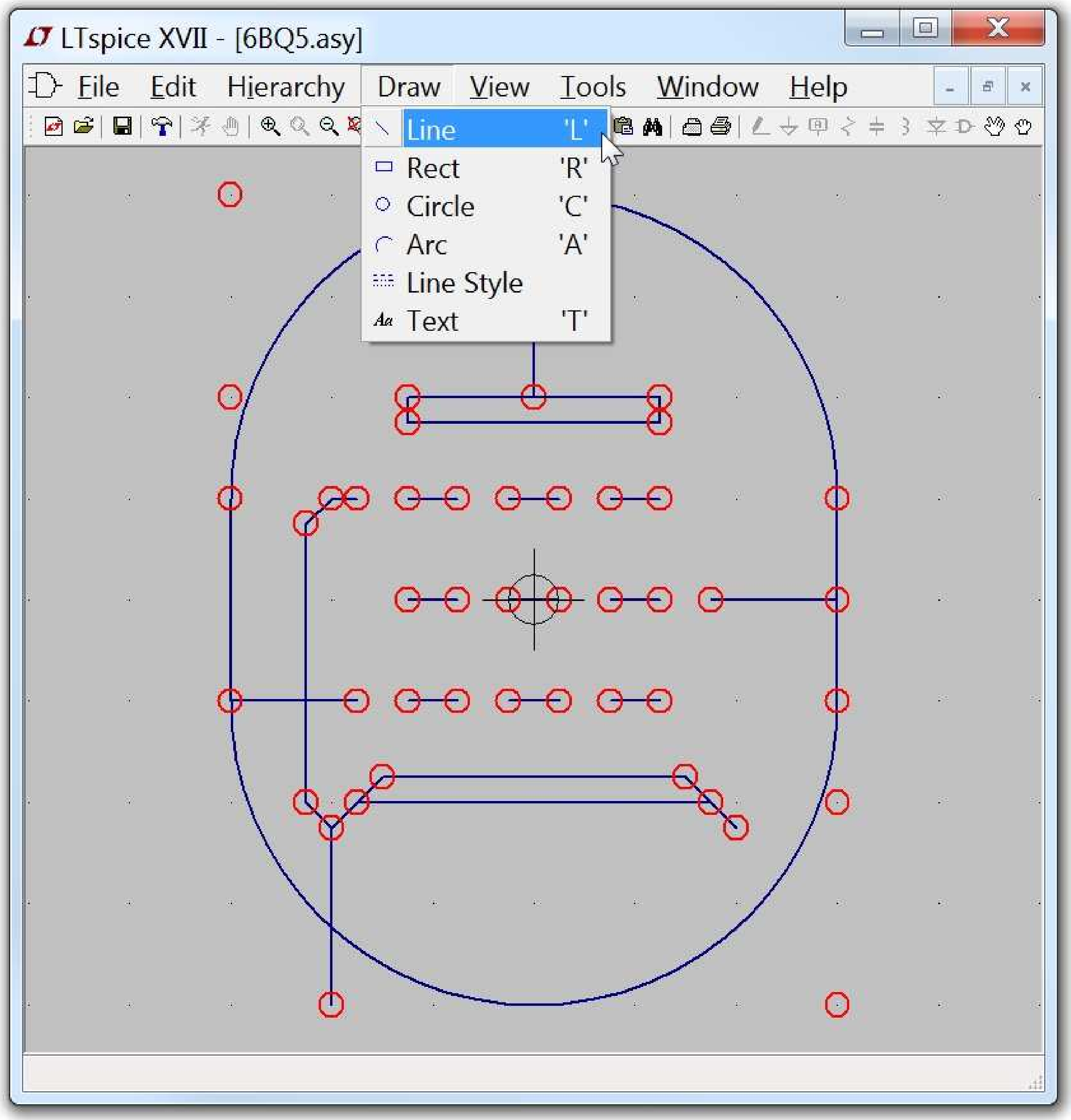
Use this method when you want to automatically include a library file yet still want to have an instance of this symbol editable. If the symbol attribute SpiceModel exists and is the name of a subcircuit in the file specified as <ModelFile> then a drop list of all subcircuits names will be available when an instance of the symbol is edited on a schematic.

Creating New Symbols

Symbols can represent a primitive device such as a resistor or a capacitor; a subcircuit libraried in a separate file; or another page of the schematic. This section describes how to define your own new symbols. To start a new symbol, use the menu command File=>New Symbol.

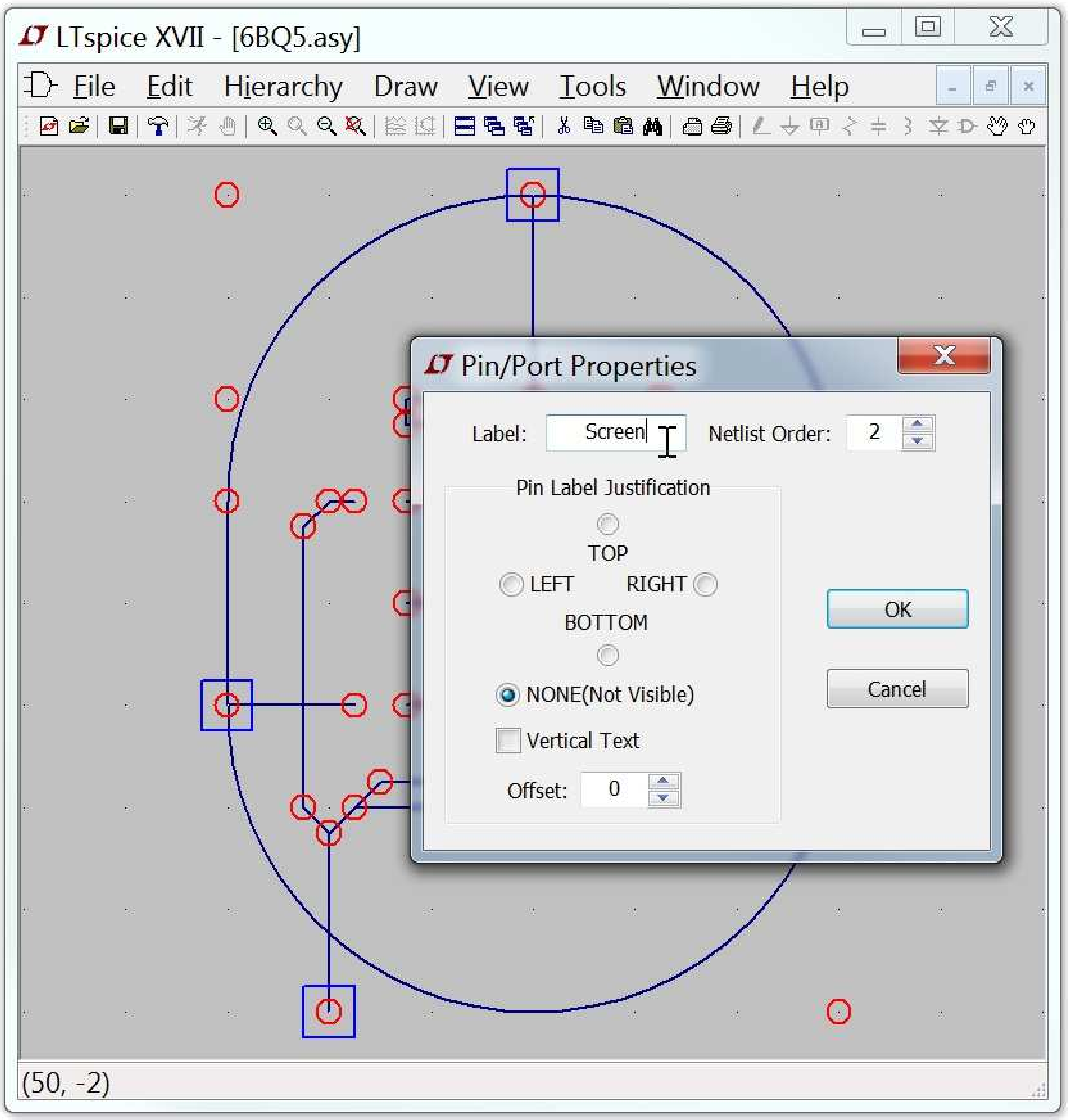
Drawing the body

You draw the body of the symbol as a series of lines, rectangles, circles, and arcs. The objects have no electrical impact on the circuit. You can also draw text on the symbol with the Draw=>Text command that has no impact on the circuit. The anchor points of these objects are drawn with small red circles so you know what to grab when dragging them about. You can toggle the red markers off and on with the menu command View=>Mark Object Anchors


LTspice XVII - [6BQ5.asy]
File
Edit
Hierarchy
Draw
View
Tools
Window
Help
Line 'L'
Rect 'R'
Circle 'C'
Arc 'A'
Line Style
Text 'T'

Adding the Pins

The pins allow electrical connection to the symbol. Use the menu command Edit=>Add Pin/Port to add a new pin.



LTspice XVII - [6BQ5.asy]
File Edit Hierarchy Draw View Tools Window Help
(50, -2)

Pin/Port Properties

Label: Screen

Netlist Order: 2

Pin Label Justification

○ TOP

○ LEFT

RIGHT ○

BOTTOM ○

● NONE(Not Visible)

□ Vertical Text

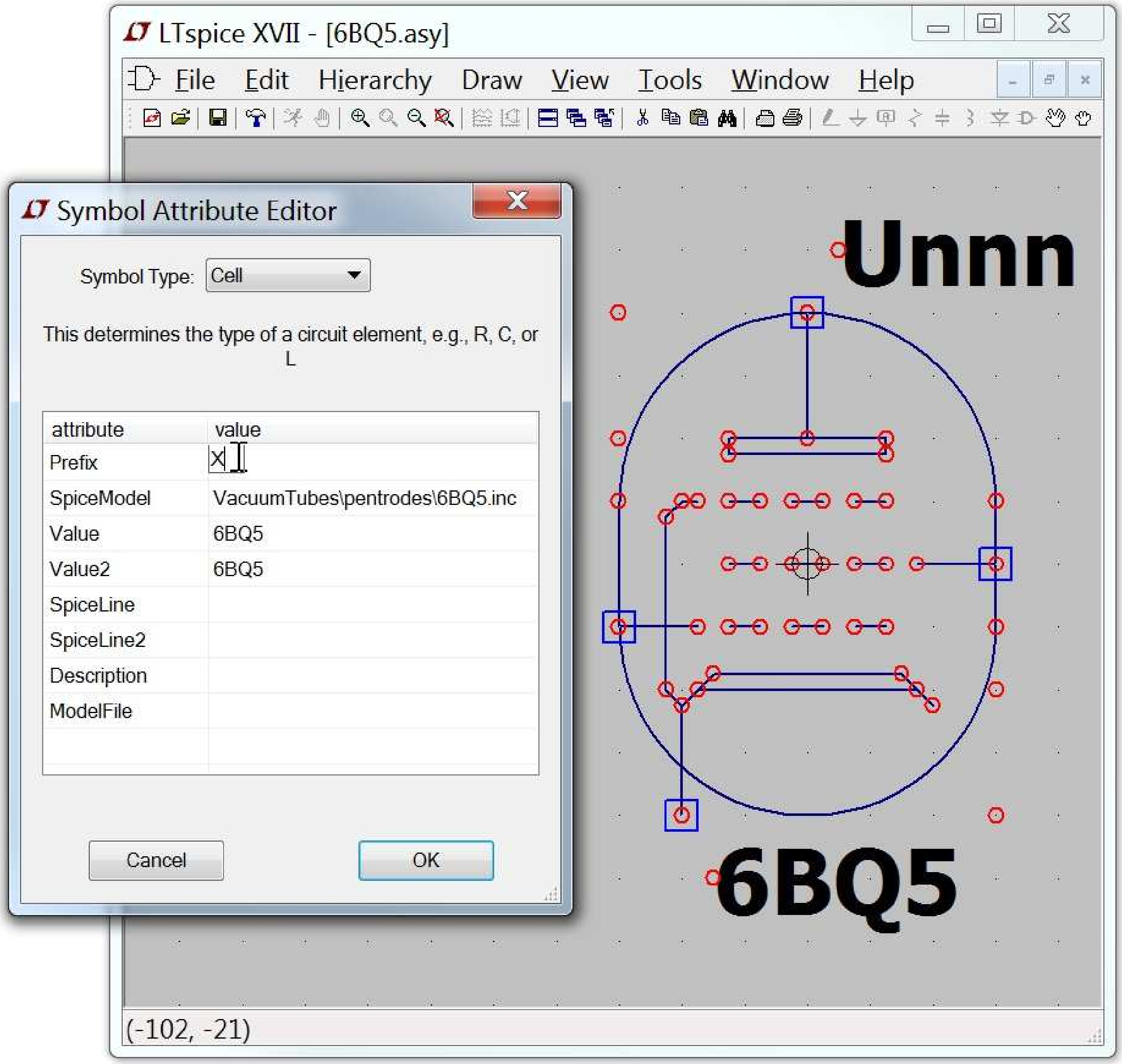
Offset: 0

The "Pin Label Position" determines how the pin label is presented. "TOP", "BOTTOM", "LEFT", and "RIGHT" are text justifications. For example, if a pin label is TOP justified, the pin(the label's text justification's anchor point) will be above the label. If the symbol represents a SPICE primitive element or a subcircuit from a library, then the pin label has no direct electrical impact on the circuit. However, if the symbol represents lower-level schematic of a hierarchical schematic, then the pin name is significant as the name of a net in the lower level schematic.

The "Netlist Order" determines the order this pin is netlisted for SPICE.

Adding Attributes

You can define default attributes for a symbol using the menu command Edit=>Attributes=>Edit Attributes. The most important attribute is called the "Prefix". This determines the basic type of symbol. If the symbol is intended to represent a SPICE primitive, the symbol should have the appropriate prefix, R for resistor, C or capacitor, M for MOSFET, etc. See the LTspice reference for a complete set of SPICE primitives available. The prefix should be 'X' if you want to use the symbol to represent a subcircuit defined in a library.



Symbol Type: Cell

This determines the type of a circuit element, e.g., R, C, or L

attribute	value
Prefix	X
SpiceModel	VacuumTubes\pentrodes\6BQ5.inc
Value	6BQ5
Value2	6BQ5
SpiceLine	
SpiceLine2	
Description	
ModelFile	

The symbol's attributes can be overridden in the instance of the symbol as a component in a schematic. For example, if you have a symbol for a MOSFET with a prefix attribute of 'M', it's possible to override the prefix to an 'X' on an instance-by-instance basis so that the transistor can be modeled as subcircuit instead.

There is a special combination of attributes that will cause a required library to be automatically included in every schematic that uses the symbol:

```
      Prefix: X
  SpiceModel: <name of file including the spicemodel>
       Value: <What ever you want visible on the schematic>
      Value2: <The value as you want in the netlist>
```

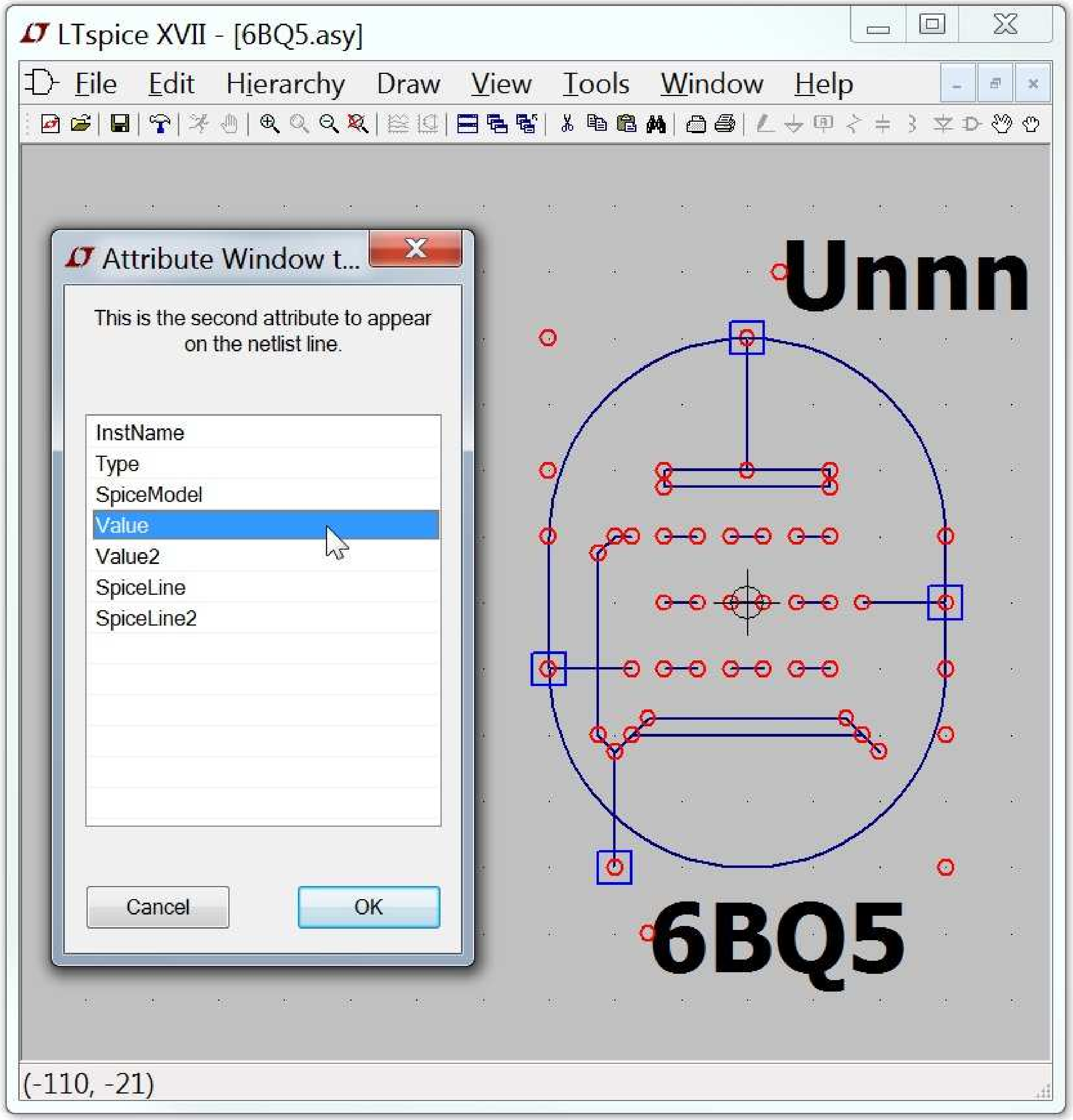
Value2 would be made to coincide with a subcircuit name defined in the file including the spicemodel and may pass additional parameters to the subcircuit. When a symbol is defined in this manner, an instance of the symbol as a component on a schematic cannot be edited to have different attributes.

If you wish the symbol to represent another page of a hierarchical schematic, all attributes should be left blank the symbol type should be changed from "Cell" to "Block". No attribute values need be set.

There is a symbol attribute, ModelFile, that may be specified. This is used for the name of a file to be included in the netlist as a library. If the prefix attribute is 'X' and there is a symbol attribute SpiceModel defined that is subcircuit defined in the model file, then a drop list of all subcircuits names will be available when an instance of the symbol is edited on a schematic.

Attribute Visibility

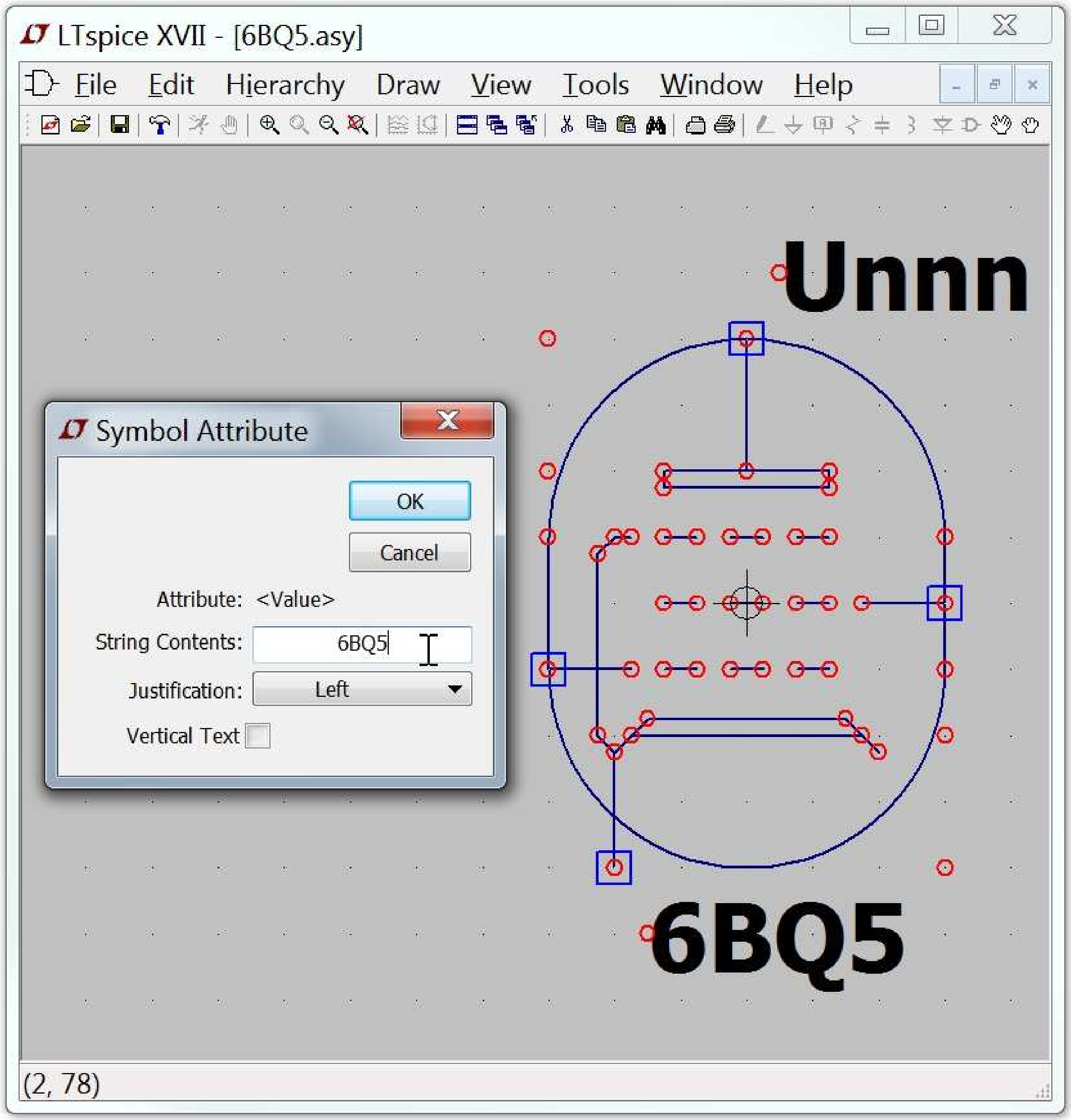
You can edit the visibility of attributes using the menu command Edit=>Attributes=>Attribute Window. After you select an attribute with this dialog you will then be able to position it as you wish with respect to the symbol.



This is the second attribute to appear on the netlist line.

- InstName
- Type
- SpiceModel
- Value
- Value2
- SpiceLine
- SpiceLine2

You can modify the text justification and contents of attributes that you've already made visible by right mouse clicking on the text of the attribute.


LTspice XVII - [6BQ5.asy]
File Edit Hierarchy Draw View Tools Window Help
Unnn
6BQ5
Symbol Attribute
OK
Cancel
(2, 78)

Attribute: <Value>

String Contents: 6BQ5

Justification: Left

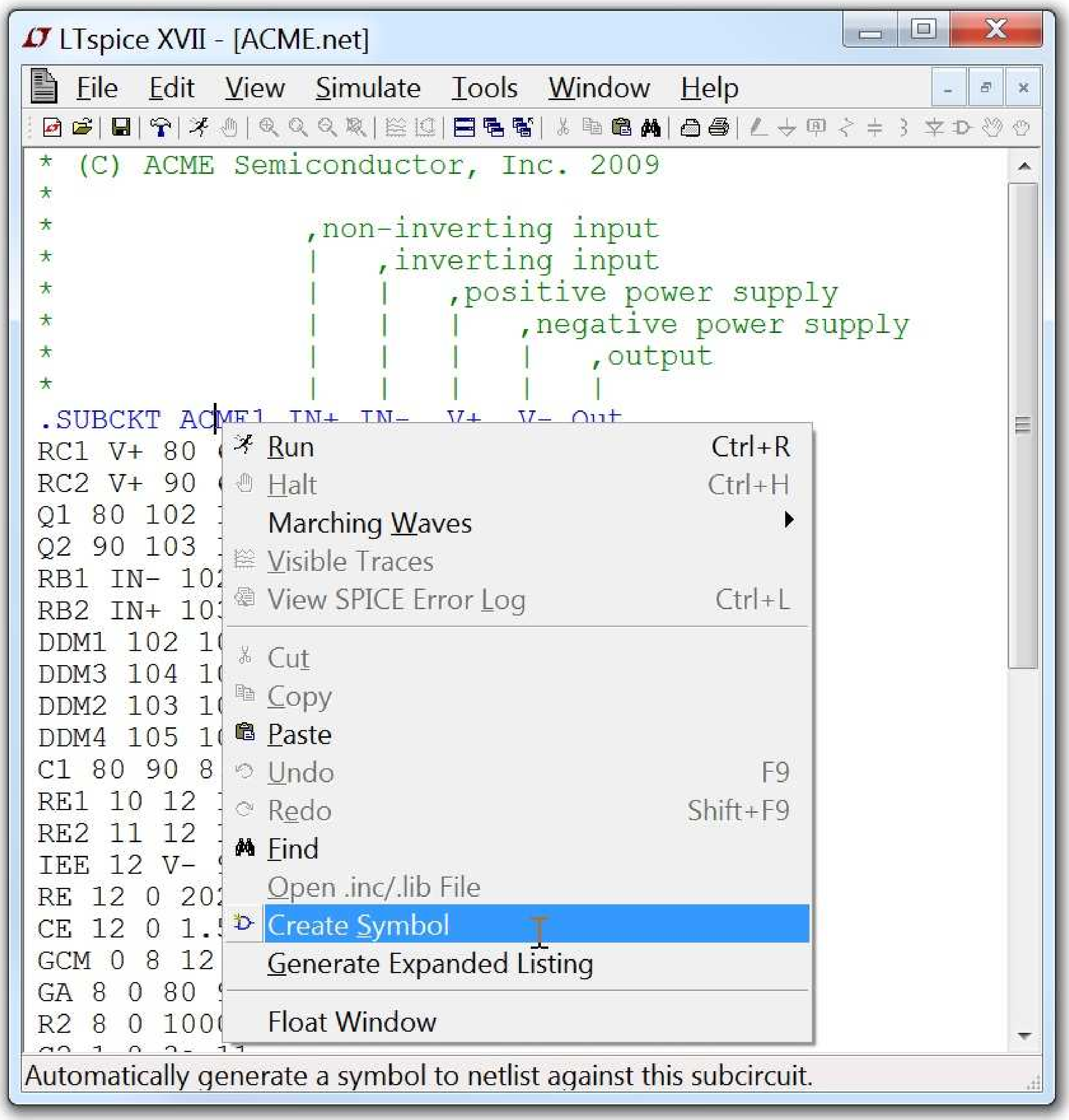
Vertical Text □

Automatic Symbol Generation

A symbol can be automatically generated in two situations:

1. When editing a schematic, you can execute menu item Hierarchy=>Open this Sheet's Symbol. When no symbol is found, LTspice will ask if you would like one automatically generated. This symbol then can be used to call this sheet of circuitry in some higher level schematic. Note that if you edit the ports of the schematic, the symbol will no longer netlist correctly against the schematic and you should delete the symbol and regenerate it.

2. When editing an ASCII netlist that contains subcircuit definitions, you place the cursor on the line containing the name of the subcircuit, right click, and execute context menu item "Create Symbol." **For most users, this is the only method you should ever consider for adding third-party models defined as subcircuits since all the details are handled for you.**



```
* (C) ACME Semiconductor, Inc. 2009
*
*              ,non-inverting input
*              |   ,inverting input
*              |   |   ,positive power supply
*              |   |   |   ,negative power supply
*              |   |   |   |   ,output
*              |   |   |   |   |
.SUBCKT ACME1 IN+ IN-  V+  V- Out
RC1 V+ 80 (
RC2 V+ 90 (
Q1 80 102 1
Q2 90 103 1
RB1 IN- 102
RB2 IN+ 103
DDM1 102 10
DDM3 104 10
DDM2 103 10
DDM4 105 10
C1 80 90 8
RE1 10 12 1
RE2 11 12 1
IEE 12 V- 9
RE 12 0 202
CE 12 0 1.5
GCM 0 8 12
GA 8 0 80 9
R2 8 0 1000
```

Automatically generate a symbol to netlist against this subcircuit.

Hierarchy

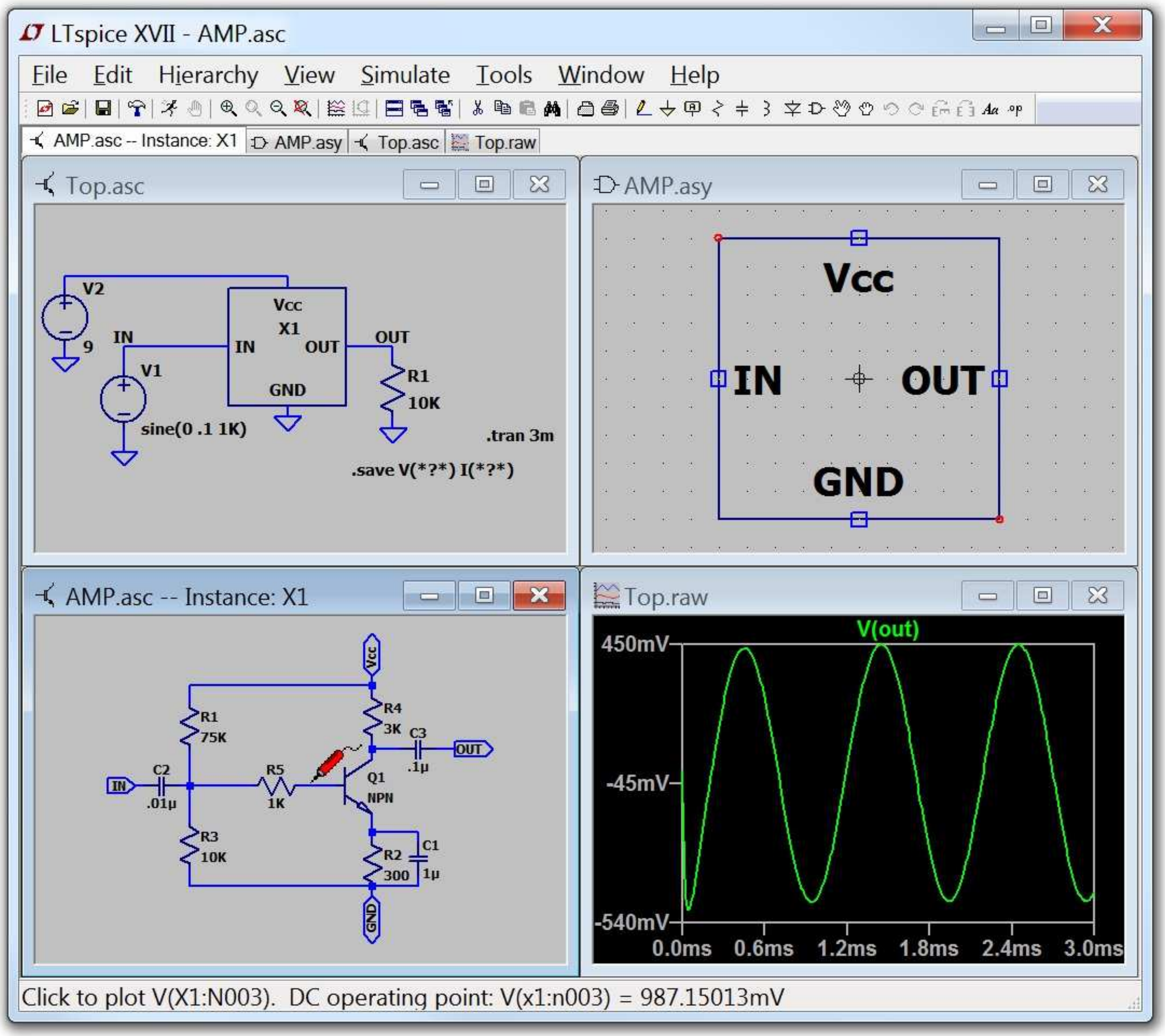
Hierarchical schematic drafting has powerful advantages. Much larger circuits can be drafted than can fit onto a one sheet schematic while retaining the clarity of the smaller schematics. Repeated circuitry to be easily handled in an abstract manner. Blocks of circuitry can be libraried for latter use in a different project.

Rules of Hierarchy

Navigating the Hierarchy

Rules of Hierarchy

The way to refer to another schematic as a block in a higher level schematic is to create a symbol with the same name as the block schematic and then by placing that symbol on the higher level schematic. For example, if you have a top-level schematic called topXYZ.asc and another schematic file called preamp.asc that you wish to place in the schematic of topXYZ then create a symbol called preamp.asy and place an instance of that symbol on the schematic of topXYZ. The electrical connectivity between the schematics is established by connecting wires of the higher-level schematic to pins on the lower level block's symbol that matches the name of a node in the lower-level schematic. As the names of symbols used as schematic blocks and the names of the schematics corresponding to those block must consist of valid characters that can be used as filenames. They also cannot contain the space character.



```
.tran 3m
.save V(*?*) I(*?*)
```

Click to plot V(X1:N003). DC operating point: V(x1:n003) = 987.15013mV

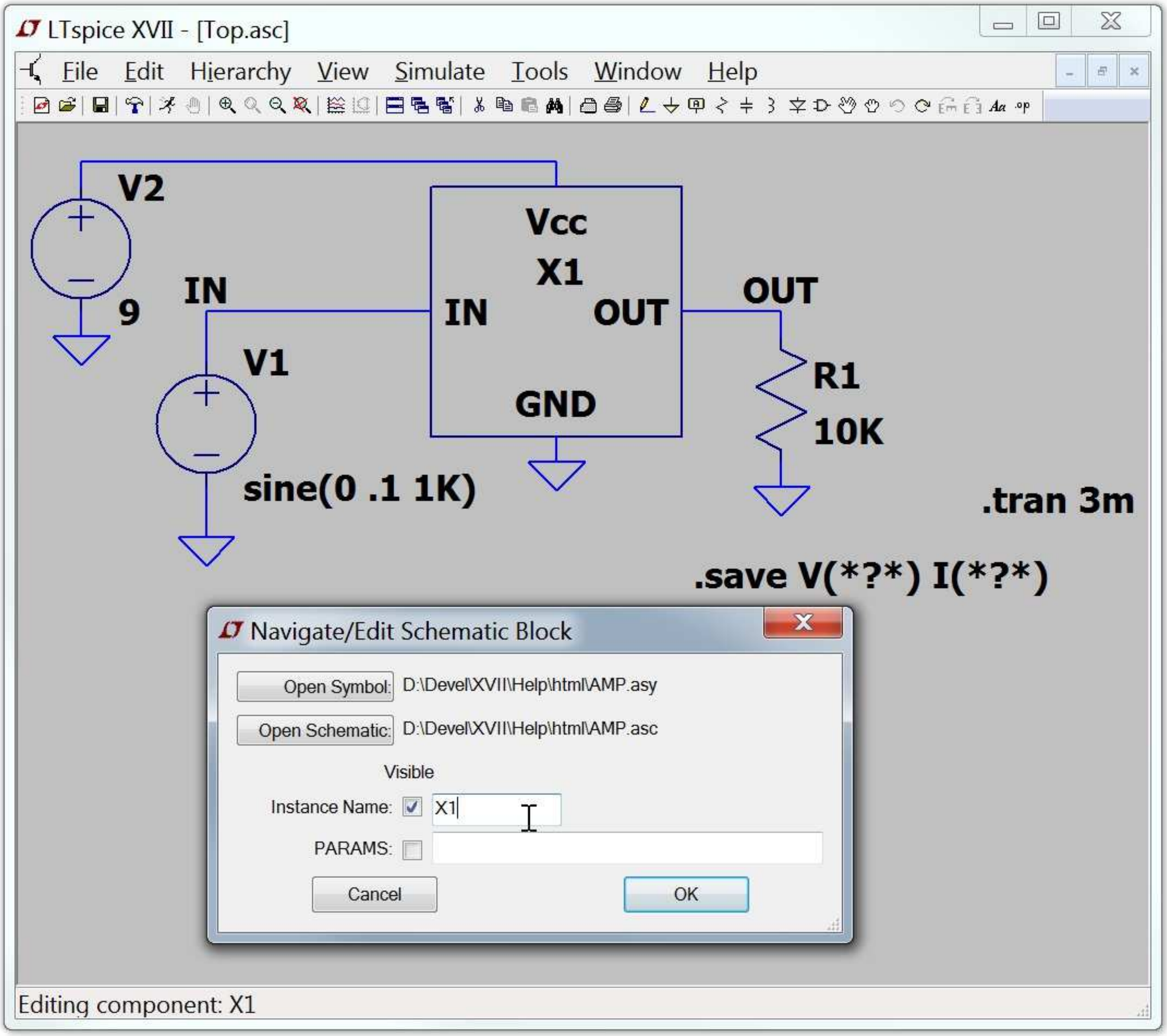
LTspice will look in the directory of the top-level schematic for symbols and blocks to complete the circuitry of the top-level schematic.

The symbol you create to represent the lower-level schematic block should have no attributes defined.

Navigating the Hierarchy

Any file opened with the File=>Open command is considered a top-level schematic. You can add SPICE directives to that block and run simulations using only it and any lower-level schematics to which it refers.

To open a schematic block as an instance of a block of a higher-level schematic, first open the higher-level schematic and then move the mouse to the body of the instance of the symbol calling the block. When you right mouse click on the body of the instance of that symbol, a special dialog appears that allows you to open the schematic. When you open the schematic in this manner, you can cross probe the nodes and current in the block. Note that you should have the options "Save Subcircuit Node Voltages" and "Save Subcircuit Device Currents" checked on the Save Defaults Pane of the Control Panel. Also, if you've highlighted a node on the top-level schematic, that node will be also highlighted in the lower level block.



Navigate/Edit Schematic Block

Open Symbol: D:\Devel\XVII\Help\html\AMP.asy

Open Schematic: D:\Devel\XVII\Help\html\AMP.asc

	Visible	
Instance Name:	☑	X1
PARAMS:	□	

Editing component: X1

Note that is dialog also allows you to enter parameters to pass to this instance of the circuitry in preamp.asc.

Waveform Viewer

LTspice XVII includes an integrated waveform viewer that allows complete control over the manner the simulation data is plotted.

Data Trace Selection

There are three basic means of selecting plotted traces.

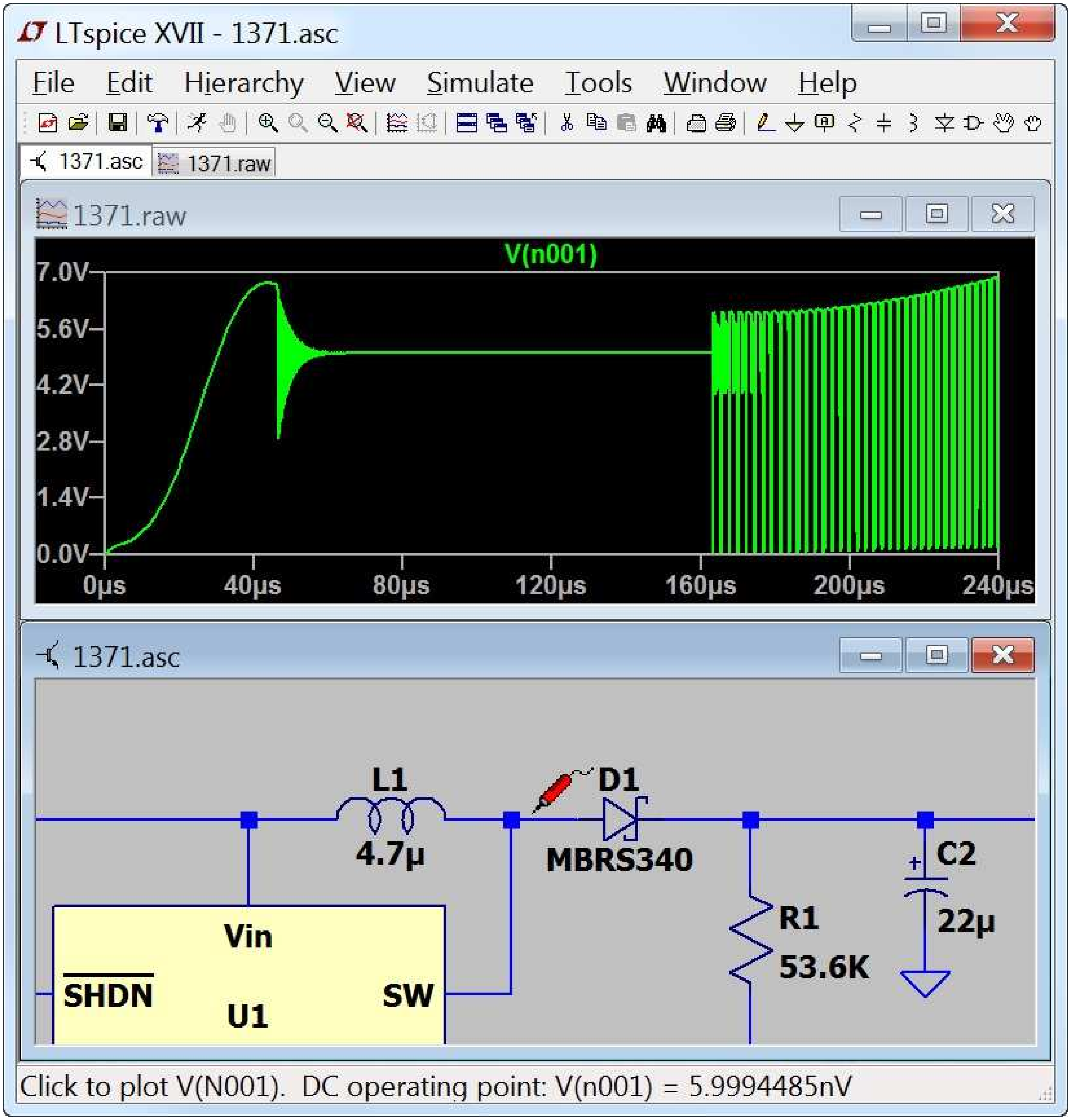
1. Probing directly from the schematic

2. Menu command Plot Settings=>Visible Traces

3. Menu command Plot Settings=>Add Trace

The undo and redo commands allow you to review the different trace selections no matter which method of selection is used.

1. Probing directly from the schematic:

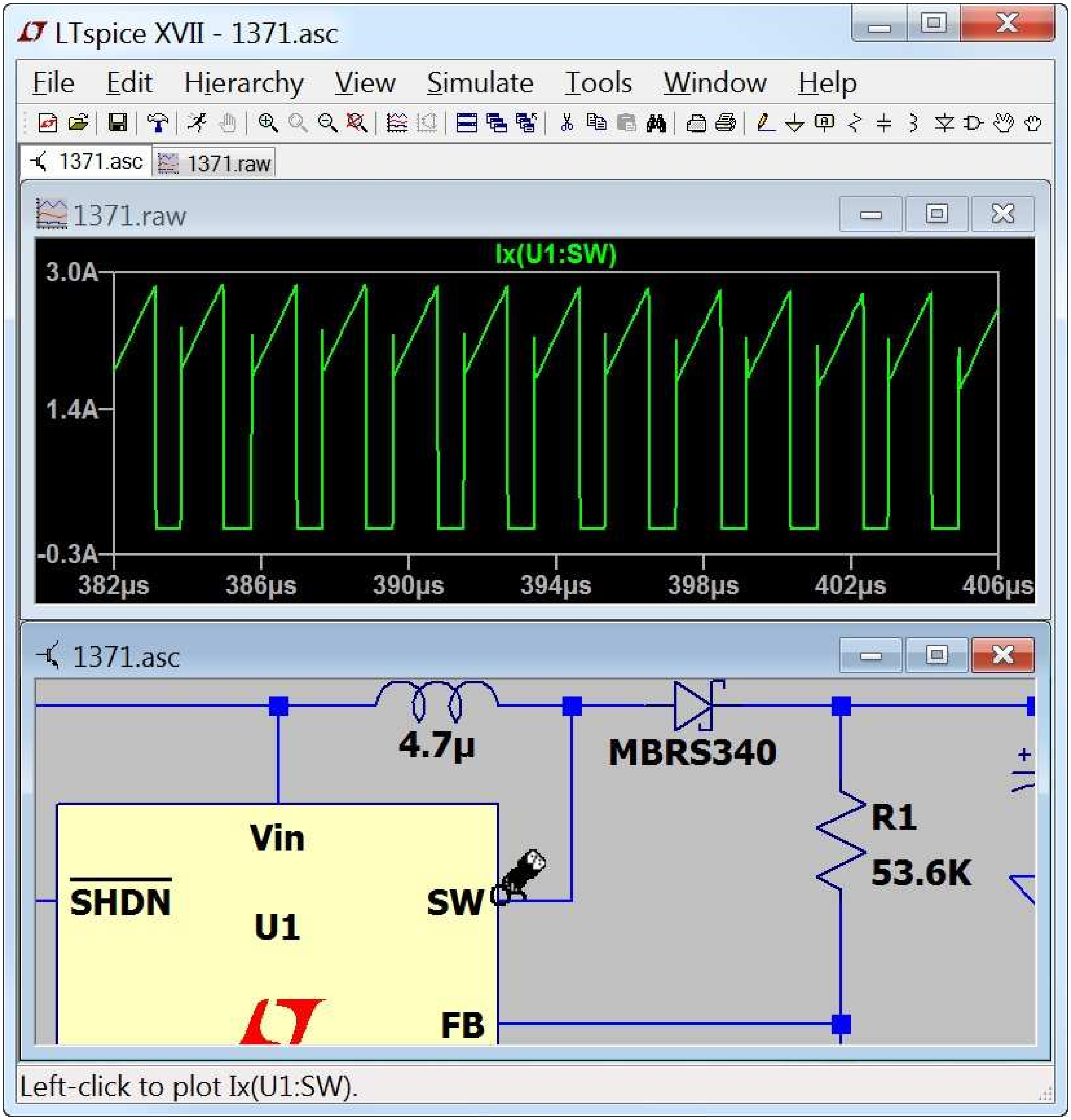


Click to plot V(N001). DC operating point: V(n001) = 5.9994485nV

The easiest method is to simply probe the schematic. You simply point and click at a wire to plot the voltage on that wire. You can plot the current through a component with two connections(like a resistor, capacitor or an inductor) by clicking on the body of the component. This works at any level of the circuit's hierarchy if you've saved all subcircuit voltages and currents. You can also plot current into a particular pin of a component by clicking on that pin of the symbol.

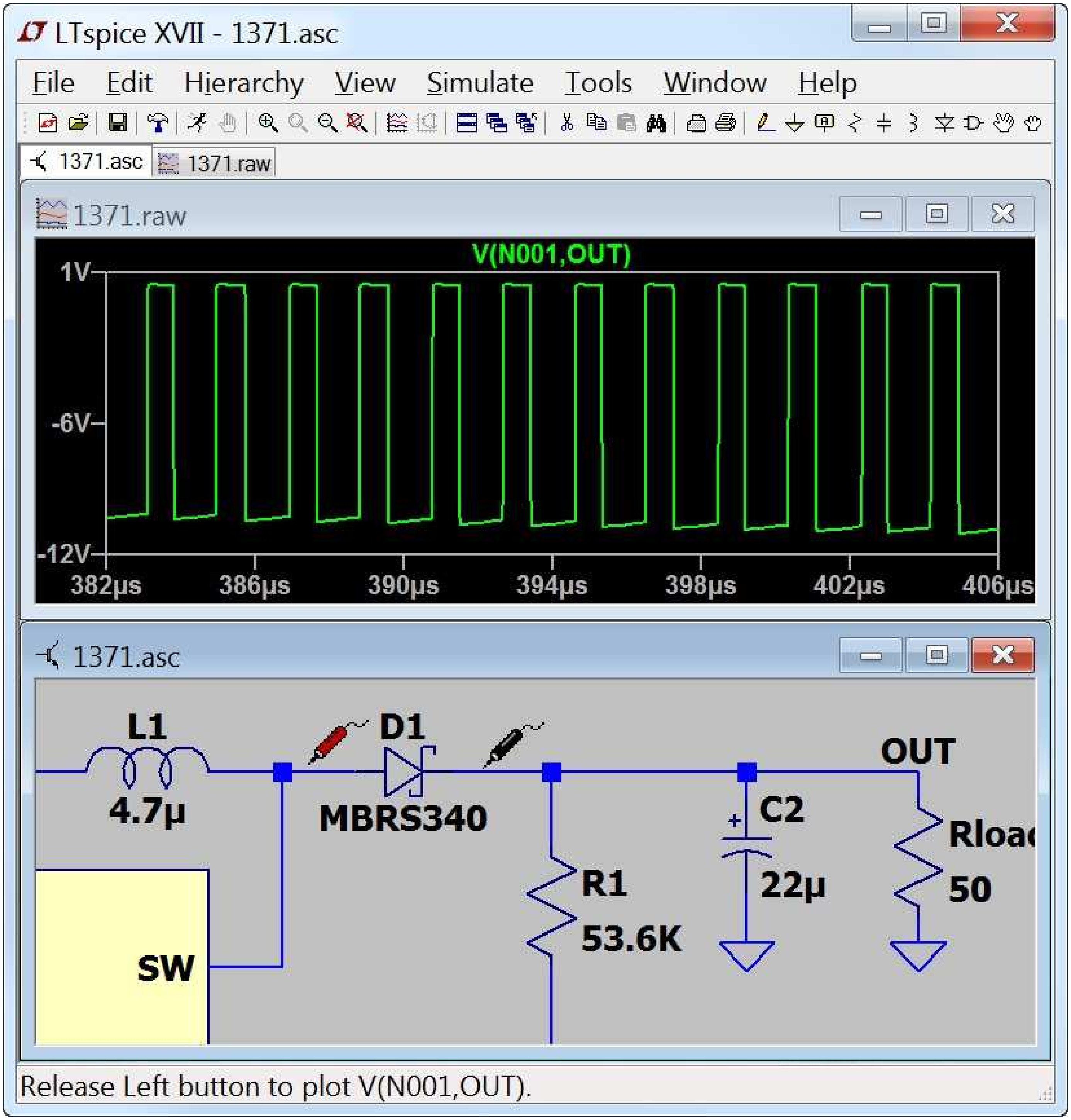
If you click the same voltage or current twice, then all other traces will be erased and the double clicked trace will be plotted by itself. You can delete individual traces by clicking on the trace's label after selecting the delete command.

The following screen shot shows how to point at a pin current. Notice that the mouse cursor turns into an icon that looks like a current clamp meter when it's pointing at a pin current. By convention, the current is reported positive if flow is into the pin.



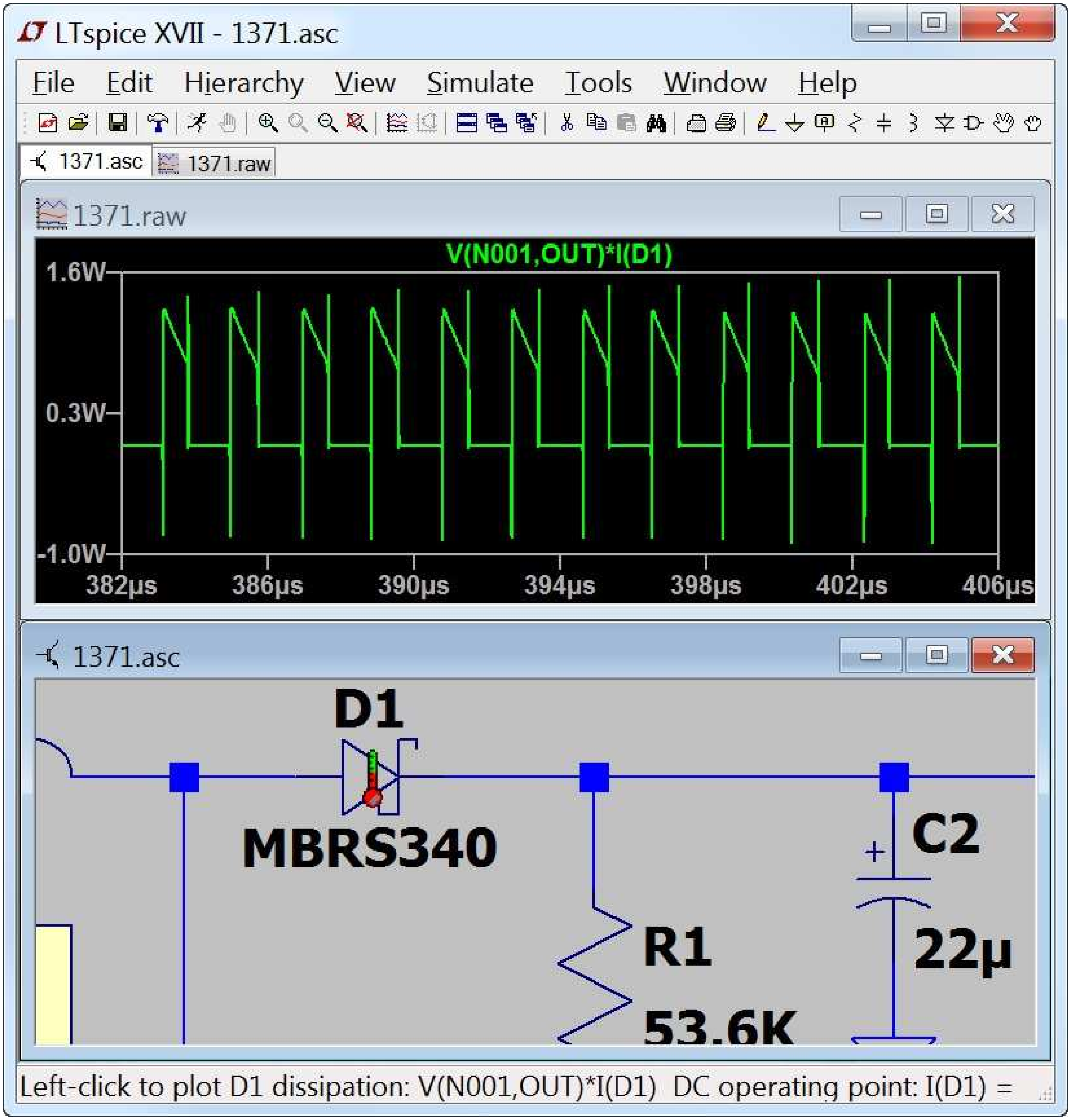
Left-click to plot Ix(U1:SW).

It is also possible to point at voltage differences with the mouse. You can click on one node and drag the mouse to another node. You will see the red voltage probe at the first node and a black probe on the second. This allows you to differentially plot voltages:



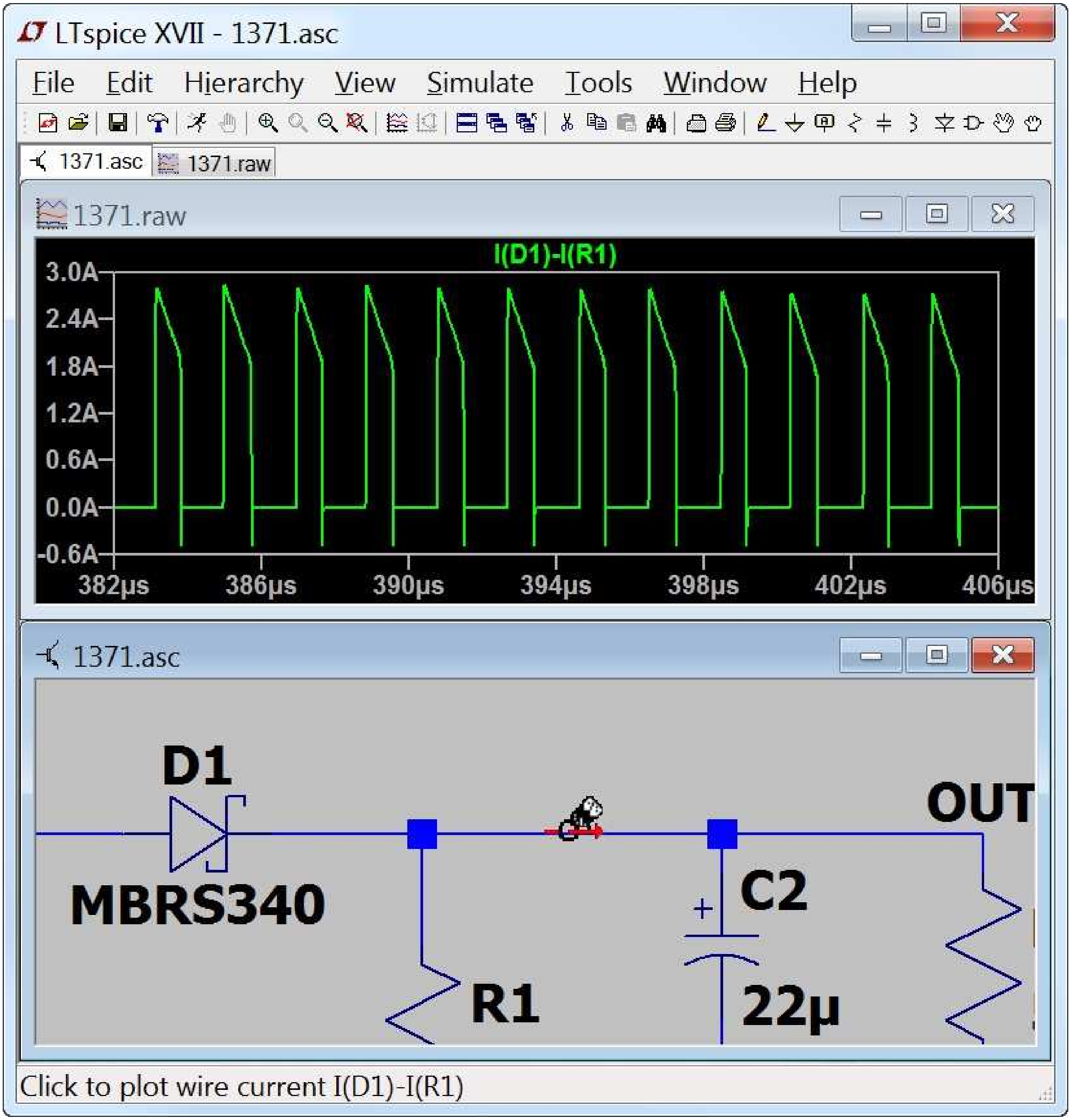
Release Left button to plot V(N001,OUT).

Yet another schematic probing technique is to plot the instantaneous power dissipation of a component. To do this, hold down the Alt key and click on the body of the symbol of the component. The instantaneous power dissipation will be plotted as an expression of voltages and currents. It will be plotted on its own scale with the units of Watts. The mouse cursor turns into an icon that looks like a thermometer when it's pointing at a dissipation that can be plotted. You can find the average power dissipation by control-clicking the trace label.



Left-click to plot D1 dissipation: V(N001,OUT)*I(D1) DC operating point: I(D1) =

You can also probe the current in a wire. To do this, hold down the Alt key and click on the wire. The mouse cursor turns into a current clamp meter to indicate it's pointing at this current and the red arrow shows the direction of positive current.

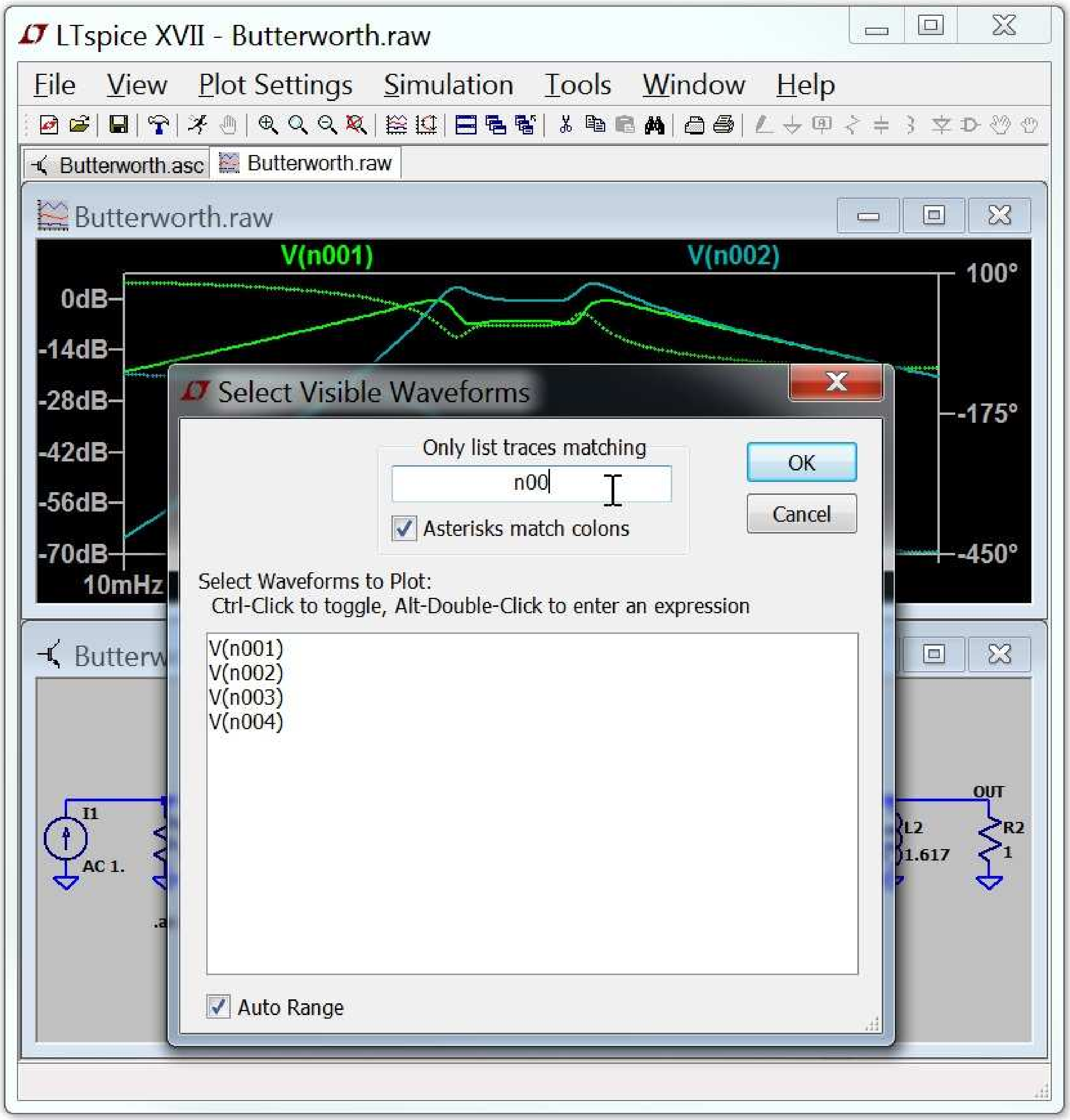


Click to plot wire current I(D1)-I(R1)

2. Menu command Plot Settings=>Visible Traces:

Notice that it is possible to list only those trace names that match a pattern. This is useful when there are many traces and you only know part of the name.

The menu command Plot Settings=>Visible Traces is the dialog seen at the beginning of plotting data from a simulation. It lets you select the initial traces to start the plot. It also gives you random access to the full list of traces plotted.



Select Visible Waveforms

Only list traces matching

n00

☑ Asterisks match colons

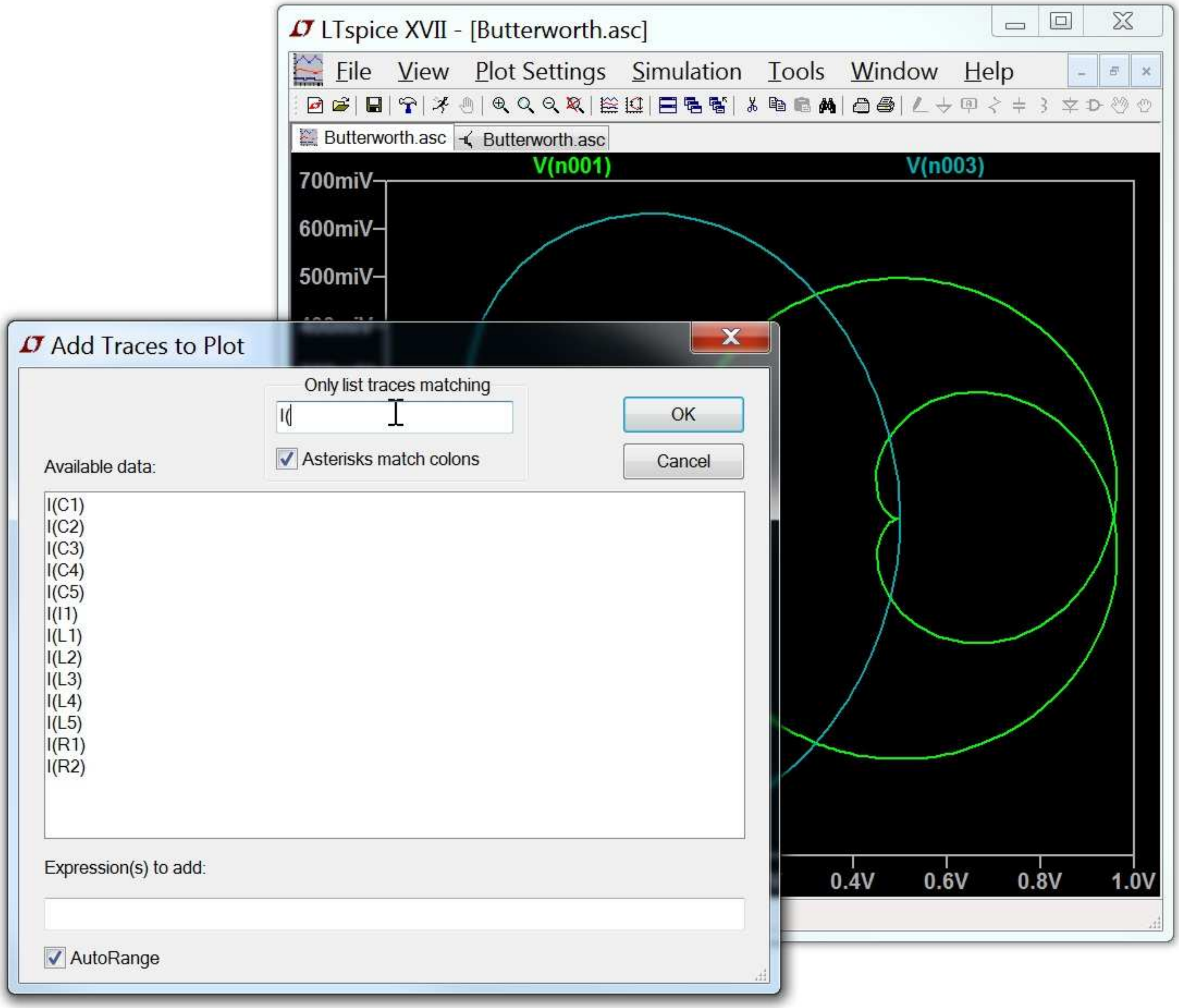
Select Waveforms to Plot:
Ctrl-Click to toggle, Alt-Double-Click to enter an expression

V(n001)
V(n002)
V(n003)
V(n004)

☑ Auto Range

3. Menu command View=>Add Trace:

The Plot Settings=>Add Trace command is similar to the Plot Settings=>Visible Traces command. However, you can not delete traces that are already visible with it. It has two useful capabilities. One is an edit box near the top of the dialog that allows you to enter a pattern of characters. Only trace names that match the pattern will be shown in the dialog. This is very useful for finding a trace when you can only partially remember the name. Also, it's a bit easier to compose an expression of trace data because you can click on a name in the dialog instead of typing out its name.



Only list traces matching: I(

☑ Asterisks match colons

Available data:

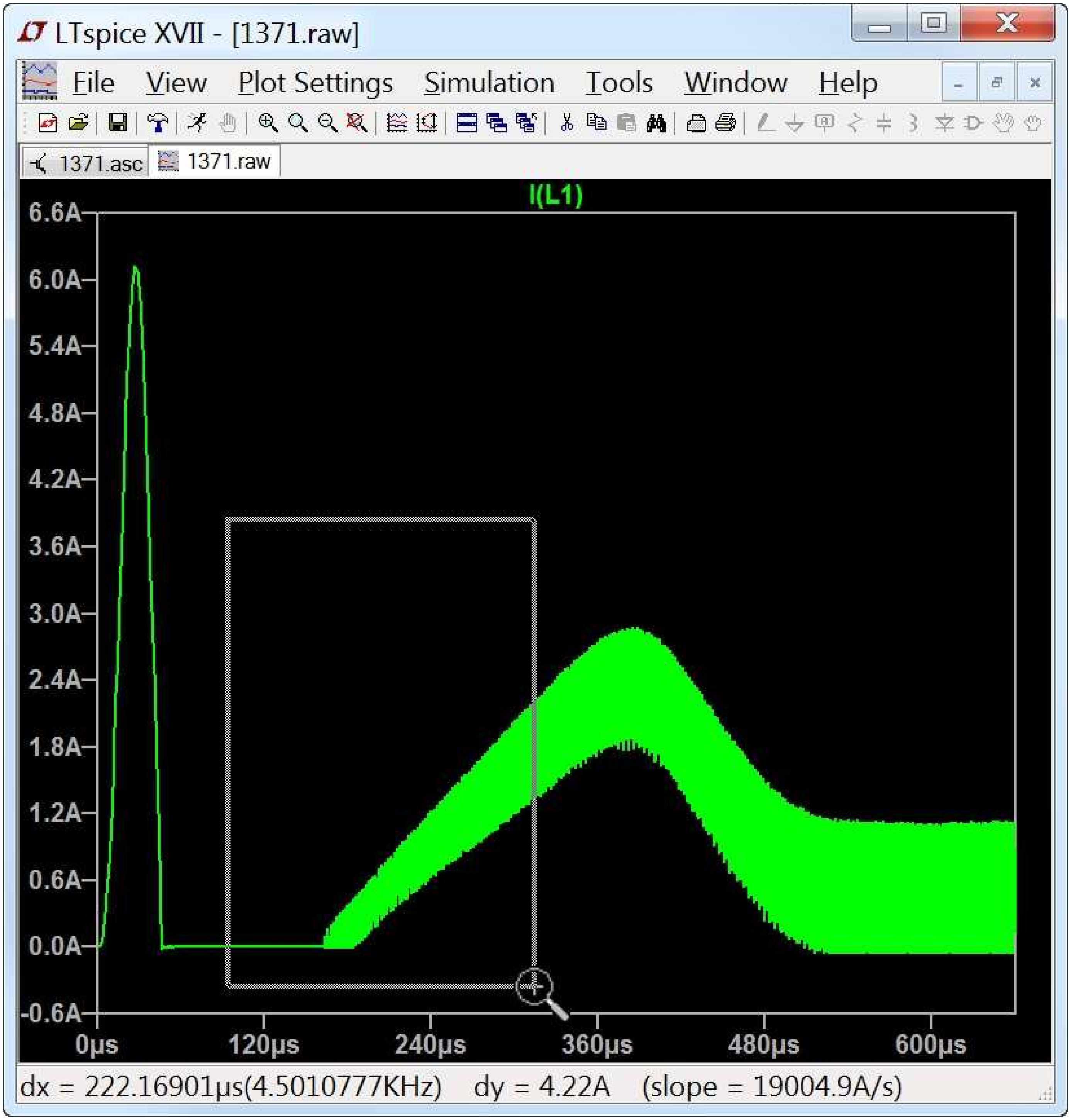
I(C1)
I(C2)
I(C3)
I(C4)
I(C5)
I(I1)
I(L1)
I(L2)
I(L3)
I(L4)
I(L5)
I(R1)
I(R2)

Expression(s) to add:

☑ AutoRange

Zooming

LTspice XVII autozooms whenever there is new data to plot. To zoom up on an area, simply drag a box about the region you wish to see drawn larger.

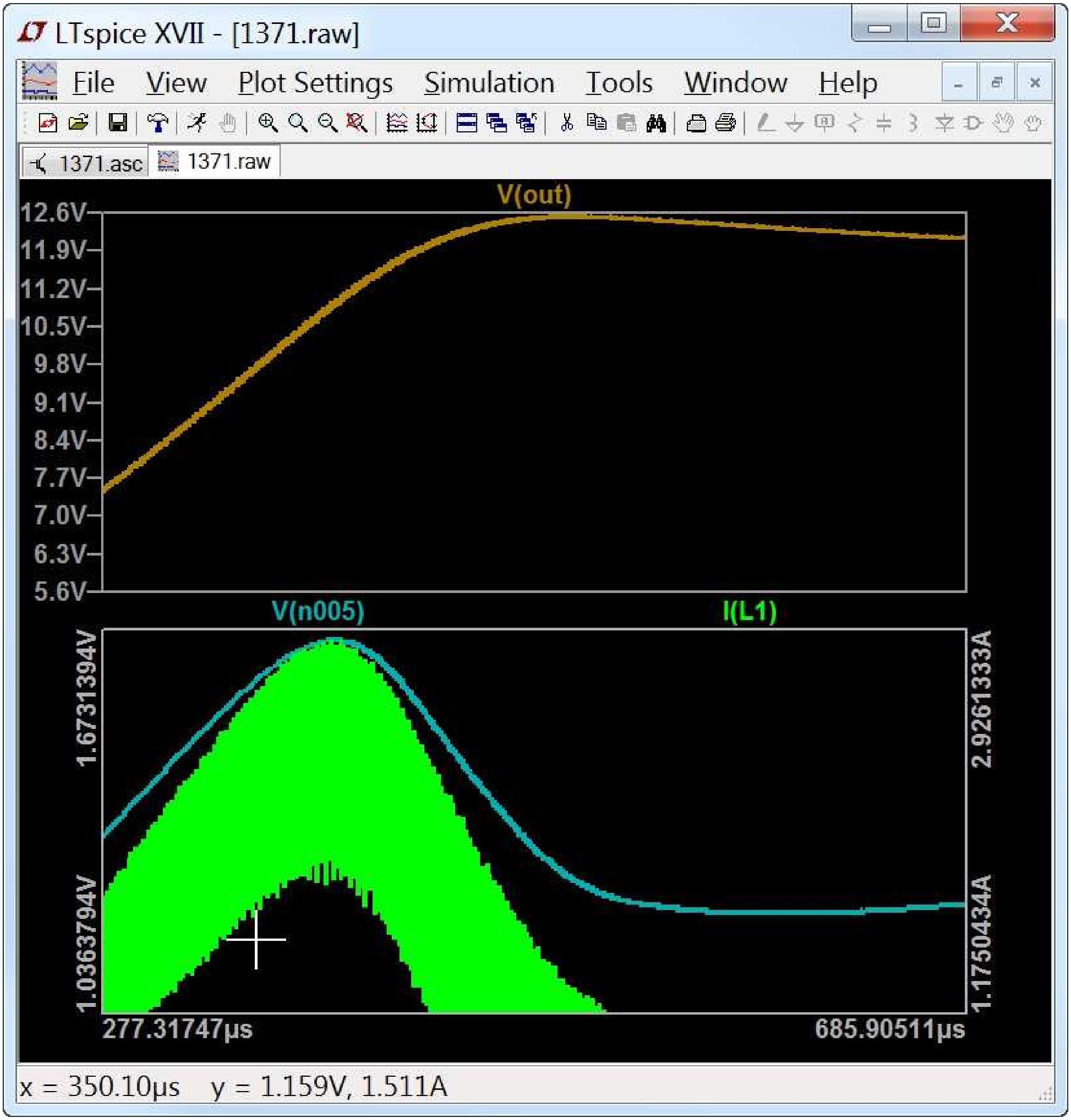


dx = 222.16901µs(4.5010777KHz) dy = 4.22A (slope = 19004.9A/s)

Note that the size of the zoom box is displayed on the status bar at the bottom so that you can quickly measure differences without setting up attached cursors.

There are toolbar buttons and menu commands for zooming out, panning, and returning to the autoranged zoom. The undo and redo commands allow you to review the different zooms used.

Another zoom mode is to hold down the control key while moving the mouse or turning the mouse wheel. The software will zoom and pan a bitmap of the current plot to give an indication of what the plot would like like when fully rendered. This mode is intended for huge waveform files that take seconds to redraw.



x = 350.10µs y = 1.159V, 1.511A

Waveform Arithmetic

There are three types of mathematical operations that can be performed on waveform data:

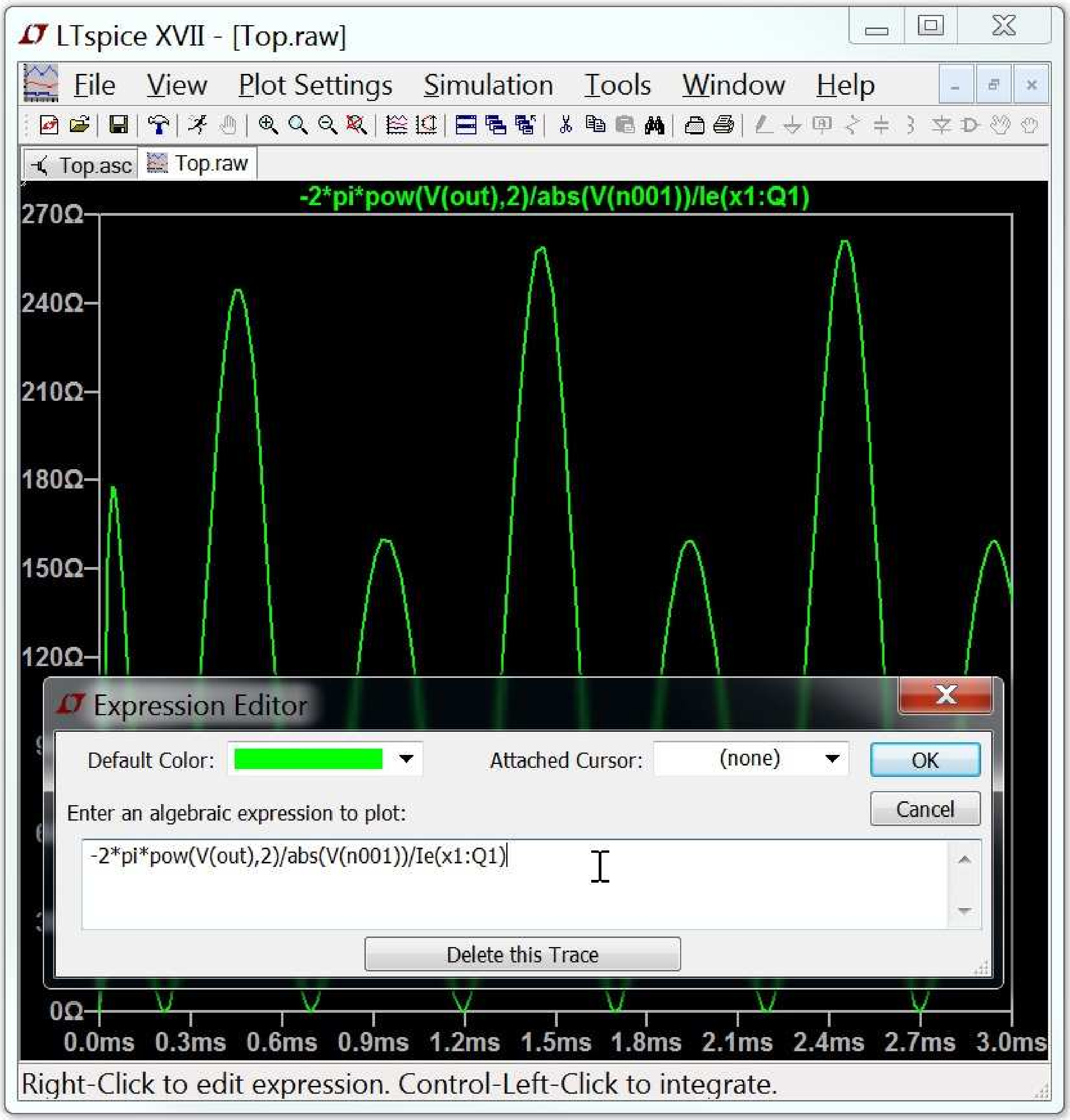
1. Plot expressions of traces.

2. Compute the average or RMS of a trace.

3. Display the Fourier Transform of a trace.

1. Plot expressions of traces.

Both the View=>Visible Traces and View=>Add Trace commands allow one to enter an expression of data. Another method to plot an expression of available simulation data traces is to move the mouse to the trace's label and right click. This dialog box also allows you to set the trace's color and allows you to attach a cursor to the waveform. LTspice will do a dimensional analysis of the expression and plot it against a vertical axis labeled with those units. For example, below you can see that LTspice identified the dimensions of -2*π*pow(V(out),2)/abs(V(n001))/Ie(x1:Q1) as Ω. All waveforms in a plotting pane with the same units are plotted on the same axis.



LTspice XVII - [Top.raw]
File View Plot Settings Simulation Tools Window Help
Top.asc
Top.raw
-2*pi*pow(V(out),2)/abs(V(n001))/Ie(x1:Q1)
270Ω
240Ω
210Ω
180Ω
150Ω
120Ω
0Ω
0.0ms 0.3ms 0.6ms 0.9ms 1.2ms 1.5ms 1.8ms 2.1ms 2.4ms 2.7ms 3.0ms
Expression Editor
OK
Cancel
Delete this Trace

Default Color: 🟩

Attached Cursor: (none)

Enter an algebraic expression to plot:

```
-2*pi*pow(V(out),2)/abs(V(n001))/Ie(x1:Q1)
```

Right-Click to edit expression. Control-Left-Click to integrate.

The difference of two voltages; e.g., V(a)-V(b); can written equivalently as V(a,b). The following functions are available for real data:

Function Name	Description
abs(x)	Absolute value of x

acos(x)	Arc cosine of x
arccos(x)	Synonym for acos()
acosh(x)	Arc hyperbolic cosine
asin(x)	Arc sine of x
arcsin(x)	Synonym for sin()
asinh(x)	Arc hyperbolic sine
atan(x)	Arc tangent of x
arctan(x)	Synonym for atan()
atan2(y,x)	Four quadrant arc tangent of y/x
atanh(x)	Arc hyperbolic tangent
buf(x)	1 if x > .5, else 0
ceil(x)	Integer equal or greater than x
cos(x)	Cosine of x
cosh(x)	Hyperbolic cosine of x
d()	Finite difference-based derivative
exp(x)	e to the x
floor(x)	Integer equal to or less than x
hypot(x,y)	sqrt(x**2 + y**2)
if(x,y,z)	If x > .5, then y else z
int(x)	Convert x to integer
inv(x)	0. if x > .5, else 1.
limit(x,y,z)	Intermediate value of x, y, and z
ln(x)	Natural logarithm of x
log(x)	Alternate syntax for ln()
log10(x)	Base 10 logarithm
max(x,y)	The greater of x or y
min(x,y)	The smaller of x or y
pow(x,y)	x**y
pwr(x,y)	abs(x)**y
pwrs(x,y)	sgn(x)*abs(x)**y
rand(x)	Random number between 0 and 1 depending on the integer value of x.
	Similar to rand(), but

random(x)	smoothly transitions between values.
round(x)	Nearest integer to x
sgn(x)	Sign of x
sin(x)	Sine of x
sinh(x)	Hyperbolic sine of x
sqrt(x)	Square root of x
table(x,a,b,c,d,...)	Interpolate a value for x based on a look up table given as a set of pairs of points.
tan(x)	Tangent of x.
tanh(x)	Hyperbolic tangent of x
u(x)	Unit step, i.e., 1 if x > 0., else 0.
uramp(x)	x if x > 0., else 0.
white(x)	Random number between -.5 and .5 smoothly transitions between values even more smoothly than random().

For complex data, the functions atan2(,), sgn(), u(), buf(), inv() uramp(), int(), floor(), ceil(), rand(), min(,), limit(,), if(,,), and table(...) are not available. The functions Re(x) and Im(x) are available for complex data and return a complex number with the real part equal to the real or imaginary part of the argument respectively and the imaginary part equal to zero. The functions Ph(x) and Mag(x) are also available for complex data and return a complex number with the real part equal to the phase angle or magnitude of the argument respectively and the imaginary part equal to zero. The function conj(x) is also available for complex data and returns the complex conjugate of x.

The following operations, grouped in reverse order of precedence of evaluation, are available for real data:

Operand	**Description**
&	Convert the expressions to either side to Boolean, then AND.
|	Convert the expressions to either side to Boolean, then OR.
^	Convert the expressions to either side

	to Boolean, then XOR.
>	TRUE if expression on the left is greater than the expression on the right, otherwise FALSE.
<	TRUE if expression on the left is less than the expression on the right, otherwise FALSE.
>=	TRUE if expression on the left is greater than or equal the expression on the right, otherwise FALSE.
<=	TRUE if expression on the left is less than or equal the expression on the right, otherwise FALSE.
+	Addition
-	Subtraction
*	Multiplication
/	Division
**	Raise left hand side to power of right hand side.
!	Convert the following expression to Boolean and invert.
@	Step selection operator

TRUE is numerically equal to 1 and FALSE is 0. Conversion to Boolean converts a value to 1 if the value is greater than 0.5, otherwise the value is converted to 0.

The step selection operator, '@' is useful when multiple simulation runs are available as in a .step, .temp, or .dc analysis. It selects the data from a specific run. For example, V(1)@3 would plot the data from the 3^{rd} run no matter what steps where selected for plotting.

For complex data, only +, -, *, /, **, and @ are available. Also with regard to complex data, the Boolean XOR operator, ^ is understood to mean exponentiation, **.

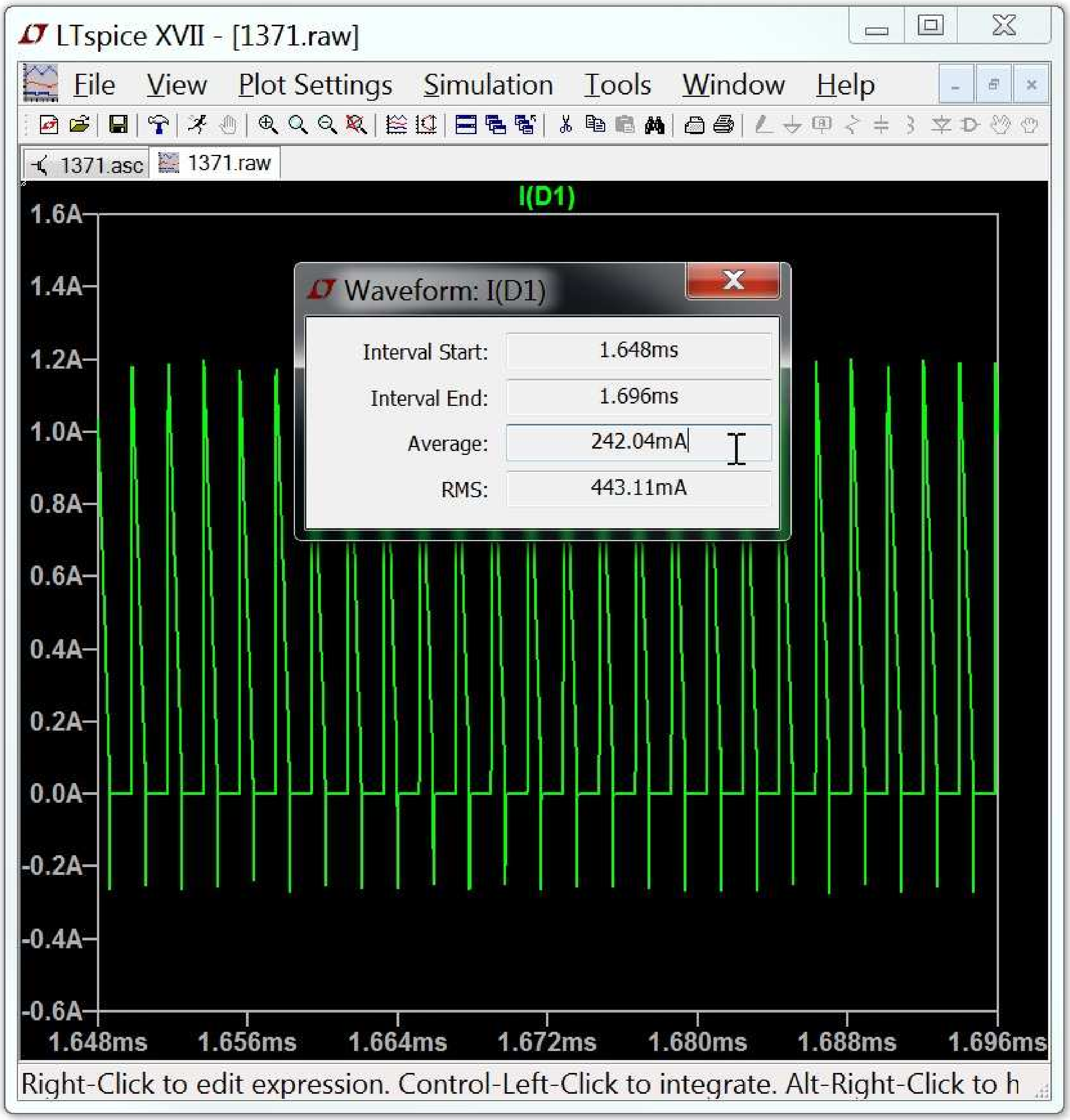
The following constants are internally defined:

Name	Value
E	2.7182818284590452354
pi	3.14159265358979323846
K	1.3806503e-23
Q	1.602176462e-19

The keyword "time" is understood when plotting transient analysis waveform data. Similarly, "freq" and "omega" are understood when plotting data from an AC analysis. "w" can be used as a synonym for omega.

2. Compute the average or RMS of a trace.

The waveform viewer can integrate a trace to obtain the average and RMS value over the displayed region. First zoom the waveform to the region of interest, then move the mouse to the label of the trace, hold down the control key and left mouse click.



LTspice XVII - [1371.raw]

Waveform: I(D1)

Interval Start: 1.648ms

Interval End: 1.696ms

Average: 242.04mA

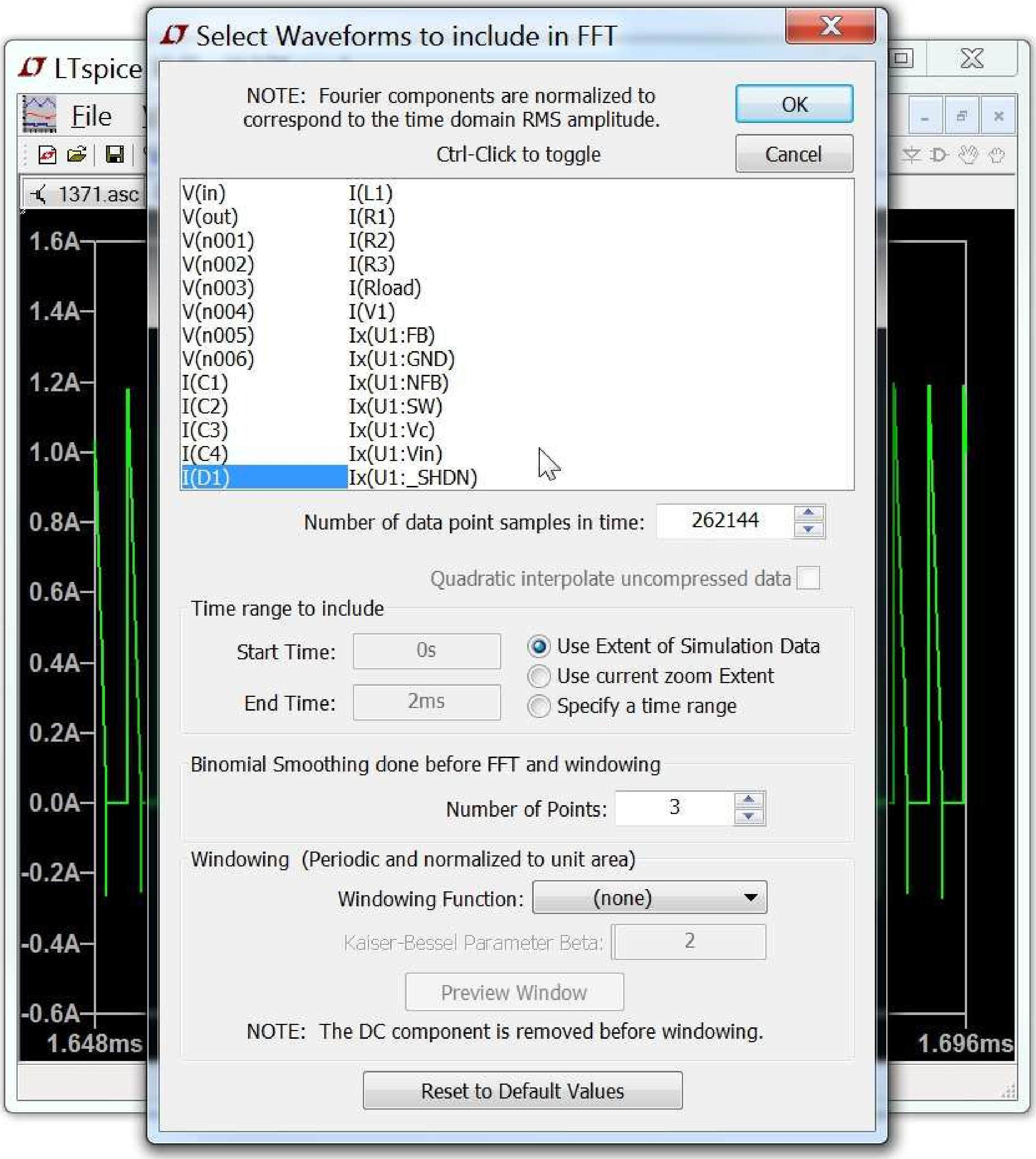
RMS: 443.11mA

Right-Click to edit expression. Control-Left-Click to integrate. Alt-Right-Click to h

Note RMS average is reported only if the physical units of the integrated quantity is volts or amps to avoid confusing people that need the average when power is integrated.

3. Display the Fourier Transform of a Trace.

You can use the menu command View=>FFT to perform a Fast Fourier transform on various data traces.



Select Waveforms to include in FFT

NOTE: Fourier components are normalized to correspond to the time domain RMS amplitude.

Ctrl-Click to toggle

V(in)
V(out)
V(n001)
V(n002)
V(n003)
V(n004)
V(n005)
V(n006)
I(C1)
I(C2)
I(C3)
I(C4)
I(D1)
I(L1)
I(R1)
I(R2)
I(R3)
I(Rload)
I(V1)
Ix(U1:FB)
Ix(U1:GND)
Ix(U1:NFB)
Ix(U1:SW)
Ix(U1:Vc)
Ix(U1:Vin)
Ix(U1:_SHDN)

Number of data point samples in time: 262144

Quadratic interpolate uncompressed data □

Time range to include

Start Time: 0s

End Time: 2ms

● Use Extent of Simulation Data
○ Use current zoom Extent
○ Specify a time range

Binomial Smoothing done before FFT and windowing

Number of Points: 3

Windowing (Periodic and normalized to unit area)

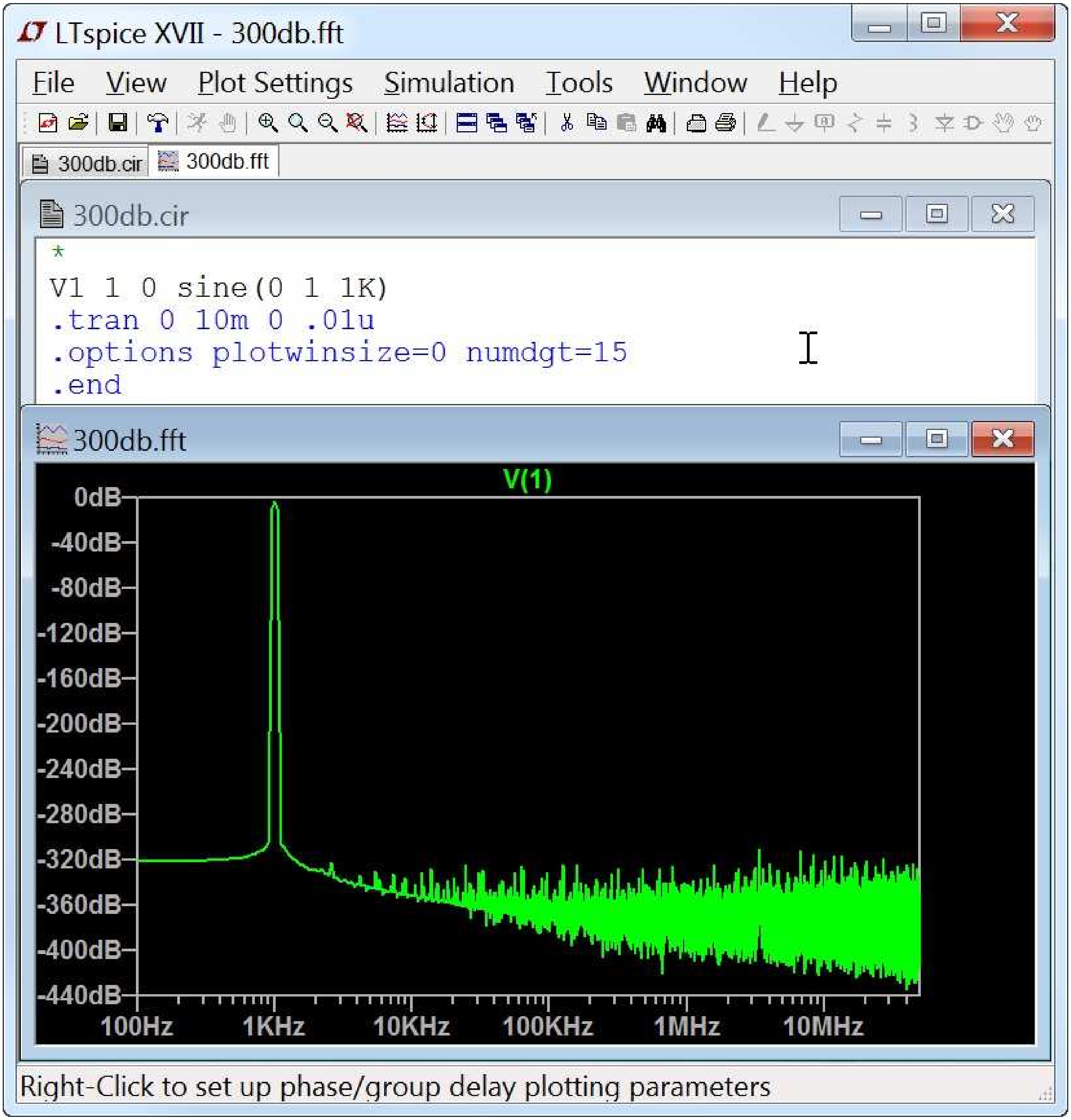
Windowing Function: (none)

Kaiser-Bessel Parameter Beta: 2

NOTE: The DC component is removed before windowing.

LTspice uses a proprietary FFT algorithm that allows an arbitrary number of datapoints, i.e., not limited to a power of 2.

When you expect to do FFT's of your simulation data, you will probably want to turn off waveform compression, stipulate a maximum time step, and possibly even use double precision waveform file format to reduce the numeric noise floor. The following netlist shows that the intrinsic noise floor of LTspice's FFT algorithm is in excess of 300dB.



```
*
V1 1 0 sine(0 1 1K)
.tran 0 10m 0 .01u
.options plotwinsize=0 numdgt=15
.end
```

Right-Click to set up phase/group delay plotting parameters

User-Defined Functions

The menu command Plot Settings=>Edit Plot Defs File allows you to enter your own function definitions and parameter definitions for use in the waveform viewer. These functions are kept in the file %HOMEPATH%\Documents\LTspiceXVII\plot.defs.

Then the syntax is the same as the .param and .func statements used for parameterized circuits. E.g., the line

```
   .func Pythag(x,y) {sqrt(x*x+y*y)}
```

defines the function Pythag() to be the square root of the sum of its two arguments.

Similarly, the line

```
   .param twopi = 2*pi
```

defines twopi to be 6.28318530717959. Note that it uses the internally defined constant pi of the waveform viewer.

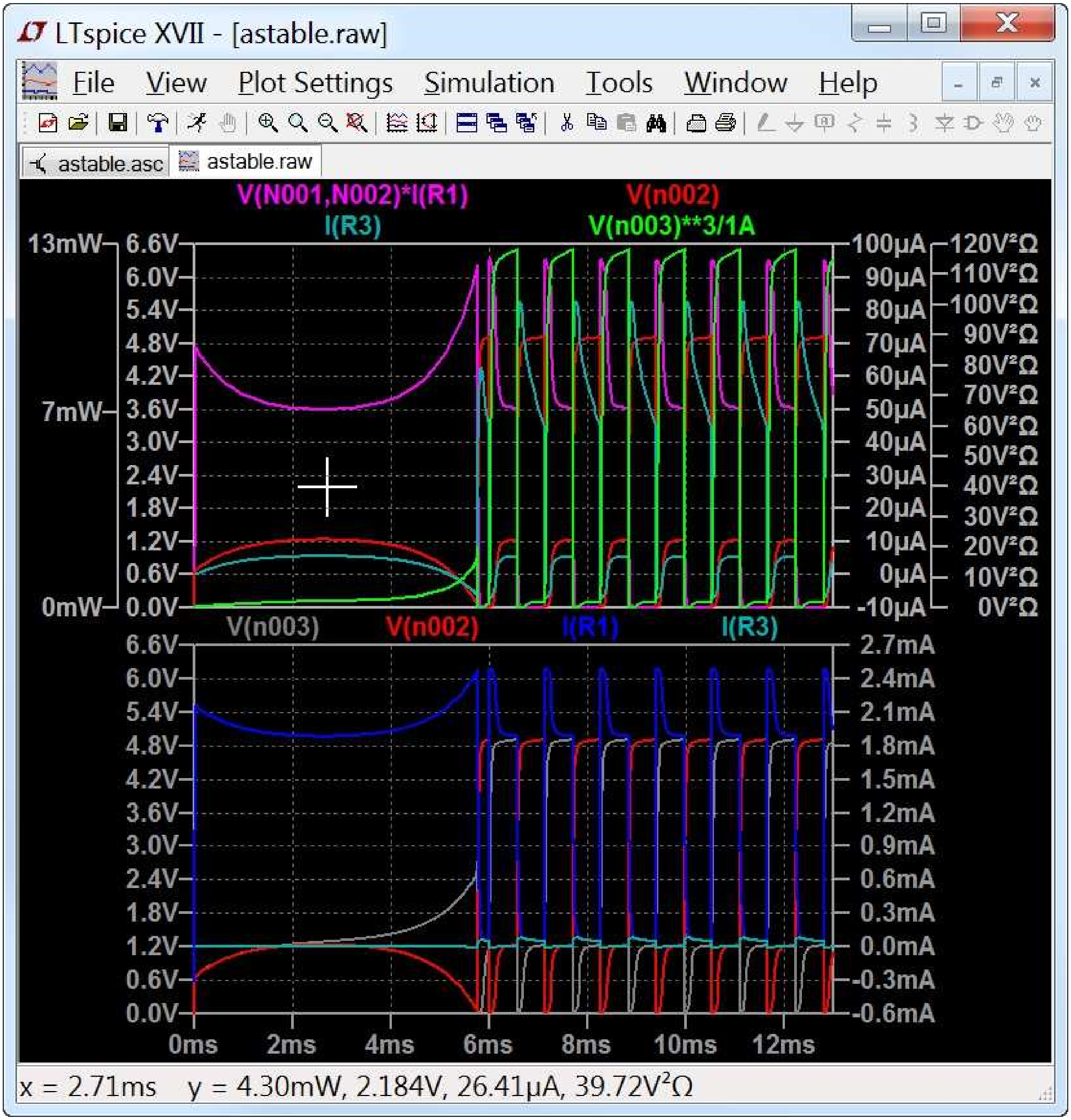
Axis Control

When you move the mouse cursor beyond the data plotting region, the cursor turns into a ruler. This indicates that you are pointing at that axis' attributes. When you left click you can enter a dialog to manually enter that axis' range and the nature of the plot. For example, for real data, if you move the mouse to the bottom of the screen and left click, you can enter a dialog to change the horizontal quantity plotted. This lets you make parametric plots.

For complex data, you can choose to plot either phase, group delay, or nothing against the right vertical axis. You can change the representation of complex data from Bode to Nyquist or Cartesian by moving the mouse to the left vertical axis of complex data.

Plot Panes

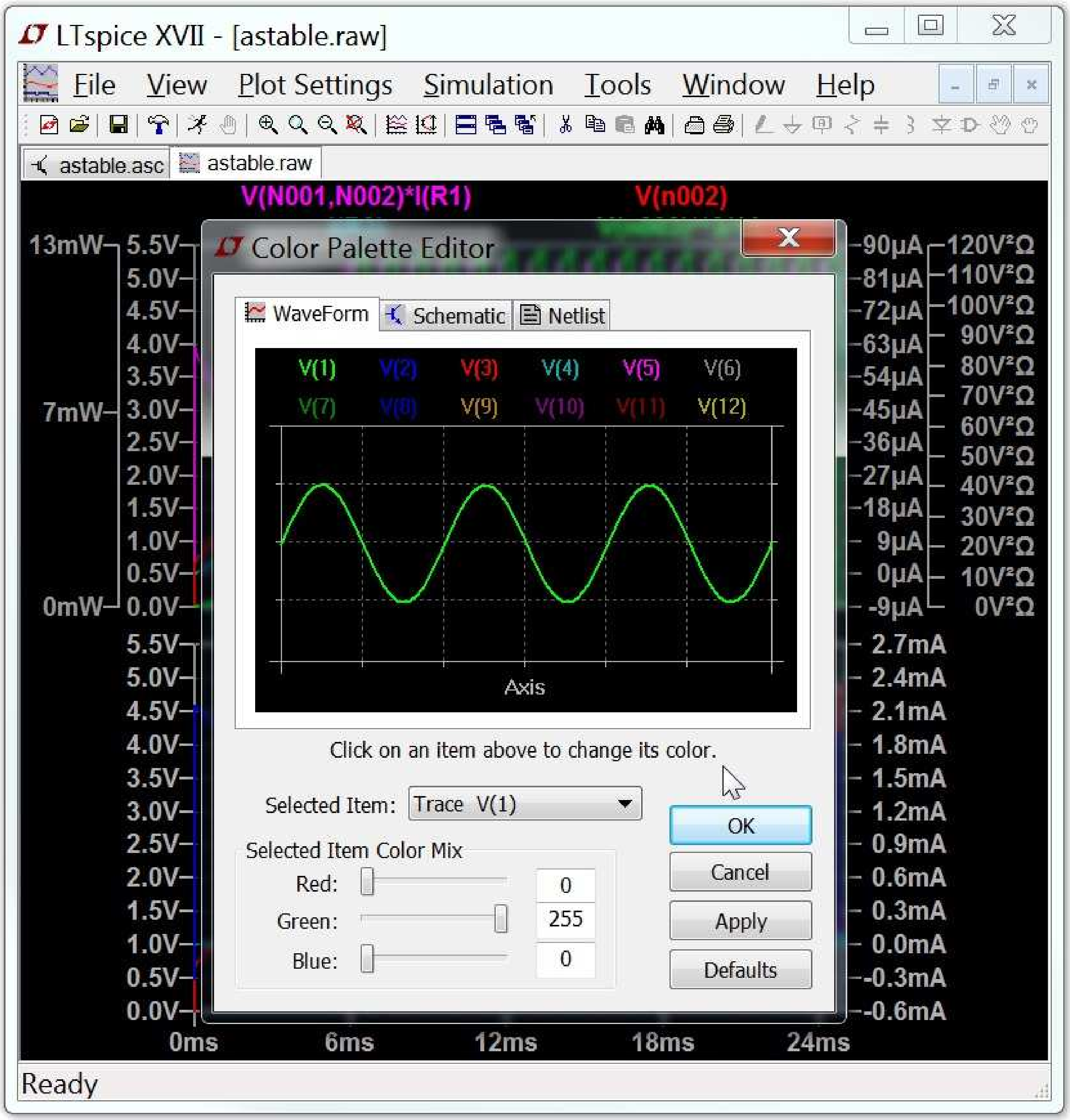
Multiple plot panes can be displayed on one window. This allows better separation between traces and allows different traces to be independently autoscaled. Traces can be dragged between panes by dragging the label. A copy of a trace can be made on another pane by holding down the control key when you release the mouse button.



x = 2.71ms   y = 4.30mW, 2.184V, 26.41µA, 39.72V²Ω

Color Control

The menu command Tools=>Color Preferences colors allows you to set the colors used for plotting data. You click on an object in the sample plot and use the red, green and blue sliders to adjust the colors to your preferences.



LTspice XVII - [astable.raw]
File View Plot Settings Simulation Tools Window Help
astable.asc
astable.raw
V(N001,N002)*I(R1)
V(n002)
Color Palette Editor
WaveForm
Schematic
Netlist
V(1) V(2) V(3) V(4) V(5) V(6)
V(7) V(8) V(9) V(10) V(11) V(12)
Axis
OK
Cancel
Apply
Defaults
13mW 7mW 0mW
5.5V 5.0V 4.5V 4.0V 3.5V 3.0V 2.5V 2.0V 1.5V 1.0V 0.5V 0.0V
90µA 81µA 72µA 63µA 54µA 45µA 36µA 27µA 18µA 9µA 0µA -9µA
120V²Ω 110V²Ω 100V²Ω 90V²Ω 80V²Ω 70V²Ω 60V²Ω 50V²Ω 40V²Ω 30V²Ω 20V²Ω 10V²Ω 0V²Ω
2.7mA 2.4mA 2.1mA 1.8mA 1.5mA 1.2mA 0.9mA 0.6mA 0.3mA 0.0mA -0.3mA -0.6mA
0ms 6ms 12ms 18ms 24ms
Ready

Click on an item above to change its color.

Selected Item: Trace V(1)

Selected Item Color Mix

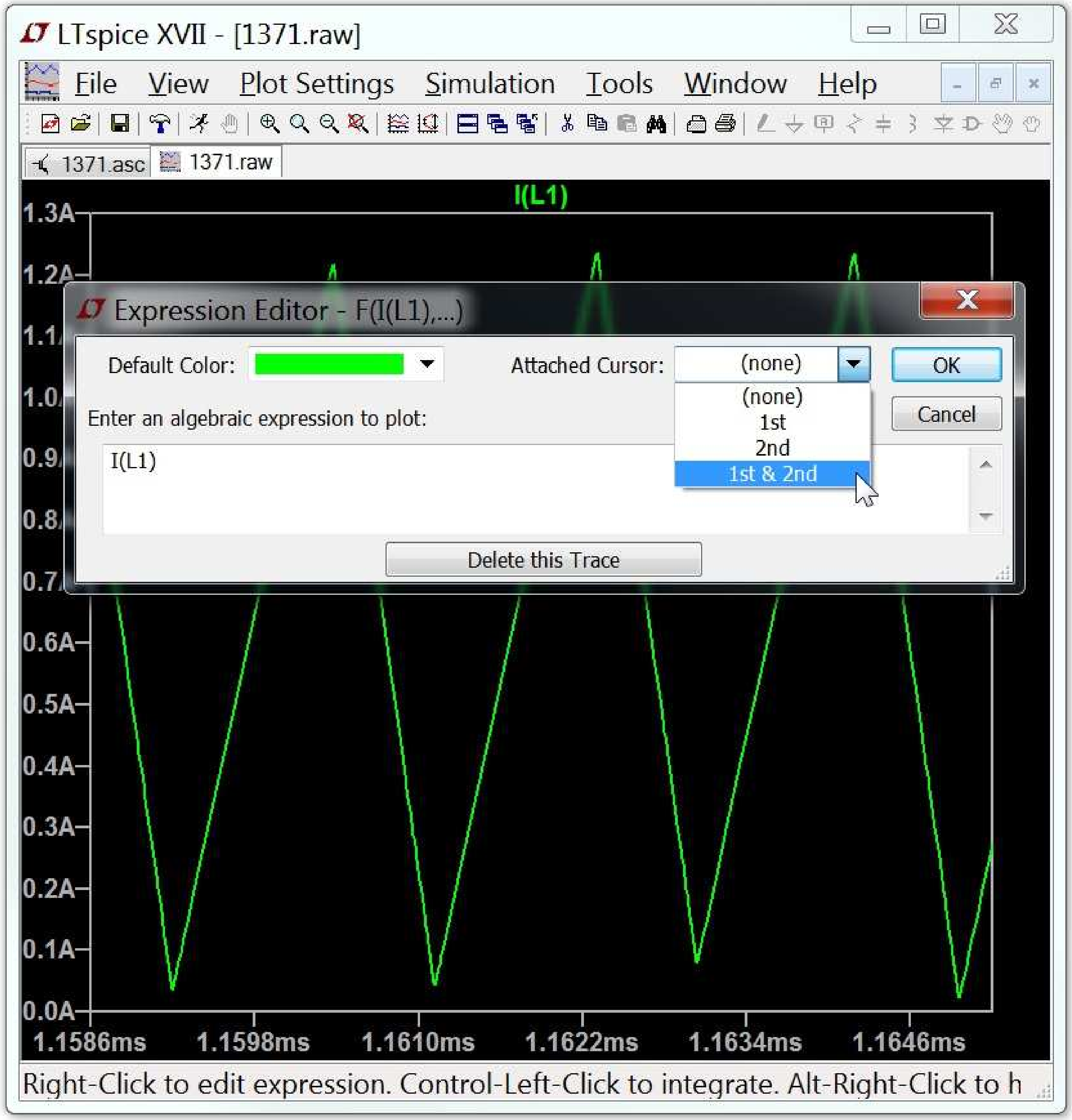
Red: 0

Green: 255

Blue: 0

Attached Cursors

There are up to two attached cursors per plotting pane available. You can attach a cursor to a trace by left mouse clicking on the trace label. You can attach both cursors to a single trace by right clicking on the trace label and selecting "1st & 2nd". You can also attach the 1st or 2nd cursor or both cursors to any trace by right clicking on that trace's label and using the Attached Cursor drop down box. The attached cursors can be dragged about with the mouse or moved with the cursor keys.



Default Color: 🟩

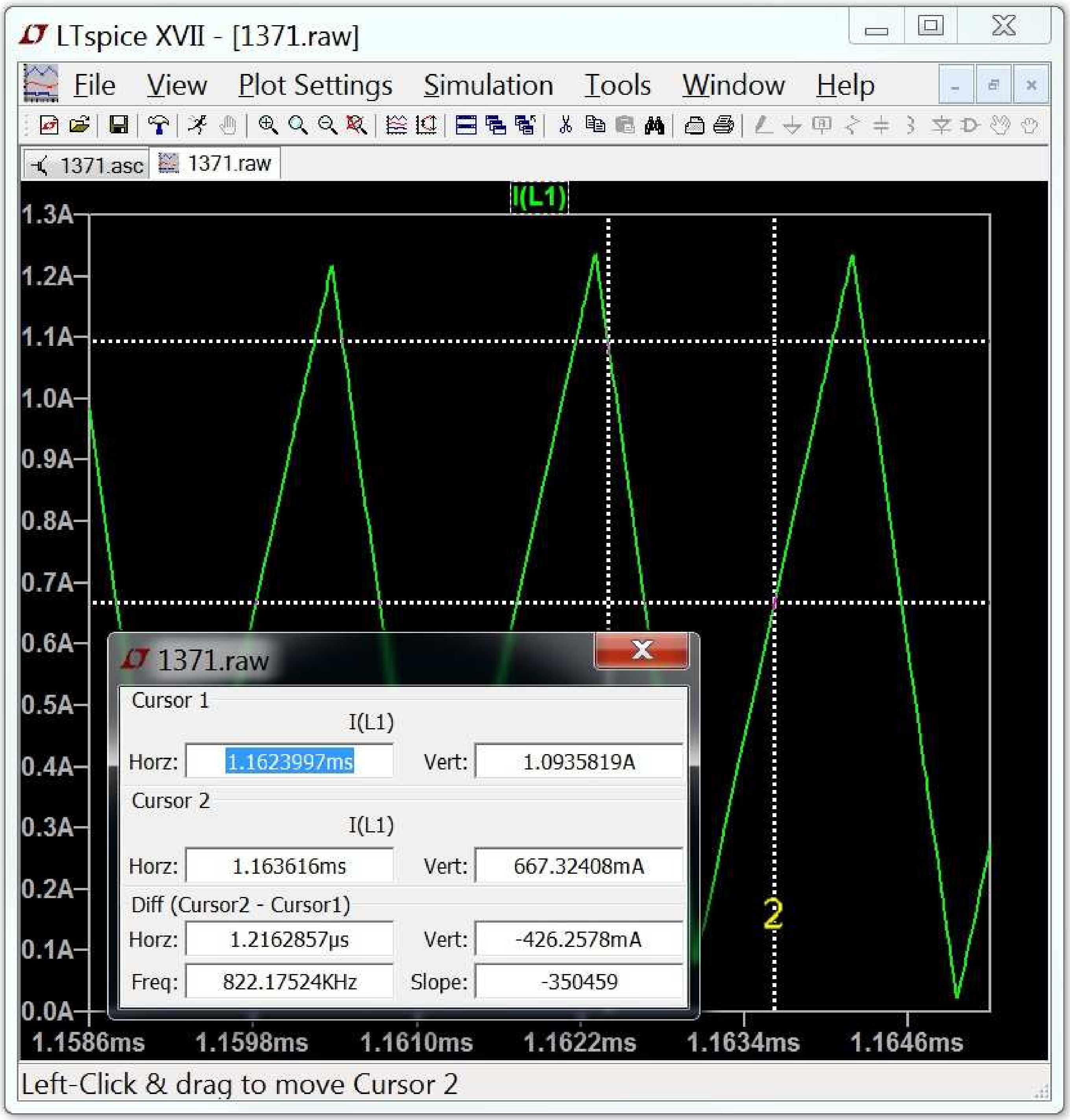
Attached Cursor: (none)

Enter an algebraic expression to plot:

I(L1)

Right-Click to edit expression. Control-Left-Click to integrate. Alt-Right-Click to h

When there are active attached cursors, a readout display becomes visible that will readout the data at the cursors and report the difference.



Cursor 1

I(L1)

Horz: 1.1623997ms Vert: 1.0935819A

Cursor 2

I(L1)

Horz: 1.163616ms Vert: 667.32408mA

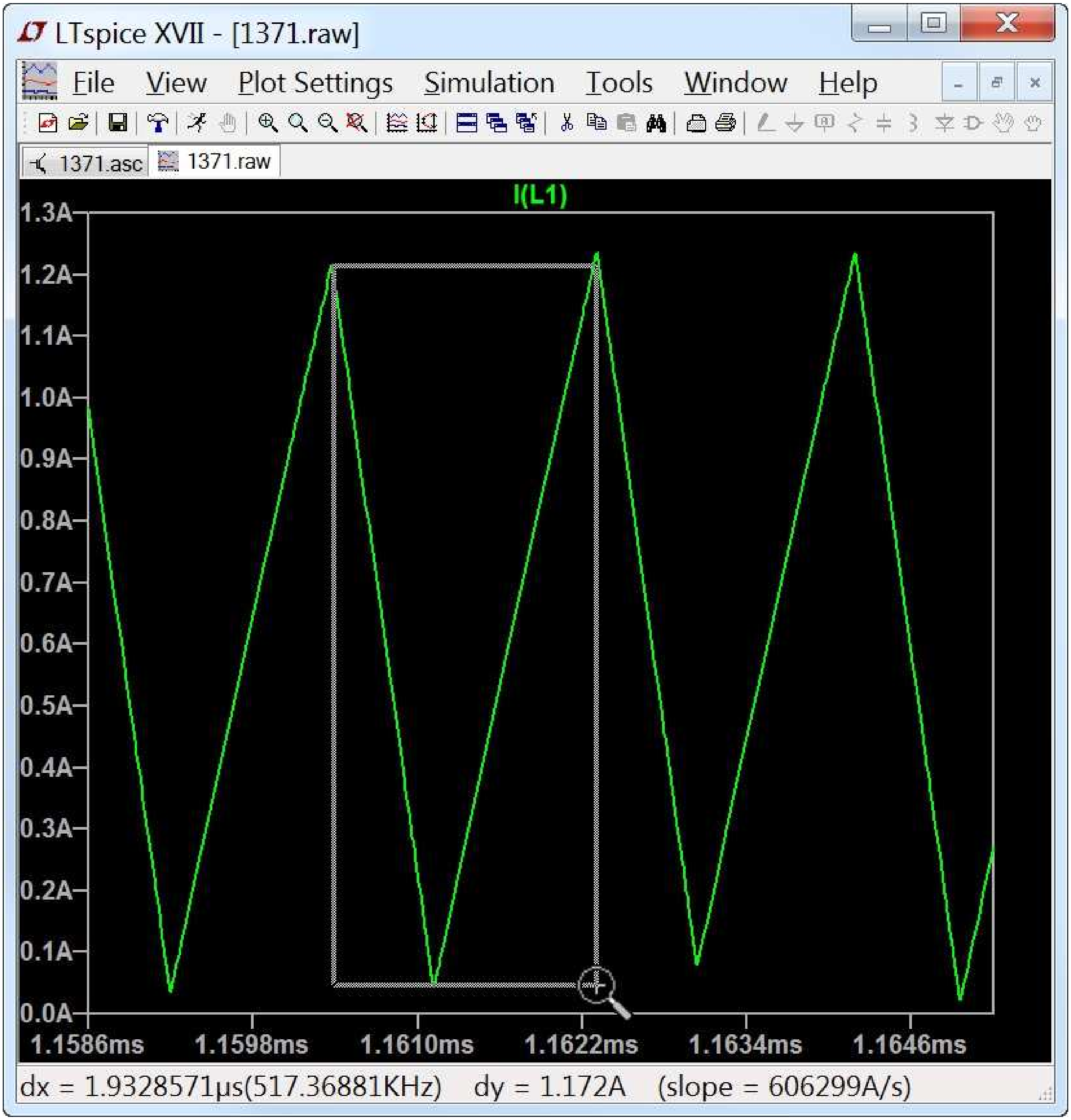
Diff (Cursor2 - Cursor1)

Horz: 1.2162857µs Vert: -426.2578mA

Freq: 822.17524KHz Slope: -350459

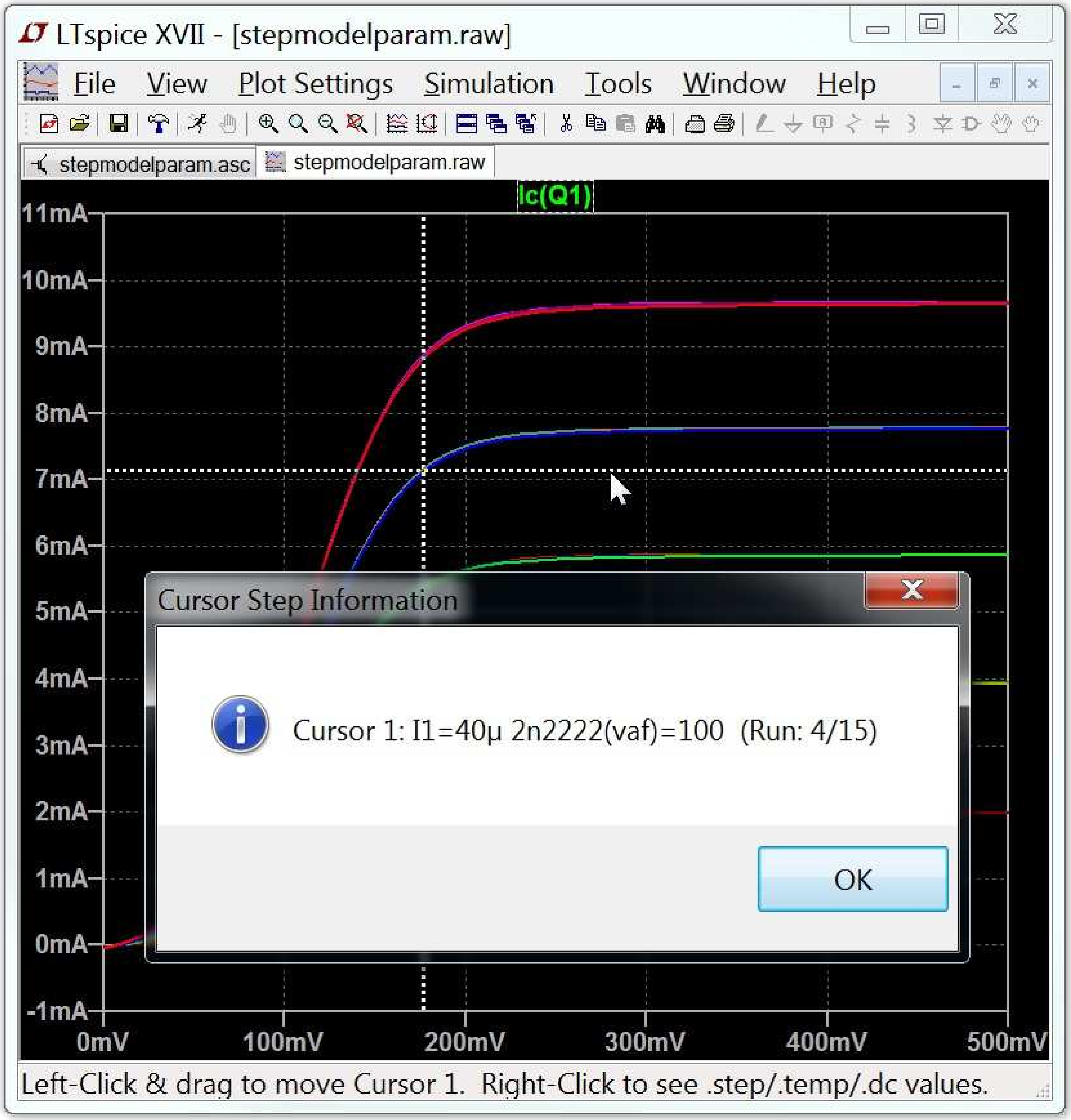
Left-Click & drag to move Cursor 2

Note that there is also mouse cursor readout independent of the above attached cursor readout. As you move the mouse over the waveform window, the mouse position is readout on the status bar. If you drag the mouse as if you were going to zoom, the size of box is displayed on the status bar. This lets you quickly measure differences with the mouse cursor. If the horizontal axis is time, then this time difference is also converted to frequency.



dx = 1.9328571µs(517.36881KHz) dy = 1.172A (slope = 606299A/s)

You can measure differences in this manner without performing the zoom by either pressing the Esc key or right mouse button before releasing the left mouse button.



Cursor 1: I1=40µ 2n2222(vaf)=100 (Run: 4/15)

Left-Click & drag to move Cursor 1. Right-Click to see .step/.temp/.dc values.

The attached cursors can also be used to readout which trace belongs to which run of a .step/.dc/.temp set of simulation runs. You can navigate the cursor from dataset to dataset with the up/down keyboard cursor keys and then right-click on the cursor to see the step information for that run.

Save Plot Configurations

The menu commands Plot Settings=>Save Plot Settings/Open Plot Settings files allow you to read and write plot configurations to disk. Plot setting files are ASCII files that have a file extension of .plt. The default filename is computed from the name of the data file by replacing the data file's ".raw" extension with ".plt" If such a file name exists when a data file is first opened, that plot settings file is read for initial plot configuration.

Each analysis type; .tran, .ac, .noise, etc.; has its own entry in the plot settings file. It isn't possible to load the settings from one analysis type to another. But you can use the plot settings file from another simulation of the same analysis type.

Fast Access File Format

During simulation, LTspice usually uses a compressed binary file format that allows additional simulation data to be appended without modifying the rest of the file. But once the simulation is completed, this file format can be slow to access for the purposes of adding a single new plot trace from the file.

To reduce this time, you can convert the file to an alternative, Fast Access, format. This format can only be done after the simulation is completed when no new data will be added to the file. But once the file is converted to this format, the load time of a new traces will be reduced typically by a factor equal to the number of data traces that have been saved in the file. For example, if you have a 5GB file with 2000 data traces, it might take 4min to add a new trace. But after you convert it to Fast Access format, this four minute load time would be reduced to a single second. This makes cross probing large circuits with huge simulation data files interactive. The exact time it takes to load a trace from a Fast Access format file will depend more on the amount of physical memory you have than your hard disk speed.

To convert a waveform window to Fast Access format, make the waveform window the active window and execute menu command =>Files=>Convert to Fast Access. The conversion process will require an amount of free disk space equal to the file size to be converted, but the converted file will be only 11 bytes larger than the original file.

The conversion process can take a long time and use up to one quarter of your physical memory. In fact, it can take more time to convert the file to Fast Access format then was required for the initial simulation. The exact time the conversion requires will depend on such factors as the state of the hard disk fragmentation and the amount of physical memory you have. During conversion, you may find your machine is not very responsive to your mouse and keyboard. It is possible to convert files in a batch command with the following command line syntax:

```
  XVIIx64.exe -FastAccess <file>
```

Where <file> is the name of the .raw file you wish to convert to Fast Access format.

This format is only supported for real data, not the complex data from a .AC analysis.

Memory, RAM, and Address Space

LTspice was the first PC-based SPICE program to implement its own 64bit address space on the hard disk to allow one to view waveform data files of essentially unlimited file size.

The 32-bit version of LTspice XVII(XVIIx86.exe) can address data files containing many Gigabytes of data but page in only about three Gigabytes at a time for plotting in the waveform viewer. No data trace can contain more than 2GB of data.

The 64-bit version of LTspice XVII(XVIIx64.exe) can use as much memory as you have installed but still keeps most of the data on the disk to allow you to view waveform files that exceed your physical memory. Note that current 64bit processors only have a few extra address lines over 32 bit processors.

LTspice® XVII

LTspice XVII is a schematic-driven circuit simulation program. The LTspice simulator was originally based years ago on Berkeley SPICE 3F4/5. The simulator has gone through a complete re-write in order to improve the performance of the simulator, fix bugs, and extend the simulator so that it can run industry standard semiconductor and behavioral models. A digital simulation capability, including co-simulation, has been added. Extensive enhancements have been made to the analog SPICE simulator, such as parallel processing and dynamic assembly and object code generation in the SPARSE matrix solver to make LTspice XVII the industry superlative analog simulator.

Many Linear Technology products are modeled with proprietary building blocks and/or proprietary hardware description languages that accurately encapsulate realistic behavior with custom macromodels. This allows a SMPS to be prototyped rapidly via simulation.

LTspice is intended to be used as your general-purpose SPICE simulator. New circuits can be drafted with the built-in schematic capture. Simulation commands and parameters are placed as text on the schematic using established SPICE syntax. Waveforms of circuit nodes and device currents can be plotted by clicking the mouse on the nodes in the schematic during or after simulation.

An invaluable reference that complements this documentation is the 2nd Edition of Semiconductor Device Modeling with SPICE by Giuseppe Massobrio and Paolo Antognetti, McGraw Hill, 1993 and later reprints. That book documents the semiconductor device equations and extensions that have been used in various commercial SPICE programs including those used in this one. For BSIM 3 and 4 devices, see the relevant documentation available from the UC Berkeley CAD group.

LTspice is a registered trademark of Linear Technology Corporation.

Circuit Description

Circuits are defined by a text netlist. The netlist consists of a list of circuit elements and their nodes, model definitions, and other SPICE commands.

The netlist is usually graphically entered. To start a new schematic, select the File=>Open menu item. A windows file browser will appear. Either select an existing schematic and save it under a new name or type in a new name to create a new blank schematic file. LTspice uses many different types of files and documents. You will want to make a file with a file name extension of ".asc". The schematic capture commands are under the Edit menu. Keyboard shortcuts for the commands are listed under Schematic Editor Overview.

When you simulate a schematic, the netlist information is extracted from the schematic graphical information to a file with the same name as the schematic but with a file extension of ".net". LTspice reads in this netlist.

You can also open, simulate, and edit a text netlist generated either by hand or externally generated. Files with the extensions ".net", ".cir", or ".sp" are recognized by LTspice as netlists.

This section of the help documents the syntax used in netlists, but occasionally gives schematic-level advice.

General Structure and Conventions

General Structure and Conventions

The circuit to be analyzed is described by a text file called a netlist. The first line in the netlist is ignored, that is, it is assumed to be a comment. The last line of the netlist is usually simply the line ".END", but this can be omitted. Any lines after the line ".END" are ignored.

The order of the lines between the comment and end is irrelevant. Lines can be comments, circuit element declarations or simulation directives. Let's start with an example:

```
* This first line is ignored
* The circuit below represents an RC circuit driven
* with a 1MHz square wave signal
R1 n1 n2 1K ; a 1KOhm resistor between nodes n1 and n2
C1 n2 0 100p ; a 100pF capacitor between nodes n2 and ground
V1 n1 0 PULSE(0 1 0 0 0 .5µ 1µ) ; a 1Mhz square wave
.tran 3µ ; do a 3µs long transient analysis
.end
```

The first two lines are comments. Any line starting with a "*" is a comment and is ignored. The line starting with "R1" declares that there is a 1K resistor connected between nodes n1 and n2. Note that the semicolon, ";", can be used to start a comment in the middle of a line. The line starting with "C1" declares that there is a 100pF capacitor between nodes n2 and ground. The node "0" is the global circuit common ground.

Below is an overview of the lexicon of LTspice:

- Leading spaces, blanks, and tabs are ignored.

- Case is ignored for A-Z as well as the following Microsoft OEM characters: Ŕ, Á, Â, Ă, Ä, Ĺ, Ć, Ç, Č, É, Ę, Ë, Ě, Í, Î, Ď, Đ, Ń, Ň, Ó, Ô, Ő, Ö, Ř, Ů, Ú, Ű, Ü, Ý, and Ţ.

- The first non-blank character of a line defines the type of circuit element.

Leading Character	**Type of line**
*****	Comment
A	Special function device
B	Arbitrary behavioral source

C	Capacitor
D	Diode
E	Voltage dependent voltage source
F	Current dependent current source
G	Voltage dependent current source
H	Current dependent voltage source
I	Independent current source
J	JFET transistor
K	Mutual inductance
L	Inductor
M	MOSFET transistor
O	Lossy transmission line
Q	Bipolar transistor
R	Resistor
S	Voltage controlled switch
T	Lossless transmission line
U	Uniform RC-line
V	Independent voltage source
W	Current controlled switch
X	Subcircuit Invocation
Z	MESFET or IGBT transistor
.	A simulation directive, For example: .options reltol=1e-4
+	A continuation of the previous line. The "+" is removed and the remainder of the line is considered part of the prior line.

Numbers can be expressed not only in scientific notation; e.g., 1e12; but also using engineering multipliers. That is, 1000.0 or 1e3 can also be written as 1K. Below is a table of understood multipliers:

Suffix	Multiplier
T	1e12
G	1e9
Meg	1e6
K	1e3

mil	25.4e-6
m	1e-3
u(or µ)	1e-6
n	1e-9
p	1e-12
f	1e-15

The suffixes are not case sensitive. Unrecognized letters immediately following a number or engineering multiplier are ignored. Hence, 10, 10V, 10Volts, and 10Hz all represent the same number, and M, MA, MSec, and MMhos all represent the same scale factor(.001). A common error is to draft a resistor with value of 1M, thinking of a one Megaohm resistor, however, 1M is interpreted as a one milliohm resistor. This is necessary for compatibility with standard SPICE practice.

LTspice will accept numbers written in the form 6K34 to mean 6.34K. This works for any of the multipliers above. It can be turned off by going to Tools=>Control Panel=>SPICE and unchecking "Accept 3K4 as 3.4K".

Nodes names may be arbitrary character strings. Global circuit common node(ground) is "0", though "GND" is special synonym. Note that since nodes are character strings, "0" and "00" are distinct nodes.

Throughout the following sections of the manual, angle brackets are placed around data fields that need to be filled with specific information; for example, "<srcname>" would be the name of some specific source. Square brackets indicate that the enclosed data field is optional.

Simulator Directives -- Dot Commands

To run a simulation, not only must the circuit be defined, but also the type of analysis to be performed. There are six different types of analyses: linearized small signal AC, DC sweep, noise, DC operating point, small signal DC transfer function and transient analysis. Precisely one of these six analyses must be specified when you run a simulation.

Whereas the circuit topology is typically schematically drafted, the commands are usually placed on the schematic as text. All such commands start with a period and are called "dot commands".

.AC -- Perform an Small Signal AC Analysis Linearized About the DC Operating Point

.BACKANNO -- Annotate the Subcircuit Pin Names on Port Currents

.DC -- Perform a DC Source Sweep Analysis

.END -- End of Netlist

.ENDS -- End of Subcircuit Definition

.FOUR -- Compute a Fourier Component

.FUNC -- User Defined Functions

.FERRET -- Download a File Given the URL

.GLOBAL -- Declare Global Nodes

.IC -- Set Initial Conditions

.INCLUDE -- Include Another File

.LIB -- Include a Library

.LOADBIAS -- Load a Previously Solved DC Solution

.MACHINE -- Arbitrary State Machine

.MEASURE -- Evaluate User-Defined Electrical Quantities

.MODEL -- Define a SPICE Model

.NET -- Compute Network Parameters in a .AC Analysis

.NODESET -- Supply Hints for Initial DC Solution

.NOISE -- Perform a Noise Analysis

.OP -- Find the DC Operating Point

.OPTIONS -- Set Simulator Options

.PARAM -- User-Defined Parameters

.SAVE -- Limit the Quantity of Saved Data

.SAVEBIAS -- Save Operating Point to Disk

.STEP -- Parameter Sweeps

.SUBCKT -- Define a Subcircuit

.TEMP -- Temperature Sweeps

.TF -- Find the DC Small-Signal Transfer Function

.TRAN -- Do a Nonlinear Transient Analysis

.WAVE -- Write Selected Nodes to a .Wav File

.AC -- Perform an Small Signal AC Analysis Linearized About the DC Operating Point

The small signal(linear) AC analysis of LTspice computes the AC complex node voltages as a function of frequency. First, the DC operating point of the circuit is found. Next, linearized small signal models for all of the nonlinear devices in the circuit are found for this operating point. Finally, using independent voltage and current sources as the driving signal, the resultant linearized circuit is solved in the frequency domain over the specified range of frequencies.

This mode of analysis is useful for filters, networks, stability analysis, and noise considerations. You can find a basic example here.

```
Syntax: .ac <oct, dec, lin> <Nsteps> <StartFreq> <EndFreq>
        .ac list <FirstFreq> [<NextFreq> [<NextFreq> ...]]
```

The frequency is swept between frequencies StartFreq and EndFreq. The number of steps is defined with the keyword "oct", "dec", or "lin" and Nsteps according to the following table:

Keyword	Nsteps
oct	No. of steps per octave
dec	No. of steps per decade
lin	Total number of linearly spaced steps between StartFreq and EndFreq

The syntax ".ac list <Freq>" with a single analysis frequency useful in combination with .step. It allows you to plot complex data as a function of a stepped parameter as shown in this example.

.BACKANNO -- Annotate the Subcircuit Pin Names to the Port Currents

Syntax: .backanno

This directive is automatically included in every netlist LTspice XVII generates from a schematic. It directs LTspice to include information in the .raw file that can be used to refer to port currents by the pin name. This allows you to cross probe pin currents by clicking on the symbol's pin.

.DC -- Perform a DC Source Sweep Analysis

This performs a DC analysis while sweeping the DC value of a source. It is useful for computing the DC transfer function of an amplifier or plotting the characteristic curves of a transistor for model verification.

```
Syntax: .dc <srcnam> <Vstart> <Vstop> <Vincr>
        + [<srcnam2> <Vstart2> <Vstop2> <Vincr2>]
```

The <srcnam> is either an independent voltage or current source that is to be swept from <Vstart> to <Vstop> in <Vincr> step sizes. In the following example, the default BSIM3v3.2.4 characteristic curves are plotted:

```
  * Example .dc sweep
  *
  M1 2 1 0 0 nbsim
  Vgs 1 0 3.5
  Vds 2 0 3.5
  .dc Vds 3.5 0 -0.05 Vgs 0 3.5 0.5
  .model nbsim NMOS Level=8
  .save I(Vds)
  .end
```

.END -- End of Netlist

This directive marks the end of the textual netlist. All lines after this one are ignored. Do not place this as text on the schematic, as the netlist extractor supplies it at the end.

.ENDS -- End of Subcircuit Definition

This directive marks the end of a subcircuit definition. See .SUBCKT for more information.

.FOUR -- Compute a Fourier Component after a .TRAN Analysis

Syntax: .four <frequency> [Nharmonics] [Nperiods] <data trace1> [<data trace2> ...]

Example: .four 1kHz V(out)

This command is performed after a transient analysis. It's supplied in order to be compatible with legacy SPICE simulators. The output from this command is printed in the .log file. Use the menu item "View=>Spice Error Log" to see the output. For most purposes, the FFT capability built into the waveform viewer is more useful.

If the integer Nharmonics is present, then the analysis includes that number of harmonics. The number of harmonics defaults to 9 if not specified.

The Fourier analysis is performed over the period from the final time, Tend, to one period before Tend unless an integer Nperiods is given after Nharmonics. If Nperiods is given as -1, the Fourier analysis is performed over the entire simulation data range.

.FUNC -- User Defined Functions

Syntax: .func <name>([args]) {<expression>}

Example: .func Pythag(x,y) {sqrt(x*x+y*y)}

The .func directive allows the creation of user-defined functions for use with user parameterized circuits and behavioral sources. This is useful for associating a name with a function for the sake of clarity and parameterizing subcircuits so that abstract circuits can be saved in libraries.

The .func statement can be included inside a subcircuit definition to limit the scope the function to that subcircuit and the subcircuits invoked by that subcircuit.

To invoke parameter substitution and expression evaluation with these user-defined functions, enclose the expression in curly braces. The enclosed expression will be replaced with the floating-point value.

Below is an example using both a .func and .param statements.

```
* Example deck using a .func statement
.func myfunc(x,y) {sqrt(x*x+y*y)}
.param u=100 v=600
V1 a 0 pulse(0 1 0 1n 1n .5µ 1µ)
R1 a b {myfunc(u,v/3)}
C1 b 0 100p
.tran 3µ
.end
```

All parameter substitution evaluation is done before the simulation begins.

.FERRET -- Download a File Given the URL

This command allows you to download files in batch mode by specifying the urls. This is handy when you don't want to have to point your browser at every file or want to prevent your browser from doing any translation of the file. The downloaded file will be in the same directory as the source schematic or netlist. This command has no effect on the simulation.

.GLOBAL -- Declare Global Nodes

Syntax: .global <node1> [node2 [node3] [...]]

Example: .global VDD VCC

The .global command allows you to declare that certain nodes mentioned in subcircuits are not local to subcircuit but are absolute global nodes.

Note that global circuit common is node "0" and that a .global statement is not required. Also, node names that of the form "$G_" are also global nodes without being declared in a .global statement.

.IC -- Set Initial Conditions

The .ic directive allows initial conditions for transient analysis to be specified. Node voltages and inductor currents may be specified. A DC solution is performed using the initial conditions as constraints. Note that although inductors are normally treated as short circuits in the DC solution in other SPICE programs, if an initial current is specified, they are treated as infinite-impedance current sources in LTspice.

Syntax: .ic [V(<n1>)=<voltage>] [I(<inductor>)=<current>]

Example: .ic V(in)=2 V(out)=5 V(vc)=1.8 I(L1)=300m

.INCLUDE -- Include Another File

Syntax: .include <filename>

This directive includes the named file as if that file had been typed into the netlist instead of the .include command. This is useful for including libraries of models or subcircuits.

An absolute path name may be entered for the filename. Otherwise LTspice looks first in the directory %HOMEPATH%\Documents\LTspiceXVII\lib\sub and then in the directory that contains the calling netlist and finally in the list of directories listed in the Library Search Path.

No file name extension is assumed. You must use ".inc myfile.lib" not ".inc myfile" if the file is called "myfile.lib"

It is possible to specify a url of the following form as a file name:

```
.inc http://www.company.com/models/library.lib
```

The file "library.lib" will be http-transferred to the circuit directory and included. For subsequence simulations, in the interest of avoiding downloading the file each time you run the simulation, you can edit the .inc statement to

```
.inc library.lib
```

Note that if the url you specify doesn't exist, most web servers don't return an error, but return a html web page to be displayed in your web browser that explains the error. LTspice can't always read these pages as error conditions so you may get some cryptic error message when the simulation tries to proceed with the included html language error page included in the simulation as valid SPICE syntax.

.LIB -- Include a Library

Syntax: .lib <filename>

This directive includes the model and subcircuit definitions of the named file as if that file had been typed into the netlist instead of the .lib command. Circuit elements at global scope are ignored.

An absolute path name may be entered for the filename. Otherwise LTspice looks first in the directory %HOMEPATH%\Documents\LTspiceXVII\lib\cmp and then %HOMEPATH%\Documents\LTspiceXVII\lib\sub and then in the directory that contains the calling netlist.

No file name extension is assumed. You must use ".lib myfile.lib" not ".lib myfile" if the file is called "myfile.lib"

It is possible to specify a URL of the following form as a file name:

```
.lib http://www.company.com/models/library.mod
```

The file "library.mod" will be http-transferred to the circuit directory and included as a library. For subsequence simulations, in the interest of avoiding downloading the file each time you run the simulation, you can edit the .lib statement to

```
.lib library.mod
```

Note that if the URL you specify doesn't exist, most web servers don't return an error, but return a html web page to be displayed in your web browser that explains the error. LTspice can't always read these pages as error conditions so you may get some cryptic error message when the simulation tries to proceed with the included html language error page included in the simulation as valid SPICE syntax.

Encrypted Libraries

LTspice can generate and read a special form of encrypted libraries. This allows one user to prepare a library that another user can use in a simulation without revealing the implementation of the library. A reasonable attempt has been made to make the encrypted library difficult to decode by unauthorized concerns, but it cannot be considered perfectly secure if for no other reason than it is implemented in software.

To prepare an encrypted library, you need to invoke LTspice from the command line with the command line option "-encrypt". You will need to first backup the library because it will be replaced with the encrypted version. THERE EXISTS NO UTILITY TO CONVERT AN ENCRYPTED LIBRARY BACK TO CLEAR TEXT. Below summarizes the two steps:

1. Make a backup copy of the library. The version you encrypt is deleted.

2. From a command line, type

```
\XVIIx64.exe -encrypt <filename>
```

The file <filename> will be replaced with an encrypted version. The encryption process will take a few minutes.

One this process is finished, you have an encrypted ASCII file. It's possible to add a copyright notice above the "* Begin:" line, but the first 9 lines of the file must remain unchanged and each line of copyright notice you add must begin with the character '*'.

That is, here an encrypted file written by LTspice:

```
* LTspice Encrypted File
*
* This encrypted file has been supplied by a 3rd
* party vendor that does not wish to publicize
* the technology used to implement this library.
*
* Permission is granted to use this file for
* simulations but not to reverse engineer its
* contents.
*
* Begin:
50 3E 46 0F FA 6E 67 FF B8 4D D9 62 14 32 60 24
36 71 35 0B 66 4F AD 52 B8 F5 9E 22 9F C0 18 8B
FB FE 1D...
```

which you can change to

```
* LTspice Encrypted File
*
* This encrypted file has been supplied by a 3rd
* party vendor that does not wish to publicize
* the technology used to implement this library.
*
```

```
* Permission is granted to use this file for
* simulations but not to reverse engineer its
* contents.
*
* Copyright © 2005 Acme SPICE Modeling
* For additional information, see
* www.acmespicemodels.com
*
* Begin:
50 3E 46 0F FA 6E 67 FF B8 4D D9 62 14 32 60 24
36 71 35 0B 66 4F AD 52 B8 F5 9E 22 9F C0 18 8B
FB FE 1D...
```

.LOADBIAS -- Load a Previously Solved DC Solution

Syntax: .loadbias <filename>

The loadbias command is the compliment to the .savebias command. First run a simulation that executes a .savebias command. Then change the .savebias command to a .loadbias command.

.MACHINE -- Arbitrary State Machine

LTspice XVII includes an arbitrary state machine and introduces a new programming language called Contraption Programming Language. There are five new commands:

```
   .mach[ine] [<tripdt>] ; tripdt is an optional temporal
tolerance
   .state <name> <value>
   .rule <old state> <new state> <condition>
   .output (node) <expression>
   .endmach[ine] ; end of block
```

The order of statements between the .mach and .endmach statements makes no difference as is common in a declarative language(vs a procedural language like C), with the exception that the first state declared is the initial state and the rules are checked in order.

The point to assigning a value to a state is so that it can be mentioned in the expression of an output.

There can be as many or few rules as you wish. If the machine is in <old state> and the expression of <condition> evaluates to something larger that .5, the machine advances to <new state>. Only one rule executes per timestep. The character '*' as the value of <old state> matches any state. Such rules are checked first.

The .output statements implement current sources that require external devices to readout the current. As shown in the examples below, 1K ground-referenced resistors are normally used, but you might want to add some parallel capacitance to ground to slow transitions. The <expression> can combine combinatorial logic and/or state.

The simplest example of a Arbitrary State Machine would be one with no states. This is an example of an inverter:

```
   * inverter state machine example
   V1 1 0 pulse(0 1 0 1u 1u .5m 1m)
   R1 2 0 1K ; an impedance for the .output statement
   .machine
   .output (2) V(1) < .5
   .endmachine
   .tran 3m
   .end
```

Here is an example of a divide by 2 with a reset:

```
* divide by 2 example
V1 1 0 pulse(0 1 0 1u 1u .5m 1m)
V2 c 0 pulse(0 1 0 1u 1u 5m 10m)
R1 2 0 1K
R2 3 0 1K
R3 4 0 1K
.machine
.state S0a 0
.state S0b 0
.state S1a 1
.state S1b 1
.rule S0a S0b V(1) < .5
.rule S0b S1a V(1) > .5
.rule S1a S1b V(1) < .5
.rule S1b S0a V(1) > .5
.rule * S0a V(c) > .5
.output (2) V(1) < .5
.output (3) V(1) > .5
.output (4) state
.endmachine
.tran 30m
.end
```

Note: Current monitoring is not implemented. If a state machine output is connected to a pin, current monitoring for the pin will not be correct.

.MEASURE -- Evaluate User-Defined Electrical Quantities

There are two basic different types of .MEASURE statements. Those that refer to a point along the abscissa (the independent variable plotted along the horizontal axis, i.e., the time axis of a .tran analysis) and .MEASURE statements that refer to a range over the abscissa. The first version, those that point to one point on the abscissa, are used to print a data value or expression thereof at a specific point or when a condition is met. The following syntax is used:

```
Syntax: .MEAS[SURE] [AC|DC|OP|TRAN|TF|NOISE] <name>
+ [<FIND|DERIV|PARAM> <expr>]
+ [WHEN <expr> | AT=<expr>]]
+ [TD=<val1>] [<RISE|FALL|CROSS>=[<count1>|LAST]]
```

Note one can optionally state the type of analysis to which the .MEAS statement applies. This allows you to use certain .MEAS statements only for certain analysis types. The name is required to give the result a parameter name that can be used in other .MEAS statements. Below are example .MEAS statements that refer to a single point along the abscissa:

```
.MEAS TRAN res1 FIND V(out) AT=5m
```

Print the value of V(out) at t=5ms labeled as res1.

```
.MEAS TRAN res2 FIND V(out)*I(Vout) WHEN V(x)=3*V(y)
```

Print the value of the expression V(out)*I(Vout) the first time the condition V(x)=3*V(y) is met. This will be labeled res2.

```
.MEAS TRAN res3 FIND V(out) WHEN V(x)=3*V(y) cross=3
```

Print the value of V(out) the third time the condition V(x)=3*V(y) is met. This will be labeled res3.

```
.MEAS TRAN res4 FIND V(out) WHEN V(x)=3*V(y) rise=last
```

Print the value of V(out) the last time the condition V(x)=3*V(y) is met when approached as V(x) increasing wrt 3*V(y). This will be labeled res4.

```
.MEAS TRAN res5 FIND V(out) WHEN V(x)=3*V(y) cross=3 TD=1m
```

Print the value of V(out) the third time the condition

V(x)=3*V(y) is met, but don't start counting until the time as elapsed to 1ms. This will be labeled res5.

```
.MEAS TRAN res6 PARAM 3*res1/res2
```

Print the value of 3*res1/res2. This form is useful for printing expressions of other .meas statement results. It's not intended that expressions based on direct simulation data, such as V(3), are present in the expression to be evaluated, but if they are, the data is taken from the last simulated point. The result will be labeled res6.

Note that the above examples, while referring to one point along the abscissa, the requested result is based on ordinate data(the dependent variables). If no ordinate information is requested, then the .MEAS statement prints point on the abscissa that the measurement condition occurs:

```
.MEAS TRAN res6 WHEN V(x)=3*V(y)
```

Print the first time the condition V(x)=3*V(y) is met. This will be labeled res6.

The other type of .MEAS statement refers to a range over the abscissa. The following syntax is used:

```
Syntax: .MEAS [AC|DC|OP|TRAN|TF|NOISE] <name>
+ [<AVG|MAX|MIN|PP|RMS|INTEG> <expr>]
+ [TRIG <lhs1> [[VAL]=]<rhs1>] [TD=<val1>]
+ [<RISE|FALL|CROSS>=<count1>]
+ [TARG <lhs2> [[VAL]=]<rhs2>] [TD=<val2>]
+ [<RISE|FALL|CROSS>=<count2>]
```

The range over the abscissa is specified with the points defined by "TRIG" and "TARG". The TRIG point defaults to the start of the simulation if omitted. Similarly, the TARG point defaults to the end of simulation data. If all three of the TRIG, TARG, and the previous WHEN points are omitted, then the .MEAS statement operates over the entire range of data. The types of measurement operations that can be done over an interval are

Keyword	**Operation performed over interval**
AVG	Compute the average of <expr>
MAX	Find the maximum value of <expr>
MIN	Find the minimum value of <expr>
PP	Find the peak-to-peak of <expr>

RMS	Compute the root mean square of <expr>
INTEG	Integrate <expr>

If no measurement operation is specified, the result of the .MEAS statement is the distance along the abscissa between the TRIG and TARG points. Below are example interval .MEAS statements:

```
.MEAS TRAN res7 AVG V(NS01)
+ TRIG V(NS05) VAL=1.5 TD=1.1u FALL=1
+ TARG V(NS03) VAL=1.5 TD=1.1u FALL=1
```

> Print the value of average value of V(NS01) from the 1^{st} fall of V(NS05) to 1.5V after 1.1us and the 1st fall of V(NS03) to 1.5V after 1.1us. This will be labeled res7.

For .AC analyses, the conditional expressions of complex data are translated to real conditions by considering only the real part of the complex value of the expression.

Also, the result of a .MEAS statement can be used in another .MEAS statement. In this example, the 3dB bandwidth is computed:

```
.MEAS AC tmp max mag(V(out)); find the peak response and call it "tmp"

.MEAS AC BW trig mag(V(out))=tmp/sqrt(2) rise=1
+ targ mag(V(out))=tmp/sqrt(2) fall=last
```

> Print the difference in frequency between the two points 3dB down from peak response. NOTE: The data from a .AC analysis is complex and so are the .measurement statements results. However, the equality refers only to the real part of the complex number, that is, "mag(V(out))=tmp/sqrt(2)" is equivalent to Re(mag(V(out)))=Re(tmp/sqrt(2)).

When testing a condition such as "when <cond1> = <cond2>" you will want the condition to go through the equality, not just meet it. This relates to the fact that floating point equality should never be required due to the finite precession used in storing numbers.

The AVG, RMS, and INTEG operations are different for .NOISE analysis than the analysis types since the noise is more meaningfully integrated in quadrature over frequency. Hence AVG and RMS both give the RMS noise voltage and INTEG gives the integrated total noise. Hence, if you add the SPICE directives

```
.MEAS NOISE out_totn INTEG V(onoise)
```

```
.MEAS NOISE in_totn INTEG V(inoise)
```

the total integrated input and output referenced rms noise will be printed in the .log file.

.MEAS statements are done in post processing after the simulation is completed. This allows you to write a script of .MEAS statements and execute them on a dataset. To do this, make the waveform window the active window and execute menu command File=>Execute .MEAS Script. Another consequence of .MEAS statements being done in post processing after the simulation is that the accuracy of the .MEAS statement output is limited by the accuracy of the waveform data after compression. You may want to adjust the compression settings for more precise .MEAS statement output.

.MEAS statements are usually just put on the schematic as a SPICE directive or in the netlist with the rest of the simulation commands and circuit definition. The output is put in the .log file which can be viewed with menu command View=>SPICE Error Log. If the simulation includes a .step command, the .measure statements are executed for each step and the results are printed as tables in the .log file. These tables for .measure results can be plotted like normal waveforms by this procedure:

i) After the simulation completes, execute menu menu command View=>SPICE Error Log

ii) Right click in the .log file and, execute context menu command Plot .step'ed .meas data.

.MODEL -- Define a SPICE Model

Defines a model for a diode, transistor, switch, lossy transmission line or uniform RC line

Some circuit elements, for example, transistors, have many parameters. Instead of defining every transistor parameter for every instance of a transistor, transistors are grouped by model name and have parameters in common. The transistors of the same model can have different sizes and the electrical behavior is scaled to the size of the instance.

Syntax: .model <modname> <type>[(<parameter list>)]

The parameter list depends on the type of model. Below is a list of model types:

Type	Associated Circuit Element
SW	Voltage Controlled Switch
CSW	Current Controlled Switch
URC	Uniform Distributed RC Line
LTRA	Lossy Transmission Line
D	Diode
NPN	NPN Bipolar Transistor
PNP	PNP Bipolar Transistor
NJF	N-channel JFET model
PJF	P-channel JFET model
NMOS	N-channel MOSFET
PMOS	P-channel MOSFET
NMF	N-channel MESFET
PMF	P-channel MESFET
NIGBT	N-channel IGBT
PIGBT	P-channel IGBT
VDMOS	Vertical Double Diffused Power MOSFET

See the description of the circuit element for a list of which parameters are instance specific and which are common to a model.

.NET -- Compute Network Parameters in a .AC Analysis

This statement is used with a small signal(.AC) analysis to compute the input and output admittance, impedance, Y-parameters, Z-parameters, H-parameters, and S-parameters of a 2-port network. It can also be used to compute the input admittance and impedance of a 1-port network. This must be used with a .AC statement, which determines the frequency sweep of the network analysis.

Syntax: .net [V(out[,ref])|I(Rout)] <Vin|Iin> [Rin=<val>] [Rout=<val>]

The network input is specified by either an independent voltage source, <Vin>, or an independent current source, <Iin>. The optional output port is specified either with a node, V(out), or a resistor, I(Rout). The ports will be terminated with resistances Rin and Rout. If unspecified, the termination impedances default to 1 Ohm except in the case of the Voltage source with an Rser specified or an output port specified with a resistor. In those two cases the termination resistances defaults to the device impedance. Termination values specified on the .NET statement will override device impedances for the .NET calculation, but not for the normal .AC node voltages and currents. That is, the .NET statement will not impose terminating impedances on the network for the normal voltages and currents computed as part of the .AC analysis.

See the example file installed at %HOMEPATH%\Documents\LTspiceXVII\examples\Educational\S-param.asc. It recommends using a voltage source, V4, with Rser set the desired source impedance and a resistor, Rout, to set the output termination with a .NET statement reading simply ".net I(Rout) V4." No Rin or Rout values specified on the .net statement and the input/output devices supply default termination values. This arrangement makes the node voltages and currents of the .AC analysis correspond to the network being terminated in the same manner as in the .NET statement.

.NODESET -- Supply Hints for Initial DC Solution

The .nodeset directive supplies hints for finding the DC operating point. If a circuit has multiple possible DC states as, for example, a flipflop, the iteration process for finding the DC solution may never converge. A .nodeset directive can be used to lead the circuit to one or another state. Basically, after a solution pass is done with the voltage specified on the nodeset directive, the constraint is removed for subsequent iterative passes.

```
Syntax: .NODESET V(node1)=<voltage> [V(node2)=<voltage [...]]
```

.NOISE -- Perform a Noise Analysis

This is a frequency domain analysis that computes the noise due to Johnson, shot and flicker noise. The output data is noise spectral density per unit square root bandwidth.

Syntax: .noise V(<out>[,<ref>]) <src> <oct, dec, lin> <Nsteps> <StartFreq> <EndFreq>
        .noise V(<out>[,<ref>]) <src> list <FirstFreq>[<NextFreq> [<NextFreq> ...]]

V(<out>[,<ref>]) is the node at which the total output noise is calculated. It can be expressed as V(n1, n2) to represent the voltage between two nodes. <src> is the name of an independent source to which input noise is referred. <src> is the noiseless input signal. The parameters <oct, dec, lin>, <Nsteps>, <StartFreq>, and <EndFreq> define the frequency range of interest and resolution in the manner used in the .ac directive.

Output data trace V(onoise) is the noise spectral voltage density referenced to the node(s) specified as the output in the above syntax. If the input signal is given as a voltage source, then data trace V(inoise) is the input-referred noise voltage density. If the input is specified as a current source, then the data trace inoise is the noise referred to the input current source signal. The noise contribution of each component can be plotted. These contributions are referenced to the output. You can reference them to the input by dividing by the data trace "gain".

The waveform viewer can integrate noise over a bandwidth by <Ctrl-Key> + left mouse button clicking on the corresponding data trace label.

The syntax ".noise V(<out>[,<ref>]) <src> list <Freq>" with a single analysis frequency is useful in combination with .step. It allows you to plot noise densities as a function of a stepped parameter as shown in this example.

.OP -- Find the DC Operating Point

Perform a DC operating point solution with capacitances open circuited and inductances short circuited. Usually a DC solution is performed as part of another analysis in order to find the operating point of the circuit. Use .op if you wish only this operating point to be found. The results will appear in a dialog box. After a .OP simulation, when you point at a node or current the .OP solution will appear on the status bar.

There is no guarantee that the operating point of a general nonlinear circuit can be found with successive linear approximations as is done in Newton-Raphson iteration. Should direct Newton iteration fail, LTspice tries a number of other methods to find an operating point. Below is a table of the methods used and the options settings required to disable a particular method.

Method	Directive to disable
Direct Newton Iteration	.options NoOpIter
Adaptive Gmin Stepping	.options GminSteps=0
Adaptive Source Stepping	.options SrcSteps=0
Pseudo Transient	.options pTranTau=0

.OPTIONS -- Set Simulator Options

Keyword	Default	Description
abstol	1pA	Absolute current error tolerance
baudrate	(none)	Used for eye diagrams. Tells the waveform v wrap the abscissa time to overlay the bit t
chgtol	10fC	Absolute charge tolerance
cshunt	0.	Optional capacitance added from every node
cshuntintern	cshunt	Optional capacitance added from every devic node to ground.
defad	0.	Default MOS drain diffusion area
defas	0.	Default MOS source diffusion area
defl	100µm	Default MOS channel length
defw	100µm	Default MOS channel width
delay	0.	Used for eye diagrams. Shifts the bit trans diagram.
fastaccess	false	Convert to fastaccess file format at end of
flagloads	false	Flags external current sources as loads.
Gmin	1e-12	Conductance added to every PN junction to a convergence.
gminsteps	25	Set to zero to prevent gminstepping for the solution.
gshunt	0.	Optional conductance added from every node
itl1	100	DC iteration count limit.
itl2	50	DC transfer curve iteration count limit.
itl4	10	Transient analysis time point iteration cou
itl6	25	Set to zero to prevent source stepping for DC solution.
srcsteps	25	Alternative name for itl6.
maxclocks	Infin.	maximum number of clock cycles to save
maxstep	Infin.	Maximum step size for transient analysis
meascplxfmt	bode	Complex number format of .meas statement re "polar", "cartesian", or "bode".
measdgt	6	Number of significant figures used for .mea statement output.
method	trap	Numerical integration method, either trapez
minclocks	10	minimum number of clock cycles to save
MinDeltaGmin	1e-4	Sets a limit for termination of adaptive gr
nomarch	false	Do not plot marching waveforms

noopiter	false	Go directly to gmin stepping.
numdgt	6	Historically "numdgt" was used to set the n significant figures used for output data. I "numdgt" is set to be > 6, double precision dependent variable data.
pivrel	1e-3	Relative ratio between the largest column e acceptable pivot value.
pivtol	1e-13	Absolute minimum value for a matrix entry t as a pivot.
reltol	.001	Relative error tolerance.
srcstepmethod	0	Which source stepping algorithm to start wi
sstol	.001	Relative error for steady-state detection.
startclocks	5	Number of clock cycles to wait before looki state.
temp	27°C	Default temperature for circuit element ins don't specify temperature.
tnom	27°C	Default temperature at which device paramet measured for models that don't specify this
topologycheck	1	Set to zero to skip check for floating node voltage sources, and non-physical transform topology
trtol	1.0	Set the transient error tolerance. This par estimate of the factor by which the actual error is overestimated.
trytocompact	1	When non-zero, the simulator tries to conde transmission lines' history of input voltag currents.
vntol	1µV	Sets the absolute voltage error tolerance.
plotreltol	.0025	Sets the relative error tolerance for wavef compression.
plotvntol	10µV	Sets the absolute voltage error tolerance f compression.
plotabstol	1nA	Sets the absolute current error tolerance f compression.
plotwinsize	300	Number of data points to compress in one wi zero to disable compression.
ptrantau	.1	Characteristic source start-up time for a c transient analysis to find the operating po zero to disable pseudo transient.
ptranmax	0	If set non-zero, that time of the damped ps transient analysis is used as the operating whether the circuit has settled or not.

.PARAM -- User-Defined Parameters

The .param directive allows the creation of user-defined variables. This is useful for associating a name with a value for the sake of clarity and parameterizing subcircuits so that abstract circuits can be saved in libraries.

The .param statement can be included inside a subcircuit definition to limit the scope the parameter value to that subcircuit.

To invoke parameter substitution and expression evaluation, enclose the expression in curly braces. The enclosed expression will be replaced with the floating-point value.

Below is an example using both a .param statement and directly passing parameters on the subcircuit invocation line.

```
*
* This is the circuit definition
.params x=y y=z z=1k*tan(pi/4+.1)
X1 a b 0 divider top=x bot=z
V1 a 0 pulse(0 1 0 .5µ .5µ 0 1µ)

* this is the definition of the subcircuit
.subckt divider n1 n2 n3
r1 n1 n2 {top}
r2 n2 n3 {bot}
.ends
*

.tran 3µ
.end
```

The parameter substitution scheme is a symbolic declarative language. The parameters are not passed to the subcircuit as evaluated values, but by the expressions and relations themselves. When curly braces are encountered, the enclosed expression is evaluated on the basis of all relations available at the scope and reduced to a floating point value.

The following functions and operations are available:

Function Name	Description
abs(x)	Absolute value of x
	Real part of the arc cosine of x, e.g., acos(-5) returns

acos(x)	3.14159, not 3.14159+2.29243i
arccos(x)	Synonym for acos()
acosh(x)	Real part of the arc hyperbolic cosine of x, e.g., acosh(.5) returns 0, not 1.0472i
asin(x)	Real part of the arc sine of x, e.g., asin(-5) returns -1.57080, not -1.57080+2.29243i
arcsin(x)	Synonym for asin()
asinh(x)	Arc hyperbolic sine
atan(x)	Arc tangent of x
arctan(x)	Synonym for atan()
atan2(y,x)	Four quadrant arc tangent of y/x
atanh(x)	Arc hyperbolic tangent
buf(x)	1 if x > .5, else 0
cbrt(x)	Cube root of (x)
ceil(x)	Integer equal or greater than x
cos(x)	Cosine of x
cosh(x)	Hyperbolic cosine of x
exp(x)	e to the x
fabs(x)	Same as abs(x)
flat(x)	Random number between -x and x with uniform distribution
floor(x)	Integer equal to or less than x
gauss(x)	Random number from Gaussian distribution with sigma of x.
hypot(x,y)	sqrt(x**2 + y**2)
if(x,y,z)	If x > .5, then y else z
int(x)	Convert x to integer
inv(x)	0. if x > .5, else 1.
limit(x,y,z)	Intermediate value of x, y, and z
ln(x)	Natural logarithm of x

log(x)	Alternate syntax for ln()
log10(x)	Base 10 logarithm
max(x,y)	The greater of x or y
mc(x,y)	A random number between x*(1+y) and x*(1-y) with uniform distribution.
min(x,y)	The smaller of x or y
pow(x,y)	Real part of x**y, e.g., pow(-.5,1.5) returns 0., not 0.353553i
pwr(x,y)	abs(x)**y
pwrs(x,y)	sgn(x)*abs(x)**y
rand(x)	Random number between 0 and 1 depending on the integer value of x.
random(x)	Similar to rand(), but smoothly transitions between values.
round(x)	Nearest integer to x
sgn(x)	Sign of x
sin(x)	Sine of x
sinh(x)	Hyperbolic sine of x
sqrt(x)	Real part of the square root of x, e.g., sqrt(-1) returns 0, not 0.707107i
table(x,a,b,c,d,...)	Interpolate a value for x based on a look up table given as a set of pairs of points.
tan(x)	Tangent of x.
tanh(x)	Hyperbolic tangent of x
u(x)	Unit step, i.e., 1 if x > 0., else 0.
uramp(x)	x if x > 0., else 0.

The following operations are grouped in reverse order of precedence of evaluation:

Operand	**Description**
&	Convert the expressions to either side to Boolean, then AND.

|	Convert the expressions to either side to Boolean, then OR.
^	Convert the expressions to either side to Boolean, then XOR.
>	True if expression on the left is greater than the expression on the right, otherwise false.
<	True if expression on the left is less than the expression on the right, otherwise false.
>=	True if expression on the left is greater than or equal the expression on the right, otherwise false.
<=	True if expression on the left is less than or equal the expression on the right, otherwise false.
+	Floating point addition
-	Floating point subtraction
*	Floating point multiplication
/	Floating point division
**	Raise left hand side to power of right hand side, only real part is returned, e.g., -2**1.5 returns zero, not 2.82843i

All parameter substitution evaluation is done before the simulation begins.

.SAVE -- Limit the Quantity of Saved Data

Some simulations, particularly time domain simulations, can generate large amount of data. The amount of output can be restricted by using the .save directive to save only the specific node voltages and device current of interest.

```
Syntax: .save V(out) [V(in) [I(L1) [I(S2)]]] [dialogbox]
```

The directive .save I(Q2) will save the base, collector and emitter currents of bipolar transistor Q2. To save a single terminal current, specify Ic(Q2).

The wildcard characters '*' and '?' can be used to specify data traces matching a pattern. For example, ".save V(*) Id(*)" will save every voltage and every drain current.

If the keyword "dialogbox" is specified, then a dialog box with a list of all-available default nodes and currents is displayed allowing the user to select from the list which should be saved. If the netlist was generated from a schematic, then nodes and devices can be pointed to and clicked on in the schematic to highlight them as selected in the dialog box.

.SAVEBIAS -- Save Operating Point to Disk

```
Syntax: .savebias <filename> [internal]
+ [temp=<value>] [time=<value> [repeat]] [step=<value>]
+ [DC1=<value>] [DC2=<value>] [DC3=<value>]
```

This command writes a text file to disk that is reloaded with a .loadbias command in a subsequent simulation. If you have a circuit that has a difficult-to-solve DC operating point, you can save that solution to disk so that the next analysis can save time finding the DC solution before proceeding to the rest of the simulation.

The keyword "internal" can be added to indicate that the internal nodes of some devices should also be kept so that a more complete version of the DC solution is kept.

If you want to save a particular DC operating point from a .tran analysis, you can give specify a time. The first solved time point after the stipulated time will be written. The modifier "repeat" will cause the DC solution to be written after every period specified by this time. The file will contain only the most recently solved DC point. DC1, DC2, and DC3 can be given to extract a single operating point from .dc sweep analysis.

The savebias command writes a text file in the form of a .nodeset command. Note that nodeset statements are only recommendations of the solution. That is, the solver will start iterating the solution with the node voltages given in the nodeset statements, but will continue iterating until it's satisfied that the solution is valid. If you want to restart a .tran solution from the DC operating point, you can edit the file from a .nodeset to a .ic to try to coercive the solver to start from this DC state.

Since the integration state of all the circuit reactances isn't saved in the .savebias file, success with this technique varies.

.STEP -- Parameter Sweeps

This command causes an analysis to be repeatedly performed while stepping the temperature, a model parameter, a global parameter, or an independent source. Steps may be linear, logarithmic, or specified as a list of values.

Example: .step oct v1 1 20 5

Step independent voltage source V1 from 1 to 20 logarithmically with 5 points per octave.

Example: .step I1 10u 100u 10u

Step independent current source I1 from 10u to 100u in step increments of 10u.

Example: .step param RLOAD LIST 5 10 15

Perform the simulation three times with global parameter Rload being 5, 10 and 15.

Example: .step NPN 2N2222(VAF) 50 100 25

Step NPN model parameter VAF from 50 to 100 in steps of 25.

Example: .step temp -55 125 10

Step the temperature from -55°C to 125°C in 10-degree step. Step sweeps may be nested up to three levels deep.

.SUBCKT -- Define a Subcircuit

As an aid to defining a circuit, repetitive circuitry can be enclosed in a subcircuit definition and used as multiple instances in the same circuit. Before the simulation runs, the circuit is expanded to a flat netlist by replacing each invocation of a subcircuit with the circuit elements in the subcircuit definition. There is no limit on the size or complexity of subcircuits.

The end of a subcircuit definition must be a .ends directive.

Here is an example using a subcircuit:

```
*
* This is the circuit definition
X1 a b 0 divider
V1 a 0 pulse(0 1 0 .5µ .5µ 0 1µ)
* this is the definition of the subcircuit
.subckt divider n1 n2 n3r1 n1 n2 1k
r2 n2 n3 1k
.ends
.tran 3
.end
```

Which runs after expanding to

```
* Expand X1 into two resistor network
r:1:1 a b 1k
r:1:2 b 0 1k
*
v1 a 0 pulse(0 1 0 .5µ .5µ 0 1µ)
.tran 3µ
.end
```

Note that unique names based on the subcircuit name and the subcircuit definition element names are made for the circuit elements inserted by subcircuit expansion.

.TEMP -- Temperature Sweeps

This is an archaic form for the step command for temperature. It performs the simulation for each temperature listed.

The syntax

```
   .TEMP <T1> <T2> ...
```

is equivalent to

```
   .STEP TEMP LIST <T1> <T2> ...
```

.TF -- Find the DC Small Signal Transfer Function

This is an analysis mode that finds the DC small signal transfer function of a node voltage or branch current due to small variations of an independent source.

```
Syntax: .TF V(<node>[, <ref>]) <source>
        .TF I(<voltage source>) <source>
```

Examples:

```
   .TF V(out) Vin
   .TF V(5,3) Vin
   .TF I(Vload) Vin
```

.TRAN -- Perform a Nonlinear Transient Analysis

Perform a transient analysis. This is the most direct simulation of a circuit. It basically computes what happens when the circuit is powered up. Test signals are often applied as independent sources.

```
Syntax: .TRAN <Tstep> <Tstop> [Tstart [dTmax]] [modifiers]
        .TRAN <Tstop> [modifiers]
```

The first form is the traditional .tran SPICE command. Tstep is the plotting increment for the waveforms but is also used as an initial step-size guess. LTspice uses waveform compression, so this parameter is of little value and can be omitted or set to zero. Tstop is the duration of the simulation. Transient analyses always start at time equal to zero. However, if Tstart is specified, the waveform data between zero and Tstart is not saved. This is a means of managing the size of waveform files by allowing startup transients to be ignored. The final parameter dTmax, is the maximum time step to take while integrating the circuit equations. If Tstart or dTmax is specified, Tstep must be specified.

Several modifiers can be placed on the .tran line.

.WAVE -- Write Selected Nodes to a .Wav File

LTspice can write .wav audio files. These files can then be listened to or be used as the input of another simulation.

Syntax: .wave <filename.wav> <Nbits> <SampleRate> V(out) [V(out2) ...]

Example: .wave C:\output.wav 16 44.1K V(left) V(right)

<filename.wav> is either a complete absolute path for the .wav file you wish to create or a relative path computed from the directory containing the simulation schematic or netlist. Double quotes may be used to specify a path containing spaces. <Nbits> is the number of sampling bits. The valid range is from 1 to 32 bits.
<SampleRate> is the number of samples to write per simulated second. The valid range is 1 to 4294967295 samples be second. The remainder of the syntax lists the nodes that you wish to save. Each node will be an independent channel in the .wav file. The number of channels may be as few as one or as many as 65535. It is possible to write a device current, e.g., Ib(Q1) as well as node voltage. The .wav analog to digital converter has a full scale range of -1 to +1 Volt or Amp.

Note that it is possible to write .wav files that cannot be played on your PC sound system because of the number of channels, sample rate or number of bits due to limitations of your PC's codec. But these .wav files may still be used in LTspice as input for another simulation. See the sections LTspice=>Circuit Elements=>V. Voltage Source and I. Current source for information on playing a .wav file into an LTspice simulation. If you want to play the .wav file on your PC sound card, keep in mind that the more popularly supported .wav file formats have 1 or 2 channels; 8 or 16 bits/channel; and a sample rate of 11025, 22050, or 44100 Hz.

.TRAN Modifiers

UIC: Skip the D.C. operating solution and use user-specified initial conditions.

steady: Stop the simulation when steady state has been reached.

nodiscard: Don't delete the part of the transient simulation before steady state is reached.

startup: Solve the initial operating point with independent voltage and current sources turned off. Then start the transient analysis and turn these sources on in the first 20 us of the simulation.

step: Compute the step response of the circuit.

UIC

Use Initial Conditions. Normally, a DC operating point analysis is performed before starting the transient analysis. This directive suppresses this initialization. The initial conditions of some circuit elements can be can be specified on an instance-per-instance basis. Uic is not a particularly recommended feature of SPICE. Skipping the DC operating point analysis leads to a nonphysical initial condition. For example, consider a voltage source connected in parallel to a capacitance. The node voltage is taken as zero if not specified. Then, in the first time step, an infinite current is required to charge the capacitor. The simulator cannot find a short enough time step to make the current nonsingular, and a "time step too small convergence fail" message is issued.

startup

This is similar to SPICE's original "uic". It means that independent sources should be ramped on during the first 20µs of the simulation. However, a DC operating point analysis is performed using the constraints specified on a .ic directive.

steady

Stop the simulation when steady state has been reached. This is required for an efficiency calculation report. Steady state detection is written into the SMPS macromodels. Typically they are written to look for zero error amp output current averaged over a clock cycle. The algorithm takes the error amp's output compliance range into consideration. The fraction of peak current that is considered zero current is specified with the sstol option.

The automatic steady state detection can fail either by being too critical or not critical enough. You can interactively specify steady state in the following manner: As soon as the simulation starts, execute menu command Simulate=>Efficiency Calculation=>Mark Start. The first time you execute this command you tell LTspice you're going to manually specify the integration limits. After the circuit looks like it's reached steady-state, execute that command again. That will clear the history and restart the Efficiency Calculation. Then, after awhile, as in you see well more than 10 clock cycles, execute Simulate=>Efficiency Calculation=>Mark End. Each time you execute Simulate=>Efficiency Calculation=>Mark Start you restart the efficiency calculation and clear the waveform history. This is a good method of preventing the data file from becoming too large and slowing down plotting, so it's recommended that you periodically execute Simulate=>Efficiency Calculation=>Mark Start whenever it is clear that you've accumulated substantial data that you don't want to be included in the integration of efficiency.

Use the .ic directive to specify node voltages and inductor currents to reduce the length of the transient analysis required to find the steady state.

nodiscard

Don't delete the part of the transient simulation before steady state is reached.

step

Compute the step response of the circuit. This function works with a current source used as a load with a list of step currents. The procedure is:

1. compute to steady state and discard the history unless nodiscard is set.

2. ramp the step load to the next value in the list of currents at the rate of 20A/µs.

3. compute to steady state

4. change the step load to the next value in the list or quit if there is none.

Due to the circuit complexity, the automatic STEP transition might not be detectable. Under this circumstance, it is best to use the .TRAN command to run the transient simulation and observe the starting and ending periods of the desired step load response. Use PWL command to program the output load current and switches to different levels at desired time periods. For example:

```
PWL(0 0.5 1m 0.5 1.01m 0.1 3m 0.1 3.01m 0.5)
```

The load current starts with 0.5A at time 0, stays at 0.5A at 1ms, switches to 0.1A at time 1.01ms, stays at 0.1A until 3ms, and switches to 0.5A at 3.01ms and stays at 0.5A.

The PWL can have almost unlimited pairs of (time, value) sequence.

Circuit Element Quick Reference

Component	Syntax
Special functions	Axx n1 n2 n3 n4 n5 n6 n7 n8 + <model> [extra parameters]
Arbitrary behavioral source	Bxx n+ n- <V=... or I=...>
Capacitor	Cxx n+ n- <capacitance> + [ic=<val.>] [Rser=<val.>] + [Lser=<val.>] [Rpar=<val.>] + [Cpar=<val.>] [m=<val.>]
Diode	Dxx A K <model> [area]
Voltage dependent voltage	Exx n+ n- nc+ nc- <gain>
Current dependent current	Fxx n+ n- <Vnam> <gain>
Voltage dependent current	Gxx n+ n- nc+ nc- <transcond.>
Current dependent voltage	Hxx n+ n- <Vnam> <transres.>
Independent current source	Ixx n+ n- <current>
JFET transistor	Jxx D G S <model> [area] [off] +[IC=<Vds,Vgs>] [temp=<T>]
Mutual inductance	Kxx L1 L2 L3... <coeff.>
Inductance	Lxx n+ n- <inductance> + [ic=<val.>] [Rser=<val.>] + [Rpar=<val.>] + [Cpar=<val.>] [m=<val.>]
MOSFET transistor	Mxx D G S B <model> [L=<len>] + [W=<width>] [AD=<area>] + [AS=<area>] [PD=<perim>] + [PS=<perim>] [NRD=<value>] + [NRS=<value>] [off] + [IC=<Vds, Vgs, Vbs> + [temp=<T>]
Lossy transmission line	Oxx L+ L- R+ R- <model>
Bipolar transistor	Qxx C B E [S] <model> [area] + [off] [IC=Vbe,Vce][temp=<T>]
Resistor	Rxx n1 n2 <value>
Voltage controlled	Sxx n1 n2 nc+ nc- <model>

switch	+ [on,off]
Lossless transmission line	Txx L+ L- R+ R- ZO=<value> + TD=<value>
Uniform RC-line	Uxx n1 n2 ncommon <model> + L=<len> [N=<lumps>]
Independent voltage source	Vxx n+ n- <voltage>
Current controlled switch	Wxx n1 n2 <Vnam> <model> + [on,off]
Subcircuit	Xxx n1 n2 n3... <subckt name>
MESFET or IGBT transistors	Zxx D G S model [area] [off] + [IC=<Vds,Vgs>]

A. Special Functions

Symbol names: INV, BUF, AND, OR, XOR, SCHMITT, SCHMTBUF, SCHMTINV, DFLOP, VARISTOR, and MODULATE

Syntax: Annn n001 n002 n003 n004 n005 n006 n007 n008 <model> [instance parameters]

These are Linear Technology Corporation's proprietary special function/mixed mode simulation devices. Most of these and their behavior are undocumented as they frequently change with each new set of models available for LTspice. However, here we document some of them because of their general interest.

INV, BUF, AND, OR, and XOR are generic idealized behavioral gates. All gates are netlisted with eight terminals. These gates require no external power. Current is sourced or sunk from the complementary outputs, terminals 6 and 7, and returned through device common, terminal 8. Terminals 1 through 5 are inputs. Unused inputs and outputs are to be connected to terminal 8. The digital device compiler recognizes that as a flag that that terminal is not used and removes it from the simulation. This leads to the potentially confusing situation where AND gates act differently when an input is grounded or at zero volts. If ground is the gate's common, then the grounded input is not at a logic false condition, but simply not part of the simulation. The reason that these gates are implemented like that is that this allows one device to act as 2-, 3-, 4- or 5- input gates with true, inverted, or complementary output with no simulation speed penalty for unused terminals. That is, the AND device acts as 12 different types of AND gates. The gates default to 0V/1V logic with a logic threshold of .5V, no propagation delay, and a 1Ohm output impedance. Output characteristics are set with these instance parameters:

Name	Default	Description
Vhigh	1	Logic high level
Vlow	0	Logic low level
Trise	0	Rise time
Tfall	Trise	Fall time
Tau	0	Output RC time constant
Cout	0	Output capacitance
Rout	1	Output impedance
Rhigh	Rout	Logic high level impedance

Rlow	Rout	Logic low level impedance

Note that not all parameters can be specified on the same instance at the same time, e.g., the output characteristics are either a slewing rise time or an RC time constant, not both.

The propagation delay defaults to zero and is set with instance parameter Td. Input hold time is equal to the propagation delay.

The input logic threshold defaults to .5*(Vhigh+Vlow) but can be set with the instance parameter Ref. The hold time is equal to the propagation delay.

The exclusive XOR device has non-standard behavior when more than two inputs are used: The output is true only when exactly one of all inputs is true. Use the associative property of XOR's with multiple XOR devices to implement an XOR block with more than two inputs.

The Schmitt trigger devices have similar output characteristics as the gates. Their trip points are specified with instance parameters Vt and Vh. The low trip point is Vt-Vh and the high trip point is Vt+Vh.

The gates and Schmitt trigger devices supply no timestep information to the simulation engine by default. That is, they don't look when they are about to change state and make sure there's a timestep close to either side of the state change. The instance parameter tripdt can be set to stipulate a maximum timestep size the simulator takes across state changes.

The VARISTOR is a voltage controlled varistor. Its breakdown voltage is set by the voltage between terminals 1 and 2. Its breakdown impedance is specified with the instance parameter rclamp. See the example schematic .\examples\Educational\varistor.asc

The MODULATE device is a voltage controlled oscillator. See the example schematic .\examples\Educational\PLL.asc. The instantaneous oscillation frequency is set by the voltage on the FM input. The conversion from voltage to frequency is linear and set by the two instance parameters, mark and space. Mark is the frequency when the FM input is at 1V and space is the frequency when the input is at 0V. The amplitude is set by the voltage on the AM input and defaults to 1V if that input is unused(connected to the MODULATE common).

The schematic capture aspect of LTspice netlists symbols for these devices in a special manner. All unconnected terminals are automatically connected to terminal 8. Also, if terminal 8 is unconnected, then it is connected to node 0.

B. Arbitrary Behavioral Voltage or Current Sources

Symbol names: BV, BI

```
Syntax: Bnnn n001 n002 V=<expression> [ic=<value>]
        + [tripdv=<value>] [tripdt=<value>]
        + [laplace=<expression> [window=<time>]
        + [nfft=<number>] [mtol=<number>]]

        Bnnn n001 n002 I=<expression> [ic=<value>]
        + [tripdv=<value>] [tripdt=<value>] [Rpar=<value>]
        + [laplace=<expression> [window=<time>]
        + [nfft=<number>] [mtol=<number>]]
```

The first syntax specifies a behavioral voltage source and the next is a behavioral current source. For the current source, a parallel resistance may be specified with the Rpar instance parameter.

Tripdv and tripdt control step rejection. If the voltage across a source changes by more than tripdv volts in tripdt seconds, that simulation time step is rejected.

Expressions can contain the following:

- Node voltages, e.g., V(n001)
- Node voltage differences, e.g., V(n001, n002)
- Circuit element currents; for example, I(S1), the current through switch S1 or Ib(Q1), the base current of Q1. However, it is assumed that the circuit element current is varying quasi-statically, that is, there is no instantaneous feedback between the current through the referenced device and the behavioral source output. Similarly, any ac component of such a device current is assumed to be zero in a small signal linear .AC analysis.
- The keyword, "time" meaning the current time in the simulation.
- The keyword, "pi" meaning 3.14159265358979323846.
- The following functions:

Function Name	**Description**
abs(x)	Absolute value of x

absdelay(x,t[,tmax])	x delayed by t. Optional max delay notification tmax.
acos(x)	Real part of the arc cosine of x, e.g., acos(-5) returns 3.14159, not 3.14159+2.29243i
arccos(x)	Synonym for acos()
acosh(x)	Real part of the arc hyperbolic cosine of x, e.g., acosh(.5) returns 0, not 1.0472i
asin(x)	Real part of the arc sine of x, asin(-5) is -1.57080, not -1.57080+2.29243i
arcsin(x)	Synonym for asin()
asinh(x)	Arc hyperbolic sine
atan(x)	Arc tangent of x
arctan(x)	Synonym for atan()
atan2(y,x)	Four quadrant arc tangent of y/x
atanh(x)	Arc hyperbolic tangent
buf(x)	1 if x > .5, else 0
ceil(x)	Integer equal or greater than x
cos(x)	Cosine of x
cosh(x)	Hyperbolic cosine of x
ddt(x)	Time derivative of x
delay(x,t[,tmax]	Same as absdelay()
dnlim(x,y,z)	Similar to max(x,y) but with a continuous 1st derivative transition width z
exp(x)	e to the x
floor(x)	Integer equal to or less than x
hypot(x,y)	sqrt(x**2 + y**2)
idt(x[,ic[,a]])	Integrate x, optional initial condition ic, reset if a is true.
idtmod(x[,ic[,m[,o]]]	Integrate x, optional initial condition ic, reset on reaching modulus m, offset output by o.
if(x,y,z)	If x > .5, then y else z
int(x)	Convert x to integer
inv(x)	0. if x > .5, else 1.
limit(x,y,z)	Intermediate value of x, y, and z
ln(x)	Natural logarithm of x

log(x)	Alternate syntax for ln()
log10(x)	Base 10 logarithm
max(x,y)	The greater of x or y
min(x,y)	The smaller of x or y
pow(x,y)	Real part of x**y, e.g., pow(-1,.5)=0, not i.
pwr(x,y)	abs(x)**y
pwrs(x,y)	sgn(x)*abs(x)**y
rand(x)	Random number between 0 and 1 depending on the integer value of x.
random(x)	Similar to rand(), but smoothly transitions between values.
round(x)	Nearest integer to x
sdt(x[,ic[,assert]])	Alternate syntax for idt()
sgn(x)	Sign of x
sin(x)	Sine of x
sinh(x)	Hyperbolic sine of x
sqrt(x)	Square root of x
table(x,a,b,c,d,...)	Interpolate a value for x based on a look up table given as a set of pairs of points.
tan(x)	Tangent of x.
tanh(x)	Hyperbolic tangent of x
u(x)	Unit step, i.e., 1 if x > 0., else 0.
uplim(x,y,z)	Similar to min(x,y) but with a continuous 1st derivative transition width z
uramp(x)	x if x > 0., else 0.
white(x)	Random number between -.5 and .5 smoothly transitions between values even more smoothly than random().
!(x)	Alternative syntax for inv(x)
~(x)	Alternative syntax for inv(x)

- The following operations, grouped in reverse order of precedence of evaluation:

Operand	**Description**
&	Convert the expressions to either side to Boolean, then AND.

|	Convert the expressions to either side to Boolean, then OR.
^	Convert the expressions to either side to Boolean, then XOR.
>	True if expression on the left is greater than the expression on the right, otherwise false.
<	True if expression on the left is less than the expression on the right, otherwise false.
>=	True if expression on the left is greater than or equal the expression on the right, otherwise false.
<=	True if expression on the left is less than or equal the expression on the right, otherwise false.
+	Floating point addition
-	Floating point subtraction
*	Floating point multiplication
/	Floating point division
**	Raise left hand side to power of right hand side. Only the real part is returned, e.g., -1**1.5 gives zero not i.
!	Convert the following expression to Boolean and invert.

True is numerically equal to 1 and False is 0. Conversion to Boolean converts a value to 1 if the value is greater than 0.5, otherwise the value is converted to 0.

Note that LTspice uses the caret character, ^, for Boolean XOR and "**" for exponentiation. Also, LTspice distinguishes between exponentiation, x**y, and the function pwr(x,y). Some 3rd party simulators have an incorrect implementation of behavioral exponentiation, evaluating -3**3 incorrectly to 27 instead of -27, presumably in the interest of avoiding the problem of exponentiating a negative number to a non-integer power. LTspice handles this issue by returning the real part of the result of the exponentiation. E.g., -2**1.5 evaluates to zero which is the real part of the correct answer of 2.82842712474619i. This means that when you import a 3rd party model that was targeted at a 3rd party simulator, you may need to translate the syntax such as x^y to x**y or even pwr(x,y).

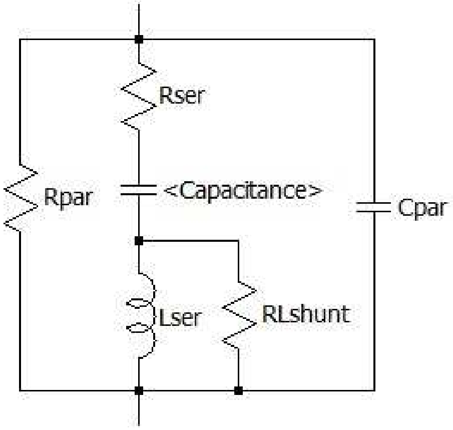
If an optional Laplace transform is defined, that transform is applied to the result of the behavioral current or voltage. The Laplace transform must be a function solely of s. The Boolean XOR operator, ^, is understood to mean exponentiation, **, when used in a Laplace expression. The frequency response at frequency f is found by substituting s with sqrt(-1)*2*pi*f. The time domain behavior is found from the sum of the instantaneous current(or voltage) with the convolution of the history of this current(or voltage) with the impulse response. Numerical inversion of a Laplace transfer function to the time domain impulse response is a potentially compute-bound process and a topic of current numerical research. In LTspice, the impulse response is found from the FFT of a discrete set points in frequency domain response. This process is prone to the usual artifacts of FFT's such as spectral leakage and picket fencing that is common to discrete FFT's. LTspice uses a proprietary algorithm that exploits that it has an exact analytical expression for the frequency domain response and chooses points and windows to cause such artifacts to diffract precisely to zero. However, LTspice must guess an appropriate frequency range and resolution. It is recommended that the LTspice first be allowed to make a guess at this. The length of the window and number of FFT data points used will be reported in the .log file. You can then adjust the algorithm's choices by explicitly setting nfft and window length. The reciprocal of the value of the window is the frequency resolution. The value of nfft times this resolution is the highest frequency considered. Note that the convolution of the impulse response with the behavioral source is also potentially a compute bound process.

C. Capacitor

Symbol names: CAP, POLCAP

```
Syntax: Cnnn n1 n2 <capacitance> [ic=<value>]
+ [Rser=<value>] [Lser=<value>] [Rpar=<value>]
+ [Cpar=<value>] [m=<value>]
+ [RLshunt=<value>] [temp=<value>]
```

It is possible to specify an equivalent series resistance, series inductance, parallel resistance and parallel shut capacitance. The equivalent circuit is given below:



Capacitor Instance Parameters

Name	Description
Rser	Equivalent series resistance
Lser	Equivalent series inductance
Rpar	Equivalent parallel resistance
Cpar	Equivalent parallel capacitance
RLshunt	Shunt resistance across Lser
m	Number of parallel units
temp	Instance temperature(for tempcos in a corresponding .model statement)
ic	Initial voltage(used only if uic is flagged on the .tran card)

It is computationally better to include the parasitic Rpar, Rser, RLshunt, Cpar and Lser in the capacitor than to explicitly draft them. LTspice uses proprietary circuit simulation technology to simulate this model of a physical capacitor without any internal nodes. This makes the simulation matrix smaller, faster to solve, and less likely to be singular at short time steps.

Note that since the capacitor element includes these parasitics, it is useful for macromodeling the fundamental of a piezoelectric crystal.

There is also a general nonlinear capacitor available. Instead of specifying the capacitance, one writes an expression for the charge.

LTspice will compile this expression and symbolically differentiate it with respect to all the variables, finding the partial derivative's that correspond to capacitances.

```
Syntax: Cnnn n1 n2 Q=<expression> [ic=<value>] [m=<value>]
```

There is a special variable, x, that means the voltage across the device. Therefore, a 100pF constant capacitance can be written as

```
Cnnn n1 n2 Q=100p*x
```

A capacitance with an abrupt change from 100p to 300p at zero volts can be written as

```
Cnnn n1 n2 Q=x*if(x<0,100p,300p)
```

This device is useful for rapidly evaluating the behavior of a new a hypothetical charge model for, e.g., a transistor.

D. Diode

Symbol Names: DIODE, ZENER, SCHOTTKY, VARACTOR, LED, TVS.

Syntax: Dnnn anode cathode <model> [area] [off] [m=<val>] [n=<val>] [temp=<value>]

Examples:

```
D1 SW OUT MyIdealDiode
```

```
.model MyIdealDiode D(Ron=.1 Roff=1Meg Vfwd=.4)
```

```
D2 SW OUT dio2
```

```
.model dio2 D(Is=1e-10)
```

Instance parameter M sets the number of parallel devices while instance parameter N sets the number of series devices.

A diode requires a .model card to specify its characteristics. There are two types of diodes available. One is a conduction region-wise linear model that yields a computationally light weight representation of an idealized diode. It has three linear regions of conduction: on, off and reverse breakdown. Forward conduction and reverse breakdown can non-linear by specifying a current limit with Ilimit(revIlimit). tanh() is used to fit the slope of the forward conduction to the limit current. The parameters epsilon and revepsilon can be specified to smoothly switch between the off and conducting states. A quadratic function is fit between the off and on state such that the diode's I-V curve is continuous in value and slope and the transition occurs over a voltage specified by the value of epsilon for the off to forward conduction and revepsilon for the transition between off and reverse breakdown.

Below are the model parameters for this type of diode:

Name	Description	Units	Default
Ron	Resistance in forward conduction	Ω	1.0
Roff	Resistance when off	Ω	1/Gmin
Vfwd	Forward threshold voltage to enter conduction	V	0.0
	Reverse breakdown		

Vrev	voltage	V	Infin.
Rrev	Breakdown impedance	Ω	Ron
Ilimit	Forward current limit	A	Infin.
Revilimit	Reverse current limit	A	Infin.
Epsilon	Width of quadratic region	V	0.0
Revepsilon	Width of reverse quad. region	V	0.0

This idealized model is used if any of Ron, Roff, Vfwd, Vrev or Rrev is specified in the model.

The other model available is the standard Berkeley SPICE semiconductor diode but extended to handle more detailed breakdown behavior and recombination current. The area factor determines the number of equivalent parallel devices of a specified model. Below are the diode model parameters for this diode.

Name	**Description**	**Units**	**Default**	**Example**
Is	saturation current	A	1e-14	1e-7
Rs	Ohmic resistance	Ω	0.0	10.
N	Emission coefficient	-	1.0	1.0
Tt	Transit-time	sec	0.0	2n
Cjo	Zero-bias junction cap.	F	0.0	2p
Vj	Junction potential	V	1.0	0.6
M	Grading coefficient	-	0.5	0.5
Eg	Activation energy	eV	1.11	1.11 Si 0.69 Sbd 0.67 Ge
Xti	Sat.-current temp. exp	-	3.0	3.0 jn 2.0 Sbd
Kf	Flicker noise coeff.	-	0.0	
Af	Flicker noise exponent	1.	1.0	
Fc	Coeff. for forward-bias depletion	-	0.5	

	capacitance formula			
BV	Reverse breakdown voltage	V	Infin.	40.
nbv	Reverse breakdown emission coefficient	-	1.0	2.0
Ibv	Current at breakdown voltage	A	1e-10	
Ibvl	Low-level reverse breakdown knee current	A	0.0	
nbvl	Low-level reverse breakdown emission coefficient	-	1.0	
Tnom	Parameter measurement temp.	°C	27	50
Isr	Recombination current parameter	A	0.0	
Nr	Isr emission coeff.	-	2.0	
Ikf	High-injection knee current	A	Infin.	
Tikf	Linear Ikf temp coeff.	/°C	0.0	
Trs1	linear Rs temp coeff.	/°C	0.0	
Trs2	Quadratic Rs temp coeff.	/°C^2	0.0	
Tbv1	Breakdown voltage temp coeff.	/°C	0.0	
Tbv2	Quadratic breakdown voltage temp coeff.	/°C^2	0.0	
Perim	Default perimeter	m	0.0	
Isw	Sidewall Is	A	0.0	
ns	Sidewall emission coefficient	-	1.0	
Rsw	Sidewall series resistance	Ω	0.0	
Cjsw	Sidewall Cjo	F	0.0	
Vjsw	Sidewall Vj	V	Vj	

mjsw	Sidewall mj	-	0.33	
Fcs	Sidewall Fc	-	Fc	
Vp	Soft reverse recovery parameter	-	0.0	0.65

The soft reverse recovery parameter, Vp, adds a dQ/dt damping to diode charge as suggested by K.J. Teng and S. Pan in 'Modified charge-control equation for simulation of diode reverse recovery', Electronics Letters, 15th February 1996 Vol. 32 No. 4.

It is possible to annotate a model with voltage and current ratings. This information is displayed in the schematic capture GUI to assist in selecting a device but does not directly impact the electrical behavior in simulation. The following parameters may be specified.

Name	**Description**	**Units**
Vpk	Peak voltage rating	V
Ipk	Peak current rating	A
Iave	Ave current rating	A
Irms	RMS current rating	A
diss	Maximum power dissipation rating	W

E. Voltage Dependent Voltage Source

Symbol Names: E, E2

There are three types of voltage-dependent voltage-source circuit elements.

```
Syntax: Exxx n+ n- nc+ nc- <gain>
```

This circuit element asserts an output voltage between the nodes n+ and n- that depends on the input voltage between nodes nc+ and nc-. This is a linearly dependent source specified solely by a constant gain.

```
Syntax: Exxx n+ n- nc+ nc- table=(<value pair>, <value pair>, ...)
```

A look-up table is used to specify the transfer function. The table is a list of pairs of numbers. The second value of the pair is the output voltage when the control voltage is equal to the first value of that pair. The output is linearly interpolated when the control voltage is between specified points. If the control voltage is beyond the range of the look-up table, the output voltage is extrapolated as a constant voltage of the last point of the look-up table.

```
Syntax: Exxx n+ n- nc+ nc- Laplace=<func(s)>
+ [window=<time>] [nfft=<number>] [mtol=<number>]
```

The transfer function of this circuit element is specified by its Laplace transform. The Laplace transform must be a function of s. The frequency response at frequency f is found by substituting s with sqrt(-1)*2*pi*f. The time domain behavior is found from the impulse response found from the Fourier transform of the frequency domain response. LTspice must guess an appropriate frequency range and resolution. The response must drop at high frequencies or an error is reported. It is recommended that LTspice first be allowed to make a guess at this and then check the accuracy by reducing reltol or explicitly setting nfft and the window. The reciprocal of the value of the window is the frequency resolution. The value of nfft times this resolution is the highest frequency considered. The Boolean XOR operator, "^" is understood to mean exponentiation "**" when used in a Laplace expression.

```
Syntax: Exxx n+ n- value={<expression>}
```

This is an alternative syntax of the behavioral source, arbitrary behavioral voltage source, B.

```
Syntax: Exxx n+ n- POLY(<N>) <(node1+,node1-) (node2+,node2-)+ ...
```

```
(nodeN+,nodeN-)> <c0 c1 c2 c3 c4 ...>
```

This is an archaic means of arbitrary behavioral modeling with a polynomial. It is useful for running legacy opamp models.

Note: It is better to use a G source shunted with a resistance to approximate an E source than to use an E source. A voltage controlled current source shunted with a resistance will compute faster and cause fewer convergence problems than a voltage controlled voltage source. Also, the resultant nonzero output impedance is more representative of a practical circuit.

F. Current Dependent Current Source

Symbol Name: F

Syntax: Fxxx n+ n- <Vnam> <gain>

This circuit element applies a current between nodes n+ and n-. The current applied is equal to the value of the gain times the current through the voltage source specified as <Vnam>.

Syntax: Fxxx n+ n- value={<expression>}

This is an alternative syntax of the behavioral source, arbitrary behavioral voltage source, B.

Syntax: Fxxx n+ n- POLY(<N>) <V1 V2 ... VN> <c0 c1 c2 c3 c4 ...>

This is an archaic means of arbitrary behavioral modeling with a polynomial. It is useful for running legacy behavioral models.

G. Voltage Dependent Current Source

Symbol Names: G, G2

There are three types of voltage dependent current-source circuit elements.

Syntax: Gxxx n+ n- nc+ nc- <gain>

This circuit element asserts an output current between the nodes n+ and n- that depends on the input voltage between nodes nc+ and nc-. This is a linearly dependent source specified solely by a constant gain.

Syntax: Gxxx n+ n- nc+ nc- table=(<value pair>, <value pair>, ...)

Here a lookup table is used to specify the transfer function. The table is a list of pairs of numbers. The second value of the pair is the output current when the control voltage is equal to the first value of that pair. The output is linearly interpolated when the control voltage is between specified points. If the control voltage is beyond the range of the look-up table, the output current is extrapolated as a constant current of the last point of the look-up table.

Syntax: Gxxx n+ n- nc+ nc- Laplace=<func(s)> [window=<time>] [nfft=<number>] [mtol=<number>]

The transfer function of this circuit element is specified by its Laplace transform. The Laplace transform must be a function of s. The frequency response at frequency f is found by substituting s with sqrt(-1)*2*pi*f. The time-domain behavior is found from the impulse response, which is found from the Fourier transform of the frequency-domain response. LTspice must guess an appropriate frequency range and resolution. The response must drop at high frequencies or an error is reported. It is recommended that the LTspice first be allowed to make a guess at this and then check the accuracy my reducing reltol or explicitly setting nfft and window. The reciprocal of the value of window is the frequency resolution. The value of nfft times this resolution is the highest frequency considered. The Boolean XOR operator, "^" is understood to mean exponentiation "**" when used in a Laplace expression.

Syntax: Gxxx n+ n- value={<expression>}

This is an alternative syntax of the behavioral source, arbitrary behavioral voltage source, B.

Syntax: Gxxx n+ n- POLY(<N>) <(node1+,node1-) (node2+,node2-) ... (nodeN+,nodeN-)> <c0 c1 c2 c3 c4 ...>

This is an archaic means of arbitrary behavioral modeling with a polynomial. It is useful for running legacy behavioral models.

H. Current Dependent Voltage Source

Symbol Name: H

Syntax: Hxxx n+ n- <Vnam> <transresistance>

This circuit element applies a voltage between nodes n+ and n-. The voltage applied is equal to the value of the transresistance times the current through the voltage source <Vnam>.

Syntax: Hxxx n+ n- value={<expression>}

This is an alternative syntax of the behavioral source, arbitrary behavioral voltage source, B.

Syntax: Hxxx n+ n- POLY(<N>) <V1 V2 ... V3> <c0 c1 c2 c3 c4 ...>

This is an archaic means of arbitrary behavioral modeling with a polynomial.

I. Current Source

Symbol Name: CURRENT

Syntax: Ixxx n+ n- <current> [AC=<amplitude>] [load]

This circuit element sources a constant current between nodes n+ and n-. If the source is flagged as a load, the source is forced to be dissipative, that is, the current goes to zero if the voltage between nodes n+ and n- goes to zero or a negative value. The purpose of this option is to model a current load on a power supply that doesn't draw current if the output voltage is zero.

For AC analysis, the value of AC is used as the amplitude of the source at the analysis frequency.

Syntax: Ixxx n+ n- PULSE(Ioff Ion Tdelay Trise Tfall Ton Tperiod Ncycles)

Time-dependent pulsed current source

Name	Description	Units
Ioff	Initial value	A
Ion	Pulsed value	A
Tdelay	Delay	sec
Tr	Rise time	sec
Tf	Fall time	sec
Ton	On time	sec
Tperiod	Period	sec
Ncycles	Number of cycles(Omit for free-running pulse function)	cycles

Syntax: Ixxx n+ n- SINE(Ioffset Iamp Freq Td Theta Phi Ncycles)

Time-dependent sine wave current source.

Name	Description	Units
Ioffset	DC offset	A
Iamp	Amplitude	A
Freq	Frequency	Hz
Td	Delay	sec
Theta	Damping factor	1/sec

Phi	Phase of sine wave	degrees
Ncycles	Number of cycles(Omit for free-running pulse function)	cycles

For times less than Td or times after completing Ncycles, have run, the output current is given by Ioffset+Iamp*sin(pi*phi/180)Otherwise the current is given by

Ioffset+Iamp*exp(-(time-Td)*Theta)*sin(2*π*Freq*(time-Td)+π*phi/180)

The damping factor, Theta, is the reciprocal of the decay time constant.

Syntax: Ixxx n+ n- EXP(I1 I2 Td1 Tau1 Td2 Tau2)

Time-dependent exponential current source

Name	Description	Units
I1	Initial value	A
I2	Pulsed value	A
Td1	Rise delay time	sec
Tau1	Rise-time constant	sec
Td2	Fall delay time	sec
Tau2	Fall-time constant	sec

For times less than Td1, the output current is I1. For times between Td1 and Td2 the current is given by>

I1+(I2-I1)*(1-exp(-(time-Td1)/Tau1))

For times after Td2 the current is given by

I1+(I2-I1)*(1-exp(-(time-Td1)/Tau1))-(I2-I1)*(1-exp(-(time-Td2)/Tau2))

Syntax: Ixxx n+ n- SFFM(Ioff Iamp Fcar MDI Fsig)

Time-dependent single-frequency FM current source.

Name	Description	Units
Ioff	DC offset	A
Iamp	Amplitude	A

Fcar	Carrier frequency	Hz
MDI	Modulation index	-
Fsig	Signal frequency	Hz

The current is given by

Ioff+Iamp*sin((2.*π*Fcar*time)+MDI*sin(2.*π*Fsig*time)).

Syntax: Ixxx n+ n- tbl=(<voltage, current>, <voltage, current>, ...)

The current can also be specified as a function of the voltage across the output nodes with a look-up table. This is useful for modeling the characteristics of a load.

Syntax: Ixxx n+ n- <value> step(<value1>, [<value2>], [<value3>, ...]) [load]

This is a special form for the current source. The current is specified as a list of currents to use in a step load response transient analysis. In this mode, the simulation is computed until steady state is reached at the first current in the list, <value1>. Then the current is stepped to the next value in the list, <value2>. The simulation proceeds until steady state is achieved at that current. Then the current is stepped to the next value and the process repeats until the list is exhausted. If the .tran command doesn't specify "step", then the original <value> is used.

Syntax: Ixxx n+ n- R=<value>

This is not a current source at all, but a resistor. It is used to model a resistive load when the load is netlisted as a current source.

Syntax: Ixxx n+ n- PWL(t1 i1 t2 i2 t3 i3...)

Arbitrary Piece-wise linear current source.

For times before t1, the current is i1. For times between t1 and t2, the current varies linearly between i1 and i2. There can be any number of time, current points given. For times after the last time, the current is the last current.

Syntax: Ixxx n+ n- wavefile=<filename> [chan=<nnn>]

This allows a .wav file to be used as an input to LTspice.

<filename> is either a full, absolute path for the .wav file or a relative path computed from the directory containing the simulation schematic or netlist. Double quotes may be used to specify a path containing spaces. The .wav file may contain up to 65536 channels, numbered 0 to 65535. Chan may be set to specify which channel is used. By default, the first channel, number 0, is used. The .wav file is interpreted as having a full scale range from -1A to 1A.

This source only has meaning in a .tran analysis.

J. JFET Transistor

Symbol Names: NJF, PJF

Syntax: Jxxx D G S <model> [area] [off] [IC=Vds, Vgs] [temp=T]

Examples:

```
J1 0 in out MyJFETmodel
.model MyJFETmodel NJF(Lambda=.001)
```

```
J2 0 in out MyPJFETmodel
.model MyPJFETmodel PJF(Lambda=.001)
```

A JFET transistor requires a .model card to specify its characteristics. Note that the model card keywords NJF and PJF specify the polarity of the transistor. The area factor determines the number of equivalent parallel devices of a specified model.

The JFET model is derived from the FET model of Shichman and Hodges extended to include Gate junction recombination current and impact ionization. The DC characteristics are defined by the parameters VTO and BETA, 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. Two ohmic resistances, Rd and Rs, are included. Charge storage is modeled by nonlinear depletion layer capacitances for both gate junctions; which vary as the -1/2 power of junction voltage and are defined by the parameters Cgs, Cgd, and PB. A fitting parameter B has been added. See A. E. Parker and D. J. Skellern, An Improved FET Model for Computer Simulators, IEEE Trans CAD, vol. 9, no. 5, pp. 551-553, May 1990.

Name	Description	Units	Default	Example
Vto	Threshold voltage	V	-2.0	-2.0
Beta	Transconductance parameter	A/V^2	1e-4	1e-3
Lambda	Channel-length modulation parameter	1/V	0.0	1e-4
Rd	Drain ohmic resistance	Ω	0.0	100
Rs	Source ohmic resistance	Ω	0.0	100
Cgs	Zero-bias G-S junction	F	0.0	5p

	capacitance			
Cgd	Zero-bias G-D junction capacitance	F	0.0	1p
Pb	Gate junction potential	V	1.0	0.6
m	Gate junction grading coefficient	-	0.5	0.8
Is	Gate junction saturation current	A	1e-14	1e-14
B	Doping tail parameter	-	1.0	1.1
KF	Flicker noise coefficient	-	0.0	
Nlev	Noise equation selector	-	0.0	3.0
Gdsnoi	Shot noise coefficient for nlev=3	-	1.0	2.0
AF	Flicker noise exponent	-	1.0	
Fc	Coefficient for forward-depletion capacitance	-	0.5	
Tnom	Parameter measurement temperature	°C	27	50
BetaTce	Transconductance parameter exponential temperature coefficient	%/°C	0.0	
VtoTc	Threshold voltage temperature coefficient	V/°C	0.0	
N	Gate junction emission coefficient	-	1.0	
Isr	Gate junction recombination current parameter	A	0.0	
	Emission			

Nr	coefficient for Isr	-	2.0	
alpha	Ionization coefficient	1/V	0.0	
Vk	Ionization knee voltage	V	0.0	
Xti	Saturation current temperature coefficient	-	3.0	
mfg	Annotation of manufacturer	-	3.0	ACME Semi Ltd.

K. Mutual Inductance

Symbol Names: None, this is placed as text on the schematic.

Syntax: Kxxx L1 L2 [L3 ...] <coefficient>

L1 and L2 are the names of inductors in the circuit. The mutual coupling coefficient must be in the range of -1 to 1.

The line

```
   K1 L1 L2 L3 L4 1.
```

is synonymous with the six lines

```
   K1 L1 L2 1.
   K2 L2 L3 1.
   K3 L3 L4 1.
   K4 L1 L3 1.
   K5 L2 L4 1.
   K6 L1 L4 1.
```

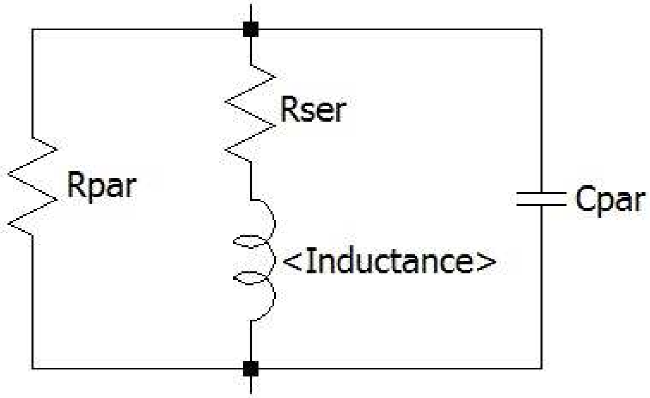
It is recommended to start a design with a mutual coupling coefficient equal to 1. This will eliminate leakage inductance that can ring at extremely high frequencies if damping is not supplied and slow the simulation.

L. Inductor

Symbol Names: IND, IND2

```
Syntax: Lxxx n+ n- <inductance> [ic=<value>]
+ [Rser=<value>] [Rpar=<value>]
+ [Cpar=<value>] [m=<value>] [temp=<value>]
```

It is possible to specify an equivalent series resistance, series inductances, parallel resistance and parallel shut capacitance. The equivalent circuit is given below:



Inductor Instance Parameters

Name	Description
Rser	Equivalent series resistance
Rpar	Equivalent parallel resistance
Cpar	Equivalent parallel capacitance
m	Number of parallel units
ic	Initial current(used only if uic flagged on the .tran card)
tc1	Linear inductance temperature coeff.
Tc1	Quadratic inductance temperature coeff.
temp	Instance temp

It is better to include the device parasitics Rpar, Rser, and Cpar in the inductor than to explicitly draft them. LTspice uses proprietary circuit simulation technology to simulate this physical inductor without any internal nodes. This makes the simulation matrix smaller, faster to compute and less likely to be singular over all time-step sizes.

By default, LTspice will supply losses to inductors to aid SMPS transient analysis. For SMPS, these losses are of usually of no consequence, but may be turned off if desired. On the "Tools=> Control Panel=>Hacks!" page, uncheck "Supply a min. inductor damping if no Rpar is given." This setting will be remembered between invocations of the program. There is also a default series resistance of 1 milliohm for inductors that aren't mentioned in a mutual inductance statement. This Rser allows LTspice XVII to integrate the inductance as a Norton equivalent circuit instead of Thevenin equivalent in order to reduce the size of the circuit's linearized matrix. If you don't want LTspice to introduce this minimum resistance, you must explicitly set Rser=0 for that inductor. This will require LTspice to use the more cumbersome Thevenin equivalent of the inductor during transient analysis.

There are two forms of non-linear inductors available in LTspice. One is a behavioral inductance specified with an expression for the flux. The inductor's current is referred to by the keyword "x" in the expression. Below is an example in a netlist.

```
*
L1 N001 0 Flux=1m*tanh(5*x)
I1 0 N001 PWL(0 0 1 1)
.tran 1
.end
```

In the above example, I1 supplies a unity dI/dT so that the inductance can be read off as the voltage on node N001.

There other non-linear inductor available in LTspice is a hysteretic core model based on a model first proposed in by John Chan et la. in IEEE Transactions On Computer-Aided Design, Vol. 10. No. 4, April 1991 but extended with the methods in United States Patent 7,502,723. This model defines the hysteresis loop with only three parameters:

Name	**Description**	**Units**
Hc	Coercive force	Amp-turns/meter
Br	Remnant flux density	Tesla
Bs	Saturation flux density	Tesla

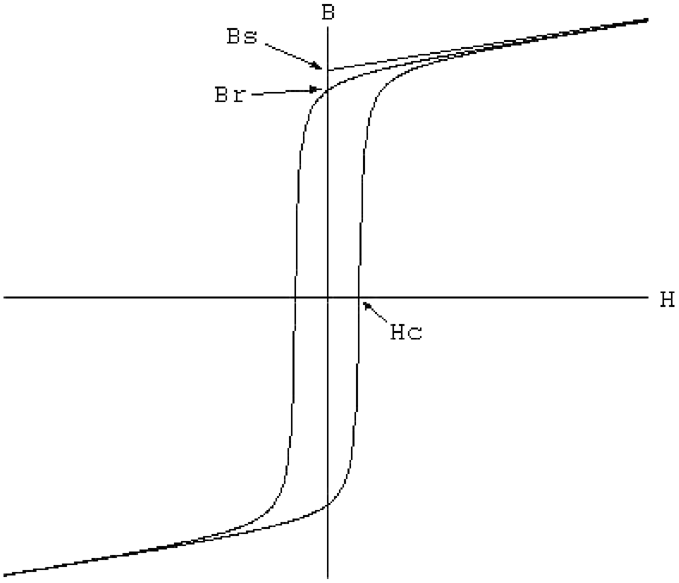
The upper and lower branches of the hysteresis major loop are given by

$$\mathbf{Bup(H) = Bs \cdot \frac{H + Hc}{|H+Hc| + Hc\cdot(Bs/Br-1)} + \mu0\cdot H}$$

and

$$\mathbf{Bdn(H) = Bs \cdot \frac{H - Hc}{|H-Hc| + Hc\cdot(Bs/Br-1)} + \mu0\cdot H}$$

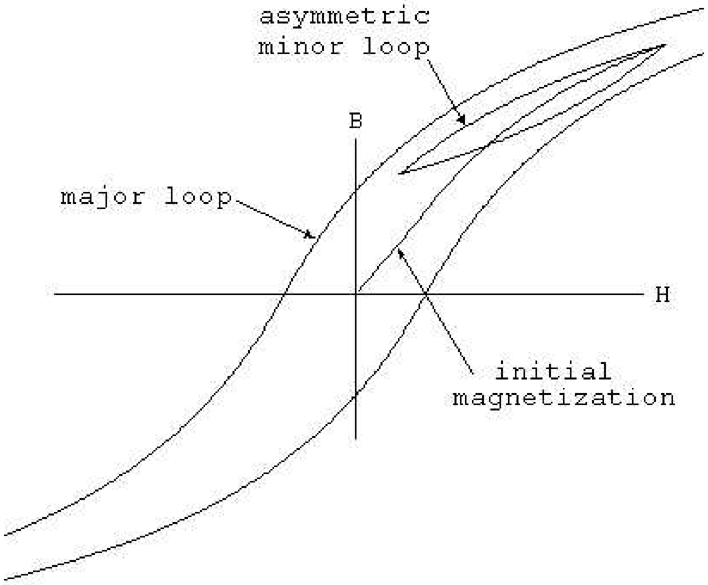
These functions are plotted in following figure. Hc and Br are the intersections of the major hysteresis loop with the H- and B-axes. Bs is the B-axis intersection of the asymptotic line, Bsat(H) = Bs + µ0 · H, approached as H goes to infinity.



The initial magnetization curve is given by

Bmag(H) = .5 · (Bup(H) + Bdn(H))

Minor loops are obtained by various translations of the above equations per the cited reference. The core's absolute and differential permeabilities are a function of H and the history of values of H. The plot below shows the path taken by an asymmetrical minor loop for a typical power ferrite(Hc=16 A-turns/m, Bs=.44T, Br= .10T).



In addition to the core property parameters Hc, Br, and Bs, mechanical dimensions of the core are required:

Name	Description	Units
Lm	Magnetic Length(excl. gap)	meter
Lg	Length of gap	meter
A	Cross sectional area	meter**2
N	Number of turns	-

Note that if specifying a non-zero gap the magnetic field, H, is not proportional to the current in the windings. LTspice solves for the magnetic fields in the core and gap under the assumption of uniform cross sectional area and thin or uniformly distributed gap.

Below is an example that shows inductance vs. current for L1, an inductor wound on a gapped core. You can read out the inductance as V(n001) since current source I1 supplies a unity dI/dt. The core follows the initial magnetization curve, so you can see that the permeability first increases from the initial value as the current is ramped and then drops as it saturates. Since the gap makes the inductance insensitive to the exact permeability of the core, you have to really zoom in on V(n001) to see that it does increase. The peak is when H inside the core is equal to its Hc.

```
*
L1 N001 0 Hc=16. Bs=.44 Br=.10 A=0.0000251
+ Lm=0.0198 Lg=0.0006858 N=1000
```

```
I1 0 N001 PWL(0 0 1 1)
.tran .5
.options maxstep=10u
.end
```

M. MOSFET

Symbol Names: NMOS, NMOS3, PMOS, PMOS3There are two fundamentally different types of MOSFETS in LTspice, monolithic MOSFETs and a new vertical double diffused power MOSFET model.

Monolithic MOSFET:

```
Syntax: Mxxx Nd Ng Ns Nb <model> [m=<value>] [L=<len>]
+ [W=<width>] [AD=<area>] [AS=<area>]
+ [PD=<perim>] [PS=<perim>] [NRD=<value>]
+ [NRS=<value>] [off] [IC=<Vds, Vgs, Vbs>]
+ [temp=<T>]
```

```
M1 Nd Ng Ns 0 MyMOSFET
.model MyMOSFET NMOS(KP=.001)
```

```
M1 Nd Ng Ns Nb MypMOSFET
.model MypMOSFET PMOS(KP=.001)
```

Vertical double diffused power MOSFET:

```
Syntax: Mxxx Nd Ng Ns <model> [L=<len>] [W=<width>]
+ [M=<area>] [m=<value>] [off]
+ [IC=<Vds, Vgs, Vbs>] [temp=<T>]
```

Example:

```
M1 Nd Ng Ns Si4410DY
.model Si4410DY VDMOS(Rd=3m Rs=3m Vto=2.6 Kp=60
+ Cgdmax=1.9n Cgdmin=50p Cgs=3.1n Cjo=1n
+ Is=5.5p Rb=5.7m)
```

The MOSFET's model card specifies which type is intended. The model card keywords NMOS and PMOS specify a monolithic N- or P-channel MOSFET transistor. The model card keyword VDMOS specifies a vertical double diffused power MOSFET.

Monolithic MOSFETS are four terminal devices. Nd, Ng, NS, and Nb are the drain, gate, source, and bulk; i.e., substrate; nodes. L and W are the channel length and width, in meters. AD and AS are the areas of the drain and source diffusions, in square meters. Note that the suffix u specifies µm and p square µm. If any of L, W, AD, or AS are not specified, default values are used. PD and PS are the perimeters of the drain and source junctions, in meters. NRD and NRS designate the equivalent number of squares of the drain and source diffusions; these values multiply the sheet resistance RSH specified on the .MODEL control line. PD and PS default to zero while NRD and NRS to one. OFF indicates an initial condition on the device for DC analysis. The initial condition specification using IC=VDS, VGS, VBS is for use with the UIC option on the .TRAN control line, when a transient analysis is desired starting from other than the quiescent operating point. The optional TEMP value is the temperature at which this device is to operate, and overrides the temperature specification on the .OPTION control line. The temperature specification is ONLY valid for level 1, 2, 3, and 6 MOSFETs, not for level 4, 5 or 8 BSIM devices.

LTspice contains seven different types of monolithic MOSFET's and one type of vertical double diffused Power MOSFET.

There are seven monolithic MOSFET device models. The model parameter LEVEL specifies the model to be used. The default level is one.

level	model
1	Shichman-Hodges
2	MOS2(see A. Vladimirescu and S. Liu, The Simulation of MOS Integrated Circuits Using SPICE2, ERL Memo No. M80/7, Electronics Research Laboratory University of California, Berkeley, October 1980)
3	MOS3, a semi-empirical model(see reference for level 2)
4	BSIM (see B. J. Sheu, D. L. Scharfetter, and P. K. Ko, SPICE2 Implementation of BSIM. ERL Memo No. ERL M85/42, Electronics Research Laboratory University of California, Berkeley, May 1985)
5	BSIM2 (see Min-Chie Jeng, Design and Modeling of Deep-Submicrometer MOSFETs ERL Memo Nos. ERL M90/90, Electronics Research Laboratory University of California, Berkeley, October 1990)
6	MOS6 (see T. Sakurai and A. R. Newton, A Simple MOSFET Model for Circuit Analysis and its application to CMOS gate delay analysis and series-connected MOSFET Structure, ERL Memo No. ERL M90/19, Electronics Research Laboratory, University of California, Berkeley, March 1990)
8	BSIM3v3.3.0 from University of California, Berkeley as of

July 29, 2005

9 BSIMSOI3.2 (Silicon on insulator) from the BSIM Research Group of the University of California, Berkeley, February 2004.

12 EKV 2.6 based on code from Ecole Polytechnique Federale de Lausanne. See http://legwww.epfl.ch/ekv and "The EPFL-EKV MOSFET Model Equations for Simulation, Version 2.6", M. Bucher, C. Lallement, F. Theodoloz, C. Enz, F. Krummenacher, EPFL-DE-LEG, June 1997.

14 BSIM4.6.1 from the University of California, Berkeley BSIM Research Group, May 18, 2007.

73 HiSIMHV version 1.2 from the Hiroshima University and STARC.

The DC characteristics of the level 1 through level 3 MOSFETs are defined by the device parameters VTO, KP, LAMBDA, PHI and GAMMA. These parameters are computed if the process parameters(NSUB, TOX,...) are given, but user-specified values always override. 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 which represent overlap capacitances, by the non-linear thin-oxide capacitance which is distributed among the gate, source, drain, and bulk regions, and by the nonlinear 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. Channel capacitance is implemented as a Yang-Chatterjee charge-based model in all aspects of simulation but legacy Meyer capacitances are reported in the SPICE log file. The thin-oxide charge-storage effects are treated slightly different for the Level=1 model. These voltage dependent capacitances are included only if Tox is specified.

There is some overlap among the parameters describing the junctions, e.g., the reverse current can be specified either through Is[Amp] or through Js[Amp/m/m]. Whereas the first is an absolute value the second is multiplied by Ad and As to give the reverse current of the drain and source junctions respectively. The same idea applies also to the zero-bias junction capacitances CBD and CBS[Farad] on one hand, and CJ[Farad/m/m] on the other. The parasitic drain and source series resistance can be expressed as either RD and RS[Ohms] or RSH[Ohms/square], the latter being multiplied by the number of squares NRD and NRS input on the device line.

MOSFET level 1, 2, and 3 parameters:

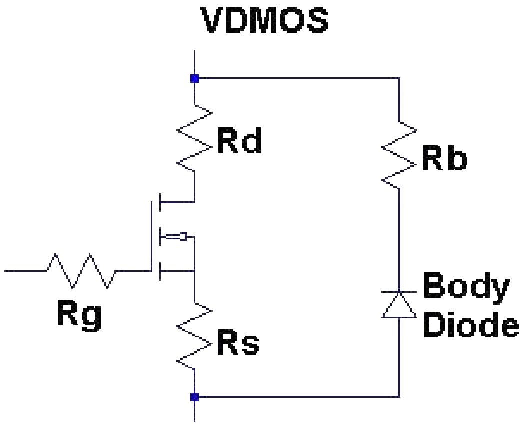
Name	Description	Units	Default	Example
Vto	Zero-bias threshold voltage	V	0	1.0
Kp	Transconductance parameter	A/V^2	2e-5	3e-5
Gamma	Bulk threshold parameter	$V^{½}$	0.	0.37
Phi	Surface inversion potential	V	0.6	0.65
Lambda	Channel-length modulation (level 1 and 2)	1/V	0.	0.02
wd	Lateral diffusion width reduction	m	0.	0.5u
Rd	Drain ohmic resistance	Ω	0.	1.
Rs	Source ohmic resistance	Ω	0.	1.
Rg	Gate ohmic resistance	Ω	0.	1.
Rb	Bulk ohmic resistance	Ω	0.	1.
Rds	Drain-Source shunt resistance	Ω	0.	1Meg
Cbd	Zero-bias B-D junction capacitance	F	0.	20f
Cbs	Zero-bias B-S junction capacitance	F	0.	20f
Is	Bulk junction saturation current	A	1e-14	1e-15
N	Bulk diode emission coefficient	-	1.	
Pb	Bulk junction potential	V	0.8	0.87
	Bulk junction			

tt	transit time	s	0	1n
Cgso	Gate-source overlap capacitance per meter channel width	F/m	0.	4e-11
Cgdo	Gate-drain overlap capacitance per meter channel width	F/m	0.	4e-11
Cgbo	Gate-bulk overlap capacitance per meter channel width	F/m	0.	2e-10
Rsh	Drain and source diffusion sheet resistance	Ω	0.	10.
Cj	Zero-bias bulk junction bottom capacitance per square meter of junction area	F/m^2	0.	2e-4
Mj	Bulk junction bottom grading coefficient	-	0.5	0.5
Cjsw	Zero-bias bulk junction sidewall capacitance per meter of junction perimeter	F/m	0.	1p
Mjsw	Bulk junction sidewall grading coefficient	-	.50 level 1 .33 level 2,3	
Js	Bulk junction saturation current per square-meter of junction area	A/m^2	0.	1u
Jssw	Bulk junction saturation current per meter of sidewall	A/m	0.	1n
Tox	Oxide thickness	m	1e-7	1e-7
Nsub	Substrate doping	1/cm^3	0.	4e15

Nss	Surface state density	1/cm^2	0.	1e+10
Nfs	Fast surface state (levels 2 & 3)	1/cm^2	0.	1e+10
Xd	Depletion layer width (level 3)	m	0.	100n
TPG	Type of gate material: +1 opp. to substrate -1 same as substrate 0 Al gate	-	1	
Xj	Metallurgical junction depth (levels 2 & 3)	m	0.	1µ
Ld	Lateral diffusion	m	0.	0.8µ
Uo	Surface mobility	cm^2/V/s	600	700
Ucrit	Critical field for mobility degradation (level 2)	V/cm	1e4	1e4
Uexp	Critical field exponent in mobility degradation (level 2)	-	0.	0.1
Utra	Transverse field coefficient (level 2)	-	0.	0.3
Vmax	Maximum carrier drift velocity (levels 2 & 3)	m/s	0.	5e4
Neff	Total channel-charge exponent (level 2)	-	1.	5.
Kf	Flicker noise coefficient	-	0.	1e-26
Af	Flicker noise exponent	-	1.	1.2
Nlev	Noise equation selector	-	0	1

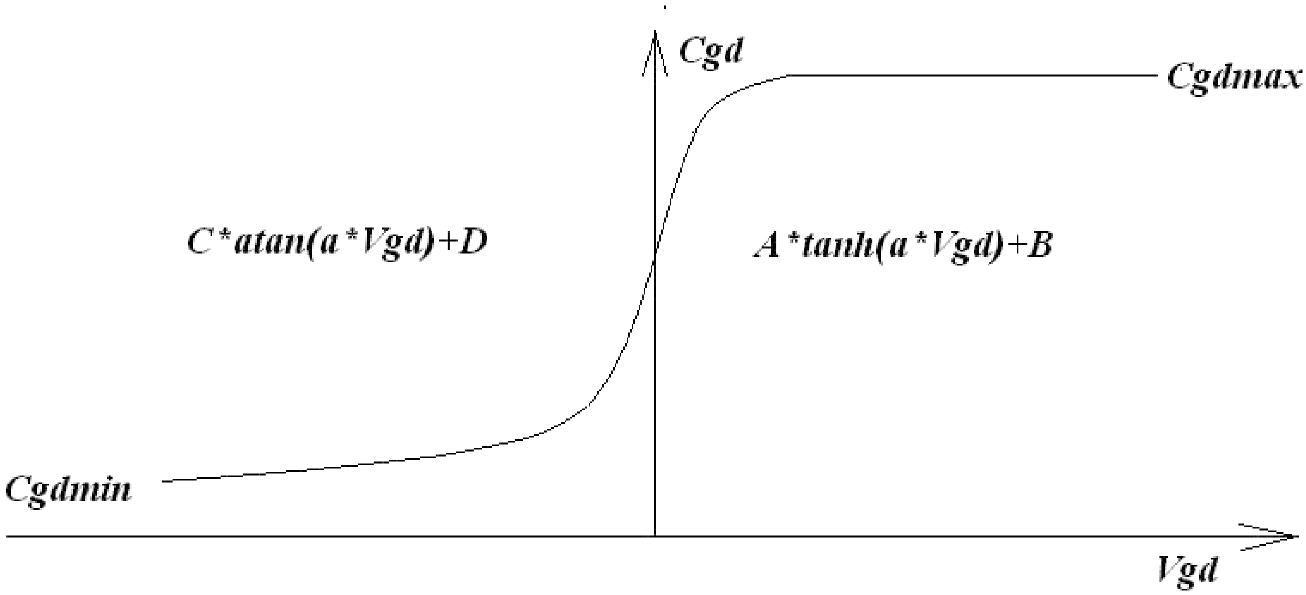
Gdsnoi	Shot noise coefficient for nlev=3	-	1.	2.
Fc	Coefficient for forward-bias depletion capacitance formula	-	0.5	
Delta	Width effect on threshold voltage (levels 2 & 3)	-	0.	1.
Theta	Mobility modulation (level 3)	-	0.	0.1
Eta	Static feedback (level 3)	-	0.	1.
Kappa	Saturation field (level 3)		0.2	0.5
Tnom	Parameter measurement temperature	°C	27	50
L	Default length	m	defl	20u
W	Default width	m	defw	20u
Ad	Default drain area	m^2	defad	200p
As	Default source area	m^2	defas	200p
Pd	Default drain perimeter	m	0.	20u
Ps	Default source perimeter	m	0.	20u
Nrd	Default drain squares	-	0.	1
Nrs	Default source squares	-	0.	1
Nrg	Default gate squares	-	0.	1
Nrb	Default bulk squares	-	0.	1
Lmin	Bin length lower limit	m	0.	10u
Lmax	Bin length upper limit	m	0.	20u

Wmin	Bin width lower limit	m	0.	10u
Wmax	Bin width upper limit	m	0.	20u



The discrete vertical double diffused MOSFET transistor(VDMOS) popularly used in board level switch mode power supplies has behavior that is qualitatively different than the above monolithic MOSFET models. In particular, (i) the body diode of a VDMOS transistor is connected differently to the external terminals than the substrate diode of a monolithic MOSFET and (ii) the gate-drain capacitance(Cgd) non-linearity cannot be modeled with the simple graded capacitances of monolithic MOSFET models. In a VDMOS transistor, Cgd abruptly changes about zero gate-drain voltage(Vgd). When Vgd is negative, Cgd is physically based a capacitor with the gate as one electrode and the drain on the back of the die as the other electrode. This capacitance is fairly low due to the thickness of the non-conducting die. But when Vgd is positive, the die is conducting and Cgd is physically based on a capacitor with the thickness of the gate oxide.

Traditionally, elaborate subcircuits have been used to duplicate the behavior of a power MOSFET. A new intrinsic spice device was written that encapsulates this behavior in the interest of compute speed, reliability of convergence, and simplicity of writing models. The DC model is the same as a level 1 monolithic MOSFET except that the length and width default to one so that transconductance can be directly specified without scaling. The AC model is as follows. The gate-source capacitance is taken as constant. This was empirically found to be a good approximation for power MOSFETS if the gate-source voltage is not driven negative. The gate-drain capacitance follows the following empirically found form:



For positive Vgd, Cgd varies as the hyperbolic tangent of Vgd. For negative Vdg, Cgd varies as the arc tangent of Vgd. The model parameters a, Cgdmax, and Cgdmax parameterize the gate drain capacitance. The source-drain capacitance is supplied by the graded capacitance of a body diode connected across the source drain electrodes, outside of the source and drain resistances.

Name	Description	Units	Default	Example
Vto	Threshold voltage	V	0	1.0
Kp	Transconductance parameter	A/V^2	1.	.5
Phi	Surface inversion potential	V	0.6	0.65
Lambda	Channel-length modulation	1/V	0.	0.02
mtriode	Conductance multiplier in triode region(allows independent fit of triode and saturation regions	-	1.	2.
Ksubthres	subthreshold conduction parameter	-	0.	1m
BV	Vds breakdown voltage	V	Infin.	40

IBV	Current at Vds=BV	A	100pA	1u
NBV	Vds breakdown emission coefficient	-	1.	10
Rd	Drain ohmic resistance	Ω	0.	1.
Rs	Source ohmic resistance	Ω	0.	1.
Rg	Gate ohmic resistance	Ω	0.	2.
Rds	Drain-source shunt resistance	Ω	Infin.	10Meg
Rb	Body diode ohmic resistance	Ω	0.	.5
Cjo	Zero-bias body diode junction capacitance	F	0.	1n
Cgs	Gate-source capacitance	F	0.	500p
Cgdmin	Minimum non-linear G-D capacitance	F	0.	300p
Cgdmax	Maximum non-linear G-D capacitance	F	0.	1000p
A	Non-linear Cgd capacitance parameter	-	1.	.5
Is	Body diode saturation current	A	1e-14	1e-15
N	Bulk diode emission coefficient	-	1.	
Vj	Body diode junction potential	V	1.	0.87
M	Body diode grading coefficient	-	0.5	0.5
	Body diode coefficient for forward-bias			

Fc	depletion capacitance formula	-	0.5	
oneway	Behavioral modeling flag to indicate current can only flow in one direction in the channel	-	-	-
tt	Body diode transit time	sec	0.	10n
Eg	Body diode activation energy for temperature effect on Is	eV	1.11	
Xti	Body diode saturation current temperature exponent	-	3.	
L	Length scaling	-	1.	
W	Width scaling	-	1.	
Kf	Flicker noise coefficient	-	0.	
Af	Flicker noise exponent	-	1.	
nchan[*]	N-channel VDMOS	-	(true)	-
pchan[*]	P-channel VDMOS	-	(false)	-
Tnom	Parameter measurement temperature	°C	27	50
Lmin	Bin length lower limit	m	0.	10u
Lmax	Bin length upper limit	m	0.	20u
Wmin	Bin width lower limit	m	0.	1
Wmax	Bin width upper limit	m	0.	10

*]The model name VDMOS is used both for a N-channel and P-channel device. The polarity defaults to N-channel. To specify P-channel, flag the model with the keyword "pchan", e.g., ".model xyz

VDMOS(Kp = 3 pchan)" defines a P-channel transistor.

It is possible to annotate a model with a voltage rating and nominal performance. This information is displayed in the schematic capture GUI to assist in selecting a device but does not impact the electrical behavior in simulation. The following parameters may be specified.

Name	**Description**	**Units**
Vds	Drain-source voltage rating	V
Ron	Nominal on resistance	Ω
Qg	Nominal gate charge required to get to Ron	C
mfg	Name of manufacturer	-

O. Lossy Transmission Line

Symbol Name: LTLIN

Syntax: Oxxx L+ L- R+ R- <model>

Example:

```
   O1 in 0 out 0 MyLossyTline
   .model MyLossyTline LTRA(len=1 R=10 L=1u C=10n)
```

This is a single-conductor lossy transmission line. N1 and N2 are the nodes at port 1. N3 and N4 are the nodes at port 2. A model card is required to define the electrical characteristics of this circuit element.

Model parameters for Lossy Transmission Lines

Name	**Description**	**Units/Type**	Default
R	Resistance per unit length	Ω	0.
L	Inductance per unit length	H	0.
G	Conductivity per unit length	1/Ω	0.
C	Capacitance per unit length	F	0.
Len	Number of unit lengths	-	0.
Rel	Relative rate of change of derivative to set a breakpoint		1.
Abs	Absolute rate of change of derivative to set a breakpoint		1.
NoStepLimit	Don't limit time- step to less than line delay	(flag)	not set
NoControl	Don't attempt complex time-step control	(flag)	not set

LinInterp	Use linear interpolation	(flag)	not set
MixedInterp	Use linear interpolation when quadratic seems to fail	(flag)	not set
CompactRel	Reltol for history compaction		RELTOL
CompactAbs	Abstol for history compaction		ABSTOL
TruncNr	Use Newton-Raphson method for time-step control	(flag)	not set
TruncDontCut	Don't limit time-step to keep impulse-response errors low	(flag)	not set

Q. Bipolar Transistor

Symbol Names: NPN, PNP, NPN2, PNP2

Syntax: Qxxx Collector Base Emitter [Substrate Node] model [area] [off] [IC=<Vbe, Vce>] [temp=<T>]

Example:

```
  Q1 C B E MyNPNmodel.model MyNPNmodel NPN(Bf=75)
```

Bipolar transistors require a model card to specify its characteristics. The model card keywords NPN and PNP indicate the polarity of the transistor.

The bipolar junction transistor model 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, quasi-saturation, and substrate conductivity. The model automatically simplifies to the Ebers-Moll model when certain parameters are not specified. The DC model is defined by the parameters Is, Bf, Nf, Ise, Ikf, and Ne which determine the forward current gain characteristics, Is, Br, Nr, Isc, Ikr, and Nc which determine the reverse current gain characteristics, and Vaf and Var which determine the output conductance for forward and reverse regions. Three ohmic resistances Rb, Rc and Re, are included, where Rb can be high current dependent. Base charge storage is modeled by forward and reverse transit times, Tf and Tr, the forward transit time Tf being bias dependent if desired; and nonlinear depletion layer capacitances, which are determined by Cje, Vje and Mje, for the B-E junction, Cjc, Vjc, and MJC for the B-C junction and Cjs, Vjs, and Mjs for the Collector- Substrate junction. The temperature dependence of the saturation current, Is, is determined by the energy gap, Eg, and the saturation-current temperature exponent, XTI. Additionally base current temperature dependence is modeled by the beta temperature exponent XTB in the new model. The values specified are assumed to have been measured at the temperature TNOM, which can be specified on the .OPTIONS control line or overridden by a specification on the .model line.

The BJT parameters used in the modified Gummel-Poon model are listed below.

Modified Gummel-Poon BJT Parameters

Name	Description	Units	Default

Is	Transport saturation current	A	1e-16
Ibc	Base-collector saturation current	A	Is
Ibe	Base-emitter saturation current	A	Is
Bf	Ideal maximum forward beta	-	100
Nf	Forward current emission coefficient	-	1.
Vaf	Forward Early voltage	V	Infin.
Ikf	Corner for forward beta high current roll-off	A	Infin.
nk	High current roll-off coefficient	-	.5
Ise	B-E leakage saturation current	A	0.
Ne	B-E leakage emission coefficient	-	1.5
Br	Ideal maximum reverse beta	-	1.
Nr	Reverse current emission coefficient	-	1.
Var	Reverse Early voltage	V	Infin.
Ikr	Corner for reverse beta high current roll-off	A	Infin.
Isc	B-C leakage saturation current	A	0
Nc	B-C leakage emission coefficient	-	2
Rb	Zero-bias base resistance	Ω	0
Irb	Current where base resistance falls halfway to its min value	A	Infin.
Rbm	Minimum base resistance at high currents	Ω	Rb
Re	Emitter resistance	Ω	0.
Rc	Collector resistance	Ω	0.
Cje	B-E zero-bias depletion capacitance	F	0.
Vje	B-E built-in potential	V	0.75
Mje	B-E junction exponential factor	-	0.33

Tf	Ideal forward transit time	sec	0.
Xtf	Coefficient for bias dependence of Tf	-	0.
Vtf	Voltage describing Vbc dependence of Tf	V	Infin.
Itf	High-current parameter for effect on Tf	A	0.
Ptf	Excess phase at freq=1/(Tf*2*Ω)Hz	°	0.
Cjc	B-C zero-bias depletion capacitance	F	0.
Vjc	B-C built-in potential	V	0.75
Mjc	B-C junction exponential factor	-	0.33
Xcjc	Fraction of B-C depletion capacitance connected to internal base node	-	1.
Xcjc2	Fraction of B-C depletion capacitance connected between internal base node and extrinsic collector	-	0
extsub	Extrinsicness of more intrinsic collector node used for substrate capacitance charge division	-	0
Tr	Ideal reverse transit time	sec	0.
Cjs	Zero-bias collector-substrate capacitance	F	0.
Xcjs	Fraction of Cjs connected internally to Rc	F	0.
Vjs	Substrate junction built-in potential	V	0.75
Mjs	Substrate junction exponential factor	-	0.
Xtb	Forward and reverse beta temperature exponent	-	0.
Eg	Energy gap for temperature effect on Is	eV	1.11
Xti	Temperature exponent for effect on Is	-	3.
Kf	Flicker-noise coefficient	-	0.

Af	Flicker-noise exponent	-	1.
Fc	Coefficient for forward-bias depletion capacitance formula	-	0.5
subs	Geometry selector if LPNP is not used: 1 means vertical 2 means lateral	-	NPN: 1 PNP: 2
BVcbo	Collector-base breakdown voltage	-	Infin.
nBVcbo	Collector-base breakdown voltage coefficient	-	5
BVbe	Base-emitter breakdown voltage	V	Infin.
Ibvbe	Current at base-emitter breakdown voltage	A	1e-10
nbvbe	Base-emitter breakdown coefficient	-	1.
Tnom	Parameter measurement temperature	°C	27
Cn	Quasi-saturation temperature coefficient for hole mobility	2.42 NPN 2.2 PNP	
D	Quasi-saturation temperature coefficient for scattering-limited hole carrier velocity	.87 NPN .52 PNP	
Gamma	Epitaxial region doping factor		1e-11
Qco	Epitaxial region charge factor	Coul	0.
Quasimod	Quasi-saturation flag for temperature dependence	-	(not set)
Rco	Epitaxial region resistance	Ω	0.
Vg	Quasi-saturation extrapolated bandgap voltage at 0°K	V	1.206
Vo	Carrier mobility knee voltage	V	10.
Tre1	Re linear temperature coefficient	1/°C	0.
Tre2	Re quadratic temperature coefficient	$1/{}^{\circ}C^2$	0.
Trb1	Rb linear temperature coefficient	1/°C	0.
Trb2	Rb quadratic temperature coefficient	$1/{}^{\circ}C^2$	0.

Trc1	Rc linear temperature coefficient	1/°C	0.
Trc2	Rc quadratic temperature coefficient	1/°C^2	0.
Trm1	Rmb linear temperature coefficient	1/°C	0.
Trm2	Rmb quadratic temperature coefficient	1/°C^2	0.
Iss	Substrate junction saturation current	A	0.
Ns	Substrate junction emission Coefficient	-	1.
Tvaf1	Vaf linear temperature coefficient	1/°C	0.
Tvaf2	Vaf quadratic temperature coefficient	1/°C^2	0.
Tvar1	Var linear temperature coefficient	1/°C	0.
Tvar2	Var quadratic temperature coefficient	1/°C^2	0.
Tikf1	Ikf linear temperature coefficient	1/°C	0.
Tikf2	Ikf quadratic temperature coefficient	1/°C^2	0.
Trbm1	Rbm linear temperature coefficient	1/°C	0.
Trbm2	Rbm quadratic temperature coefficient	1/°C^2	0.
Tbvcbo1	BVcbo linear temperature coefficient	1/°C	0.
Tbvcbo2	BVcbo quadratic temperature coefficient	1/°C^2	0.

It is possible to annotate a model with device ratings. This information is displayed in the schematic capture GUI to assist in selecting a device but does not directly impact the electrical behavior in simulation. The following parameters may be specified.

Name	**Description**	**Units**
Vceo	Maximum collector-emitter voltage with the base floating	V

Icrating	Maximum collector current	A
mfg	Name of manufacturer	-

The model parameter "level" can be used to specify another type of BJT in LTspice.

Set Level=504 to use the MEXTRAM 504 transistor due to NXP(Philips).

Due to a generous contribution of source code from Dr.-Ing. Dietmar Warning of DAnalyse GmbH, Berlin, Germany; LTspice includes a version of VBIC. Set Level=9 to use the alternate device. Level 4 is a synonym for level 9. The following documentation has been supplied by Dr. Warning:

VBIC - Vertical Bipolar Inter Company model

The VBIC model is a extended development of the Standard Gummel-Poon (SGP) model with the focus of integrated bipolar transistors in today's modern semiconductor technologies. With the implemented modified Quasi-Saturation model from Kull and Nagel it is also possible to model the special output characteristic of switching transistors. It is a widely used alternative to the SGP model for silicon, SiGe and III-V HBT devices.

VBIC Capabilities compared to Standard Gummel-Poon Model

- Integrated Substrate transistor for parasitic devices in integrated processes

- Weak avalanche and Base-emitter breakdown model

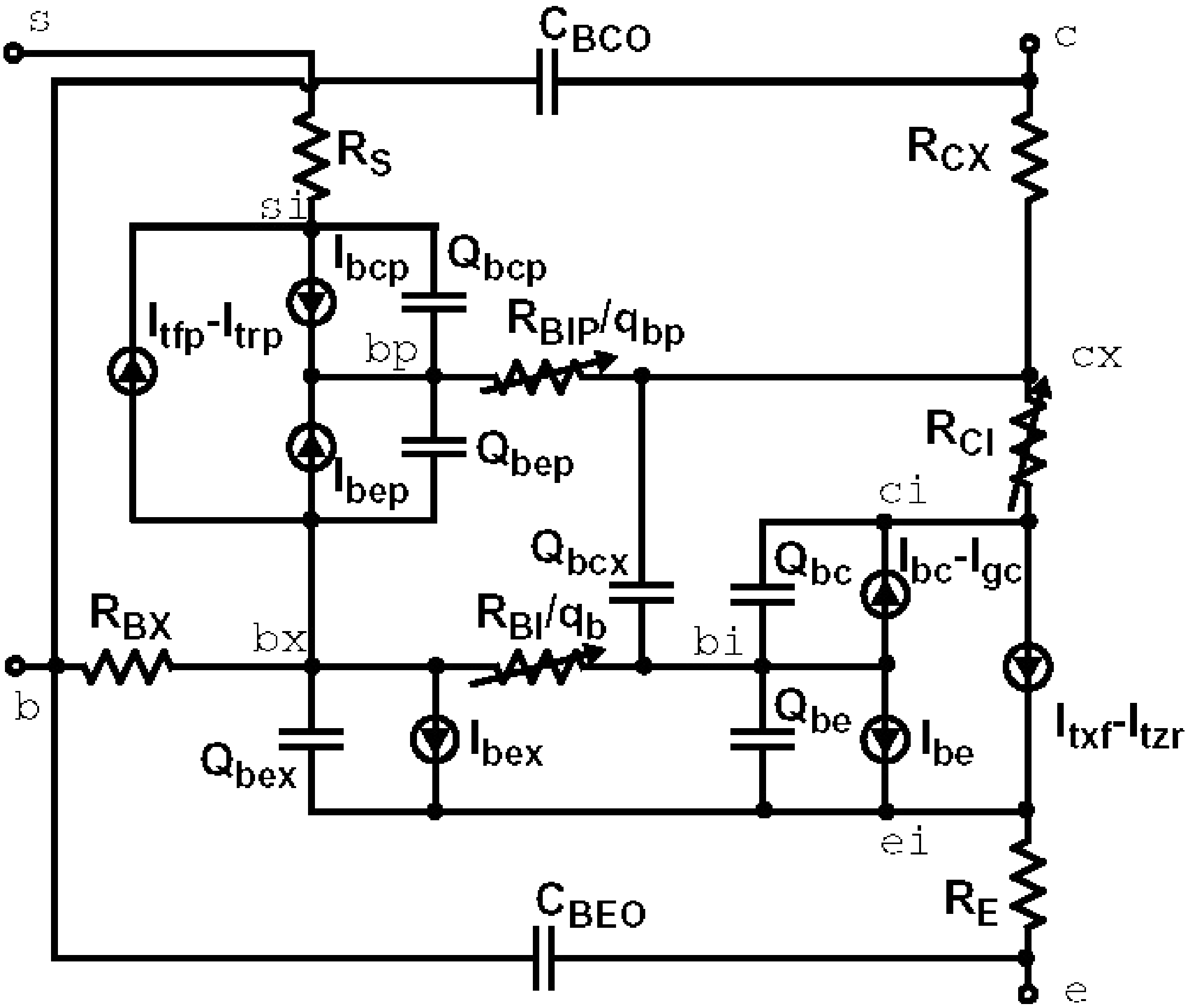
- Improved Early Effect modeling

- Physical separation of Ic and Ib

- Improved Depletion capacitance model

- Improved temperature modeling

Model Structure



VBIC Parameters

Because the VBIC model is based on SGP model it is possible to start with SGP parameters, carry out some transformations. Following parameters are from VBIC version 1.2, which is implemented in LTSpice in the 4-terminal version without excess phase network and self-heating effect. To switch from SGP to VBIC you should set the extra parameter level to 9.

Name	Description	Unit	Default
Tnom	Parameter measurement temperature	°C	27.
Rcx	Extrinsic collector resistance	Ω	0.1
Rci	Intrinsic collector resistance	Ω	0.1
	Epi drift saturation		

Vo	voltage	V	Infin.
gamm	Epi doping parameter		0.0
hrcf	High current RC factor		Infin.
Rbx	Extrinsic base resistance	Ω	0.1
Rbi	Intrinsic base resistance	Ω	0.1
Re	Intrinsic emitter resistance	Ω	0.1
Rs	Intrinsic substrate resistance	Ω	0.1
Rbp	Parasitic base resistance	Ω	0.1
Is	Transport saturation current	A	1e-16
nf	Forward emission coefficient		1.
nr	Reverse emission coefficient		1.
Fc	Fwd bias depletion capacitance limit		0.9
Cbeo	Extrinsic B-E overlap capacitance	F	0.0
Cje	Zero bias B-E depletion capacitance	F	0.0
pe	B-E built in potential	V	0.75
me	B-E junction grading coefficient		0.33
Aje	B-E capacitance smoothing factor		-0.5
Cbco	Extrinsic B-C overlap capacitance	F	0.
Cjc	Zero bias B-C depletion capacitance	F	0.
Qco	Epi charge parameter	C	0.
Cjep	B-C extrinsic zero bias capacitance	F	0.
pc	B-C built in potential	V	0.75
mc	B-C junction grading coefficient		0.33

Ajc	B-C capacitance smoothing factor		-0.5
Cjcp	Zero bias S-C capacitance	F	0.
ps	S-C junction built in potential	V	0.75
ms	S-C junction grading coefficient		0.33
Ajs	S-C capacitance smoothing factor		-0.5
Ibei	Ideal B-E saturation current	A	1e-18
wbe	Portion of IBEI from Vbei 1-WBE from Vbex		1.
nei	Ideal B-E emission coefficient		1.
Iben	Non-ideal B-E saturation current	A	0.
nen	Non-ideal B-E emission coefficient		2.
ibci	Ideal B-C saturation current	A	1e-16
Nci	Ideal B-C emission coefficient		1.
ibcn	Non-ideal B-C saturation current	A	0.
ncn	Non-ideal B-C emission coefficient		1.
avc1	B-C weak avalanche parameter 1	1/V	0.
avc2	B-C weak avalanche parameter 2	1/V	0.
isp	Parasitic transport saturation current	A	0.
wsp	Portion of ICCP		1.
nfp	Parasitic fwd emission coefficient		1.
Ibeip	Ideal parasitic B-E saturation current	A	0.
ibenp	Non-ideal parasitic B-E saturation current	A	0.

ibcip	Ideal parasitic B-C saturation current	A	0.
ncip	Ideal parasitic B-C emission coefficient		1.
Ibcnp	Non-ideal parasitic B-C saturation current	A	0.
ncnp	Non-ideal parasitic B-C emission coefficient		2.
Vef	Forward Early voltage		Infin.
Ver	Reverse Early voltage		Infin.
Ikf	Forward knee current	A	Infin.
ikr	Reverse knee current	A	Infin.
Ikp	Parasitic knee current	A	Infin.
tf	Ideal forward transit time	s	0.
Qtf	Variation of TF with base-width modulation		0.
Xtf	Coefficient for bias dependence of TF		0.
Vtf	Voltage giving VBC dependence of TF	V	Infin.
Itf	High current dependence of TF	A	Infin.
tr	Ideal reverse transit time	sec	0.
Td	Forward excess-phase delay time	Sec	0.
kfn	B-E Flicker Noise Coefficient		0.
afn	B-E Flicker Noise Exponent		1.
bfn	B-E Flicker Noise 1/f dependence		1.0
Xre	Temperature exponent of RE		0.
Xrbi	Temperature exponent of RBI		0.
Xrci	Temperature exponent of RCI		0.
Xrs	Temperature exponent of		0.

	RS		
Xvo	Temperature exponent of VO		0.
Ea	Activation energy for IS	V	1.12
Eaie	Activation energy for IBEI	V	1.12
Eaic	Activation energy for IBCI/IBEIP	V	1.12
Eais	Activation energy for IBCIP	V	1.12
Eane	Activation energy for IBEN	V	1.12
Eanc	Activation energy for IBCN/IBENP	V	1.12
Eans	Activation energy for IBCNP	V	1.12
Xis	Temperature exponent of IS		3.
Xii	Temperature exponent of IBEI,IBCI,IBEIP,IBCIP		3.
Xin	Temperature exponent of IBEN,IBCN,IBENP,IBCNP		3.
Tnf	Temperature exponent of NF		0.
Tavc	Temperature exponent of AVC2		0.
rth	Thermal resistance	K/W	0.
Cth	Thermal capacitance	Ws/K	0.
Vrt	Punch-through voltage of internal B-C junction	V	0.
Art	Smoothing parameter for reach-through		0.1
Ccso	Fixed C-S capacitance	F	0.
qbm	Select SGP qb formulation		0.
nkf	High current beta rolloff		0.5
Xikf	Temperature exponent of IKF		0.

Xrcx	Temperature exponent of RCX		0.
Xrbx	Temperature exponent of RBX		0.
Xrbp	Temperature exponent of RBP		0.
Isrr	Separate IS for fwd and rev		1.
Xisr	Temperature exponent of ISR		0.
dear	Delta activation energy for ISRR		0.
Eap	Excitation energy for ISP		1.12
Vbbe	B-E breakdown voltage	V	0.
nbbe	B-E breakdown emission coefficient		1.
Ibbe	B-E breakdown current		1e-06
Tvbbe1	Linear temperature coefficient of VBBE		0.
Tvbbe2	Quadratic temperature coefficient of VBBE		0.
Tnbbe	Temperature coefficient of NBBE		0.
ebbe	exp(-VBBE/(NBBE*Vtv))		0.
dtemp	Local Temperature difference	°	0.
Vers	Revision Version		1.2
Vref	Reference Version		0.

References:

C. C. McAndrew et al., "Vertical Bipolar Inter Company 1995: An Improved Vertical, IC Bipolar Transistor Model", Proceedings of the IEEE Bipolar Circuits and Technology Meeting, pp. 170 - 177, 1995

C. C. McAndrew et.al., VBIC95, "The Vertical Bipolar Inter-Company Model", IEEE Journal of Solid State Circuits, vol. 31, No. 10, October 1996

C. C. McAndrew, VBIC Model Definition, Release 1.2, 18. Sep. 1999

R. Resistor

Symbol Names: RES, RES2

Syntax: Rxxx n1 n2 <value> [tc=tc1, tc2, ...] [temp=<value>]

The resistor supplies a simple linear resistance between nodes n1 and n2. A temperature dependence can be defined for each resistor instance with the parameter tc. The resistance, R, at will be

```
R = R0 * (1. + dt * tc1 + dt**2 * tc2 + dt**3 * tc3 + ...)
```

where R0 is the resistance at the nominal temperature and dt is the difference between the resistor's temperature and the nominal temperature.

S. Voltage Controlled Switch

Symbol Names: SW

Syntax: Sxxx n1 n2 nc+ nc- <model> [on,off]

Example:

```
S1 out 0 in 0 MySwitch
.model MySwitch SW(Ron=.1 Roff=1Meg Vt=0 Vh=-.5 Lser=10n Vser=.6)
```

The voltage between nodes nc+ and nc- controls the switch's impedance between nodes n1 and n2. A model card is required to define the behavior of the switch. See the schematic file .\examples\Educational\Vswitch.asc to see an example of a model card placed directly on a schematic as a SPICE directive.

Voltage Controlled Switch Model Parameters

Name	Description	Units	Default
Vt	Threshold voltage	V	0.0
Vh	Hysteresis voltage	V	0.0
Ron	On resistance	Ω	1.0
Roff	Off resistance	Ω	1/Gmin
Lser	Series inductance	H	0.0
Vser	Series voltage	V	0.0
Ilimit	Current limit	A	Infin.

The switch has three distinct modes of voltage control, depending on the value of the hysteresis voltage, Vh. If Vh is zero, the switch is always completely on or off depending upon whether the input voltage is above the threshold. If Vh is positive, the switch shows hysteresis, as if it was controlled by a Schmitt trigger with trip points at Vt - Vh and Vt + Vh. Note that Vh is half the voltage between trip points which is different than the common laboratory nomenclature. If Vh is negative, the switch will smoothly transition between the on and off impedances. The transition occurs between the control voltages of Vt - Vh and Vt + Vh. The smooth transition follows a low order polynomial fit to the logarithm of the switch's conduction.

There is also a level 2 voltage-controlled switch which is an advanced version of the level 1 switch with negative hysteresis. The level 2 switch is never completely on or off. The conduction as a function of control voltage Vc is

```
   g(Vc) = exp(A * atan((Vc - Vt)/abs(Vh)) + B)
```

where

```
   A = log(Roff / Ron) / π
   B = log(1 / (Roff * Ron)) / 2
```

Also, the transition of the level 2 switch to current limit is gradual instead of abrupt. At a fixed control voltage, the I-V curve is given by the equation

```
   I(V) = Ilimit * tanh(g(Vc) * V)
```

where Ilimit defaults to 10 amperes for the level 2 switch.

The level 2 switch supports the option to conduct in only one direction by either specifying the flag "oneway" or specifying a voltage drop with parameter Vser. The transition between forward conduction and reverse open circuit can be specified to be a smooth transition by specifying the parameter epsilon to be non-zero.

T. Lossless Transmission Line

Symbol Name: TLINE

Syntax: Txxx L+ L- R+ R- Zo=<value> Td=<value>

L+ and L- are the nodes at one port. R+ and R- are the nodes for the other port. Zo is the characteristic impedance. The length of the line is given by the propagation delay Td.

This element models only one propagation mode. If all four nodes are distinct in the actual circuit, then two modes may be excited. To simulate such a situation, two transmission-line elements are required. See the schematic file .\examples\Educational\TransmissionLineInverter.asc to see an example simulating both modes of a length of coax.

U. Uniform RC-line

Symbol Names: URC

Syntax: Uxxx N1 N2 Ncom <model> L=<len> [N=<lumps>]

N1 and N2 are the two element nodes the RC line connects, whereas Ncom is the node to which the capacitances are connected. MNAME is the model name and LEN is the length of the RC line in meters. Lumps, if specified, is the number of lumped segments to use in modeling the RC line. A guess at an appropriate number of lumps to use will be made if lumps is not specified.

The URC model is derived from a model proposed by L. Gertzberrg in 1974. The model is accomplished by a subcircuit-type expansion of the URC line into a network of lumped RC segments with internally generated nodes. The RC segments are in a geometric progression, increasing toward the middle of the URC line, with K as a proportionality constant.

The URC line is made up strictly of resistor and capacitor segments unless the ISPERL parameter is given a nonzero value, in which case the capacitors are replaced with reverse-biased diodes with a zero-bias junction capacitance equivalent to the capacitance replaced, and with a saturation current of ISPERL amps per meter of transmission line and an optional series resistance equivalent to RSPERL ohms per meter.

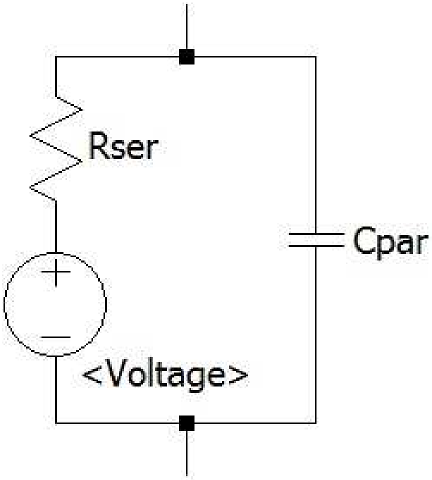
Name	Description	Units	Default
K	Propagation Constant	-	2.0
Fmax	Maximum Frequency of interest	Hz	1G
Rperl	Resistance per unit length	Ω	1K
Cperl	Capacitance per unit length	F	1e-15
Isperl	Saturation Current per unit length	A	0.0
Rsperl	Diode Resistance per unit length	Ω	0.0

V. Voltage Source

Symbol Names: VOLTAGE, BATTERY

Syntax: Vxxx n+ n- <voltage> [AC=<amplitude>] [Rser=<value>] [Cpar=<value>]

This element sources a constant voltage between nodes n+ and n-. For AC analysis, the value of AC is used as the amplitude of the source at the analysis frequency. A series resistance and parallel capacitance can be defined. The equivalent circuit is:



Voltage sources have historically been used as the current meters in SPICE and are used as current sensors for current-controlled elements. If Rser is specified, the voltage source can not be used as a sense element for F, H, or W elements. However, the current of any circuit element, including the voltage source, can be plotted.

Syntax: Vxxx n+ n- PULSE(V1 V2 Tdelay Trise Tfall Ton Tperiod Ncycles

Time-dependent pulsed voltage source

Name	Description	Units
Voff	Initial value	V
Von	Pulsed value	V
Tdelay	Delay	sec
Tr	Rise time	sec
Tf	Fall time	sec
Ton	On time	sec
Tperiod	Period	sec

Ncycles	Number of cycles(Omit for free-running pulse function)	cycles

Syntax: Vxxx n+ n- SINE(Voffset Vamp Freq Td Theta Phi Ncycles)

Time-dependent sine wave voltage source.

Name	Description	Units
Voffset	DC offset	V
Vamp	Amplitude	V
Freq	Frequency	Hz
Td	Delay	sec
Theta	Damping factor	1/sec
Phi	Phase of sine wave	degrees
Ncycles	Number of cycles(Omit for free-running sine function)	cycles

For times less than Td or times after completing Ncycles, have run, the output voltage is given by

Voffset+Vamp*sin(π*Phi/180)

Otherwise the voltage is given by

Voffset+Vamp*exp(-(time-Td)*Theta)*sin(2*π*Freq*(time-Td)+π*Phi/180)

The damping factor, Theta, is the reciprocal of the decay time constant.

Syntax: Vxxx n+ n- EXP(V1 V2 Td1 Tau1 Td2 Tau2)

Time-dependent exponential voltage source

Name	Description	Units
V1	Initial value	V
V2	Pulsed value	V
Td1	Rise delay time	sec
Tau1	Rise-time constant	sec
Td2	Fall delay time	sec

Tau2	Fall-time constant	sec

For times less than Td1, the output voltage is V1. For times between Td1 and Td2 the voltage is given by

```
V1+(V2-V1)*(1-exp(-(time-Td1)/Tau1))
```

For times after Td2 the voltage is given by

```
V1+(V2-V1)*(1-exp(-(time-Td1)/Tau1))-(V2-V1)*(1-exp(-(time-
Td2)/Tau2))
```

Syntax: Vxxx n+ n- SFFM(Voff Vamp Fcar MDI Fsig)

Time-dependent single frequency FM voltage source.

Name	**Description**	**Units**
Voff	DC offset	V
Vamp	Amplitude	V
Fcar	Carrier frequency	Hz
MDI	Modulation index	-
Fsig	Signal frequency	Hz

The voltage is given by

```
Voff+Vamp*sin((2.*π*Fcar*time)+MDI*sin(2.*π*Fsig*time))
```

Syntax: Vxxx n+ n- PWL(t1 v1 t2 v2 t3 v3...)

Arbitrary Piece-wise linear voltage source.

For times before t1, the voltage is v1. For times between t1 and t2, the voltage varies linearly between v1 and v2. There can be any number of time, voltage points given. For times after the last time, the voltage is the last voltage.

Syntax: Vxxx n+ n- wavefile=<filename> [chan=<nnn>]

This allows a .wav file to be used as an input to LTspice. <filename> is either a full, absolute path for the .wav file or a relative path computed from the directory containing the simulation schematic or netlist. Double quotes may be used to specify a path containing spaces. The .wav file may contain up to 65536 channels, numbered 0 to 65535. Chan may be set to specify which channel is used. By default, the first channel, number 0, is used. The .wav file is interpreted as having a full scale range from -1V to 1V.

This source only has meaning in a .tran analysis.

W. Current Controlled Switch

Symbol Names: CSW

Syntax: Wxxx n1 n2 Vnam <model> [on,off]

Example:

```
W1 out 0 Vsense MySwitch
Vsense a b 0.
.model MySwitch CSW(Ron=.1 Roff=1Meg It=0 Ih=-.5)
```

The current through the named voltage source controls the switch's impedance. A model card is required to define the behavior of the current controlled switch.

Current Controlled Switch Model Parameters

Name	Description	Units	Default
It	Threshold current	A	0.0
Ih	Hysteresis current	A	0.0
Ron	On resistance	Ω	1.0
Roff	Off resistance	Ω	1/Gmin

The switch has three distinct modes of current control, depending on the value of the hysteresis current, Ih. If Ih is zero, the switch is always completely on or off according to whether the control current is above threshold. If Ih is positive, the switch shows hysteresis with trip point currents at It - Ih and It + Ih. If Ih is negative, the switch will smoothly transition between the on and off impedances. The transition occurs between the control currents of It - Ih and It + Ih. The smooth transition follows a low order polynomial fit to the logarithm of the switch's conduction.

X. Subcircuit

Syntax: Xxxx n1 n2 n3... <subckt name> [<parameter>=<expression>]

Subcircuits allow circuitry to be defined and stored in a library for later retrieval by name. Below is an example of defining and calling a voltage divider and invoking it in a circuit.

```
*
* This calls the circuit
X1 in out 0 divider top=9K bot=1K
V1 in 0 pulse(0 1 0 .5m .5m 0 1m)
*
* This is the subcircuit definition
.subckt divider A B C
R1 A B {top}
R2 B C {bot}
.ends divider
*
.tran 3m
.end
```

Z. MESFET and IGBT Transistors

Symbol Names: MESFET, NIGBT, PIGBT

Syntax: Zxxx D G S model [area] [m=<value>] [off] [IC=<Vds, Vgs>] [temp=<value>]

A MESFET transistor requires a model card to specify its characteristics. The model card keywords NMF and PMF specify the polarity of the transistor. The MESFET model is derived from the GaAs FET model described in H. Statz et al., GaAs FET Device and Circuit Simulation in SPICE, IEEE Transactions on Electron Devices, V34, Number 2, February, 1987 pp160-169.

Two ohmic resistances, Rd and Rs, are included. Charge storage is modeled by total gate charge as a function of gate-drain and gate-source voltages and is defined by the parameters Cgs, Cgd, and Pb.

Name	Description	Units	Default
Vto	Pinch-off voltage	V	-2.0
Beta	Transconductance parameter	A/V^2	1e-4
B	Doping tail extending parameter	1/V	0.3
Alpha	Saturation voltage parameter	1/V	2.0
Lambda	Channel-length modulation	1/V	0.0
Rd	Drain ohmic resistance	Ω	0.0
Rs	Source ohmic resistance	Ω	0.0
Cgs	Zero-bias G-S junction capacitance	F	0.0
Cgd	Zero-bias G-D junction capacitance	F	0.0
Pb	Gate junction potential	V	1.0
Kf	Flicker noise coefficient	-	0.0
Af	Flicker noise exponent	-	1.0
Fc	Forward-bias depletion coefficient	-	0.5
Is	Junction saturation current	A	1e-14

A device with a Z as prefix can also mean an IGBT transistor. Disambiguation between the MESFET and IGBT is via the model statement.

Syntax: Zxxx C G E MNAME [area] [m=<value>] [off] [temp=<value>]

```
        .model MNAME NIGBT
```

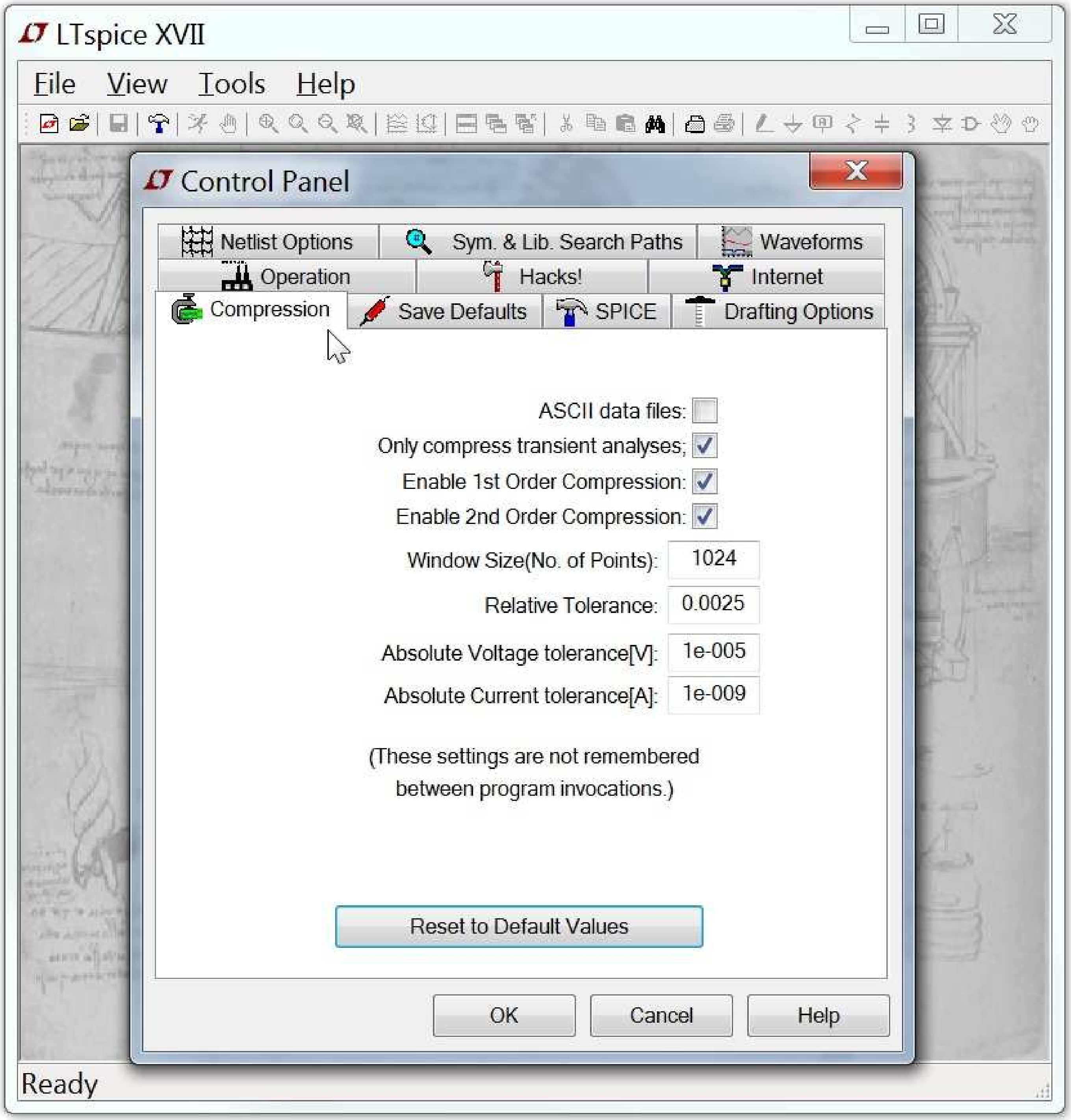
The LTspice IGBT implementation is based on original work by Robert Ritchie of Linear Technology Corporation. It uses device equations out of a series of papers by Allen Hefner of NIST et al. with some exceptions, e.g., the LTspice implementation includes subthreshold conduction and stochastic noise mechanisms.

Name	Description	Units	Default
Agd	Gate-Drain overlap area	A/V^2	5e-6
area	Active area	m^2	1e-5
BVF	Avalanche uniformity factor	-	1.0
BVN	Avalanche multiplication exponent	-	4.0
Cgs	Gate-Source capacitance per unit area	F/cm^2	1.24e-8
Coxd	Gate-Drain oxide capacitance per unit area	F/cm^2	3.5e-8
Jsne	Emitter saturation current density	A/cm^2	6.5e-13
KF	Triode region factor	-	1.0
KP	MOSFET transconductance	A/V^2	0.38
MUN	Electron mobility	cm^2/(V新)	1500
MUP	Hole mobility	cm^2/(V新)	450
NB	Base doping	1/cm^3	2e14
Tau	Ambipolar recombination lifetime	sec	7.1e-6
Theta	Transverse field factor	1/V	0.02
Vt	Threshold voltage	V	4.7
Vtd	Gate-Drain overlap depletion threshold	V	1e-3
WB	Metallurgical base width	m	9e-5
subthres	Subthreshold current parameter	-	0.02
Kfn	Flicker noise coefficient	-	0.0
Afn	Flicker noise exponent	-	1.0
tnom	Parameter measurement temperature	°C	27

Accessing the Control Panel

To get to the Control Panel, use the menu command Tools=>Control Panel. There you can configure many aspects of LTspice XVII.

Compression



ASCII data files: □
Only compress transient analyses: ☑
Enable 1st Order Compression: ☑
Enable 2nd Order Compression: ☑
Window Size(No. of Points): 1024
Relative Tolerance: 0.0025
Absolute Voltage tolerance[V]: 1e-005
Absolute Current tolerance[A]: 1e-009

(These settings are not remembered between program invocations.)

LTspice compresses the raw data files as they are generated. A compressed file can be 50 times smaller than the un-compressed one. This is a lossy compression. This pane of the control panel allows you to control how lossy the compression runs.

Window Size(No. of Points): Maximum number of points that can be compressed into two end points. Set to zero to turn off waveform compression.

Relative Tolerance: The relative error allowed between the compressed data and the uncompressed data.

Absolute Voltage tolerance[V]: The voltage error allowed by the compression algorithm.

Absolute Current tolerance[A]: The current error allowed be the compression algorithm.

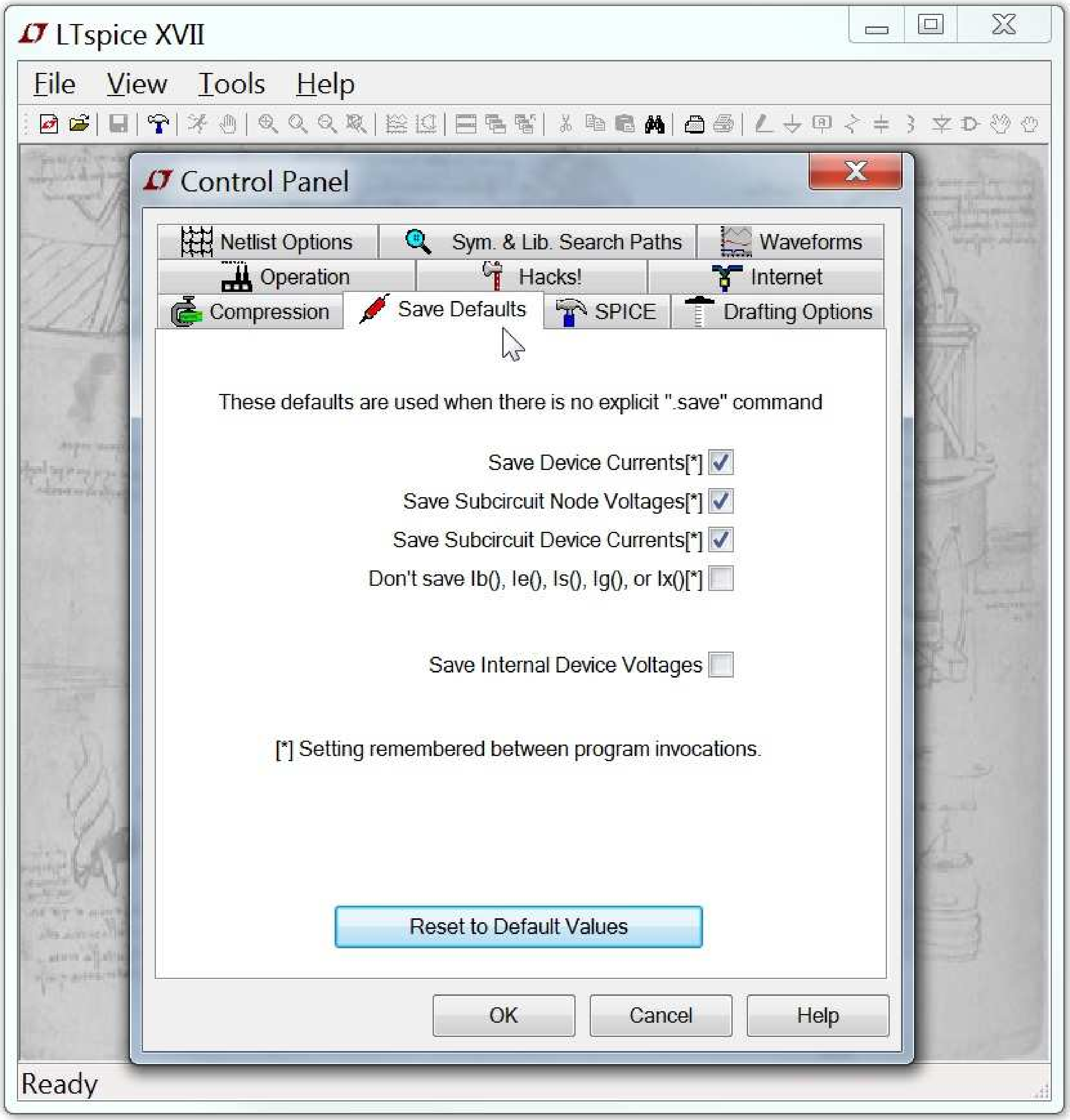
These compression settings are not remembered between program invocations to encourage use of the defaults. They are available on the control panel for diagnostic purposes. The tolerances and window size can be specified with option parameters plotreltol, plotvntol, plotabstol and plotwinsize in .option statements placed as SPICE directives on the schematic.

You will probably want to turn of compression when using .four statements or doing FFT's in post analysis of your data with the SPICE directive:

```
.options plotwinsize=0
```

Save Defaults

These settings are used when you don't explicitly state which nodes should be saved in a simulation. Useful settings are "Save Device Currents", "Save Subcircuit Node Voltages", and "Save Subcircuit Device Currents". Device voltages and internal device voltages are only of internal program development use.



These defaults are used when there is no explicit ".save" command

Save Device Currents[*] ☑

Save Subcircuit Node Voltages[*] ☑

Save Subcircuit Device Currents[*] ☑

Don't save Ib(), Ie(), Is(), Ig(), or Ix()[*] □

Save Internal Device Voltages □

[*] Setting remembered between program invocations.

Save Device Currents: Check this so that you can plot device and terminal currents. You will also need it to be able to plot dissipation.

Save Subcircuit Node Voltages: You will need to check this to plot voltages in hierarchical designs.

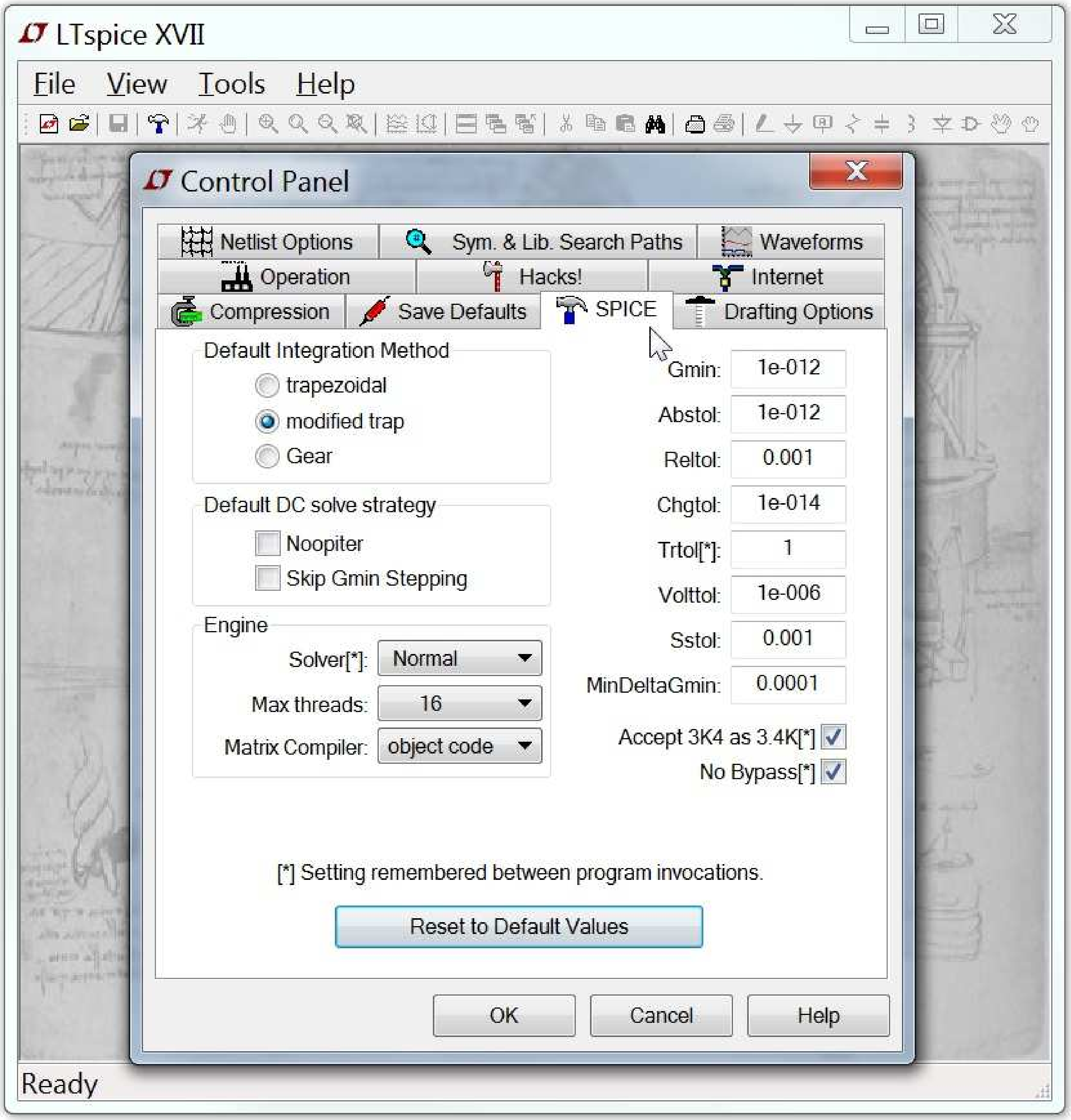
Save Subcircuit Device Currents: You will need to check this to plot currents in hierarchical designs.

Don't save Ib(), Ie(), Is(), Ig(): This saves only the collector(drain) currents of transistors in the interest of reducing the size of the output .data file. This is useful for IC design, but it using it means that there isn't enough data available to compute transistor dissipation.

Save Device Currents: Check this so that you can plot device and terminal currents. You will also need it to be able to plot dissipation.

SPICE

This pane allows you to define the various defaults for LTspice. These defaults can be overridden in any simulation by specifying the options in a .option statement in that simulation. Usually you can leave these options as they are.



Default Integration Method

○ trapezoidal
● modified trap
○ Gear

Default DC solve strategy

□ Noopiter
□ Skip Gmin Stepping

Engine

Solver[*]: Normal
Max threads: 16
Matrix Compiler: object code

Gmin: 1e-012
Abstol: 1e-012
Reltol: 0.001
Chgtol: 1e-014
Trtol[*]: 1
Volttol: 1e-006
Sstol: 0.001
MinDeltaGmin: 0.0001

Accept 3K4 as 3.4K[*] ☑
No Bypass[*] ☑

[*] Setting remembered between program invocations.

One default you may want to change is Trtol. Most SPICE programs default this to 7. In LTspice this defaults to 1 so that simulations are extremely unlikely to show any simulation artifacts in their waveforms. Trtol affects the timestep strategy more than directly affecting the accuracy of the simulation. For transistor-level simulations, a value larger than 1 is usually a better overall solution. You might find that you get a speed of 2x if you increase trtol with out adversely affecting simulation accuracy. Your trtol setting is remembered between program invocations. However, most of the traditional SPICE tolerance parameters, gmin, abstol, reltol, chgtol, vntol are not remembered between program invocations in order to encourage use of the default values. If you want to use something other than the default values, you will have to write a .option statement specifying the values you want to use and place it on the schematic or keep the settings in a file and .inc that file.

Also interesting is which solver is used. LTspice contains two complete implementations of SPICE. One is called the Normal Solver and the other is called the Alternate Solver. The Alternate Solver uses a different sparse matrix package with reduced roundoff error. Typically the Alternate Solver will simulate at half the speed of the Normal Solver but with one thousand times more internal accuracy. This can be a useful diagnostic to have available. There is no .option to specify which solver is used, the choice must be made before the netlist is parsed because the two solvers use different parsers.

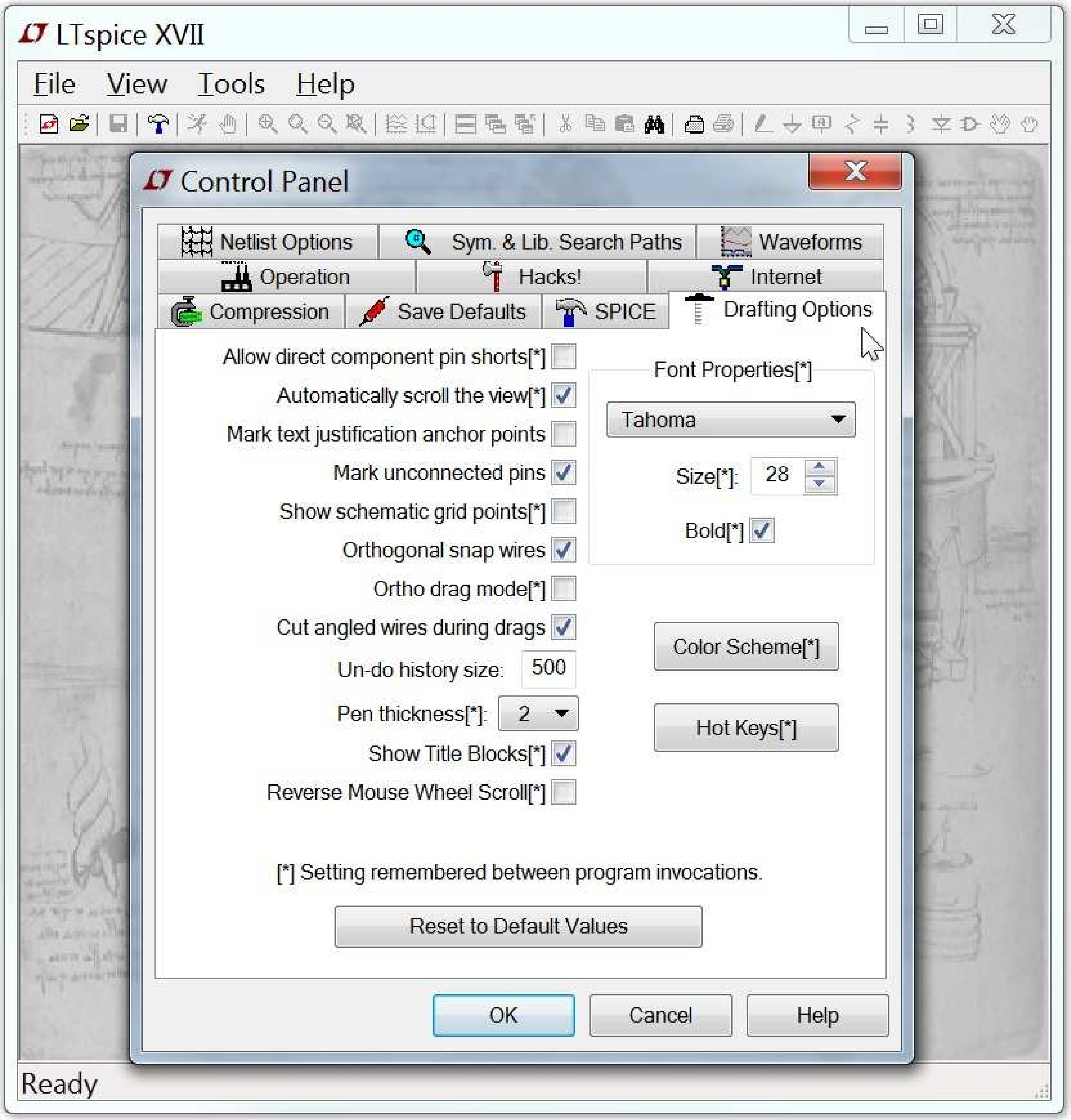
The maximum number of threads is set to the maximum number of concurrently executing threads your OS and CPU hardware supports. The actual number used in any given simulation depends on the nature of the circuit. While LTspice introduces stochastically cooled threads[1] to define the state of the art of multi-threaded SPICE simulation, there are some circuits that cannot benefit from multiple threads. LTspice will not tie up additional threads that don't in the end make the simulation run faster.

The matrix compiler defaults to object code. That means that as LTspice solves your circuit, it will, on the fly, author an assembly language listing optimized for your circuit. It will then assemble, link, and execute this code instead of the normal LU factorization codes that are written in a mix of C++ and hand-coded assembly.

Check the box next to "Accept 3K4 as 3.4K" to force LTspice to understand a number written as 4K99 to be equal to 4.99K. Normal SPICE practice does not allow this, but it is available in LTspice by popular request.

[1] Multithreaded solvers are hindered by inter-thread communication timing. LTspice improves the coherency of thread execution by dynamically adjusting the cache available to each thread so they run at the same real time speed.

Drafting Options



Allow direct component pin shorts[*] □

Automatically scroll the view[*] ☑

Mark text justification anchor points □

Mark unconnected pins ☑

Show schematic grid points[*] □

Orthogonal snap wires ☑

Ortho drag mode[*] □

Cut angled wires during drags ☑

Un-do history size: 500

Pen thickness[*]: 2

Show Title Blocks[*] ☑

Reverse Mouse Wheel Scroll[*] □

Font Properties[*]

Tahoma

Size[*]: 28

Bold[*] ☑

[*] Setting remembered between program invocations.

Allow direct component pin shorts: Normally you can draw a wire directly through a component and the wire segment shorting pins is deleted. If you check it, the shorting wire will not be automatically deleted.

Automatically scroll the view: Checking this box makes the view of the schematic scroll as you move the mouse close the edge while editing the schematic.

Mark text Justification anchor points: Draw a small circle to indicate the reference point of text blocks.

Mark unconnected pins: Draw a small square at each unconnected pin to flag it as unconnected.

Show schematic grid points: Start with visible grid enabled.

Orthogonal snap wires: Force wires to be drawn in vertical and horizontal segments while drawing. If not checked, a wire can drawn at any angle and will snap to any grid. Holding down the control key will momentarily toggle the current setting while drawing wires.

Ortho drag mode: Force wires to be drawn in vertical and horizontal segments while dragging. If not checked, a wire can dragged at any angle. Holding down the control key will momentarily toggle the current setting while dragging wires.

Cut angled wires during drags: During the Drag command, a non-orthogonal wire will be broken into two connected wires if you click along the middle of the wire

Undo history size: Set the size of the undo/redo buffer.

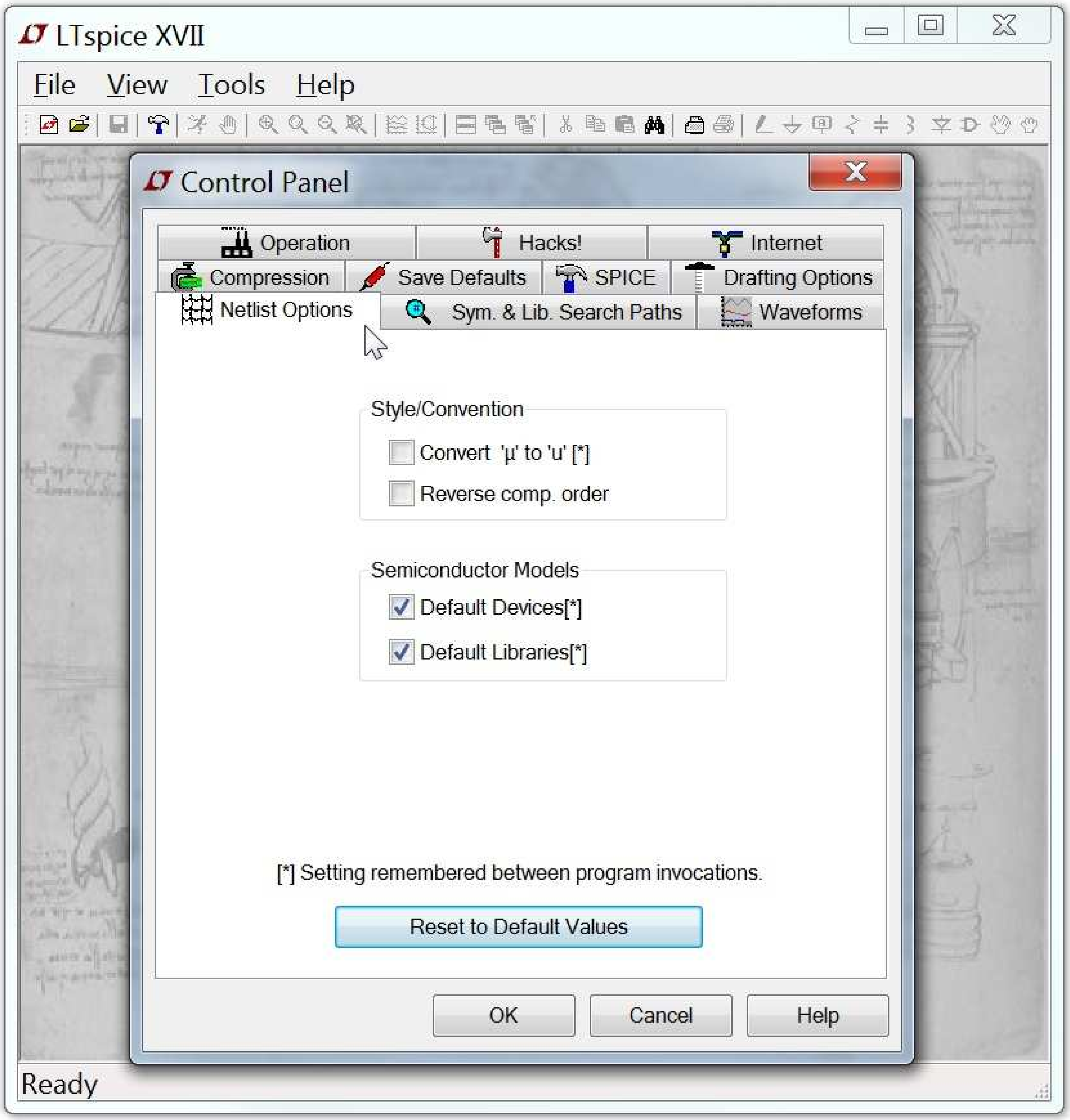
Pen thickness: Width of pen in pixels.

Draft with thick lines: Increases the all line widths. Useful for generating images for publication.

Show Title Block: For internal use.

Reverse Mouse Wheel Scroll: Normally, pulling the mouse wheel towards yourself moves the paper closer. If this option is checked, the same mouse syntax will pull your head further back.

Netlist Options



Style/Convention

□ Convert 'µ' to 'u' [*]

□ Reverse comp. order

Semiconductor Models

☑ Default Devices[*]

☑ Default Libraries[*]

[*] Setting remembered between program invocations.

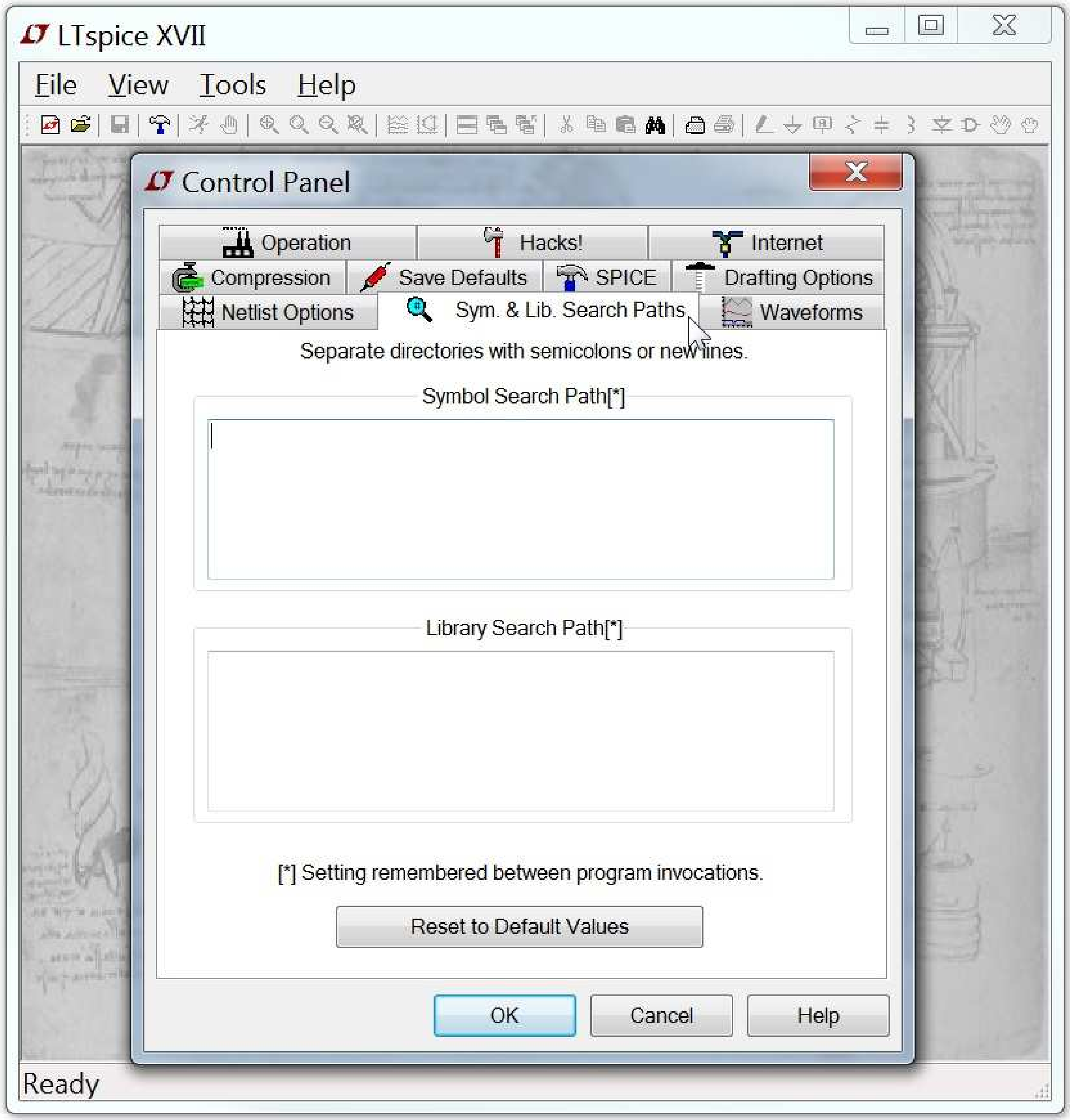
Convert 'µ' to 'u': Replace all instances of 'µ' to 'u'. Useful for generating netlists for SPICE simulators that don't understand the 'µ' character as the metric multiplier of 1e-6.

Reverse comp. order: Circuit elements are normally netlisted in the order in which they were added to the schematic. Checking this box causes this order to be reversed.

Default Devices: Whenever a diode is used in an LTspice schematic, the default model statement ".model D D" is added to the netlist to suppress messages about using the default model. Unchecking this option suppresses inclusion of this line as well as the analogous model statements for bipolar, MOSFET, and JFET transistors.

Default Libraries: Whenever a diode is used in an LTspice schematic, the default library, standard.dio, is included in the simulation by a .lib statement. Unchecking this option suppresses inclusion of this library as well as the analogous library statements for bipolar, MOSFET, and JFET transistors.

Symbol and Library Search Paths



Separate directories with semicolons or new lines.

Symbol Search Path[*]

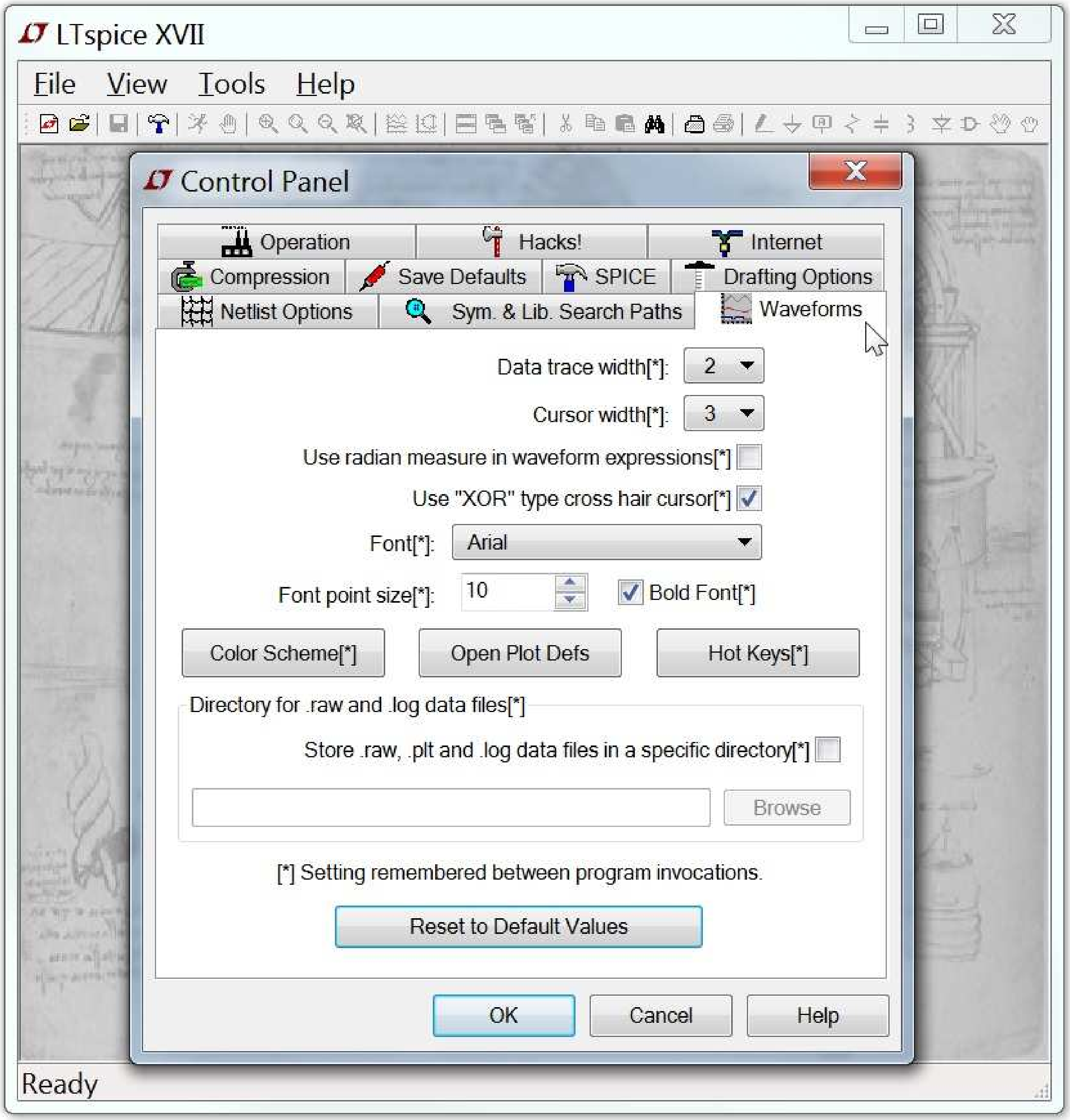
Library Search Path[*]

[*] Setting remembered between program invocations.

This panel allows you to enter additional paths than the default to find symbols and libraries. List each directory on it's own line.

Waveforms

This pane allows you to configure some aspects of the waveform viewer.



Data trace width[*]: 2

Cursor width[*]: 3

Use radian measure in waveform expressions[*] □

Use "XOR" type cross hair cursor[*] ☑

Font[*]: Arial

Font point size[*]: 10 ☑ Bold Font[*]

Directory for .raw and .log data files[*]

Store .raw, .plt and .log data files in a specific directory[*] □

[*] Setting remembered between program invocations.

Data trace width: Plot pen thickness in pixels.

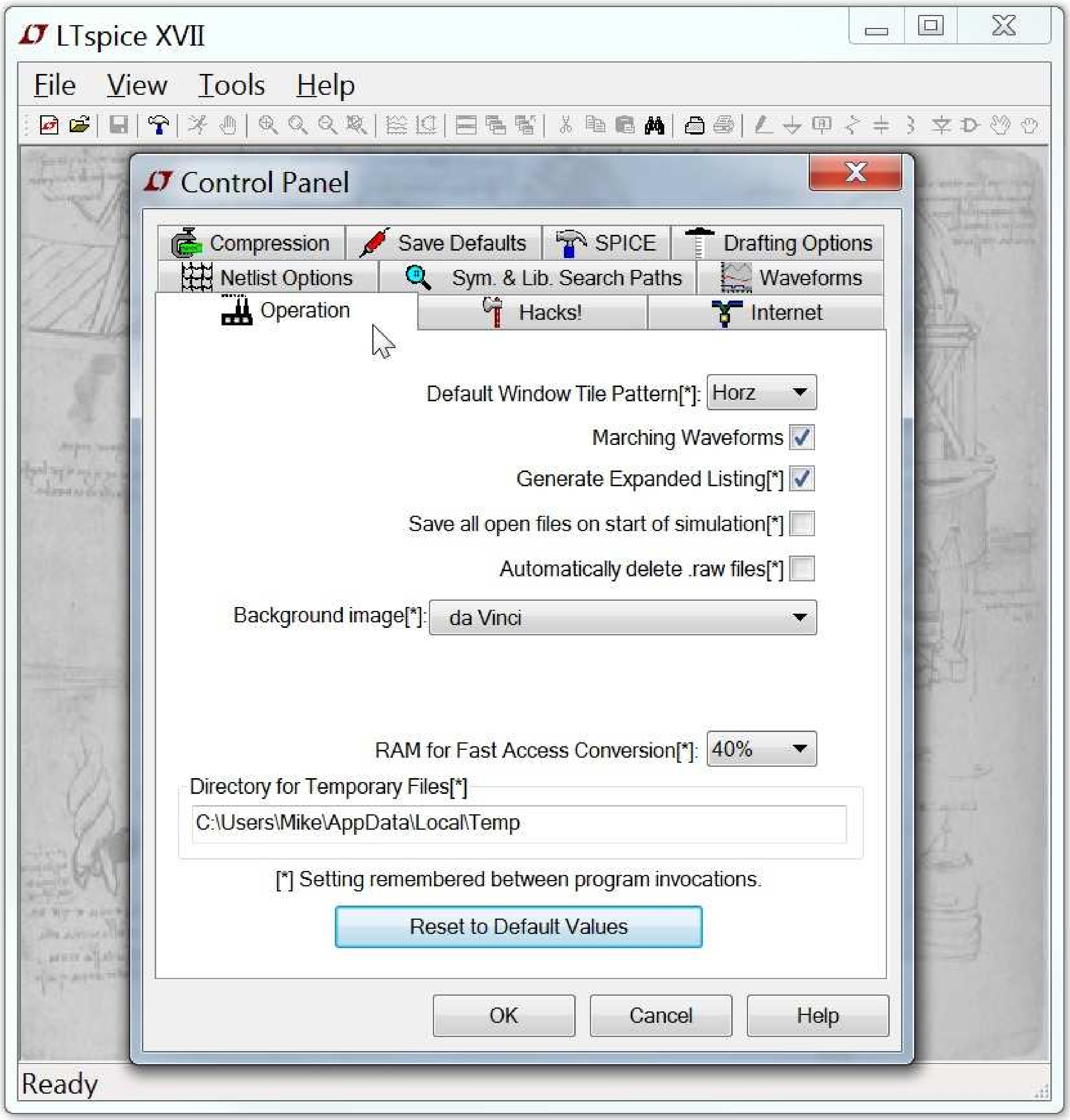
Cursor width: Thickness of the attached cursor and zoom rectangle dithered lines.

Use radian measure in waveform expressions: Determines where sin(90) is 1 or 0.8939966636005579.

Use "XOR" type cross hair cursor: The XOR cursor is a cross hair that is visible no matter what screen color is behind the cursor. For example, if the background is black, the cursor is white and visa versa. But note that it actually isn't an XOR function, because the cursor is still highly visible even against a grey background where the XOR'ed value has all inverted bits but the color is not distinguishably different. Using the "XOR" cursor is highly desirable, but not all video hardware and drivers get this right. Hence LTspice supports an opaque cursor that is in high contrast to the waveform window background but not necessarily in high contrast to the plot data. This less desirable cursor is the installation default since some hardware can't do the "XOR" cursor properly.

Directory for .raw and .log datafiles: This is useful if you want to specify a RAM disk or SSD drive for waveform data.

Operation



Default Window Tile Pattern[*]: Horz

Marching Waveforms ☑

Generate Expanded Listing[*] ☑

Save all open files on start of simulation[*] □

Automatically delete .raw files[*] □

Background image[*]: da Vinci

RAM for Fast Access Conversion[*]: 40%

Directory for Temporary Files[*]

C:\Users\Mike\AppData\Local\Temp

[*] Setting remembered between program invocations.

Settings marked with an asterisk [*] are remembered between program invocations.

Default Window Tile Pattern: Allows you to set a preference for windows tiled side by side instead of one above the other.

Marching Waveforms: Check to enable simulation results to be incrementally plotted during the simulation.

Generate Expanded Listing: Dump the flat netlist after expanding subcircuits to the SPICE Error Log file.

Save all open files on start of simulation: LTspice simulates the schematic in memory, not the one on the disk. This option forces the two to be in sync at the start of every simulation.

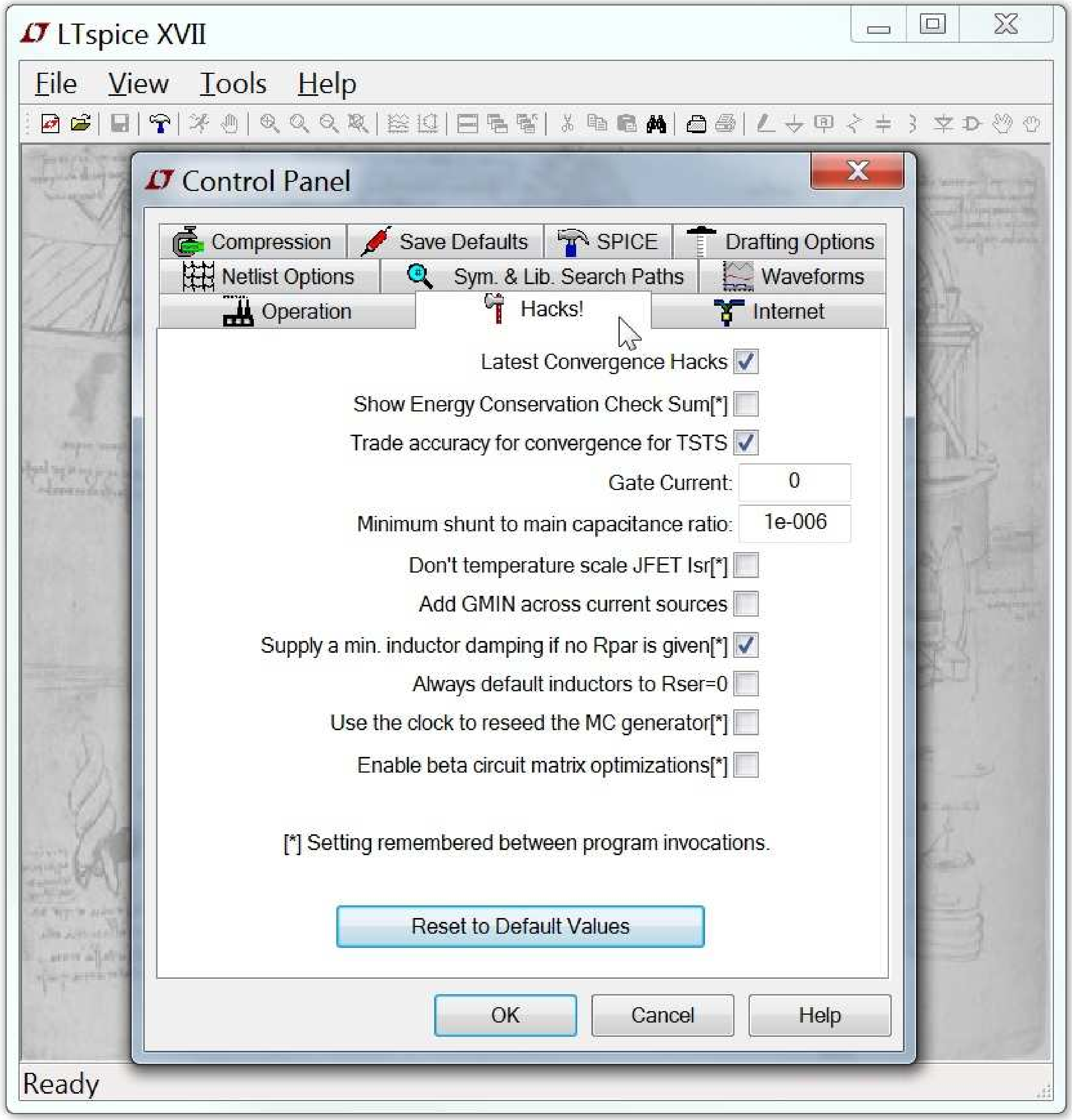
Automatically delete .raw files: This allows waveform data files to be deleted automatically after closing a simulation. This dramatically reduces the amount of disk space used by LTspice but requires the simulation to be rerun when you reopen the simulation.

Background image: Select the background of the main application window frame. The default is a stipple pattern so that that background isn't confused with a blank schematic. Other choices include a da Vinci drawing of a collection of his inventions or a user-supplied jpep file that needs to be stored at path location %USERPROFILE%\LTspiceXVII.jpg

RAM for Fast Access Conversion: This allows you to tune memory usage when you convert waveform data to fastaccess files format.

Directory for Temporary Files: Directory for temporary storage of update files downloaded during a Sync Release.

Hacks



Latest Convergence Hacks ☑

Show Energy Conservation Check Sum[*] □

Trade accuracy for convergence for TSTS ☑

Gate Current: 0

Minimum shunt to main capacitance ratio: 1e-006

Don't temperature scale JFET Isr[*] □

Add GMIN across current sources □

Supply a min. inductor damping if no Rpar is given[*] ☑

Always default inductors to Rser=0 □

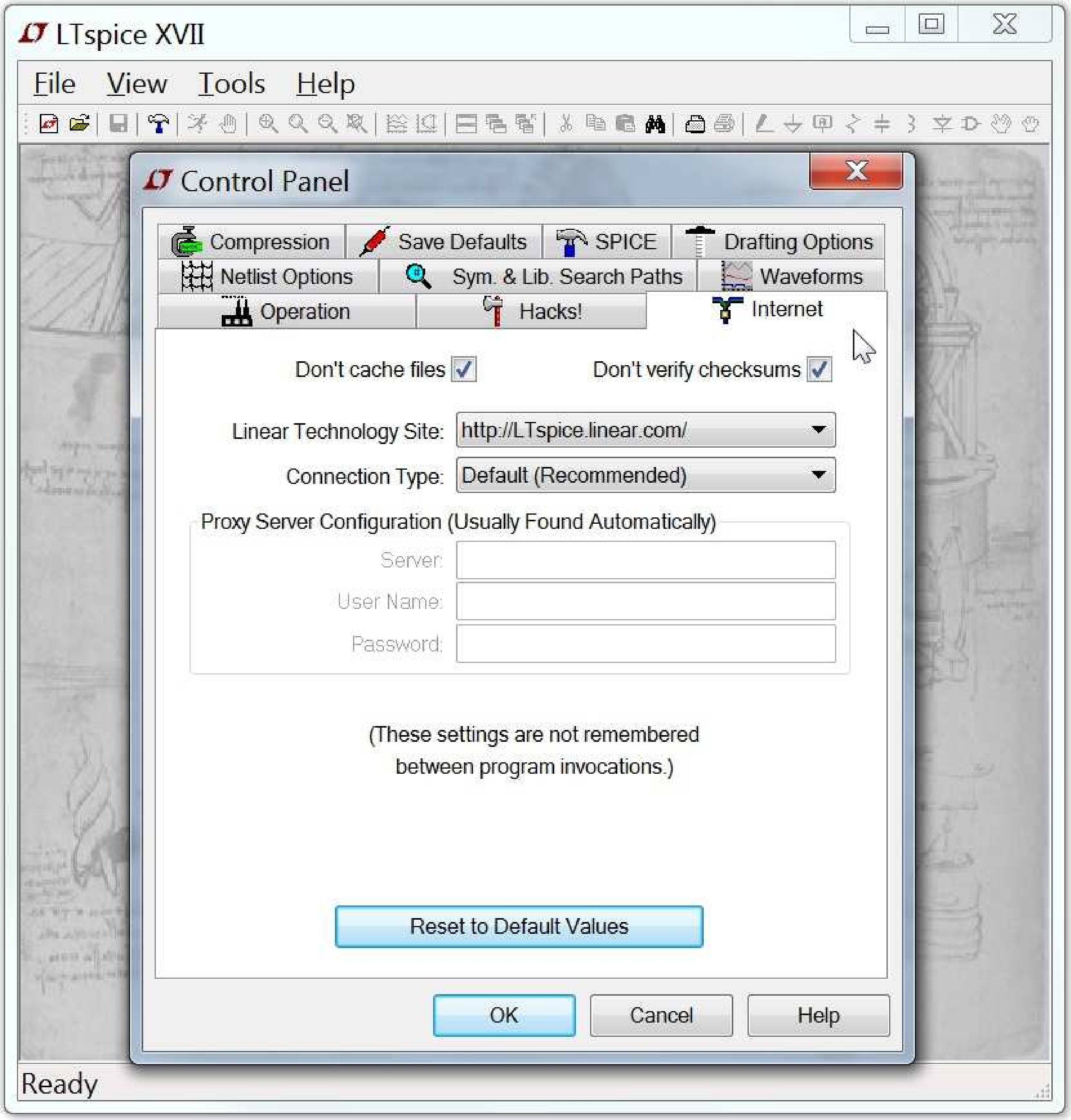
Use the clock to reseed the MC generator[*] □

Enable beta circuit matrix optimizations[*] □

[*] Setting remembered between program invocations.

This pane was used for internal program development, but is currently almost obsolete.

Internet Options



Don't cache files ☑ Don't verify checksums ☑

Linear Technology Site: http://LTspice.linear.com/

Connection Type: Default (Recommended)

Proxy Server Configuration (Usually Found Automatically)

Server:

User Name:

Password:

(These settings are not remembered between program invocations.)

This pane of the Control Panel is used for the incremental updates obtained from the web. LTspice is often updated with new features and models. Use the menu command Tools=>Sync Release to update to the current version. If you don't update for a couple months, LTspice will begin to ask if you would like to check for updates. LTspice never accesses the web without asking for your permission to do so. LTspice contains no spyware or transmits any type of data while obtaining the files it needs for updating.

Don't cache files: Neither cache nor use files cached on your machine for the update.

Don't verify checksums: For security reasons, LTspice uses a proprietary and confidential 128bit checksum algorithm to authenticate the files it receives off the web for updating. This authentication can be disabled in case there's an error in that algorithm. However, no problem with this has ever been reported, so it is not recommended that you ever defeat this security feature.

LTspice uses only high-level operating system calls for its Internet access. It should not be required to make any adjustments to these settings except in rare cases when you need to specify the Proxy server and password since LTspice is not managing the Internet access, but your computer and operating system. Settings on this pane are not remembered between program invocations.

Frequently Asked Questions(FAQ):

SPICE Differentiation

So just why is LTspice better than other SPICE programs?

Economics. If I write a SPICE program for an EDA company, it's possible to gross some smallish number of millions of dollars revenue. But if I write the same program for an IC company and it's used to both design and sell the ICs, then that same simulator is mission path to 1.5 billion dollars in revenue.

Sure, but what's the big TECHNICAL difference between LTspice and any other SPICE out there? Aren't all SPICE's based on Berkeley SPICE and essentially the same solver?

LTspice is a dramatically improved version of SPICE over any other SPICE program available. While most of these improvements are proprietary, here is an article that discusses a few of the differences between LTspice and other SPICE programs:

SPICE Differentiation, LT Journal of Analog Innovation, 2015

Note that article uses screen dumps from an old version of PSpice for comparison to the better solver in LTspice, but the netlists are machine readable and you can run them in the current versions of commercial simulators to see that (i) the problems persist and (ii) SPICE solver development as stopped at SPICE software companies.

Fine. I understand that article. It's pretty lucid, really. But when a boutique SPICE company comes by my office to promote their $100K/seat SPICE, they use a bunch of terms about numerical methods I don't understand. Do you have anything like that?

Sure, LTspice includes the following original methods:

- Stochastic Cooling of process threads: Eliminate inter-thread communication deadlocks -- it's the difference between a light bulb and a laser.
- Node Reduction: simulations run both faster and **more accurately** (like an FFT is both faster and more accurate than computing the DFFT from definition because there's less accumulated round off error because there so many fewer floating point operations to do).
- Predication Coalescence: Behavioral expressions are analyzed. Duplicate content is cached. Statistical analysis is applied to conditional evaluations. The result is that the speed of evaluation of runtime-parsed behavioral expressions approaches

that of SPICE devices written in native C-code. The global optimizer is based in graph theory.
- Self-authoring Assembly Language: Because of dynamic memory allocation, memory access is slow because when software is written, the return result from malloc() isn't know until runtime. Hence LTspice write a unique solver for each simulation run at runtime after the addresses returned malloc() are known. The speeds things up 3x.
- Opportunistic 32-bit Addressing: 64-bit programs need 64-bit pointers. For equal quality object code generation from the compiler, the 32-bit version of a program will run faster than the 64-bit version. However, since LTspice doesn't write the solver until malloc() is called, it can use 32-bit addressing if malloc() returned an address sufficiently close to the program counter(it almost always does). Not only does the smaller code execute faster, more of it fits in CPU cache, so opportunistic 32-bit addressing in a 64-bit program is a big win.
- Routines optimized to execute entirely in CPU cache: Disciplined guidance for writing fast numerical methods focuses on using CPU cache over CPU cores.

How can you possibly know whether a routine runs in on-chip CPU cache or the PCB mounted RAM?

Formerly I'd slice open the outer layer of insulation of my computer power cord so I could clamp an ammeter over one conductor to measure the current draw for different algorithms. That's not so effective on current machines -- even if you increase the computer's SMPS input filter capacitance(to reduce the power factor correction so a small change in power draw makes an exaggerated change in RMS current and needle deflection) -- so I just profile the code.

OK, I get it. But what about that GUI?

LTspice's GUI was based on a statistical analysis of the keyboard entry and mouse motion required to enter a schematic. Besides the overwhelming empirical success of LTspice's GUI, it actually is the easiest GUI to use for schematic entry.

Installation Problems

How do I install LTspice XVII?

1. Go to http://www.linear.com and download the file LTspiceXVII.exe into a temporary directory on your PC.

2. Execute the file LTspiceXVII.exe to install. You will need to run this as Administrator.

The download file, LTspiceXVII.exe, is digitally signed by Linear Technology Corporation.

I downloaded the LTspice installation program, but it tells me that it's not compatible with the version of Windows I'm running. Can I get a copy that will run under my Windows version?

You need to be using Windows 7, 8 or 10. Windows XP[1] won't run LTspice XVII. If you are using Windows 7, 8 or 10 and got a error message about the installation file, LTspiceXVII.exe, then it got corrupted during download. This can happen if the file on the server is revised(due to web site maintenance) while you're downloading it with a slow connection. You'll have to download it anew. Windows 7 x64 currently is the most popular platform for LTspice XVII.

1] LTspice IV is still available for Windows XP users, but LTspice IV is no longer being updated.

Program Updates

How do I get the latest version?

Once installed, there are two ways to getting the latest version. You can always reinstall the program again as mentioned in Installation Problems. The installer can optionally do an update instead or full install so you don't lose your preference settings. You don't have to remove the old version before installing. The other possibility to get the latest release is simply by using the Sync Release feature.

How do I know what new features are added?

After you have updated your file to the latest version, the file Changelog.txt in your root installation directory, usually at C:\Program Files\LTC\LTspiceXVII\Changelog.txt, has a detailed program revision list.

Can I go back to the old version after I execute the Sync Release command?

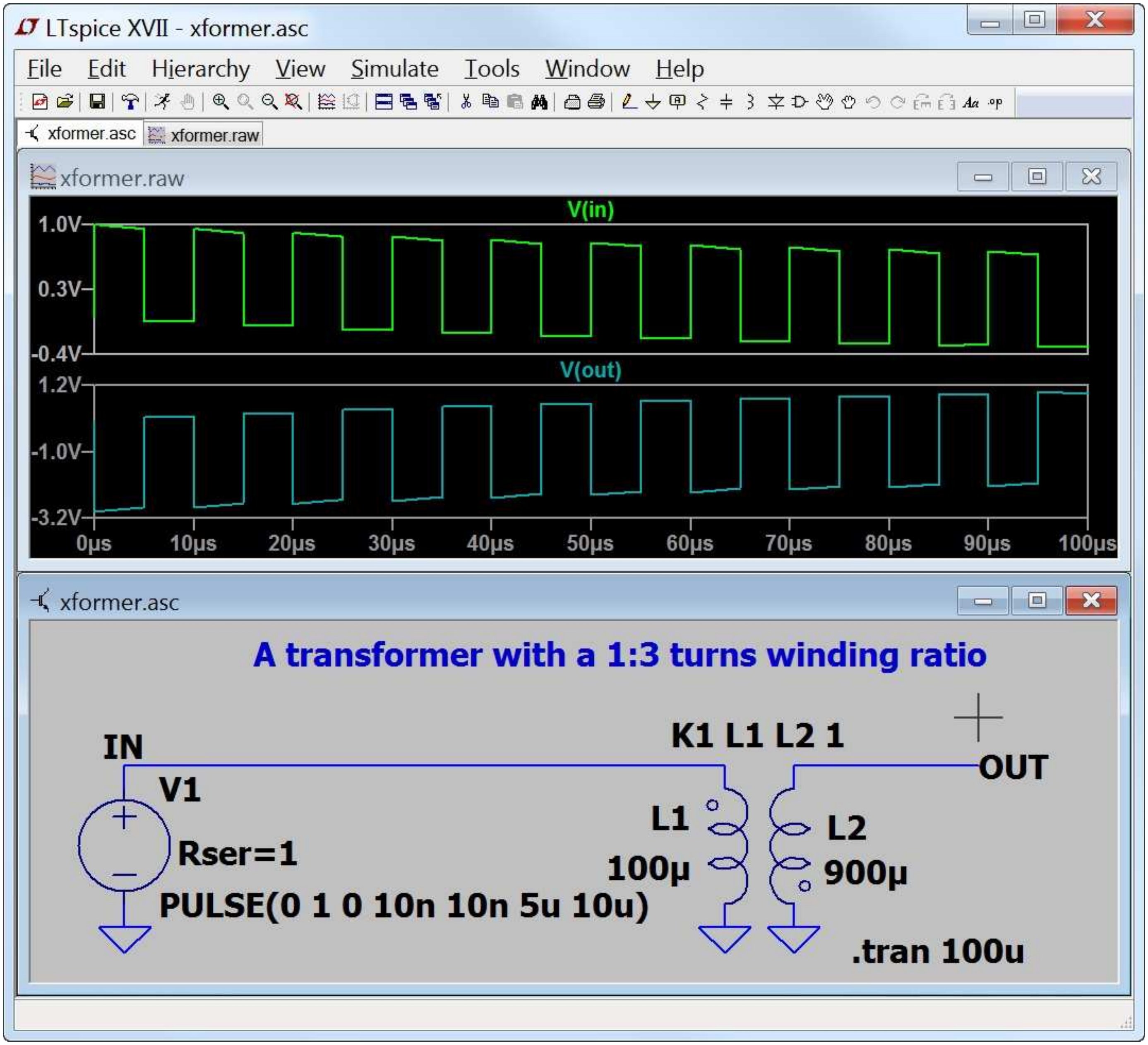
No. All symbols, models, and programs are updated with the current version. The component databases, standard.*, will be merged with the new ones automatically. If you added new inductors or capacitors, your devices will be preserved and merged with the new ones from the update.

Simulating Transformers

How do I simulate a transformer?

Draft coupled inductors as independent inductors and then add a mutual inductance statement placed as a SPICE directive on the schematic. See the section on mutual Inductance for more information. Inductors participating in a mutual inductance will be drawn with a phasing dot.

The following example illustrates a transformer with 1:3 turns ratio (one to nine inductance ratio) with a square wave input. The mutual coupling coefficient is set to 1 to model a transformer with no leakage inductance.



No, that is too easy. Don't you have anything more complicated?

Sure:

i) Measure winding L's with a DC LCR meter. Use these values directly in simulation.

ii) Measure winding ESR with an ohmmeter. Use twice these values in simulation since the resistance at frequency is higher than the DC value. My experience with ferrite and switchers is that it is usually nearly a factor of two higher.

iii) Short all but most inductive windings and measure the leakage inductance with the DC LCR meter. Adjust the coupling coefficient to match this, or for the case of two windings:

```
K = sqrt(1-Lleak/sqrt(L1*L2))
Lleak = sqrt(L1*L2)*(1-K*K)
```

iv) Find the transformer's resonate frequency and Q. Specify a Cpar and Rpar for the most inductive winding to match this resonance.

v) Enjoy.

What about non-linear Transformers?

Please see the example installed as
%HOMEPATH%\Documents\LTspiceXVII\examples\Educational\NonLinearTran

Third-party Models

This section explains the basics to adding a third-party model to LTspice XVII.

Basically there are two types of third party SPICE models, those described with a .MODEL statement and those defined with a .SUBCKT.

Models given as .MODEL statements are for intrinsic SPICE devices like diodes and transistors. The .MODEL statement gives the parameters for the specific component. The behavior of the device is already known by SPICE, only the parameters need to be given to finish specifying the component's electrical characteristics.

On the other hand, models given by .SUBCKT statements define the modeled component by a collection of circuitry of intrinsic SPICE devices. For example, the SPICE model of an opamp would be given as a subcircuit.

The way how to include the model in LTspice depends on whether the model is given as a .MODEL statement or a .SUBCKT.

Example for an NPN transistor defined with a .MODEL statement:

1. Add an instance of the symbol NPN to your schematic.

2. Edit the value "NPN" to be "BC547C" to coincide with the name used in the target .MODEL statement.

3. Now either

   3a) Add the .MODEL BC547C... statement as a SPICE directive on your schematic.

   or

   3b) If you have a file bipol.lib containing your .MODEL BC547C... (other models may be too in this file), then add the SPICE directive ".INCLUDE bipol.lib" on your schematic. Note that "bipol.lib" must be the complete name with any file extensions and that Windows Explorer defaults to not showing the file extension. So you if you have a file called "bipol.lib.txt", which you can edit/view in notepad, and Windows Explorer shows you the file exits as "bipol.lib" The SPICE directive to include this file is ".inc bipol.lib.txt" If you used, ".inc bipol.lib" you will get an error message that that file can't be found.

or

3c) You can alternatively add the .MODEL BC547C... statement to the file typically installed at %HOMEPATH%\Documents\LTspiceXVII\lib\cmp\standard.bjt. If you do that you will automatically see the model as a choice was editing the NPN transistor. If you edit this standard.bjt file outside of LTspice, you will have to restart LTspice for it to notice that the file has changed.

Example for a 5-pin opamp. This will be defined with a .SUBCKT statement:

1. Add an instance of symbol opamp2 to your schematic.

2. Edit the value "opamp2" to "TL072" on the schematic to coincide with the name of the .SUBCKT.

3. Either

3a) Paste the ".SUBCKT TL072 ..... .ENDS" definition as one multi-line SPICE directive to your schematic.

or

3b) If you have a file called "TI.lib" containing the definition of subcircuit TL072(It will look like a line that starts out as ".SUBCKT TL072...") add the SPICE directive ".INCLUDE TI.lib" to the schematic.

It is possible to create a new symbol and program it to automatically include the necessary model for the simulation whenever it is used on a schematic. See help section Schematic Capture=>Creating New Symbols.

It is possible to create an automatically generated symbol that netlists correctly against an arbitrary third party model and have it programmed so that it includes the necessary model for the simulation whenever it appears on a schematic. See help section Schematic Capture=>Creating New Symbols. For most users, this is the only method you should consider for adding new models defined as subcircuits since all the details are handled for you.

Example for a 3-pin NPN transistor but defined with a .SUBCKT statement:

1. Add an instance of symbol NPN to your schematic.

2. Move the cursor over the body of the newly-placed NPN symbol instance. Press <Ctrl>RightMouseButton. A dialog box will appear. Change Prefix: QN to Prefix: X. This causes this instance of the symbol to netlist as a subcircuit instead of an intrinsic bipolar transistor.

3. Edit the value "NPN" to be "BFG135" to coincide with the name given on the .SUBCKT line.

4. Then either

   4a) Add the .SUBCKT BFG135 lines to your schematic

   or

   4b) If you have a file Phil.lib containing your .SUBCKT BFG135 .... (others may be too in this file) then you have to add a SPICE directive .INCLUDE Phil.lib

One aspect of adding a .SUBCKT model to LTspice is that you need have the symbol used to call the subcircuit and the model agree on the same pin/port netlist order. The above examples assume the 3rd party model you're adding follows popular pin order conventions.

Further related information is in the help sections Schematic Capture and LTspice. The basic idea is that the schematic capture program generates a netlist that the simulator, LTspice reads. Any aspect of importing 3rd party models can be resolved by understanding SPICE netlist syntax and how the schematic capture program generates that syntax. There are also tutorials prepared on this topic archived at the independent users' group at http://groups.yahoo.com/group/LTspice.

Inductor Models

How do I design a coupled inductor?

You first (i) draw at least two inductors and then (ii) define the K coefficient between the two inductors. See mutual inductance section.

How do I control the inductor parasitic resistance?

By default, LTspice will supply losses to inductors to aid SMPS transient analysis. For SMPS, these losses are of usually of no consequence, but may be turned off if desired. On the "Tools=> Control Panel=>Hacks!" page, uncheck "Supply a min. inductor damping if no Rpar is given" This setting will be remembered between invocations of the program. There is also a default series resistance of 1 milliohm for inductors that aren't mentioned in a mutual inductance statement. This Rser allows LTspice XVII to integrate the inductance as a Norton equivalent circuit instead of Thevenin equivalent in order to reduce the size of the circuit's linearized matrix. If you don't want LTspice to introduce this minimum resistance, you must explicitly set Rser=0 for that inductor. This will require LTspice to use the more cumbersome Thevenin equivalent of the inductor during transient analysis.

Can I add/edit my own inductor model?

Open the file %HOMEPATH%\Documents\LTspiceXVII\lib\cmp\standard.ind to add or edit inductor models.

MOSFET Models

What is the difference between LTspice XVII MOSFET and standard SPICE MOSFET models?

Besides the standard SPICE MOSFET models, LTspice XVII also includes a proprietary MOSFET model that is not implemented in other SPICE programs. It directly encapsulates the charge behavior of the vertical double diffused MOS transistor. This allows a power device to be modeled with an intrinsic VDMOS device LTspice instead of a subcircuit as in other SPICE programs. See the MOSFET section for details.

Can I add my own MOSFET models?

Yes, you can add your own model in the file %HOMEPATH%\Documents\LTspiceXVII\lib\cmp\standard.mos. This file is only for devices defined with a .model statement, not as subcircuits. If you want to use a subcircuit, follow the following steps:

1. Change the "Prefix" attribute of the component instance of the symbol to be an 'X'. Don't change the symbol, just the instances of the symbol as a component on a schematic. You can access this attribute by holding down the control key and right clicking on the body of the component.

2. Edit the "Value" attribute of the component to coincide with the name of the subcircuit you wish to use.

   Add a SPICE directive on the schematic such as ".inc filename" where filename is the name of the file containing the definition of the subcircuit. Note that this must be the complete name with any file extension and Windows Explorer defaults to not showing the file extension. So you if you have a file called "mylib.sub.txt", which you can edit/view in notepad, and Windows Explorer shows you the file exists as "mylib.sub" The SPICE directive to include this file is ".inc mylib.sub.txt" If you used, ".inc mylib.sub" you will get an error message that that file can't be found.

License and Distribution

Can I re-distribute the software?

Yes, you can distribute the software freely whether you are a Linear Technology customer or not. See the license section for more details.

Is it a shareware, freeware or demo?

This program is not a shareware or a demo. It is fully functional freeware. The purpose of this software is to help our customers use our products. It can also be used as a general-purpose circuit design package with schematic capture and SPICE simulation. We do encourage students using the program to become familiar with the analog design process. We cannot guarantee support for non Linear Technology related program usage, but we'll fix all general program bugs and appreciate such reports. We do extensive in-house testing and believe the program has superior convergence capability. There are no known outstanding bugs.

Who can I contact at Linear Technology for help?

For all software issues, e-mail LTspice@linear.com. Linear Technology Corporation does not offer phone support for general LTspice usage questions.

For all hardware issues, such as additional application information for Linear Technology IC's, call Linear Technology application department at (408) 954-8400 during normal US Pacific business hours.

Circuit Efficiency Calculation

How can I get an efficiency report for my circuit?

You need to add the keyword "steady" steady to the .tran command, e.g., ".TRAN <time> steady". The program will detect the steady state by checking the internal state of the switcher macromodel. It doesn't work when a switching regulator part is not part of the circuit because the steady state detection is implemented in the model -- usually by looking for the the current flowing out of the error amplifier to drop to zero as integrated over a switching cycle. There must be exactly one voltage source in the circuit. This will be identified as the input. There must be exactly one current source in the circuit, though a resistor with the name of Rload can be used instead. This will be identified as the load. After the simulation is done, you can select the 'Efficiency Report' under the 'View' menu to see the report on the schematic.

Custom Symbols

Can I create my own symbols?

Yes, you can create your own symbols.

How do I create my own symbol?

Start with the menu command File=>New Symbol.

Can I create my own switching regulator models?

Not very easily. The switching regulator models that ship with LTspice XVII use a new hardware description language and new intrinsic SPICE devices designed to encapsulate the behavior of LTC's switching regulator products. Even if you succeed in making a model with standard SPICE primitives, the simulation will run orders of magnitude slower. Note that some people have made such switching regulator models with standard SPICE devices. LTspice can run these models and will usually outperform the simulator for which they were targeted.

Memory Problems

How much memory do I need to run the program?

If you can run Windows, you can run LTspice XVII. We have spent a great deal of effort in minimizing the memory requirement of this program. There are no memory leaks. But waveform data requires memory and that is where people run into trouble. An x64 OS will be the best choice in this regard.

Where is the waveform stored during simulation?

All the waveform data are stored on disk. Only the plotted traces are loaded into RAM. Turning off the marching waveforms can reduce the memory requirement. Note that for most analysis types, there is no particular file size limit. You can generate and view .raw files that are very many Gigabytes in size.

What if I don't have enough disk space for long simulation?

The waveform data has been compressed, but it is still proportional to the run time and the number of traces saved. The easiest way to save memory is to select only the desired traces for storing before the simulation starts.

OK, I've done everything and I'm still running out of memory. What can I do?

During a transient analysis, you can interactively throw away the past waveforms by pressing the '0' key. That will retrigger the simulation time to t=0 as the present time.

Model Compatibility

Are the switching regulator models compatible with PSpice models and others?

The LTspice SMPS macromodels are implemented in a combination of new proprietary native LTspice devices and/or a proprietary hardware description language. While it is possible, in principle, to develop generic SPICE or PSpice macromodels, the resultant simulation speed would not be viable. LTspice can, however, run PSpice semiconductor and behavioral models and is generally a much higher performance simulator, so you might move your PSpice simulations to LTspice. Most SPICE users have already upgraded from PSpice to LTspice.

SPICE Netlist

How do I create a SPICE netlist?

A netlist can be created with any text editor capable of generating an ASCII file. You can view the SPICE netlist of any schematic in LTspice XVII with the command View=>SPICE netlist. From this view you can copy the netlist to the clipboard by selecting all text and typing Ctrl-C to bring the netlist to a different editor.

How do I run a netlist?

Just open the text file first and then run it. LTspice XVII will recognize the file as a netlist if it has file extension of ".cir"

Exporting/Merging Waveform Data

Can I export the waveform data to other applications?

You can copy a plot as bitmap by making a waveform window the active window and typing Ctrl-C. Then, in an application that accepts bitmap pastes from the clipboard like Word or Paint, type Ctrl-V. Note that this also works for bitmaps of schematics. These images can also be exported as Windows metafiles(Menu command Tools=>Write to a .wmf file) which writes the image as vector graphics to a .wmf file that can be imported in various desktop publishing tools. When exporting a metafile of waveform data, you first go to Tools=>Control Panel=>Waveform=>Font and select Arial. The default, System, is highly legible on a CRT, but is a fixed font that does not scale correctly in metafiles.

OK, that works for bitmaps, but can I get the data itself to an application like Excel?

There is an export utility(Waveform Menu: File=>Export) that allows data to be exported to an ASCII file. There is also a 3rd party free utility written by Helmut Sennewald. It is available from the independent users' group http://groups.yahoo.com/group/LTspice. This utility allows various forms of manipulation of the data including the ability to merge waveforms from different simulation runs.

Who is Helmut Sennewald?

The guy on the right:

Elektronika, Munich, Nov 17, 2006

Helmut Sennewald is also the moderator of the independent LTspice Users' Group.

But isn't there any way to export the waveform data to other applications without resorting to 3rd party software?

Yes. Make the waveform window the active window and use menu command File=>Export.

But I want the data in equally spaced timesteps. Is there anyway to do that?

Yes. Do the the FFT of the desired data. Before the FFT, the data is interpolated to equally spaced time steps. Now do the FFT on the FFT'ed data. That will recover the equally spaced time-step data and export that.

But if I do the FFT twice, won't I lose accuracy?

No. LTspice's FFT algorithm is bit-accurate to double precision.

But what if I don't want the data interpolated to a number of FFT bins that is a power of 2?

Then enter the number of bins you want to use. LTspice uses a proprietary FFT algorithm that works for an arbitrary number of bins.

How is it possible that LTspice's FFT is bit-accurate to double

precision and works for an arbitrary number of bins?

That is a trade secret.

Extracting Switch Mode Power Supply Loop Gain in Simulation and Why You Usually Don't Need To

The gain of a negative feedback loop needs to fall with frequency below unity before too much phase shift occurs unless your aim actually is to make an oscillator[1]. This idea can be applied to the stability analysis of a Switch Mode Power Supply(SMPS). Even though a SMPS is an intrinsically non-linear circuit with no small-signal linear equivalent circuit, there typically is an analog feedback loop operating on the filtered, switched output.

There are two issues involved in determining the loop gain of a SMPS (i) obtaining the open loop gain from the closed loop system and (ii) ignoring the switching waveform by averaging over a switching cycle and/or using Fourier analysis to ignore the switching frequency components. The first issue is common to most stability analysis of feedback loops. Stability analysis is based on the open loop response but if you break the feedback loop to measure open loop response directly, the circuit doesn't work anymore which was why feedback was used in the first place. The second issue arises from the fact that a SMPS is an intrinsically non-linear circuit and linear feedback theory is basically restricted to the hypothetical waveform averaged over a switching cycle.

Determining the open loop response of a linear, closed loop system is a problem solved well by Middlebrook's method[2]. That method uses test signals injected into the closed loop system to independently solve for the voltage and current gains. These two gains are then convoluted together to get the true loop gain. If a point in the feedback loop can be identified where a low impedance drives a high impedance, then the current gain is zero and it is sufficient to measure only the voltage gain and identify that as the loop gain. Such a point can normally be found in a SMPS since you have a power supply output driving an error amplifier input.

Laboratory measurement of a SMPS loop gain is automated with commercial instrumentation pioneered by Venable Corporation and now also available from other companies. The technique of using injected test signals and Fourier analysis is called Frequency Response Analysis(FRA). While this method is routine in the lab, not everyone is aware of how to use it simulation. This article explains how to do FRA in LTspice XVII. The method uses the voltage gain part of the Middlebrook method, .measure statements to do the Fourier transform, a step statement to sweep frequency, and the feature in LTspice that allows one to plot the results of .measure statements. In reading through the steps below, you might

want to refer to the working FRA examples that are part of the general LTspice XVII release typically installed in directory
%HOMEPATH%\Documents\LTspiceXVII\examples\Educational\FRA\

Step 1: Identify a point in the SMPS feed back loop where a low impedance source is driving a high impedance input. Two places are useful for this, either in series with the feedback pin of the SMPS controller or between the output to the top of the resistor divider going to the feedback pin.

Step 2: Insert a voltage source here. This will be a time-domain sine wave that perturbs the feedback loop. Give it a value of "SINE(0 10m {Freq})" The choice of amplitude(here 10mV) will impact accuracy and the signal to noise of the method. The smaller the amplitude, the lower the signal to noise. But if the amplitude is too large, the system is not operating linearly and frequency response becomes less relevant since the frequencies are no longer independent.

Step 3: Label the nodes to either end of this voltage source "A" and "B" The direction of feedback should be from node A to node B. For example, if the voltage source is connected directly to the feedback pin, node B is the feedback pin and node A is the one on the other side of the voltage source.

Step 4: Paste the following .measure statements on the schematic as a SPICE directive:

```
 .meas Aavg avg V(a)
 .meas Bavg avg V(b)
 .meas Are avg (V(a)-Aavg)*cos(360*time*Freq)
 .meas Aim avg -(V(a)-Aavg)*sin(360*time*Freq)
 .meas Bre avg (V(b)-Bavg)*cos(360*time*Freq)
 .meas Bim avg -(V(b)-Bavg)*sin(360*time*Freq)
 .meas GainMag param
20*log10(hypot(Are,Aim)/hypot(Bre,Bim))
 .meas GainPhi param mod(atan2(Aim,Are)-
atan2(Bim,Bre)+180,360)-180
```

These .measure statements perform the Fourier transform of nodes A and B and then compute the ratio of the resultant complex voltages. The result is the complex open loop gain of the system. The magnitude is given by GainMag in dB and phase as GainPhi in degrees.

Step 5: Paste the following on the simulation command on the schematic as a SPICE directive:

```
.param t0=.2m
.tran 0 {t0+10/freq} {t0}
```

Parameter t0 is the length of time required for the system to come to steady state. You will probably have to run a few simulations to determine an appropriate value for t0. It occurs as the third parameter on the .tran command, meaning is it the time the simulator should start saving data. This prevents the .meas statements of Step 4 from using this data in the analysis. This is done because initial transient conditions might not be operating within the small perturbations from regulation that could be considered small signal response.

Notice that t0 appears in both the 2nd and 3rd parameters of the .tran command. The 2nd parameter is the stop time. The difference between start and stop times has been chosen as 10/freq, i.e., an integral number of perturbation cycles. Ideally, the Fourier analysis would be done over a period that is both an integral number of perturbation cycles and switching cycles, but his isn't always possible. Since loop gain must drop to less than unity at a frequency that is a fraction of the switching frequency, there are always more switching cycles than perturbation cycles and an integral number of perturbation cycles is used with the hope the error from a non-integral number of switching cycles will be small since many switching cycles are included.

Step 6: Choose which frequency or frequencies at which to perform the analysis. To do a single frequency, simply add this SPICE directive:

```
.param Freq=15K
```

and run the simulation. The output of the .meas statements are in the error log which you can view after running the simulation with menu command View=>SPICE Error Log. You can run the simulation at multiple frequencies by placing the following SPICE directive on the schematic:

```
.step oct param freq 50K 100K 5
```

This directive tells LTspice to run the simulation at frequencies from 50kHz to 100kHz using 5 points per octave. To plot this as a Bode plot, after the simulations complete, execute menu command View=>SPICE Error Log and then right click menu "Plot .step'ed .meas data" At this point, the Bode plot will not have any data plotted. so right click again and execute menu command "Visible Traces" and then select gain.

Armed with the above technique, one might feel ready to go and conquer SMPS design with Bode analysis of the feedback loop. I understand the temptation. It'd be rewarding if one could traverse the feedback loop identifying the components that gave rise to the poles and zeros, strategize which zeros to move to cancel which poles, and synthesize component values for the compensation network components to achieve a stable feedback loop. But that's pretty much exactly what you can't do with this technique or any other frequency domain technique. Let me explain why.

Let's consider a typical fixed-frequency, peak-current mode switcher such as that in

```
%HOMEPATH%\Documents\LTspiceXVII\examples\Educational\FRA\Eg3.asc
```

The controller uses a flip-flop which is set by a clock pulse and turns on the switch which ramps the inductor current up. Once the peak switch current is proportional to the voltage on the output of the error amplifier, the flip-flop is reset, the switch turns off and the controller sits idle while until the next clock pulse sets the flip-flop again. Since average current is proportional to peak current up to a geometrical factor, if we average over one clock cycle this flip-flop controlled-switch behaves like a transconductance. That is the current through the switch is proportional to the voltage on the output of the error amplifier. Now if continue on along the feedback path, we have the inductor in series with the switch current. Since the switch is a current source, the series impedance of the inductor, even though it is reactive, causes no phase shift. This is actually the point to current-mode control and why you buy that controller. Continuing on the feedback path, we are now at the output of the SMPS. The output filter capacitor(C4) gives rise to one pole. The output is then divided by the feedback resistive divider and compared to a reference voltage at the feedback pin. The difference between the divided output and the reference voltage is the error voltage. This error voltage is amplified the the error amplifier to be a current which flows out of the error amplifier. But it is the voltage on the output of the error amplifier and not the current flowing out of it that determines switch current so to complete traversing the feedback loop, we need to convert that current to a voltage. We could do that with a resistor and that would work, but a much better idea is to use a capacitor(C1) because that will maximize the open loop DC gain to keep the output regulated to a stiff voltage. That capacitor makes a second pole.

Now, since each pole can cause a phase shift infinitesimally close to 90° and the controller must cause some additional delay, one might think that some circuit design is necessary to ensure a stable feedback loop. But that's not really the case particularly if one is using an aluminum electrolytic cap output filter capacitor because it has ESR and that will put a zero in the response. Also, since we buy compensation cap C1 a series resistor, R1, that also puts another zero in the response. Further, the delay from the controller is a very small fraction of the switching frequency. At the loop crossover frequency and below, that delay is negligible. This all means that the loop is stable and it isn't possible to synthesize component values since the loop is stable for all component values. This argument basically is pointing out that as soon as the signal regulated by the feedback loop of a current mode SMPS is well- described by the current averaged over one switching cycle, that loop is stable.

If the output filter cap isn't an aluminum electrolytic but a ceramic capacitor, the ESR of a ceramic capacitor isn't high enough to substantially impact the stability of the SMPS. So the loop response, per the previous discussion, would now be two poles and one zero so it should still be stable independent of the specific component values of the output filter cap or RC circuit attached to the error amplifier output. But it would be appropriate to discuss the limits of applicability of the above analysis. There is an effect that degrades from the accuracy of the above description of current mode SMPS stability. The average current isn't proportional to the peak current over variation of output voltage because the duty cycle, and hence ripple current, changes with output voltage so the same peak current that trips the controller flip-flop does not give the same average current over changes in output voltage. This means that the transfer function from the voltage on the output of the error amplifier to the current flowing into the inductor isn't perfectly described as a transconductance, but a transconductance shunted with some real impedance. This impedance is typically several Ohms, which while very large compared to the on resistance of a MOSFET, it is less than infinite. This is not desirable from a stability point of view since the inductor is no longer fed from a current source and its reactance can cause some phase shift. This situation is further degraded by slope compensation. Slope compensation is the fix for a subharmonic oscillation that occurs in fixed frequency current mode controllers operating at high duty cycle. The technique entails adding a spoofed current to the measured switch current and using that quantity to reset the controller's flip-flop. The impact of using a quantity other than current to reset the flip-flop reduces the impedance of the current source feeding the inductor so the inductor's reactance causes yet more phase shift.

By and large, I find it pretty hard to make a current-mode SMPS unstable. For example, if you use an inductance value that is too high by an order of magnitude or two, then inductor ripple current becomes very small and the spoofed current of the slope compensation controls the flip-flop reset. That will reduce the impedance of the switched source driving this inductance to the impedance of the MOSFET Rds(on) so the inductor creates another pole in the loop and that causes instability. But in that situation, even though you're using a current mode controller, the power supply is actually running in voltage mode. Small signal linear analysis of voltage mode power supplies is quite useful because unless the feedback loop has been contrived to cancel one of the poles, the power supply will oscillate and may blow itself up the first time it is turned on. Current-mode supplies are quite different. While it is possible to do small-signal linear analysis of a current-mode switcher, there just isn't much engineering to be accomplished with the method since the feedback loop is stable as long as the power supply really is operating in current mode.

The last advise I can offer answers how one can be sure that a SMPS is stable and operating in current mode. The answer is to start with the schematic on the front page of the datasheet. The critical information there are the inductance value, output filter capacitance, and external compensation component values. Some datasheets give equations for computing these values, but I just start with those values and adjust using time domain simulation to evaluate the response. After all, the whole point of frequency domain analysis is to improve the time domain response. With current-mode switchers, it usually more direct to jump right to time-domain simulation to check overshoot since stability has already been achieved.

[1] The discussion is restricted to minimum-phase systems.

[2] R. David Middlebrook, "Measurement of Loop Gain in Feedback Systems", International Journal of Electronics (vol 38, no. 4, pages 485-512, April 1975).

Running Under Linux

Do you have a Linux version of this program?

No. But there are reports that LTspice XVII does run on Linux under WINE.

OK, I've never used WINE, how do I install this?

Check with http://www.winehq.com to find the current version of WINE for your system.

Copy the appropriate .rpm file to your machine and open it from nautilus.

Get the file LTspiceXVII.exe from http://www.linear.com. In an xterm, execute "wine LTspiceXVII.exe" to install LTspice.

There will now be a Linear Technology Logo on your desktop. Double click it to start or type "wine LTspiceXVII.exe" from an xterm to start the program.

The schematic fonts don't scale as smoothly under WINE as Windows. Why is that?

WINE is doing the best it can with the fonts it finds. It will do better if you tell it how to find the font files from your Windows system.

The PWL additional point editor doesn't look right under WINE?

Try using the native Windows .dll from your licensed Windows system. The command line to then invoke LTspice from WINE is wine -dll commctrl,comctl32=n XVIIx64.exe.

How does the performance running under Linux compare to running under Windows?

Every Linux user you ask will tell you that LTspice runs better under Linux than Windows.

Wow, cool! Does it really?

No.

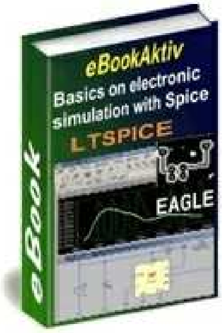
Paper Manuals

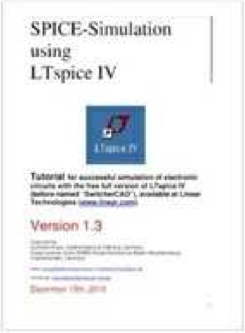
What about a Paper Manual?

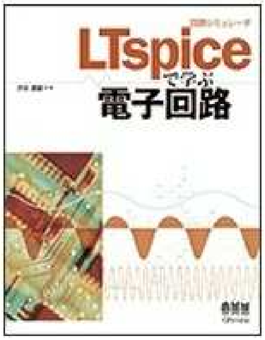
These help pages are the manual. They are set up such that you can print them in one complete set, but please think of the environment before doing so.

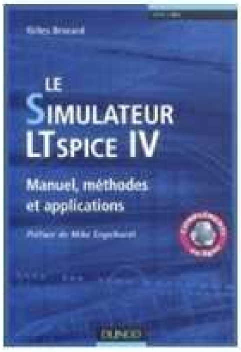
Then what about a book?

Checking Amazon, I see 10 books with LTspice in their title:









The books by Michio Shibuya, Gilles Brocard, and Mihail Pushkarev(book cover not illustrated above)are high quality. The graphic on the title page of this help document is due to Michio Shibuya.

Tutorials

What about Tutorials?

I thought you'd never ask! Please check out my coworker's site at http://www.simonbramble.co.uk. That site has numerous tutorials on LTspice and well as the broader area of analog circuit design. Recommended.

No, no, I want a movie. Where can I get an LTspice movie?

There's also numerous videos as well as a blog at http://www.linear.com/solutions/LTspice.

What about those LTspice seminars?

Linear Technology continuously offers interesting design seminars. Check the event schedule here.

Independent Users Group

What about a Users' Group?

There is an independent users' group at http://groups.yahoo.com/group/LTspice. The group has a Files section with additional tutorials, libraries, and examples. The group has 59,817 members at time of this writing.

Web Update(Sync Release)

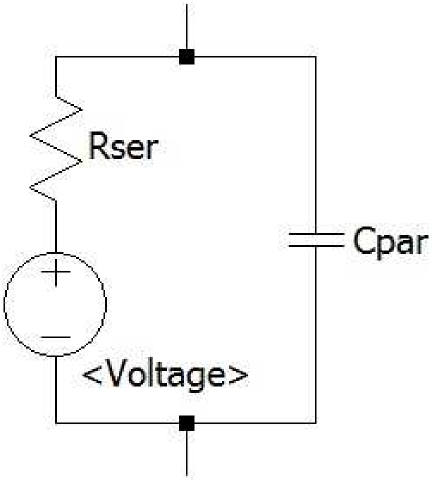
All windows named LTspice XVII must be closed before the Sync Release command can be activated. The user needs to establish the internet connection first. The LTspice XVII program will then download the master index file(release.log) from the LTC web server. The master index file contains the checksums for every file in the sub-directories. The local file's checksum is then calculated and checked against the one in the master index file. The file on the web server will then be downloaded automatically if there is a difference in checksum. LTspice XVII program files that were saved under the same name will be overwritten! Most of the macromodels are less than 3KB and can be transferred in a few seconds. During the update of the XVIIx64.exe(XVIIx86.exe on 32-bit systems), the new file is first copied to the Windows temp directory and the old XVIIx64.exe is overwritten after the download is complete. The old program is still preserved if the user cancels the file transfer. The changelog.txt file lists the program revisions.

V. Voltage Source

Symbol Names: VOLTAGE, BATTERY

Syntax: Vxxx n+ n- <voltage> [AC=<amplitude>] [Rser=<value>] [Cpar=<value>]

This element sources a constant voltage between nodes n+ and n-. For AC analysis, the value of AC is used as the amplitude of the source at the analysis frequency. A series resistance and parallel capacitance can be defined. The equivalent circuit is:



Voltage sources have historically been used as the current meters in SPICE and are used as current sensors for current-controlled elements. If Rser is specified, the voltage source can not be used as a sense element for F, H, or W elements. However, the current of any circuit element, including the voltage source, can be plotted.

Syntax: Vxxx n+ n- PULSE(V1 V2 Tdelay Trise Tfall Ton Tperiod Ncycles

Time-dependent pulsed voltage source

Name	Description	Units
Voff	Initial value	V
Von	Pulsed value	V
Tdelay	Delay	sec
Tr	Rise time	sec
Tf	Fall time	sec
Ton	On time	sec
Tperiod	Period	sec

Ncycles	Number of cycles(Omit for free-running pulse function)	cycles

Syntax: Vxxx n+ n- SINE(Voffset Vamp Freq Td Theta Phi Ncycles)

Time-dependent sine wave voltage source.

Name	Description	Units
Voffset	DC offset	V
Vamp	Amplitude	V
Freq	Frequency	Hz
Td	Delay	sec
Theta	Damping factor	1/sec
Phi	Phase of sine wave	degrees
Ncycles	Number of cycles(Omit for free-running sine function)	cycles

For times less than Td or times after completing Ncycles, have run, the output voltage is given by

```
Voffset+Vamp*sin(π*Phi/180)
```

Otherwise the voltage is given by

```
Voffset+Vamp*exp(-(time-Td)*Theta)*sin(2*π*Freq*(time-Td)+π*Phi/180)
```

The damping factor, Theta, is the reciprocal of the decay time constant.

Syntax: Vxxx n+ n- EXP(V1 V2 Td1 Tau1 Td2 Tau2)

Time-dependent exponential voltage source

Name	Description	Units
V1	Initial value	V
V2	Pulsed value	V
Td1	Rise delay time	sec
Tau1	Rise-time constant	sec
Td2	Fall delay time	sec

Tau2	Fall-time constant	sec

For times less than Td1, the output voltage is V1. For times between Td1 and Td2 the voltage is given by

```
V1+(V2-V1)*(1-exp(-(time-Td1)/Tau1))
```

For times after Td2 the voltage is given by

```
V1+(V2-V1)*(1-exp(-(time-Td1)/Tau1))-(V2-V1)*(1-exp(-(time-
Td2)/Tau2))
```

Syntax: Vxxx n+ n- SFFM(Voff Vamp Fcar MDI Fsig)

Time-dependent single frequency FM voltage source.

Name	Description	Units
Voff	DC offset	V
Vamp	Amplitude	V
Fcar	Carrier frequency	Hz
MDI	Modulation index	-
Fsig	Signal frequency	Hz

The voltage is given by

```
Voff+Vamp*sin((2.*π*Fcar*time)+MDI*sin(2.*π*Fsig*time))
```

Syntax: Vxxx n+ n- PWL(t1 v1 t2 v2 t3 v3...)

Arbitrary Piece-wise linear voltage source.

For times before t1, the voltage is v1. For times between t1 and t2, the voltage varies linearly between v1 and v2. There can be any number of time, voltage points given. For times after the last time, the voltage is the last voltage.

Syntax: Vxxx n+ n- wavefile=<filename> [chan=<nnn>]

This allows a .wav file to be used as an input to LTspice. <filename> is either a full, absolute path for the .wav file or a relative path computed from the directory containing the simulation schematic or netlist. Double quotes may be used to specify a path containing spaces. The .wav file may contain up to 65536 channels, numbered 0 to 65535. Chan may be set to specify which channel is used. By default, the first channel, number 0, is used. The .wav file is interpreted as having a full scale range from -1V to 1V.

This source only has meaning in a .tran analysis.

T. Lossless Transmission Line

Symbol Name: TLINE

Syntax: Txxx L+ L- R+ R- Zo=<value> Td=<value>

L+ and L- are the nodes at one port. R+ and R- are the nodes for the other port. Zo is the characteristic impedance. The length of the line is given by the propagation delay Td.

This element models only one propagation mode. If all four nodes are distinct in the actual circuit, then two modes may be excited. To simulate such a situation, two transmission-line elements are required. See the schematic file .\examples\Educational\TransmissionLineInverter.asc to see an example simulating both modes of a length of coax.

SPICE Error Log Command

Use this command to display the simulation log file. A typical log file is shown as follows:

```
Circuit: * LT1300-DC035A.asc
Date: Tue Oct 05 16:57:31 1999
Total elapsed time: 6.64 seconds.
tnom = 27
temp = 27
method = modified trap totiter = 14872
traniter = 14862
tranpoints = 3865
accept = 2986
rejected = 879
trancuriters = 0
matrix size = 12
fillins = 2
solver = Normal
```

K. Mutual Inductance

Symbol Names: None, this is placed as text on the schematic.

Syntax: Kxxx L1 L2 [L3 ...] <coefficient>

L1 and L2 are the names of inductors in the circuit. The mutual coupling coefficient must be in the range of -1 to 1.

The line

```
   K1 L1 L2 L3 L4 1.
```

is synonymous with the six lines

```
   K1 L1 L2 1.
   K2 L2 L3 1.
   K3 L3 L4 1.
   K4 L1 L3 1.
   K5 L2 L4 1.
   K6 L1 L4 1.
```

It is recommended to start a design with a mutual coupling coefficient equal to 1. This will eliminate leakage inductance that can ring at extremely high frequencies if damping is not supplied and slow the simulation.