S-Edit 12 User Guide—Contents

S-Edit 12 User Guide—Contents

(Continued)

S-Edit 12 User Guide—Contents

(Continued)

S-Edit 12 User Guide—Contents

(Continued)

S-Edit 12 User Guide—Contents

(Continued)

S-Edit 12 User Guide—Contents

(Continued)

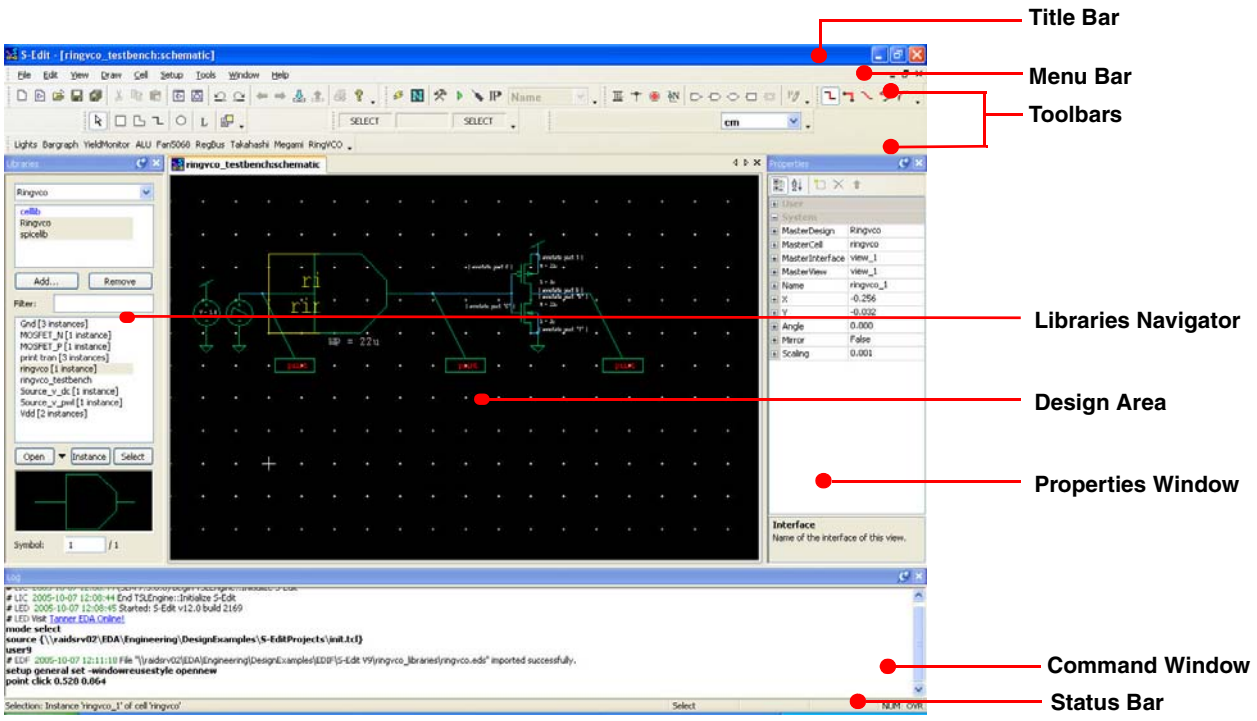
1 Interface and Setup

S-Edit is a fully hierarchical computer-aided schematic capture application for the logical design of integrated circuits. S-Edit contains integrated SPICE simulation and probing of simulation results, including voltages, currents, and noise parameters.

Launching S-Edit

To launch S-Edit, double-click on the S-Edit icon.

The user interface consists of the elements shown below. Unless you explicitly retrieve a setup file, the position, docking status and other display characteristics are saved with a design and will be restored when the design is loaded.



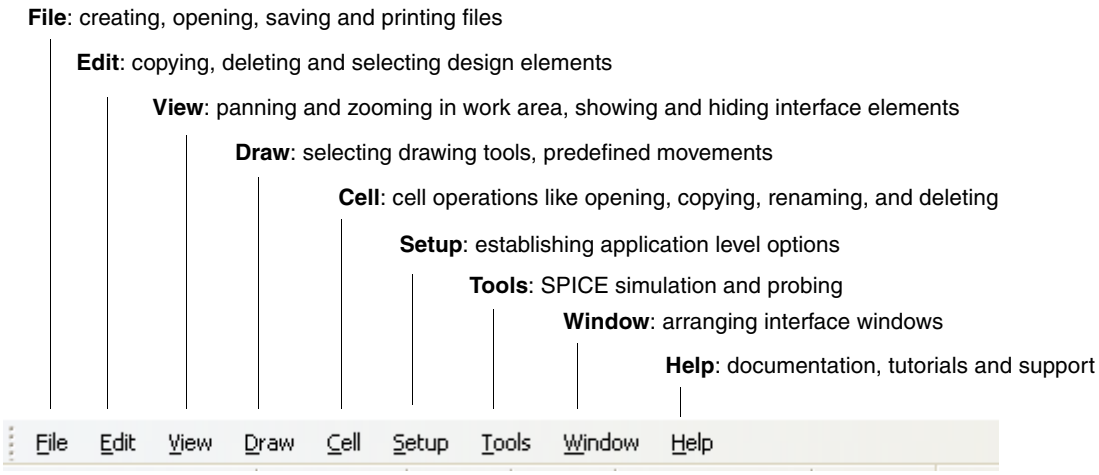
Parts of the User Interface

Title Bar

The *title bar* shows the name of the current cell and the view type (symbol, schematic, etc.).

Menu Bar

The *menu bar* contains the S-Edit menu titles. The menu displayed may vary depending on the view type that is active. See Shortcuts for Cell and View Commands on page 41 for the various methods S-Edit provides for executing commands.



Menu List Filtering

Most S-Edit menus and dialogs allow for filtering to speed the process of selecting from a drop-down list. So, when you enter a character, S-Edit will jump to the first list item that begins with that character. For example, typing **g** highlights the first list item beginning with that letter and filters the display to show only items that begin with **g**. Typing a **u** after the **g** highlights the first list item beginning with **gu**, and filters the display to show only items that begin with **gu**, and so on. The search procedure is case-insensitive.

Toolbars

You can display or hide individual toolbars using the **View > Toolbars** command, or by right-clicking in the toolbar region. Toolbars can be relocated and docked as you like. For added convenience, S-Edit displays a tool tip when the cursor hovers over an icon.

Standard Toolbar

The *Standard* toolbar provides buttons for commonly used file and editing commands, as well as operations specific to S-Edit such as “View Symbol” or “Save Design and Its Libraries.”

Draw Toolbar

The *Draw* toolbar provides tools used to create non-electrical objects, such as rectangles, circles, and lines, for illustrating and documenting a design.

Segment Toolbar

The *Segment* toolbar provides tools with which you limit the degree of angular freedom allowed when you are drawing wires. (See Drawing Tools for Electrical Objects on page 59.)

Electrical Toolbar

The *Electrical* toolbar provides the tools used to create wires, nets, and ports, and to add properties.

SPICE Simulation Toolbar

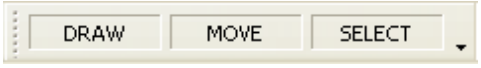
The *SPICE Simulation* toolbar lets you extract connectivity, select and probe nets, launch T-Spice and select evaluated properties.

Locator Toolbar

The *Locator* toolbar displays the coordinates of the mouse cursor and allows you to quickly change the units of measurement application-wide.

Mouse Buttons Toolbar

The *Mouse Buttons* toolbar shows the current functions of the mouse buttons.



Mouse buttons vary in function according to the tool that is active. The **Shift**, **Ctrl** and **Alt** keys can further change the function. For two-button mice, the middle-button function is accessed by clicking the left and right buttons at the same time, or by pressing **Alt** while clicking the left mouse button.

Toolbar Options

This drop-down menu lets you add or remove buttons from the S-Edit toolbars.

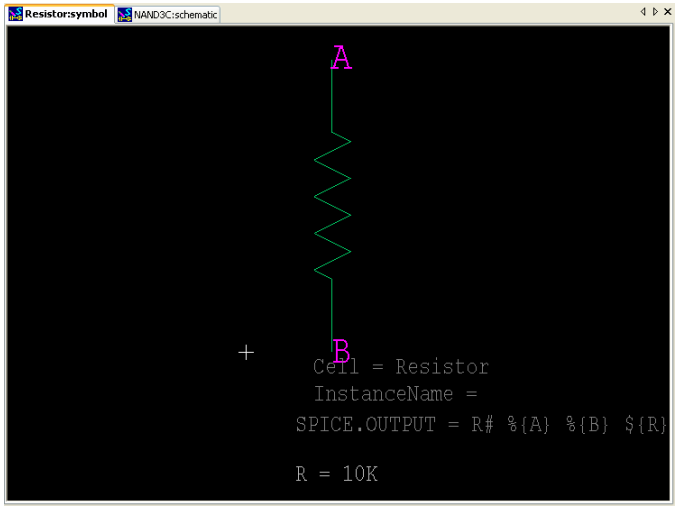
Status Bar

The *Status Bar* display varies with the type and number of objects selected and the tool in use. You can use **View > Status Bar** to toggle display on and off.

Item Selected	*Status Bar displays:*
single instance	The instance name and source cell
simple geometry	Width, height, area, pathlength, the number of vertices it contains and for ports, the X,Y coordinates and name of the object.
multiple instances	The number of instances selected.
multiple objects	The type and number of each object.
during design probing	The status of waveform probing operations.
during drawing operations	The drawing mode and any constraints (on the right side).

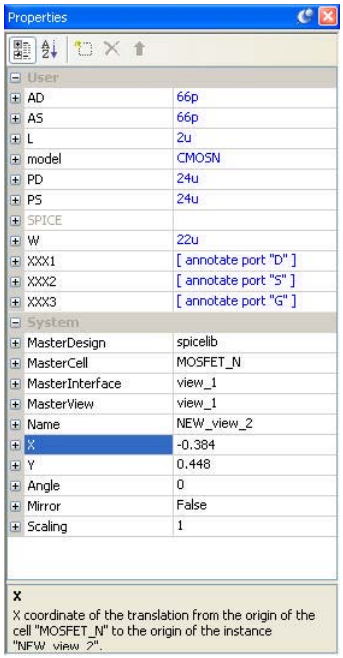
Design Area

The design region in Tanner tools, where you create, view and edit objects, is the *Design Area*. The portion of the design area currently visible is called the *Work Area*. You can move or resize design windows as you would in any other application window. Refer to The Work Area on page 46 for further information.



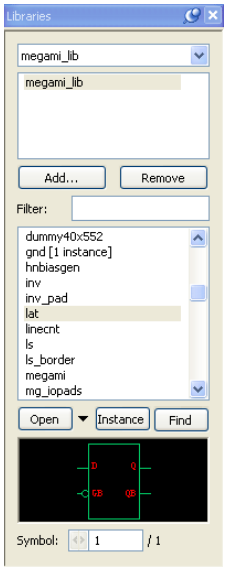
Properties Window

The Properties window provides an editable display of the characteristics of a selected object, which may be a cell, drawn geometry, port, etc. See Symbol Properties on page 88 and Editing Instance Properties on page 71.



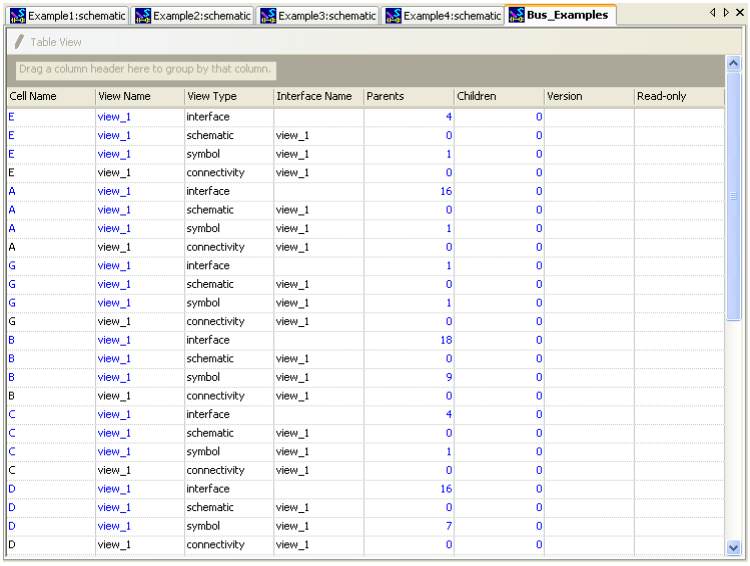
Libraries Navigator

The Libraries navigator allows you to add and remove cell libraries and lists the cells in the libraries that are highlighted. You can use this list to open, duplicate, or instance a cell, or to create a new view of a cell. See The Libraries Navigator on page 51 for more information.



View Navigator

The View navigator displays hierarchical information for the cells in a library such as the existing view types and names, their interface, any parent or child cells, the version and whether or not they are read-only.

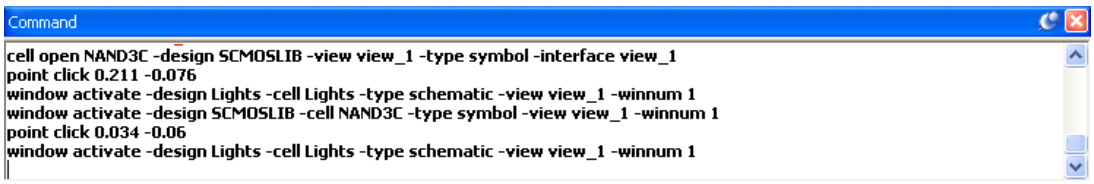


Command Window

All events that occur in the S-Edit design area are recorded to a log displayed in the *Command window*, in tool command language (TCL) format. TCL files are macro-like scripts that allow you to perform or repeat operations.

The Command window serves as both a recording and a playback device for TCL files. As such, any action or operation performed by S-Edit can be copied or replayed. Text can be typed in, copied from executed operations and then pasted back into the Command window, or written in from a saved TCL file to instantly perform the desired operations. This is especially useful for automating and simplifying difficult or repetitive tasks.

You can use **View > Activate Command Window** (shortcut backquote ‘) to open the Command window if it is not open. If it is open, backquote will shift focus to it.

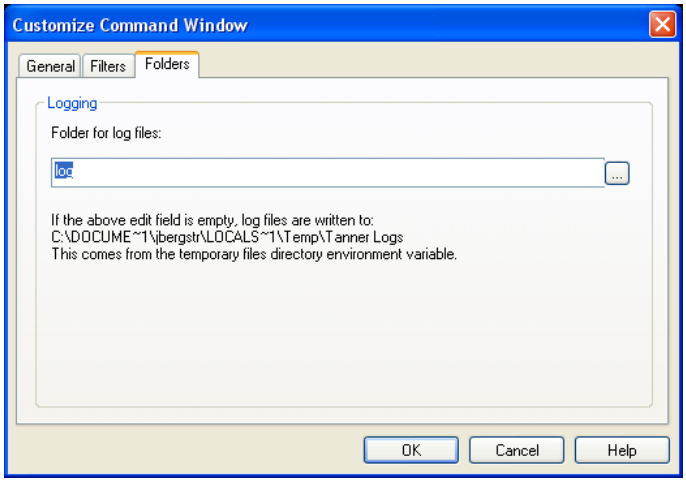


Command Window Log Files

Each time S-Edit is launched it creates a TCL log file that records each operation performed in the design windows for the entire session. The last 10,000 lines (a default value which can be changed, see Controlling Warnings and Information in the Log File on page 14) of this log are displayed in the Command window. Logs files are identified by the date and time the session was launched.

How to Locate Log Files

To set the directory where log files are stored, right-click in the Command window to open its context-sensitive menu. Select **Customize** (shortcut **F8**) to open the **Customize Command Window** dialog, then use the **Folders** tab to select or create a storage directory.

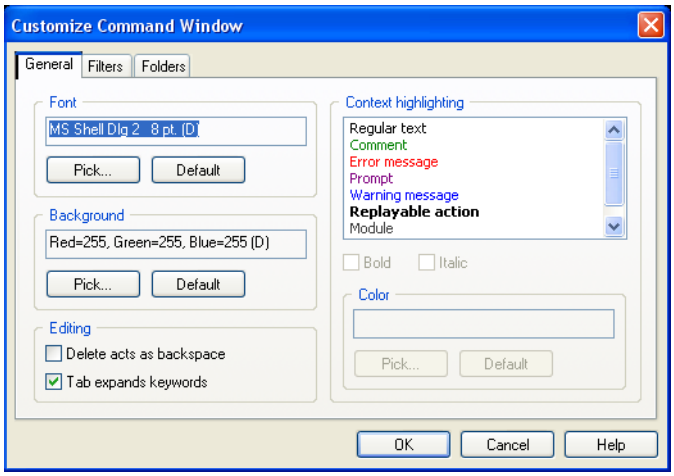


If no path is set in this dialog, then logs are stored in the location set by the TANNERLOGPATH environment variable.

If there is no TANNERLOGPATH environment variable, logs are stored in a folder “Tanner Logs” under the temp folder, which can be %TMP% or %TEMP% or even the WINDOWS folder on your hard drive, depending on the configuration of your Windows environment.

Setting Command Window Text Styles

You can set display characteristics in the S-Edit Command window for each of the categories of text—error messages, warnings, modules, etc. using the **General** tab of **Customize Command Window**.



Font Use this field to **pick** a font or set it to the **default**.

Background Use this field to **pick** a background color or set it to the **default**.

Editing

Use this field to set the behavior of the backspace and tab keys.

Delete acts as backspace When this checkbox is enabled, the **Delete** key functions like the **Backspace** key by removing text to the left of the cursor. When this checkbox is not checked, the **Delete** key removes text to the right of the cursor as usual.

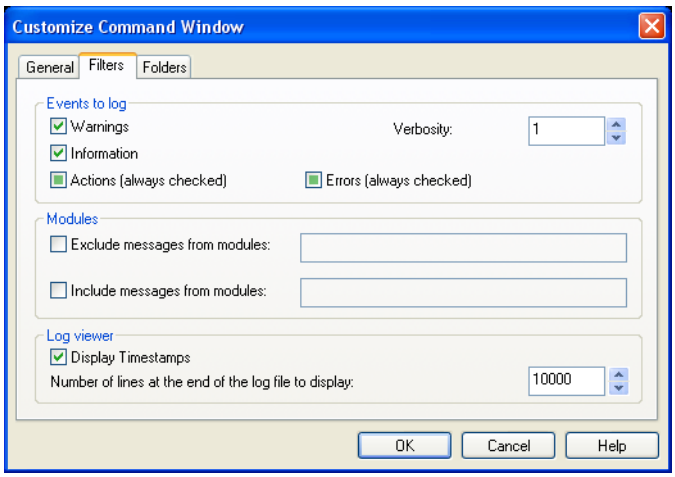
Tab expands keywords When this box is checked, you can use the **Tab** key to expand all keywords.

Context highlighting

When you select a text type in the upper pane, you can check the boxes to make it **Bold** and **Italic**, and also **Pick** a text color or set it to the **Default**.

Controlling Warnings and Information in the Log File

The **Filters** tab in **Customize Command Window** allows you to limit the type or amount of information written to the log file. You can also set certain display controls for the Command window which do not effect the log file.



Events to log

Check the boxes to include events categorized as **Warnings** and **Information** in the log file and Command window display. (**Actions** and **Errors** are always written to the log file.)

The higher the **Verbosity** value, the higher the detail included in event messages written to the log file.

Modules

Each program module in S-Edit generates a set of messages, identified by a preceding # *module_name*, where *module_name* is a three character abbreviation such as “LIC” for license related messages. Enter these abbreviations, separated by commas, to exclude or include a type of message from the log file and Command window.

When **Exclude messages from module** is checked, messages from any module abbreviations listed in the corresponding entry field will not be written to the log file.

When **Include messages from module** is checked, messages from any module abbreviations listed in the corresponding entry field will be written to the log file.

Log viewer

Note: These options effect the Command window display only, not what is written to the log file.

Check the **Display Timestamps** box to display the timestamp when one is included in a message.

Enter an integer between 100 and 100,000 to set the **Number of lines at the end of the log file to display**.

Arranging Interface Elements

The S-Edit interface is highly flexible, allowing you to display and arrange elements as you choose.

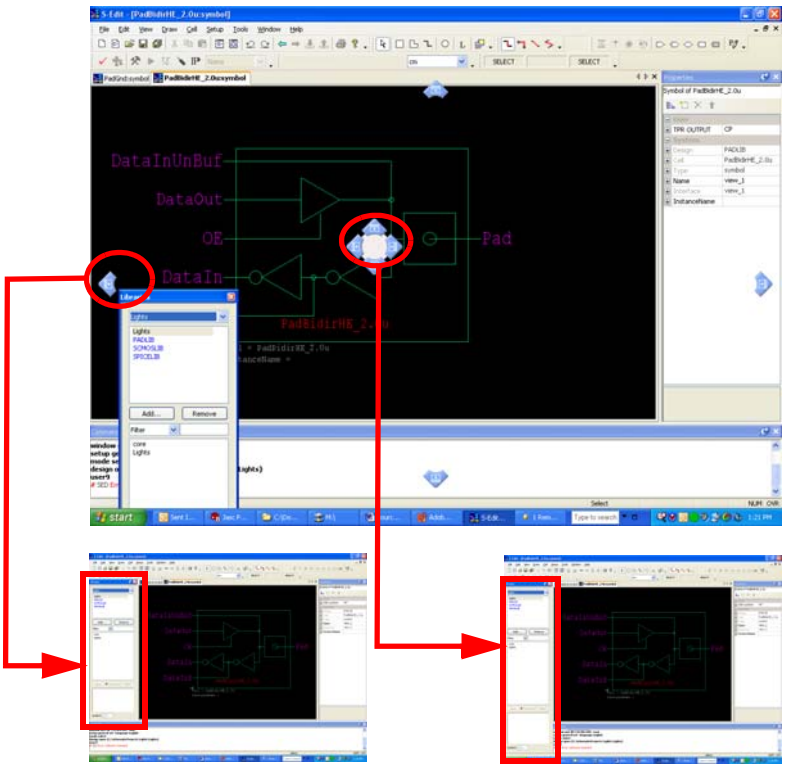
You can dock individual S-Edit toolbars, navigators and the design window, or they can float anywhere in the application window. With the exception of the locator toolbar, toolbar buttons will be arranged vertically when docked to the sides. Design windows can be shown, hidden or made dockable directly from the interface.

Display colors can be customized for objects, background, the grid, etc., and can be set differently in the text editor for each of the file types it reads. Windows reuse in the design area is configurable, as are parameters such as selection range and behavior.

Settings for these and other configuration parameters are available in the **Setup** menu.

How to Dock Windows in S-Edit

S-Edit uses standard Windows 2000/XP docking behavior. When you drag any of the interface windows and release the cursor over a blue docking arrow, it will dock as shown below.



Edge Arrows
When you release a window over a docking arrow at the edge of the screen, it will dock at that edge, in between the other docked windows.

Center Arrows
When you release a window over a docking arrow at the center of the screen, it will dock at the corresponding edge, filling the entire application area.

Setup Options

The **Setup** menu contains configuration controls as follows:

- **Colors** – Controls the color of objects in the design window.
- **Grids** – Controls the size and style of the display grids.
- **Units** – Controls the unit type and scaling factor for display units.
- **Page** – Controls page size, margins and borders.
- **Validation**– Controls the allowable entry type (e.g. alpha vs. numeric) in naming fields.
- **Protection** – Allows you to protect an entire design file to prevent editing.
- **General** – Controls design window and language display.
- **Selection** – Controls mouse selection behavior.

- **Text Editor and Styles** – Controls text file update and display characteristics.
- **SPICE Simulation** – Controls simulation parameters and options (see Running Simulations on page 126).

Setup information can be saved and reloaded as described below.

Saving and Loading Setup Options

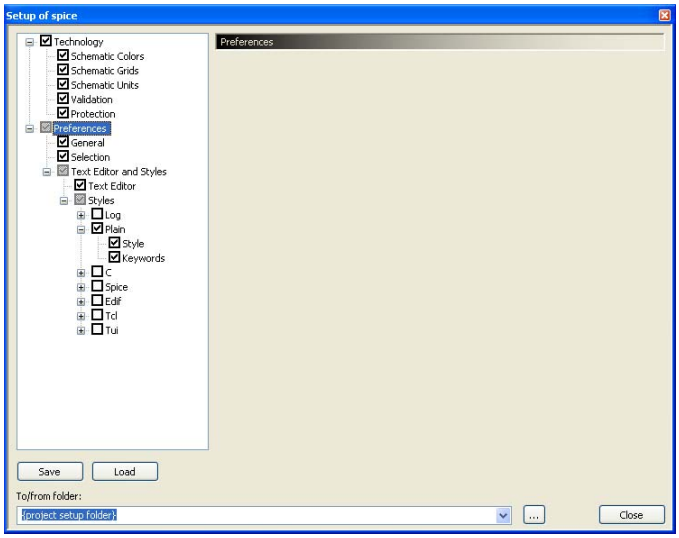
Setup values are stored as TCL scripts, with one TCL file for each page (**colors.tcl**, **grid.tcl**, etc.) is the Setup dialog.

Use **Setup > Technology** or **Setup > Preferences** to make changes to setup values. Note that changes to a sestup page will only be applied if the corresponding checkbox in the list on the left side of the window is checked. (S-Edit automatically enables the check box for a page when changes are first entered.)

If you want the changes to apply only to the current editing session, simply click on the **Close** button.

If you want to save the changes locally so they will be loaded when the design is launched again, click on the **Save** button before closing the dialog. S-Edit creates and populates a setup folder in the design directory to store those changes.

Note: Changes to setup pages are only saved if **1)** the page is checked in the list in the left pane of the setup window and **2)** the **Save** button has been pressed before the dialog is closed.



Save You must click on this button to save a setup configuration to a TCL file.

Note: Make sure there is a check in the checkbox for the attributes you want to save! Only dialog pages that are checked will be saved.

Load	Use this button to load a saved setup file. ***Note:*** Make sure there is a check in the checkbox for the attributes you want to load! Only checked dialog pages will be loaded.
To/from folder	Use this drop-down list to select the folder to which a setup file will be saved, or from which a setup file will be retrieved. **{project setup folder}** is the setup folder in the design folder of the selected design or library shown in the title of the **Setup** dialog. **{user preferences folder}** is a setup folder in the Tanner directory on a local computer from which S-Edit will automatically read user-defined setup values that overwrite the default settings for a design. You can also browse to the directory of your choice to save or load a setup file that S-Edit does not search for automatically.

The User Preferences Folder

When you open a design, S-Edit first reads TCL scripts for setup data from the project folder in the design directory, and then from the *user preferences folder*.

Settings from the user preferences folder take precedence over settings from the design project folder if the same settings are present in both locations. Setup TCL scripts are not required to exist in either location, in which case S-Edit default program settings are used.

This sequential load order allows for a universal setup intended for all users of a given design which individual users can modify by saving their own setup preferences to their user preferences folder.

To save setup options to the design folder, select **{project setup folder}** from the drop-down menu at the bottom of each setup page. To save settings to the user preferences folder, select **{user preferences folder}**.

User Preferences Folder Location

The predefined location of the user preferences folder is **C:\Documents and Settings\\<*username*>\Application Data\Tanner EDA\scripts\open.design\setup**, where *username* is the login name of the current user. This is the directory to which S-Edit saves setup TCL files when you select **{user preferences folder}**. The user preferences folder may be modified with appropriate windows environment variables.

Alternate Locations for Setup Scripts

You can use the **To/from folder** field to save setup scripts to any other location you like. This is useful when you want to save setup values that differ from those S-Edit loads automatically. For example, setup definitions can be explicitly saved and loaded in order to copy setup data to a new design file, or so that you can save multiple alternate setup schemes. Note that when you load a setup file from a user selected folder, only settings on the pages with their checkbox checked will be loaded.

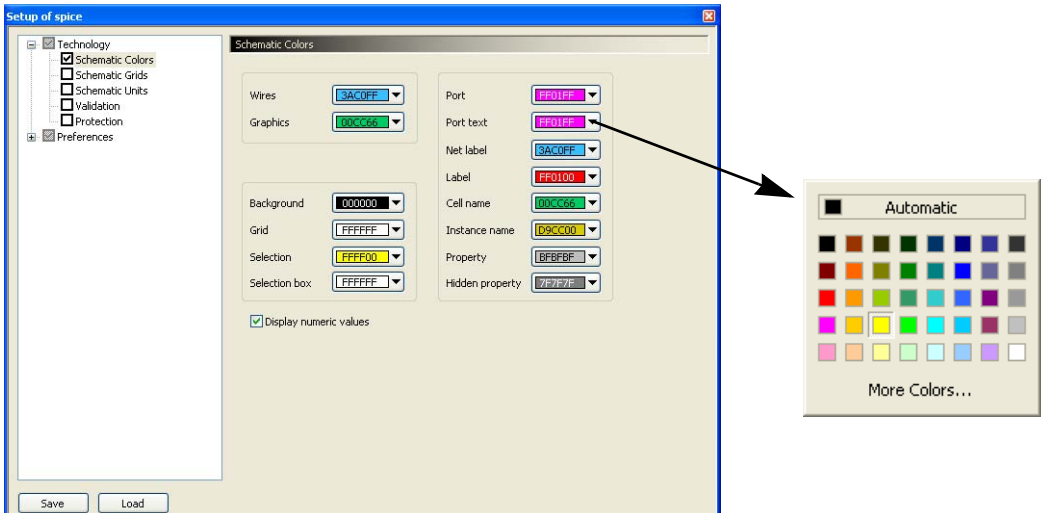
Different Setup Values by Library

When a design contains more than one library, each library will maintain its own setup values. For example, take a design *Cool*, with graphics rendered in blue, which uses a library *Warm*, with graphics drawn in red. Instances of symbols that come from cells in the *Cool* design will be rendered in blue, while instances of cells from the library *Warm* will be rendered in red.

Color Display Options

Setup > Technology > Schematic Colors

Use **Setup > Colors** to set the color scheme for your design.



Schematic Colors — Use these fields to set the color for the associated drawn element (wire, port, background, grid, etc.) by clicking on the drop-down arrow to open the standard windows color dialog.

Display numeric values — Check this box to also show the six-digit hexadecimal representation of colors in the form RRGGBB, corresponding to the red, green, and blue values of the color.

Grid and Unit Coordinates in S-Edit

S-Edit uses *display units* to report object dimensions and coordinates, to set a grid to use as a visual aid while drawing, and to establish an optional *snapping grid* for the mouse cursor.

Major and Minor Display Grid

The grid:display grid consists of two arrays in the design area, large (*major*) and small (*minor*), that can be set to any desired size to provide a drawing guide, typically to set minimum feature size or critical manufacturing measurements. The grids can be independently displayed or not, as either dots or lines.

Display units can be shown in millimeters, centimeters, meters, or inches. The choice of display units does not affect the scaling of your design. If you change the display units, for example, from millimeters to inches, S-Edit will automatically convert the unit values displayed in the locator bar to inches, but nothing in the design itself will change.

The apparent spacing of the grids will vary with the magnification of the work area. If the number of screen pixels per grid square falls below the value entered in this field, the grid is hidden.

The coordinate origin point (0,0) is indicated by a large cross-hair marker, and its display can also be toggled on and off.

Snap Grid

The snapping grid, which can be a different size than the display grid, causes all drawing and editing coordinates entered with the mouse to be placed on grid points. To achieve adequate resolution, you may wish to adjust the spacing of the mouse snap grid based on your minimum feature size.

Internal Units

For its own computation, S-Edit uses *internal units*. Before beginning your design, you must define the relation between internal units and physical units. This ratio will determine the maximum dimensions of the design area and the smallest object that can be drawn. This relation is also critical when you replace your design setup or export a design, since it sets the scale of the design file.

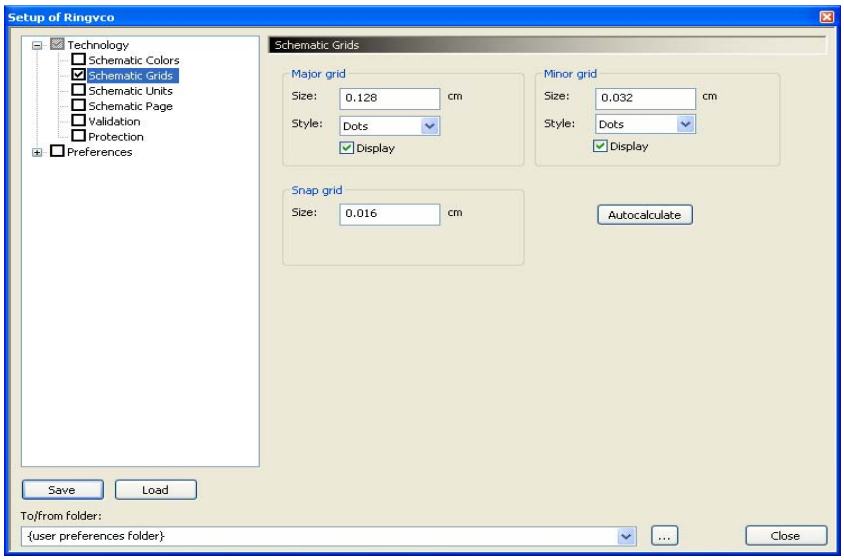
The S-Edit design area extends from -536,870,912 to +536,870,912 internal units in the *x*- (horizontal) and *y*- (vertical) directions. Thus, if 1 internal unit = 0.001 millimeters, the largest possible design would be 1,073,741 internal units (roughly 42.3 inches) on a side. Similarly, the smallest dimension S-Edit can define is 1 internal unit. If 1 internal unit = 0.001 millimeters, the smallest possible feature size would be 0.001 millimeters.

To set internal units see Unit Display Options on page 22.

Grid Options

Setup > Technology > Schematic Grids

Use **Setup > Technology > Schematic Grids** to set the size of the display grid for your design.



Major Grid (in display units)
The absolute spacing of the major grid display. The value entered in this field is the distance, in display units, between major grid points.

Minor Grid (in display units)
The absolute spacing of the minor grid display. The value entered in this field is the distance, in display units, between minor grid points.

Snap Grid
The absolute spacing of the cursor snap grid, entered in display units as the length of one side of a grid square.

The value entered in this field is the minimum resolution, in display units, allowed during drawing and editing operations. All drawing and editing coordinates are snapped to this grid size when it is not zero.

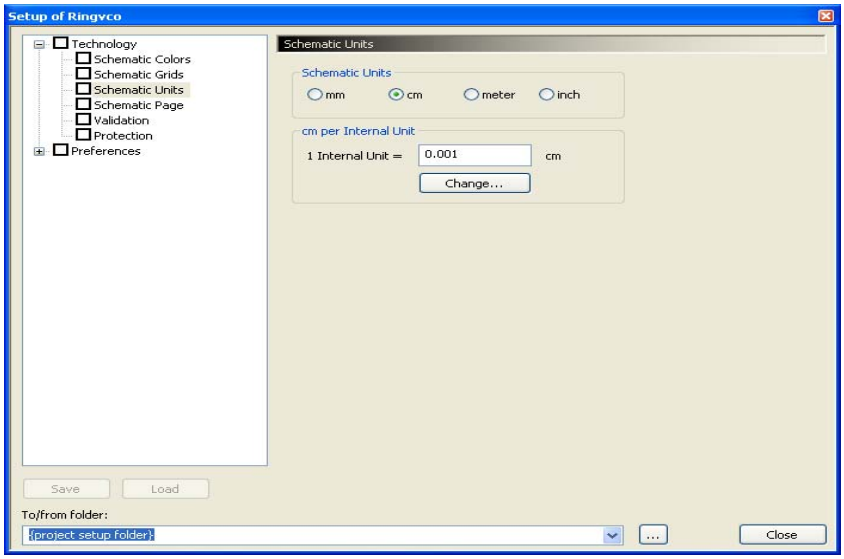
Snap cursor movements
Toggles cursor snapping on and off.

Autocalculate
Pressing this button calculates the largest possible minor grid size for the selected design based on all port positions in the design and updates the other values in the dialog. If a common grid value cannot be found S-Edit will return one (1) internal unit.

Unit Display Options

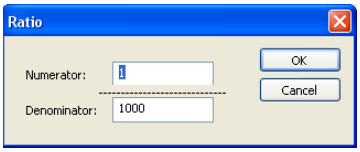
Setup > Technology > Schematic Units

Use **Setup > Technology > Schematic Units** to establish the relation between S-Edit display units and physical units.



Schematic Units — Choose a physical unit for your design.

***units* per Internal Unit** — Enter the scale of physical units to internal units as a decimal value. To enter the ratio as a fraction, click **Change** to enter a numerator and denominator.



Name Constraints

Setup > Technology > Validation

The fields in **Setup > Technology > Validation** allow you to set constraints on how cells, views, instances, port and nets can be named. For example, you might want to adhere to GDSII naming conventions by preventing the use of spaces in cell names.

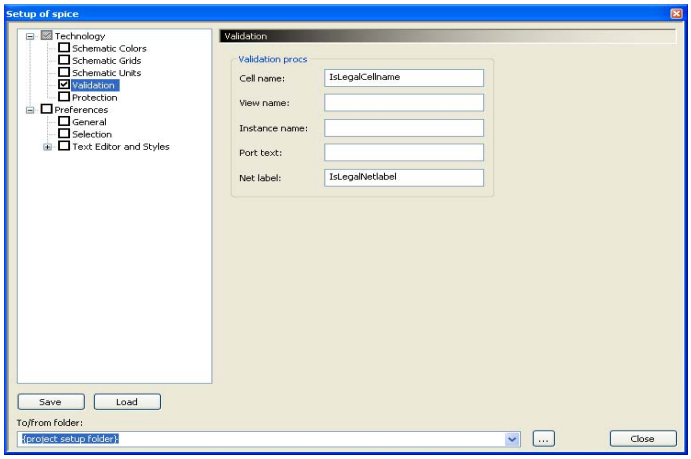
When you enter a name using a dialog from the menu, S-Edit validates it immediately, and displays any violation warnings in the Command window, highlighted in red. For example, "# SED Error: Name: bad name contains illegal characters (only alphanum _ ? $ allowed)."

However, existing cell names, and names that are entered using the Properties window, are not validated until a file undergoes a design check. After a design check, violation warnings are displayed in the Command window.

Creating TCL Procedures for Name Constraints

S-Edit uses TCL procedures ("procs") to check for naming errors. To create naming constraints for an object type (ex. cell, port, net), you write a TCL proc and then enter the procedure name in the corresponding Validation setup field. S-Edit executes the procs from these fields.

A default set of validation procs (**IsLegalCellname**, **IsLegalViewName**, **IsLegalInstanceName**, **IsLegalPortName**, **IsLegalNetName**) is provided with S-Edit in the **init.tcl** file. When you create a new design, it is automatically initialized to point to these procs. You can override this default by placing TCL procs in **C:\Documents and Settings\<*username*>\Application Data\Tanner EDA\scripts\startup**. (Note that it is strongly recommended that you do not modify the **init.tcl** file, as it is overwritten with each product upgrade.)



Cell name — Enter the name of the TCL procedure with the set of naming constraints you want to apply to cells.

View name — Enter the name of the TCL procedure with the set of naming constraints you want to apply to views.

Instance name — Enter the name of the TCL procedure with the set of naming constraints you want to apply to instances.

Port text — Enter the name of the TCL procedure with the set of naming constraints you want to apply to ports.

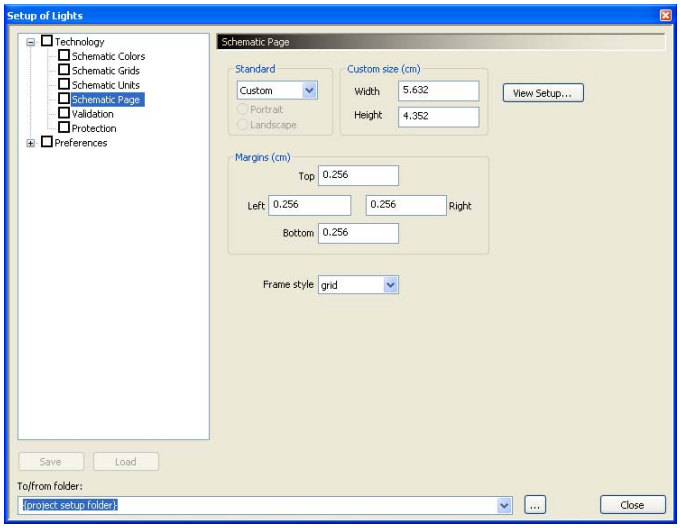
Net label — Enter the name of the TCL procedure with the set of naming constraints you want to apply to nets.

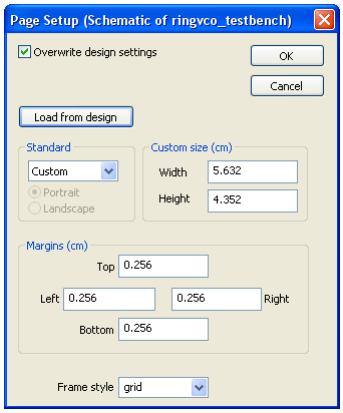
Setting Page Size

Setup > Technology > Schematic Page

Setup > Technology > Schematic Page lets you select a standard page size and frame style for your design.

S-Edit calculates the frame size with respect to the unit settings defined in **Setup > Technology > Schematic Units** so that when you print using the **Do not scale** setting the design window will print to fit the page size you have chosen.





Standard — Choose from the predefined page sizes, or select **Custom** to define your own.

Custom size — Enter the **Width** and **Height** here when using a Custom page size.

Margins — Controls the margins between the frame and paper edges, separately for **Top**, **Bottom**, **Left** and **Right.**

Frame style — Choose from the options **box**, **grid** or **none**. Grids increment as an alpha array in the x direction from A and numerically in the -y direction from 1.

All frames are placed with the lower left corner at the origin.

View Setup — Opens a dialog that allows you to enter page setup values for the currently active view that differ from those defined for the entire design.

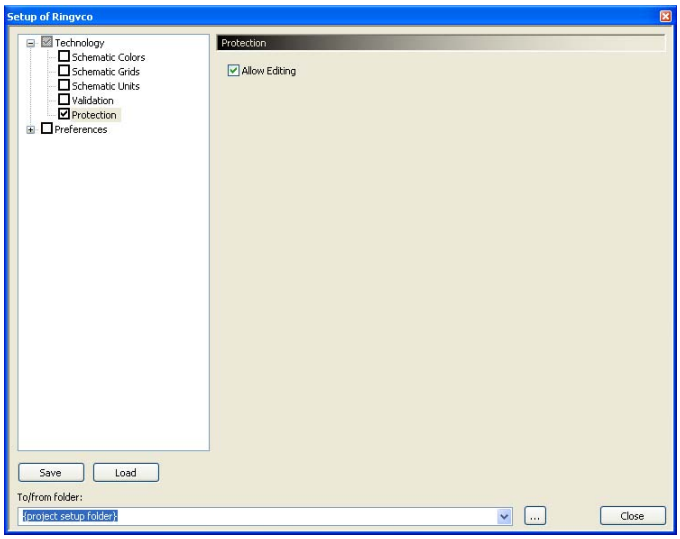
Enable **Overwrite design settings** to apply the values in this dialog to the active view.

Click on **Load from design** to reset values to those defined for the entire design.

Cell Protection Options

Setup > Technology > Protection

Setup > Technology > Protection prevents edits to the contents of all cells in a design.

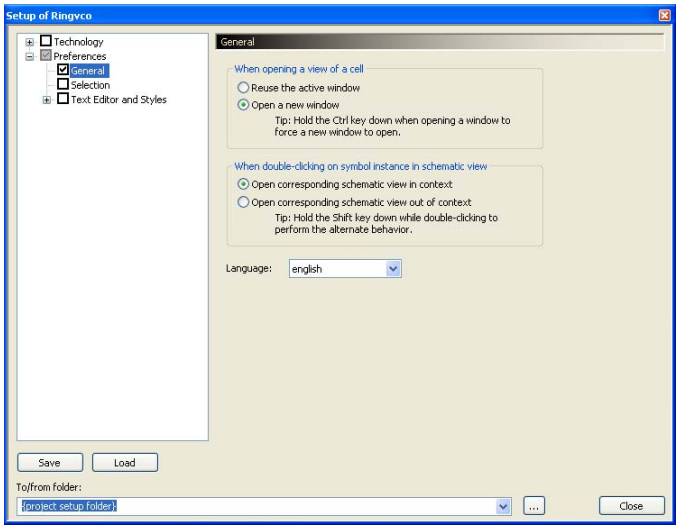


Allow editing When checked, cell contents can be edited.

Cell View and Language Options

Setup > Preferences > General

Use this dialog to set display behavior in the design area and the language display in menus and dialogs.



The options are as follows:

When opening a view of a cell

Reuse the active window sets S-Edit to replace the contents of the active design window when opening a new cell. If you do not select a view type, the new cell will clone the view type of the old.

Note: Holding the **Ctrl** key down while opening a cell will force a new window to open.

Open a new window sets S-Edit to reuse the active design window when opening a new cell view.

When double-clicking on symbol instance in schematic

Use this option to set whether you will open a unique instance or the primitive cell symbolwhen you double-click on an object in the design window.

Use **open corresponding schematic view in context** to push down to a specific instance when you double-click on a symbol.

Use **open corresponding schematic view out of context** to show the primitive symbol when you double-click on a symbol.

Note: Holding the **Shift** key down while double-clicking performs the action alternate to your setup default.

Open a new window

The default setting, where a new window opens each time a view is opened.

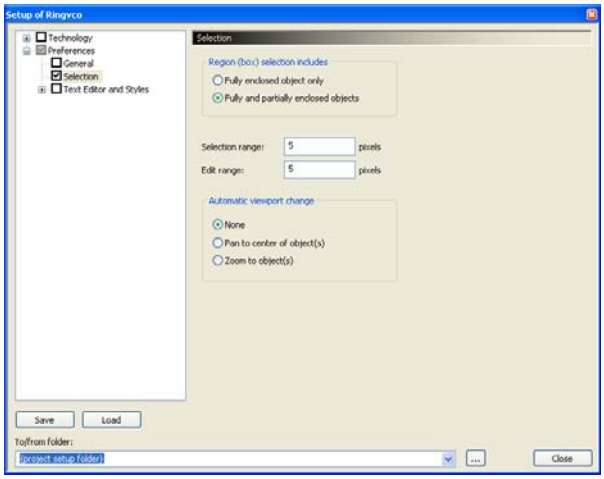
Language

Select a language for menu and dialog text from the drop-down list. Options are American English, Japanese, Russian and Simplified Chinese.

Selection Behavior Options

Setup > Preferences > Selection

This dialog governs selection behavior in the design area and also from hyperlinks in the Command window.



The options are as follows.

Region (box) selection includes	**Fully enclosed object only**—with this option enabled, only objects completely contained within a selection box will be selected. **Fully and partially enclosed objects**—with this option enabled, all objects completely or partially enclosed by a selection box will be selected.
Selection Range	A positive integer *x* such that when a mouse button is clicked less than *x* pixels from an object, though not touching it, the object will still be selected.
Edit Range	A positive integer *e* such that if the pointer is within *e* pixels of an edge or vertex of a selected object, the default operation of the MOVE-EDIT button is an **edit**. Outside this range it is a **move**.
Automatic viewport change	Controls the display behavior in the design area when you click on a hyperlink in the Command window. **None**—simply selects objects. **Pan to center of object(s)**—pans to the selected objects but does not change the zoom level. **Zoom to object(s)**—pans and zooms to the MBB of the selected objects.

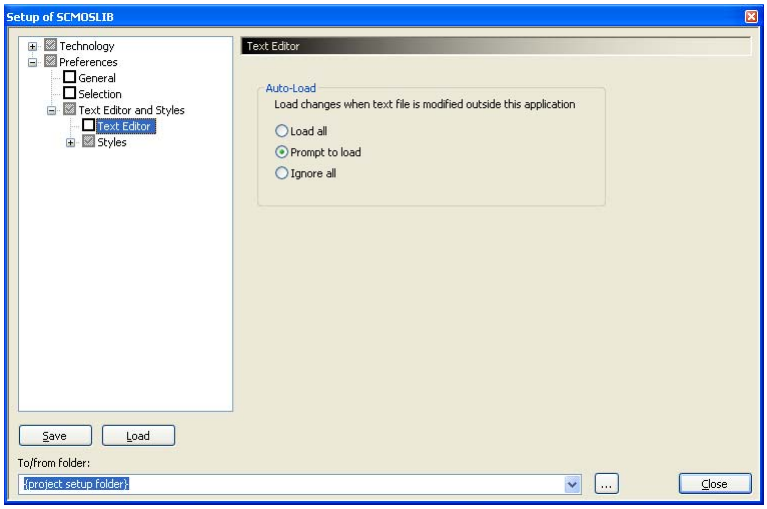
Text File Update Options

Setup > Preferences > Text Editor and Styles > Text Editor

This dialog governs if and how the text files that are open in S-Edit are updated when they are saved outside the application.

The S-Edit text editor checks the stored version of a file for modifications when files are saved, first changed, and when the text window or application becomes active or is closed. If a file has not been

modified outside S-Edit, nothing will happen. If a file has been modified outside S-Edit, the selected action will be triggered.



Auto-Load

Load all—with this option enabled, S-Edit automatically updates text files that have been modified outside of the text editor.

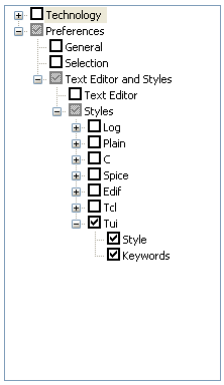
Prompt to Load—(default) this option opens a dialog indicating which files have been modified and gives you the option to save them.

Ignore all—with this option enabled, externally modified files are not updated to S-Edit.

Text File Display Options

Setup > Preferences > Text Editor and Styles > Styles

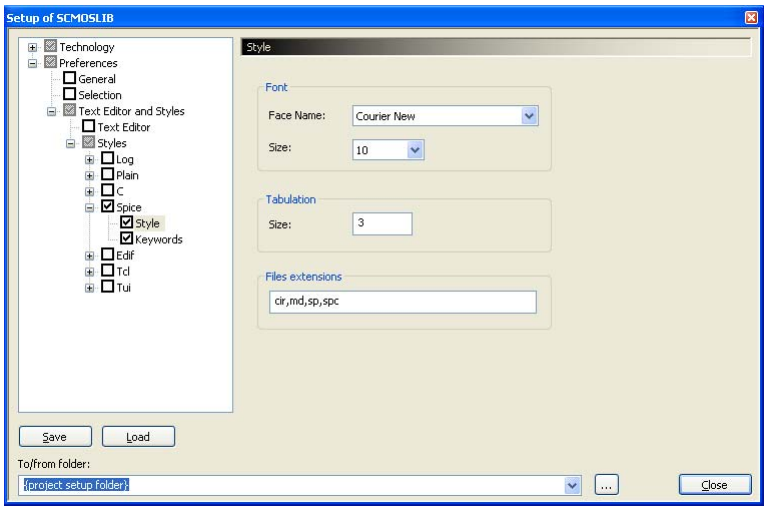
You can set individual display characteristics in the S-Edit text editor for each of these file formats using the **Style** and **Keywords** pages.



Text File Font Options

Setup > Preferences > Text Editor and Styles > Styles > {filetype} > Style

Use this dialog to set the font, tab stops and file extensions recognized for each of the file formats included in the S-Edit setups.



Font — Select or enter a font from the **Face Name** drop-down menu and a point size in the **Size** field.

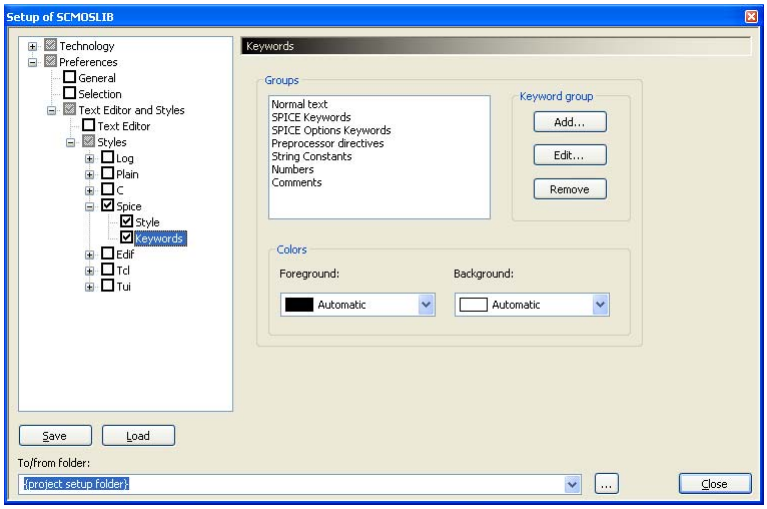
Tabulation — Enter a positive integer value to set the increment, in spaces, of the tab spacing the text editor uses.

File extensions — Enter the extensions of the files that S-Edit should include in the active category, separated by commas with no spaces.

Text File Keyword Options

Setup > Preferences > Text Editor and Styles > Styles > {filetype} > Keywords

Use this dialog to define the types of text, called *keyword groups* in S-Edit, that will be highlighted in the text editor, and their appearance. Each file type has a set of predefined keyword groups that cannot be edited or deleted.



Groups Displays the keyword groups defined for a given file type.

Keyword group Use **Add** to enter the name of a new keyword group. Use **Edit** to enter the terms belonging to a keyword group. Use **Remove** to delete a keyword group.

Colors Use **Foreground** and **Background** to set the respective colors for a keyword group.

SPICE Simulation Options

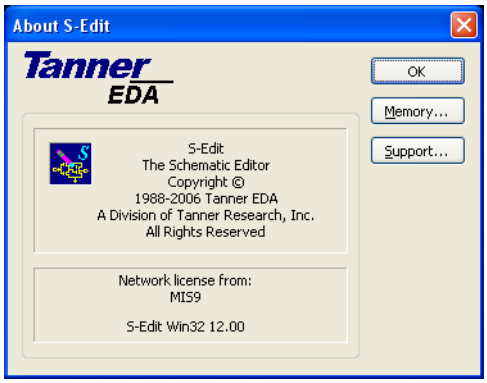
Please refer to SPICE Simulation Settings on page 116.

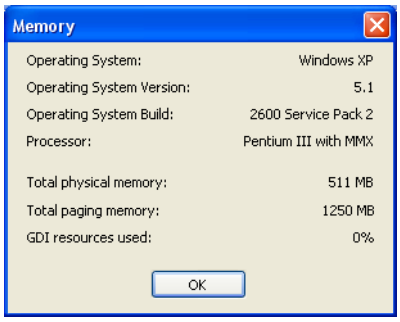
S-Edit Documentation

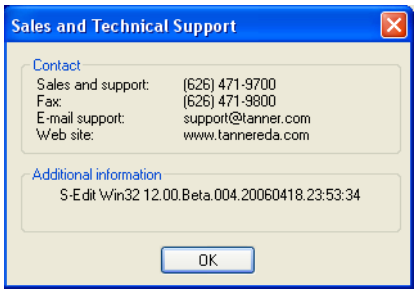
In addition to this manual in PDF format, S-Edit is shipped with application notes, release notes and a tutorial that highlights basic schematic entry and editing operations.

S-Edit Product Support

If you are in contact with Tanner EDA customer support they will sometime request the information found at **Help > About S-Edit**, **Help > About S-Edit > Memory**, or **Help > About S-Edit > Support**.

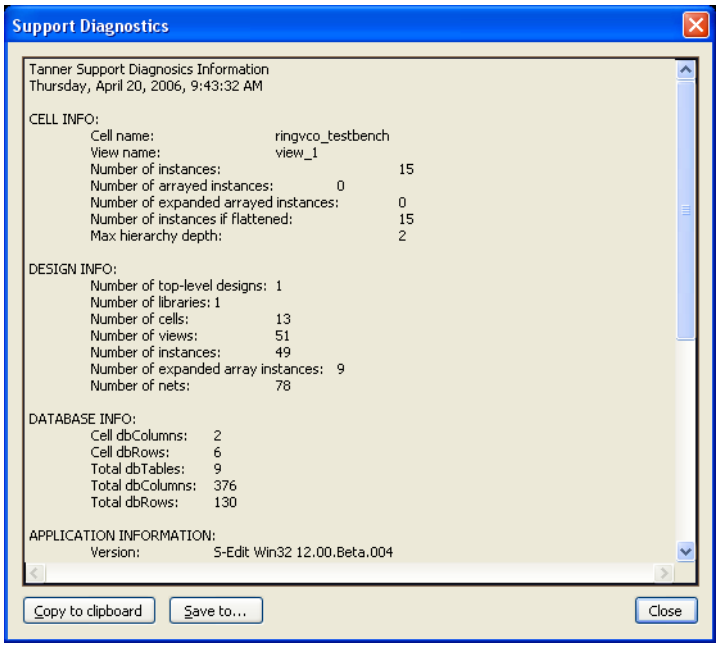






Support Diagnostics

You can view comprehensive design, system and hardware information from **Help > Support > Support Diagnostics**.



2 Creating a Project

The highest level entity in the S-Edit schematic database hierarchy is the design. A design contains many cells, some of which may be referenced from a library.

Most often a cell will contain a single *interface*, which contains a single symbol view and a single schematic view. However, a cell can contain any number of interfaces, and each interface can contain any number of symbol views and schematic views.

Project Structure in S-Edit

It is important to understand the basic project structure and terminology used in S-Edit.

Design

A *design* is the container for all elements of the design database.

Library

A *Library* is also a design, one whose cells are externally referenced by other designs. A design can reference multiple libraries, and any given library can be referenced by multiple designs.

Cell

A *cell* is the fundamental unit of design. A design contains multiple cells. Cells in turn contain multiple *views*, of different types. Cells can be instanced by other cells (an instance is a generic reference to a representation of cellB found within cellA.)

Properties

A property is an attribute of an object (cell, instance, shape). S-Edit differentiates between built-in properties (e.g., cell name, shape type) which are always defined, and user-defined properties (e.g. L= on a transistor cell) which are optional and not interpreted directly by the engine. Each type of object has its own set of parameters, which are displayed and can be edited in the Properties window. Please refer to Symbol Properties on page 88 for a complete discussion.

View

A *view* is simply a component of a cell definition. It is a depiction of the cell, a different way of depicting it. There are defined *view types*—you can view a cell as a *symbol*, a *schematic*, or an *interface*.

Cells can instance each other, but cyclical cell references are not allowed.

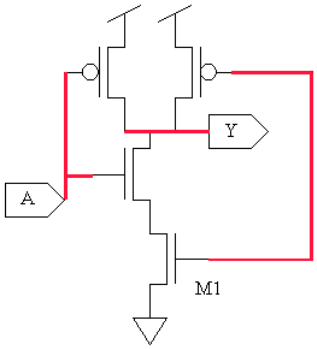
View Types

- **Symbol View**

A graphical description of a cell, for use in schematic views of other cells. A symbol view contains the ports of a cell and non-electrical geometry that is representational only. Usually the most basic design components will have only a symbol view.

- **Schematic View**

A more detailed view of a cell, showing ports, instances, references to ports in the parent cell of instanced cells, connecting wires and graphic objects (e.g. boxes, polygons, paths, text labels) that have no electrical meaning.



In this example, NAND2 is a two input NAND gate. A, B and Y are ports.

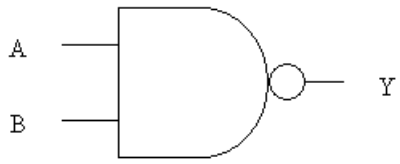
There are seven instanced symbols (including M1, three other transistors, and three power/ground symbols), and six red shapes implementing three nets.

There is a fourth net between the two NMOS devices, and a fifth (power) and sixth (ground) net as well.

- **Interface View**

The definition of the electrical interface of a cell, containing ports, optional permutability information for those ports, and a set of user-defined parameters.

Each schematic view and symbol view in a cell must be associated with a specific interface. When you create an instance of a cell you must specify the interface it belongs with. Cells may be associated with multiple interfaces. For example: the NAND2 cell shown below could have two interfaces:.



Interface 1, without power, there are three ports:
A = in, B = in, Y = out. A and B are permutable.

Interface 2, with power, there are five ports:
A = in, B = in, Y = out, Vdd = inout (virtual)
Gnd = inout (virtual). A and B are permutable.

How to Use Multiple Views

It is often useful to create a cell with multiple interface, symbol, or schematic views. The following scenarios provide examples of such situations.

Scenario 1

You want to create a 3-terminal MOSFET and a 4-terminal MOSFET for inclusion in a library. To do so, you should create a single MOSFET cell with two interfaces, each having its own symbol. Library users can instance either symbol in their designs.

Cell name	NMOS
Interface name 1	NMOS3
Symbol name 1	NMOS3
Interface name 2	NMOS4
Symbol name 2	NMOS4

Scenario 2

You have a component called CORE with 400 terminals. You want one symbol that exposes all the terminals, and another that exposes a subset of 100 of the terminals. You should create a single cell with one interface, one schematic and two symbols. One symbol has all the ports, the other symbol has the subset of 100 ports.

Cell name	Core
Interface name	Interface
Symbol name 1	Symbol_400
Symbol name 2	Symbol_100
Schematic name	Schematic

Scenario 3

A library designer has a standard cell library with a single implementation of each cell, and wishes to provide two different pictorial representations of each cell for use as symbols, to adhere to the pictorial representations from two different standards organizations. He should create a single interface, a single schematic, and two symbols for each standard cell, both symbols having the same ports.

Cell name	NOR
Interface name	Interface
Symbol name 1	ACM_symbol
Symbol name 2	IEEE_symbol
Schematic name	Schematic

Scenario 4

You have a primitive component, a transistor, that you wish to setup with two sets of default properties on the symbol. Create a cell with a single interface but multiple symbols, each symbol having a different set of properties.

Cell name	NMOS
Symbol name 1	N1
Symbol name 2	N2
Schematic name	Schematic

Scenario 5

You want to have a single symbol of an amplifier with a basic and a high precision implementation of the schematic so you can switch between the two to trade off accuracy and run time in different simulations. (The high precision schematic could contain, for example, additional parasitic information back annotated from layout.) You would create a cell with a single interface, a single symbol, and two schematics.

In a case where there are two schematics and a single symbol, there is an ambiguity in which schematic to use. S-Edit will use the schematic whose name matches the symbol name. To switch from using one schematic to another, use **Cell > Rename View** to change the symbol name to match that of the desired schematic. In this example, to switch from using the basic schematic to using the precision schematic, you would change the symbol name from “Amplifier_basic” to “Amplifier_precision.”

Cell name	Amplifier
Interface name	Interface
Symbol name 1	Amplifier_basic (or Amplifier_precision, depending on the schematic you want to use.)
Schematic name 1	Amplifier_basic
Schematic name 2	Amplifier_precision

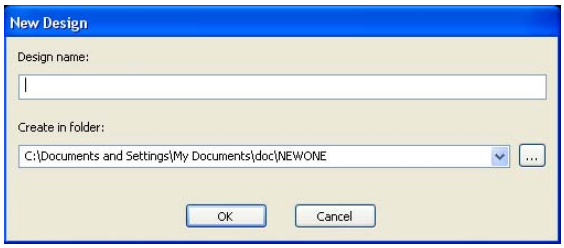
Creating a Schematic Project

An S-Edit project consists of the design itself, plus any number of libraries. The design contains cells, which may reference other cells in the same design or may reference cells in libraries. A reference to cell “A” consists of the instancing of a symbol of cell “A” within the schematic of another cell. A library is simply a design that is referenced by another design.

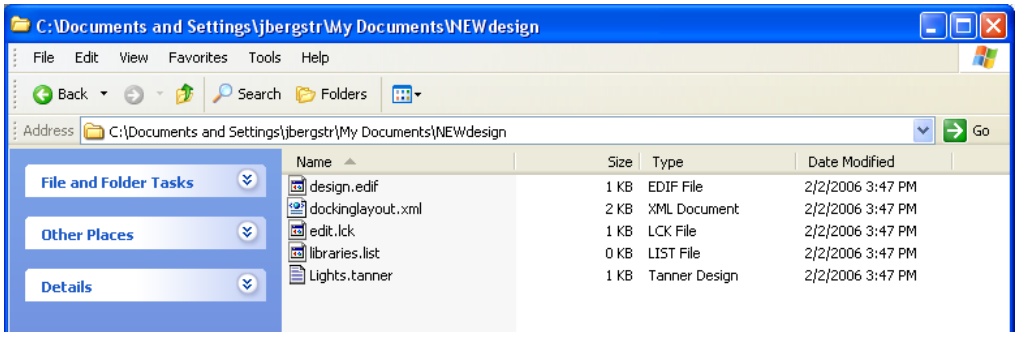
A library must be added to a list of libraries available to a design before its cells can be referenced. When you open a design, S-Edit will also open all libraries that are referenced by that design. There is no functional difference between a library and a design, so in the Libraries navigator, the design name is shown in black and library names in blue.

Creating a New Design

Use **File > New > New Design** to enter the name and directory location for a new design.



In either case, S-Edit creates a folder with the design name plus other files that comprise the complete design.



S-Edit Project File Structure

S-Edit stores design information in several different files in the {*designname*} directory.

- **design.edif** is the design itself.
- **design.old.edif** is a backup of the design that is overwritten each time the design is saved.
- **{*designname*}.tanner** is the TCL file that launches a design. It specifies the path, cell, and windows to open, and references the other three files
- **dockinglayout.xml** stores any previously used (or else the default) workspace settings
- **libraries.list** references the libraries used in the design. Library path names can be edited if desired.
- **edit.lck** is a file that prevents an open design from being opened elsewhere.
- The **setup** folder stores any changes that have been saved from the Setup dialog (see Saving and Loading Setup Options on page 17).

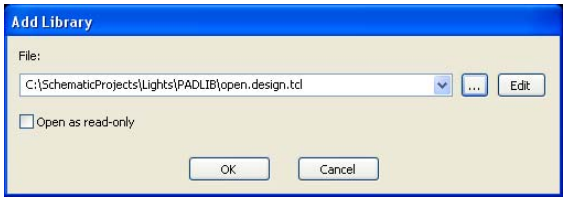
Creating a New Text File

Files in the context of S-Edit menus are always text files, typically of the type for which the S-Edit text editor provides syntax highlighting (C, log, SPICE, EDIF, etc.). Use **File > New > New File** (**Ctrl + N**) to open the text editor in the design window.

Adding a Library

When you load an existing design, any libraries referenced by that design are also loaded and will appear in the Libraries navigator. When you create a new design, or if you want to reference cells in a library that is not currently open, you will need to explicitly load that library.

Use either **File > Open > Add Library** or the **Add** button in the Libraries navigator to open the dialog shown below.



When you add a library, S-Edit will also load the other libraries it references.

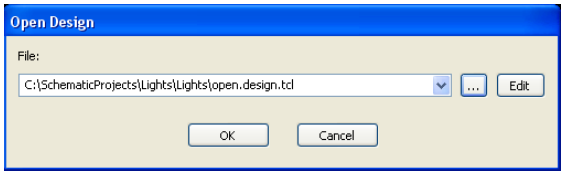
For portability, library locations are saved to the **libraries.list** file with a relative path if the library is in either a subfolder of the design or a descendant folder of the design. However, when the folder branches above the level of the design, we write an absolute path.

Opening an Existing Design or Text File

Use **File > Open > Open Design** to open an existing design.

Each design in S-Edit has a script file named "***filename*.tanner**" that opens the design and specifies the related libraries, interface and related project settings.

Enter the path to the design under **File:**, select from the drop-down menu, or use the **...** button to browse to the folder with the ***filename*.tanner** TCL file for the design you want to open.

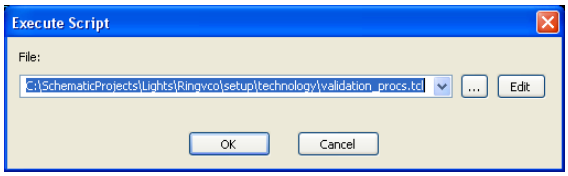


Opening a TCL File for Editing

Use **File > Open > Open File** to open a any kind of text file in the S-Edit text editor . This command lets you open a TCL file without executing it.

Executing a TCL File

Use **File > Open > Execute Script** to find and execute TCL file in the Command window. You can also drag a TCL file icon from a browser and drop it into the Command window to run it.



Closing a Design

When you close an S-Edit design file, the document is removed from memory. To close a design, use the **File > Close > Close Design [*filename*]** command.

Saving a Design, Library or Text File

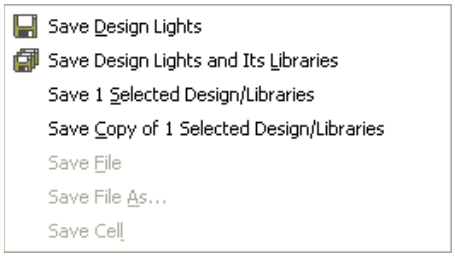
Use the **File > Save > ...** commands to save a design or libraries, in any combination. Note that simply renaming the directory will not change the name of a design! You must use the "**Save Copy of 1 Design/Libraries**" command with only the primary design selected to be able to rename a design.

When multiple designs are open, the one in the currently active window will be saved. (The design name is displayed in the save menu for confirmation.)

"**design.old.edif**" is a backup of the design that is overwritten each time the design is saved.

Note that **File > Save > Save Copy of ...** is different than the "save as" command. After the **Save Copy of Design** operation, S-Edit returns to editing the original design, *not the newly saved copy*.

If just a primary design is selected you will be able to rename the design. If libraries are also selected it will not be possible to enter a new name.



Save Design [*filename*]	Saves the specified design only, and does not save changes to libraries.
Save Design [*filename*] and Its Libraries	Saves the specified design and its libraries.
Save [*number*] Selected Design/Libraries	Saves only those designs or libraries selected in the Libraries navigator. For confirmation, the number of files selected is dynamically updated in the save menu.

Save Copy of [*number*] Selected Design/Libraries

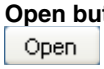
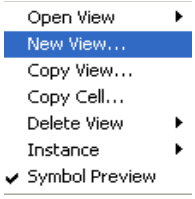
Saves a copy of the selected designs or libraries to the specified new location.

Note: Once the files are copied, S-Edit returns to the file *that was active prior to the command*.

Cell and View Operations

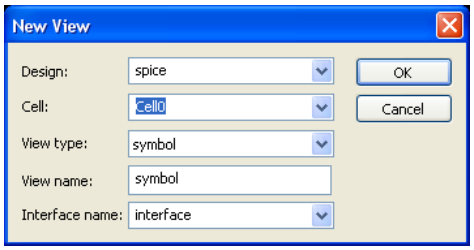
S-Edit has a highly flexible interface that offers many ways to perform key operations. In addition to the standard menu and keyboard shortcuts, you can use the following methods to perform cell and view operations.

Shortcuts for Cell and View Commands

Operation	*Interface: action*	*Behavior*
▪ **Open a cell**	**Libraries navigator: Double-click**	Highlight a cell in the list and **double-click** to open it. The view type opened will be the same as what was last opened.
▪ **Open a cell**	**Libraries navigator: Open button** 	Highlight a cell in the list and press the **Open** button to open it in schematic view.
▪ **Open a cell**	**Design area: Double-click**	Select a cell in the design area and **double-click** to open it in schematic view.
▪ **Open a view**	**Libraries navigator: drop-down menu** 	Highlight a cell in the list and use the drop-down arrow next to the **Open** button to select one of its views to open.
▪ **Open a view** ▪ **Create a new view** ▪ **Copy a view** ▪ **Copy a cell** ▪ **Delete a view** ▪ **instance a cell** ▪ **Symbol Preview**	**Libraries navigator: Right-click** and **select from menu** 	Highlight a cell in the list, **right-click** to open the context-sensitive menu, and click on the desired operation. **Symbol Preview** opens or closes the symbol preview window.

Creating a New View (Cell > New View or “N”)

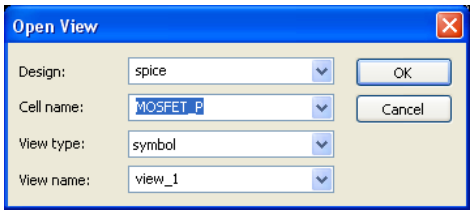
To create a new cell or a new view of an existing cell, use **Cell > New View** (shortcut **N**). Simply select from the drop-down menus in each field.



Design	From the drop-down list of those open, select a file to which the cell will be saved.
Cell	Enter or select a cell name.
View type	Specify a view type for the cell.
View name	Specify a view name for the cell.
Interface name	Specify the interface view to which it belongs.

Opening a View (Cell > Open View or “O”)

The **Cell > Open View** command (shortcut **O**) lets you open a cell view from any of the designs currently loaded in S-Edit.



Design — Select a design file or library file from the menu of those open in S-Edit.

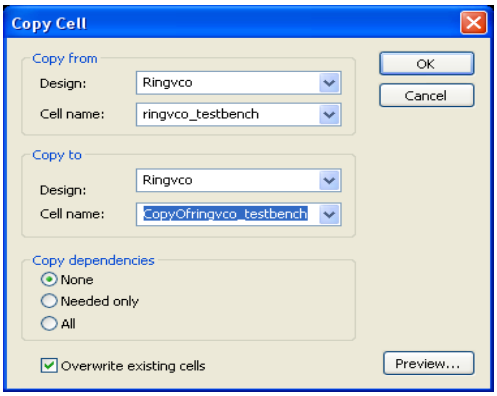
Cell name — Select the cell to open from the drop-down menu.

View type — Select the type of view to open.

View name — Select a specific view to open.

Copying a Cell

Use the **Cell > Copy Cell** command to duplicate a cell and all its views. Cells can be copied within the current design or from one design to another.



Copy from — **Design:** select a source design or library file from the menu of those open.

Cell name: select a source cell from the drop-down menu.

Copy to — **Design**: select a destination design file from the list of those open.

Cell name: enter a name for the new cell. S-Edit will prepend “CopyOf” to the cell name by default if you do not enter a name.

Copy dependencies

None: only the select cell is copied.

Needed only: only the instanced cells that do not exist in the destination file are copied.

All: all instanced cells within the cell are also copied, down through the hierarchy.

Note that S-Edit only checks for cell name during this operation, not cell contents or save date.

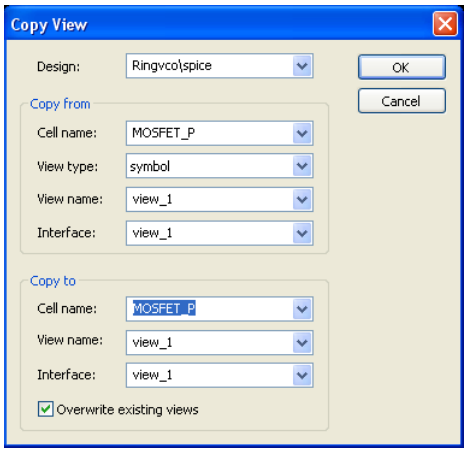
Overwrite existing cells

When checked, the copy will overwrite the existing cell(s) of the same name.

Copying a View

Use the **Cell > Copy View** command to copy a single view of a cell to the same or a different target cell. A view can only be copied within the same library of a design. See the How to Use Multiple Views on page 34 for some of the ways it can be useful to define multiple views of a given cell.

You can use this dialog as a shortcut to creating a new cell by entering a new name in the target **Cell name** field. Note that if you enter the name of an existing cell with the overwrite option active, if all other view names are the same as in the existing cell, the existing view will be replaced.



Design

Select a source design or library file from the menu of those open.

Copy from

Select a **Cell name**, **View Type**, **View name**, and **Interface** to copy.

S-Edit will preprend “CopyOf” to the cell name by default if you do not enter a name.

Copy to

Select a **Cell name**, **View Type**, **View name**, and **Interface** for the new view from the pulldown menu, or enter new names.

Overwrite existing cells

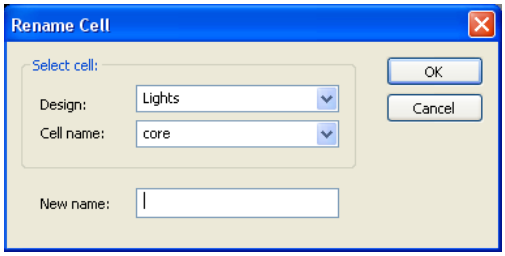
When checked, the copy will overwrite the existing view of the same name.

Instancing a Cell

Instances in a cell are references to other cells. These cells might be common library cells (such as basic circuit elements) or larger, custom-designed cells. Instances are dynamically linked to their source cell so that any change you make to a source cell is reflected in each higher-level instance of that cell. (See Placing and Naming Instances on page 69.)

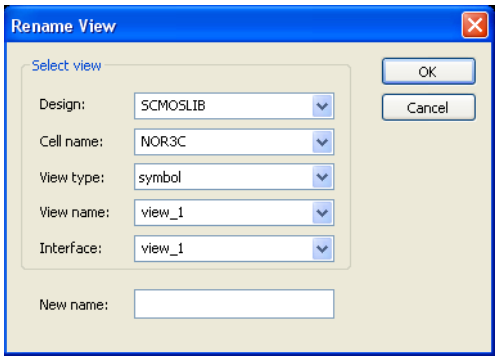
Renaming a Cell

The **Cell > Rename Cell** command will rename a cell but not its views. This operation keeps the renamed cell open.



Renaming a View

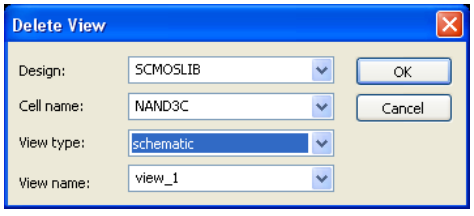
Use the **Cell > Rename View** command to rename a view.



Deleting a View

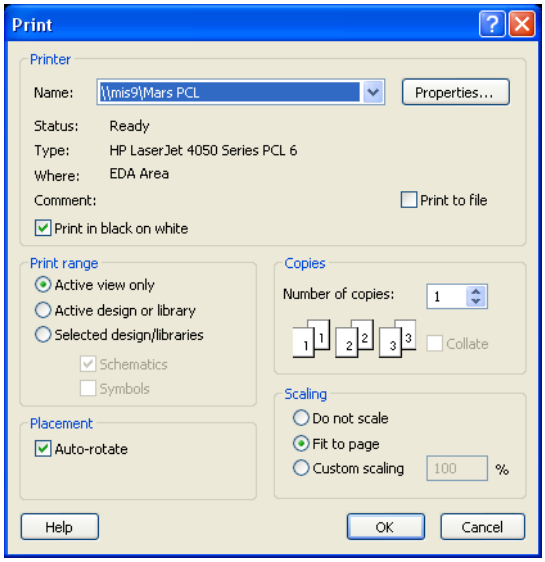
The command **Cell > Delete View** deletes a specified view, or all views of a cell. When you issue this command, S-Edit displays the **Delete View** dialog box, which prompts for the name of the cell to be deleted. A symbol view or its interface command cannot be deleted if the symbol is instanced.

When the last view of a cell is deleted the cell itself is also deleted.



Printing a Design

File > Print prints the active work area. Use **File > Page Setup** to specify the layout of printed pages (see Setting Page Size on page 23).



Printer — **Name**, **Type**, **Where**, and **Comment** all identify the active printer. **Status** describes the state of the selected printer—busy or ready.

Properties — Opens the **Printer Properties** dialog for additional printer properties.

Print in black on white — If this is not checked, the design will print just as it appears on screen.

Print to file — When this is checked, S-Edit creates a .PRN format file to the location you specify.

Print Range — Select **Active view only**, **Active design or library** or **Selected design/libraries** to specify the range of pages to print from a file. You can also print just **Schematics** or just **Symbols**.

Copies — Enter the number of copies to print.

Collate — When checked, prints copies of a file in proper page order. If not checked, each copy of the first page will be printed consecutively, then the second page, etc.

Placement — When **Auto-rotate** is checked, the image will be rotated to best fit the selected page size and scaling.

Scaling — **Do not scale** prints the design as set in in **Setup > Technology > Schematic Page**.

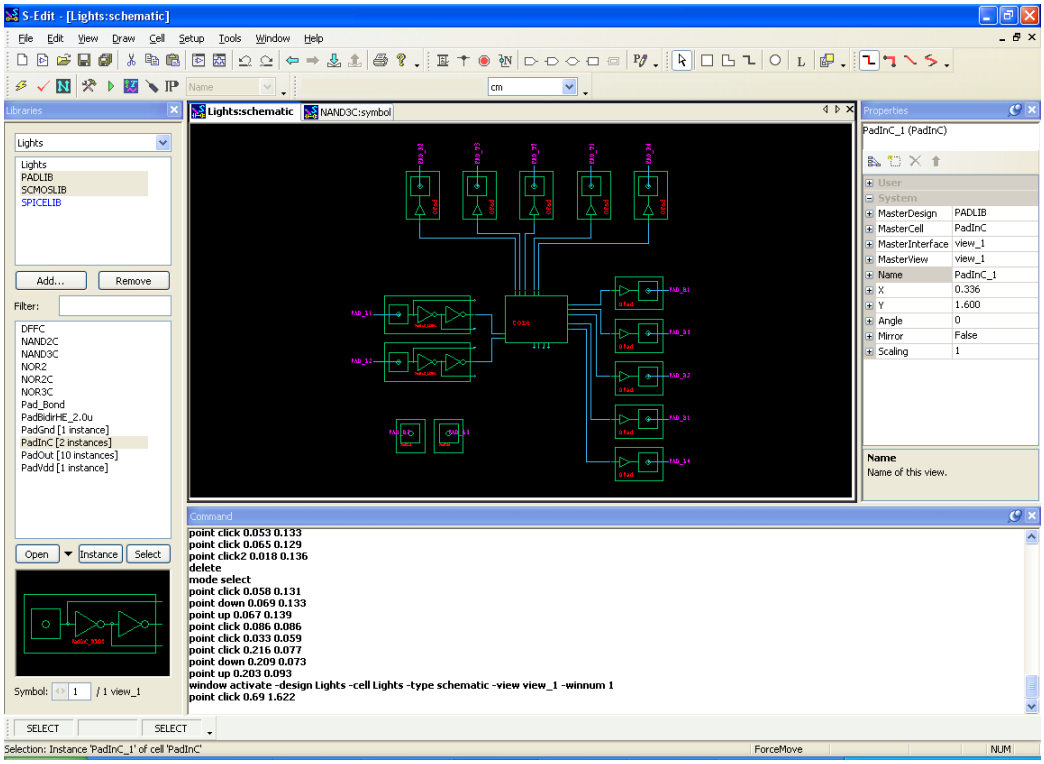
Fit to Page scales the design to fit the paper size you are printing to.

Custom scaling lets you use percentage values to set the size of the print out with respect to the page layout set in **Setup > Technology > Schematic Page**.

3 Navigating and Viewing a Design

The Work Area

The visible portion of the design area of a design, where you create, view, and edit objects, is called the *work area*. You can move or pan the work area to show a different part of the design area, or change its magnification to show a larger or smaller part of the design.



When you resize the work area, S-Edit expands or contracts the view in the direction in which you resized, without changing the magnification.

S-Edit's pan and zoom operations are always *active*: you may pan or zoom while drawing, moving, or editing an object—and *interruptible*: once you have initiated a panning or zooming operation, you need not wait for the screen to redraw completely before you initiate another one.

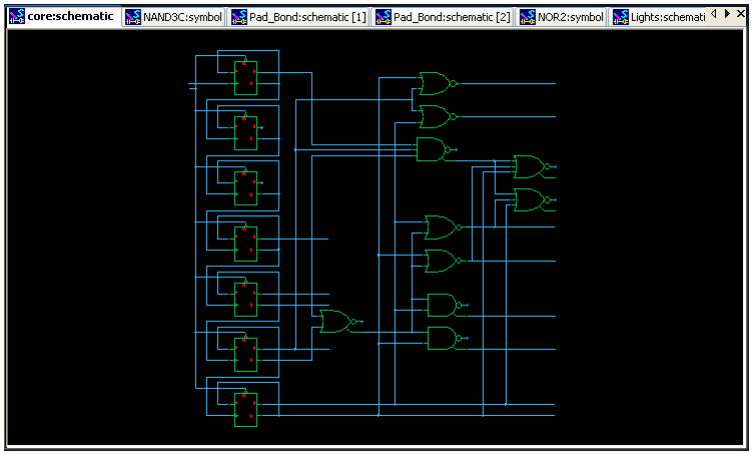
Opening Design Windows

You can open multiple design windows in S-Edit. Design windows can be tiled or cascaded, but the default view is as tabs. If you right-click in the tab area you can use the context-sensitive menu there to

"**Close All**" open views or "**Switch Tab Placement**" from the top to the bottom of the window and vice-versa.

Use the arrows in the upper right corner to scroll to open windows that are offscreen. A filled-in arrow indicates the presence of offscreen windows, otherwise is it outlined.

When you maximize one design window, they will all be maximized. You can right-click directly on a tab to open a context-sensitive menu to "**Close**" that tab or make it "**Dockable**."



Reusing Design Windows

You can also control how S-Edit reuses design windows to avoid having too many windows open at once.

By default, each time you open a new view or cell, S-Edit opens it in a new window. You can keep this default behavior, or you can use **Setup > Preferences > General** to change the default so that S-Edit opens only one window at a time, regardless of the view type. For example, if you have a schematic view of cellA and you open a symbol of cellB, S-Edit will replace the cellA display with cellB, reusing the window that is open.

You can force a new window to open at any time by holding the **Ctrl** key down while opening a view.

Shortcuts for Changing Windows and Views

Redrawing the Screen

You can refresh display in the active window at any time by using **View > Redraw** (shortcut **Space**).

Cycling Focus Between Windows

Use the **View > Goto > Previous** and **View > Goto > Next** commands to cycle focus through each pane that has been active in the current editing session.

You can also use the **Backward** ⇦ and **Forward** ⇨ icons to step backwards or forwards through the previous and next views that were opened. Note that the **Forward** button is only available after stepping **Back**.

Cycling Views Within a Window

Use **View > Exchange** (shortcut **X**) to return to the previous view within a given pane after you execute any zoom or pan command. You can use this command to toggle back and forth between two views. If you return to a previously stored view after resizing the work area, S-Edit will alter the aspect ratio of that view to fit the current window.

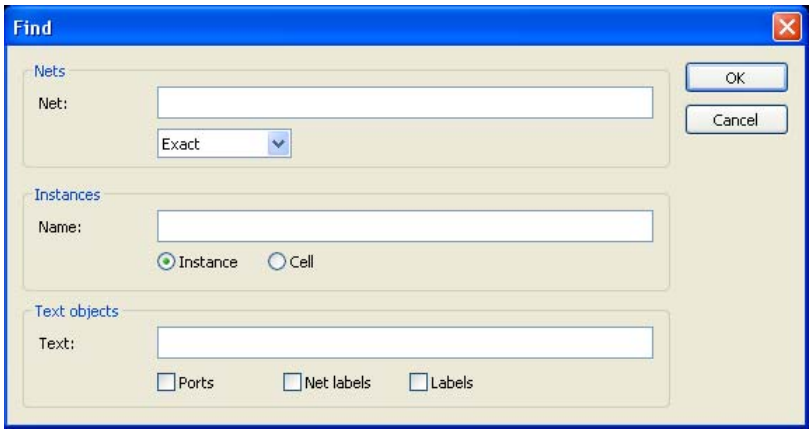
Use **View > Cell View > Symbol** and **View > Cell View > Schematic** to select one of these view types for the active cell. Or, use **View > Cell View > Cycle View** to cycle through all the views for a given cell.

Showing and Hiding Grids

You can use **View > Display > {Display Major Grid, Display Minor Grid or Display Origin}** to display, or not, each of these elements. This setting applies just to the active design.

Finding Objects

The **Edit > Find** dialog lets you search for nets, instances and text in the current view. Use a space to separate multiple entries. Note that the find operation is additive, so that objects that are selected remain highlighted unless explicitly deselected before a subsequent search



.

Nets Use **Exact** to find a unique net. Use **And** to find all nets of a name or range, such as those that originate from a bus or bundle. Use **Or** to find any of the net names entered.

Name Select the type of search that will be performed. Use **Instance** to search by instance name or **Cell** to search by cell name.

Text Specify **Ports**, **Net labels** or **Labels** in any combination to limit the search.

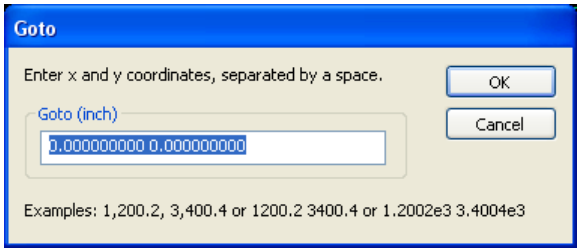
Panning

You can pan to different portions of the design area by using the arrow keys, or one of the **View > Pan** commands.

Command	*Shortcut*	*Function*
View > Pan > To Selection		Centers the view over the selected objects. Depending on the magnification, all selected objects may not be visible in the resulting view.
View > Pan > Left	(left arrow key)	Moves the work area to the left by one quarter of the width of the display.
View > Pan > Right	(right arrow key)	Moves the work area to the right by one quarter of the width of the display.
View > Pan > Up	(up arrow key)	Moves the work area up by one quarter of the width of the display.
View > Pan > Down	(down arrow key)	Moves the work area down by one quarter of the width of the display.
View > Pan > To Edge **Left Edge** **Right Edge** **Up Edge** **Down Ege**		Moves the display window in the direction you indicate, to the farthest edge of all objects in the design area.

Panning to a Specific Location

Use **View > Goto > Coordinates** to center the view on a specific coordinate. Enter x, y coordinates in display units, as measured relative to the origin, separated by a space.



Zooming

You can zoom in S-Edit using the mouse buttons, a mouse wheel, the keyboard, or menu commands.

Zooming with the Mouse

View > Zoom > Mouse (shortcut **Z**) changes the function of the left mouse button for a single operation to enable a *zoom box*. When the zoom box is enabled, the next two mouse clicks define opposing corners of a rectangle. S-Edit zooms the display window directly to the area inside the rectangle.

You can also click and drag to draw the zoom box. Note that S-Edit must maintain the correct height-to-width ratio of the display, so the new work area may not be exactly the region contained inside the rectangle.

After a **Zoom Mouse** operation the mouse buttons revert to their previous functions.

Zooming with the Mouse Wheel

Spin the mouse wheel up to zoom in and down to zoom out. In both cases the zoom will be centered on the cursor location.

Zooming with the Keyboard

Use the plus key (+) to zoom in by a factor of 1.5 and the minus key (−) to zoom out by a factor of 1.5.

Zooming to Selected Objects

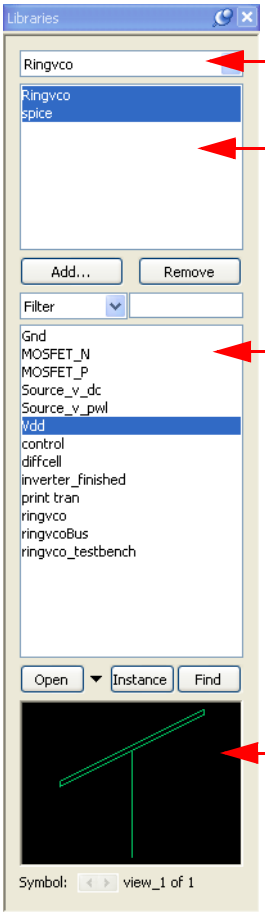
Use **View > Zoom > To Selection** (shortcut **W**) to zoom the display window to encompass only the selected object(s).

Zooming to Show the Entire Contents of a Cell

View > Fit (shortcut **Home**) zooms the display window so that all objects in the design area are visible in the work area.

The Libraries Navigator

The Libraries navigator provides an easy way to view and manage your libraries and cells. You can use it to add and remove cell libraries and to list the cells in all the libraries that are loaded. You can also use this window to open, copy, or instance a cell.



Select a design to open from this menu of the designs that are loaded.

The **Libraries** navigator shows the designs and libraries that are open.

The **Cell list** shows the cells available from the design and libraries highlighted in the **Libraries** navigator.

The **Preview Pane** displays the symbol view for the selected cell. You can right-click anywhere in the **Libraries** navigator to toggle this preview on or off.

(drop-down menu) The uppermost field is a drop-down menu that lets you select a design to open from those that are loaded.

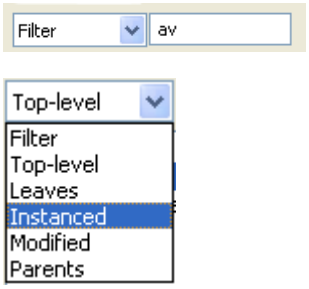
Libraries navigator This pane shows the designs and libraries that are open.

When you highlight a file in the Libraries navigator, the cell browser displays the cells it contains. Use the **Ctrl** key to select multiple libraries and list the cells in each of them. Black text indicates the design file, blue text indicates library files.

Add Use this button to **Add** designs and libraries.

Remove Use this button to **Remove** designs and libraries.

Filter



The **Filter** option lets you enter letters or numbers to filter the cell list to show only the cell names that contain those characters. For example, type “av” to show only cells with the letters “av” in their name.

The remaining options in the pull-down menu provide hierarchical filters:

- **Top-level**—filters the list to show only those cells that are not instanced in the design.
- **Leaves**—filters the list to show just primitive cells that you cannot push into.
- **Instanced**—filters the list to show only those cells that are instanced in the design.
- **Modified**—filters the list to show just those cells that have been changed but not saved.
- **Parents**—filters the list to show only the cells that instance the active cell.

Cell list

Lists the cells contained in each of the files highlighted in the Libraries navigator. Information in square brackets shows how many instances of a given cell are used in the active view.

Open

Use this button to open the highlighted cell. (See also Opening a View (Cell > Open View or “O”) on page 42.)

Instance

Use this button to quickly instance the highlighted cell. (See also Placing and Naming Instances on page 69.)

Find

Use this button to highlight all appearances of the selected cell in the view.

(preview pane)

This preview pane displays the symbol view for the selected cell. You can right-click anywhere in the Libraries navigator to toggle the preview on or off.

Symbol __ / view_x of y

Displays the number, view type and view names for the highlighted cell.

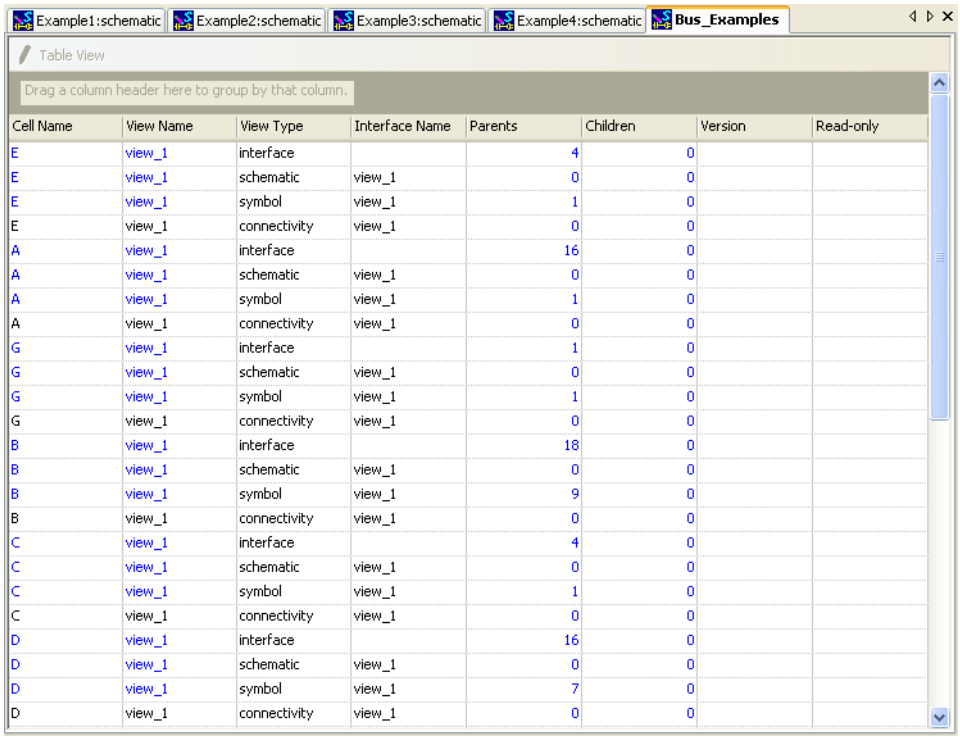
The View Navigator

The View navigator is a table that shows all the cell views in a design. (To open the View navigator for a library, right click on the library in the library list of the Libraries navigator, and select View Navigator from the context-sensitive menu.)

Use the View navigator to quickly see the structure of a design by cell, view, and view type.

For any given cell, the View navigator table also shows the number of “*children*”—the number of instances a cell contains, and “*parents*”—the number of cells in which it is contained as an instance.

.



Cell Name	View Name	View Type	Interface Name	Parents	Children	Version	Read-only
E	view_1	interface		4	0		
E	view_1	schematic	view_1	0	0		
E	view_1	symbol	view_1	1	0		
E	view_1	connectivity	view_1	0	0		
A	view_1	interface		16	0		
A	view_1	schematic	view_1	0	0		
A	view_1	symbol	view_1	1	0		
A	view_1	connectivity	view_1	0	0		
G	view_1	interface		1	0		
G	view_1	schematic	view_1	0	0		
G	view_1	symbol	view_1	1	0		
G	view_1	connectivity	view_1	0	0		
B	view_1	interface		18	0		
B	view_1	schematic	view_1	0	0		
B	view_1	symbol	view_1	9	0		
B	view_1	connectivity	view_1	0	0		
C	view_1	interface		4	0		
C	view_1	schematic	view_1	0	0		
C	view_1	symbol	view_1	1	0		
C	view_1	connectivity	view_1	0	0		
D	view_1	interface		16	0		
D	view_1	schematic	view_1	0	0		
D	view_1	symbol	view_1	7	0		
D	view_1	connectivity	view_1	0	0		

You can also click on view names from the View navigator to open the respective design windows for the schematic or symbol, or on the interface view name to open a table of the ports in that cell.

If you click on the **Parents** link on the symbol view of the cell, the View navigator will display all the cells that instance the symbol of interest. Similarly, if you select "Parents" in the filter drop-down of the Libraries list you will see the list of cells that instance the active view.

4 Drawing, Selecting and Editing Objects

Object Types

Drawn objects in S-Edit are either *electrical objects*—for example, wires, ports or solder points, or *annotation graphics*—geometric objects without electrical significance.

To draw an object, you must select a tool from either the *Drawing toolbar* (see Drawing Tools for Annotation Graphics on page 56) or the *Electrical toolbar*, (see Drawing Tools for Electrical Objects on page 59) and also a *segment type* from the *Segment toolbar*.

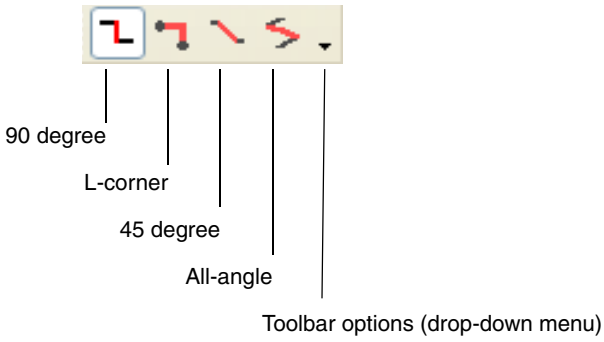
Annotation and electrical tools are "sticky"—once you select a tool, S-Edit will draw only objects of that type until you choose a different tool.

You can change segment type at any point while drawing a geometric object, for example from orthogonal to 45° segments.

Segment Types

S-Edit has various types of lines segments—90°, L-corner, 45°, all-angle, curve—you can select from when drawing. You can change segment type at any point during a drawing operation by simply selecting a different icon from the Segment toolbar.

To draw a line, select **Path** and a segment type, then position the cursor where you want to place the starting point. **Left-click** to begin drawing and to place vertices. **Double-click** to place a vertex and end drawing. **Right-click** to back up to the vertex that was placed last.



90° Segments

When you use this segment option, S-Edit will only draw geometry having 90° angles.

L-Corner Segments

The L-corner segment draws an L-shaped line from the last placed vertex to the current cursor position.

You can toggle the orientation of the 90° angle by pressing the “L” key before placing the next L-shaped segment.

The L-shaped line behavior remains active when you use the middle button, so the action of removing vertices is modified to remove two vertices with each middle-button click.

45° Segments

When you use this segment option, S-Edit will draw geometry having 45° or 90° angles.

All Angle Segments

With this segment option, S-Edit will draw geometry having any angle.

Using the Mouse to Draw

Drawing instructions are given for the most commonly-used drawing interface, a three-button mouse (with or without a mouse wheel), as default. Other drawing methods are discussed in Command Window on page 12 and Drawn Properties of Objects on page 60.

Use the mouse as follows for drawing operations:

LEFT-CLICK

Use a left-click (**DRAW**) to start a line segment or place a vertex.

MIDDLE-CLICK

Use a middle-click (**BACKUP**) to remove vertices in reverse order of placement. This back-up action is only available while an object is being drawn. The **BACKUP** button can also be used to move a vertex.

DOUBLE-CLICK

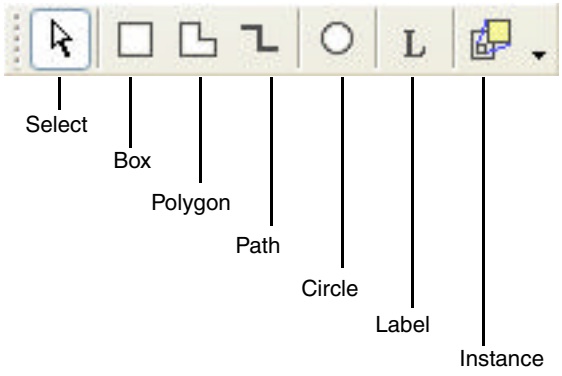
Use a double-click (**END**) to end the drawing operation by placing a vertex.

RIGHT-CLICK

Use a right-click to end the drawing operation without placing a vertex. (This results in what can appear to be "backing up" to the last vertex placed.)

Drawing Tools for Annotation Graphics

Annotation tools are available in any view type.



Boxes (Draw > Box or "B")

To draw a box, press the DRAW mouse button and drag the cursor away from the starting point to determine the opposite corner (and therefore the length and width) of the box. Release the DRAW button at the desired opposite corner.

Polygons (Draw > Polygon or "P")

To draw a polygon, click the VERTEX (left) mouse button at the starting point, move the cursor and LEFT CLICK to determine the second vertex. Repeat the process for each successive vertex. Click the BACKUP mouse button to remove the last vertex that was placed. A polygon can have any number of vertices.

When you click the END button, coincident vertices (two or more vertices occupying the same location) and colinear vertices (three or more vertices lying on the same straight line) are eliminated.

When you draw a polygon the mouse buttons become VERTEX, BACKUP, and END, respectively.

Paths (Draw > Path)

Note that this is simply a line drawing tool—a path has no electrical properties. A path can have any number of vertices.

To draw a path, click the VERTEX (left) mouse button at the starting point, move the cursor and LEFT CLICK to determine the second vertex. Repeat the process for each successive vertex. Click the

BACKUP mouse button to remove the last vertex that was placed. Click the END button to complete the operation.

The mouse buttons for this operation are VERTEX, BACKUP, and END.

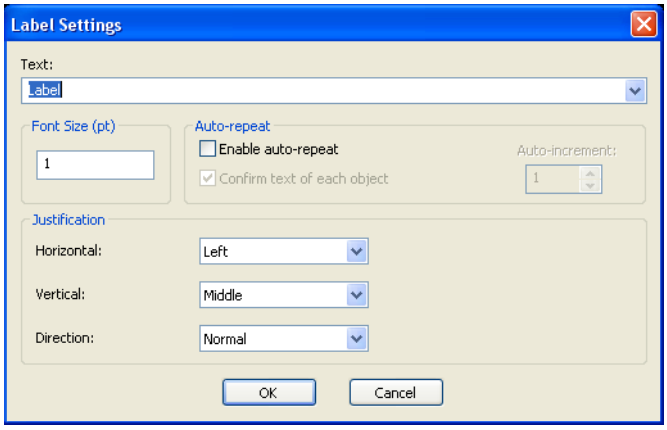
Circles (Draw > Circle)

To draw a circle, press the DRAW mouse button and drag the cursor away from the center starting point to determine the radius of the circle. Release the DRAW button at the desired radius.

The mouse buttons for this operation are DRAW and SELECT.

Labels (Draw > Label)

When you select the **Label** icon and click in the design area, S-Edit opens the **Label Settings** dialog so you can enter text and choose how it will be displayed. Click **OK** to place the label.



Text Enter your label text in this field.

Font Size Enter the text size.

Auto-repeat

If **Enable auto-repeat** is not checked, S-Edit opens the **Label Settings** dialog just once, places the label when you click **OK**, and reverts to **Select** mode.

With **Enable auto-repeat** checked you can place labels repeatedly by clicking in the design area, without having to re-open the label settings dialog each time. S-Edit remains in label mode until you select a different mode or press **ESC**. The following controls may be added:

- **Confirm text of each object**—when checked, re-opens the **Label Settings** dialog each time a label is placed so you can enter changes to the next label placed.
- **Auto-increment**—when **auto-repeat** is active and the last character in the **Text** field is an integer, S-Edit automatically increments the label text by the value entered here.

Justification

All text orientations are with respect to the origin point of the label, indicated by a cross.

Horizontal: select **Left**, **Center** or **Right** horizontal text alignment.

Vertical: select **Top**, **Middle** or **Bottom** vertical text alignment.

Direction: select **Normal**, **Upside down**, **Down** or **Up** for the text direction.

Auto-Repeat for All Types of Labels

For labels, port labels and net labels in S-Edit, when **Enable Auto-repeat** is checked, you will remain in label mode so you can place multiple consecutive labels. Otherwise, S-Edit opens the label dialog just once, places the label when you click OK, and reverts to Select mode.

If **Confirm text of each object** is checked, the label dialog will reopen with each placement, allowing you to change the any setting. If it is not checked, the dialog will not reopen, and you can place as many identical consecutive ports as desired.

If the label name ends in a numeral and **auto-repeat** is active, when you place labels in succession, the number will increment by the amount entered in the **Auto-increment** field.

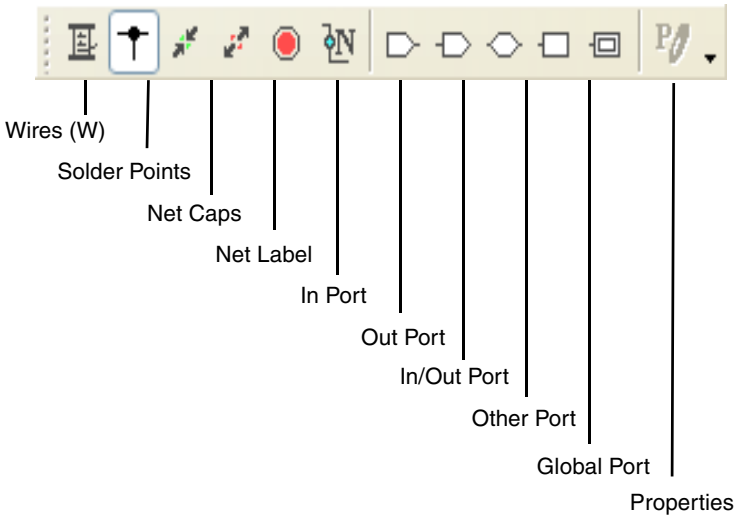
Note that you can use the **R**, **H** and **V** shortcut keys (for rotate 90°, flip horizontally and flip vertically) to change the orientation of a label before clicking to place it in the design area. This, as opposed to using the Properties window, is the suggested procedure.

Instance (Cell > Instance)

To instance a cell, highlight it in the Libraries navigator and click on the **Instance** icon (for more details, see Placing and Naming Instances on page 69).

Drawing Tools for Electrical Objects

In S-Edit, design connectivity is achieved by the proper arrangement of ports, labels, and wires to form nodes. They are drawn using the **Draw > Electrical** menu or with the electrical toolbar (only available in schematic view).



Wires (Draw > Electrical > Wire)

Note that wires and paths, while similar in appearance, are functionally different. Wires connect objects electrically; paths do not. S-Edit will draw only orthogonal wires.

The starting point is the first vertex of the wire. Wires can have any number of vertices. The mouse buttons for this operation are VERTEX, BACKUP, and END. To draw a wire, activate the Wire tool, then click the DRAW button to place the first endpoint of the wire. Continue clicking the DRAW button wherever you want to place a vertex, up to the last endpoint. Double-click the END button to place the last endpoint and finish drawing the wire, or simply place a vertex on a pin so that S-Edit will terminate the wire automatically. See also Drawing Wires on page 72.

Solder Points (Draw > Electrical > Solder Point)

Use solder points to define a point where wires cross and make an electrical connection (see Creating a Connection where Wires Intersect on page 73).

Net Caps (Draw > Electrical > Net Cap)

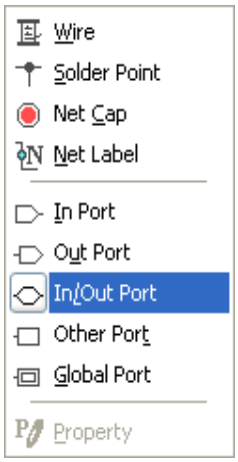
A net cap halts the propagation of a global net to any cell that instances the current cell. (See Naming Global Nets on page 82.)

Net Labels (Draw > Electrical > Net Name)

Net labels identify nets by placing text in the design area. Use the **NetLabel Port Settings** dialog to place text and control its size, position and alignment. (See Labeling Nets on page 76).

Ports

Use the toolbar icons or the **Draw > Electrical** menu to select and place ports. S-Edit will open a dialog where you can set the port name, size and alignment parameters. (See Adding Ports on page 74.)

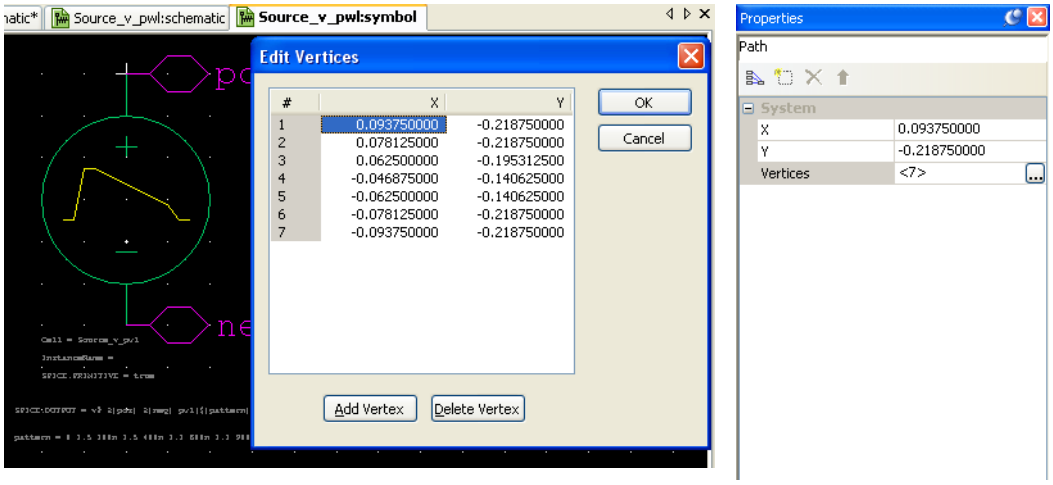


Drawn Properties of Objects

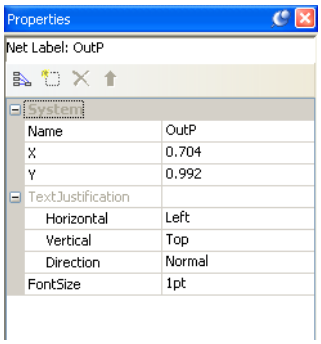
Cells and objects in S-Edit are characterized by drawn properties such as the coordinates of their vertices, the length and width of their lines, etc. You can use these properties to edit objects by changing the related values.

The set of drawn properties differs for each object. For example, a path, polygon, or wire is characterized by its vertex coordinates (**X1**, **Y1**), (**X2**, **Y2**), (**X3**, **Y3**), etc.

Note: To open a dialog like the one below, click once in the **Vertices** field of the Properties window to activate the ellipsis button and then again on the button to open the **Edit Vertices** dialog.



Similarly, a comment or net label is characterized by its name, location in coordinates, text size and orientation.



Electrical Properties of Objects

See Symbol Properties on page 88 and Editing Instance Properties on page 71 for.

Selecting Objects

You must select an object before editing it or modifying it in any way. You can select more than one object at a time to manipulate a group of objects as a single one.

When you select an object, it is highlighted. Or, if the object is an instance of a cell, a thin highlight appears defining its MBB [minimum bounding box, or perimeter]. When you deselect an object, the highlight disappears and subsequent operations do not affect that object.

S-Edit provides two ways of selecting objects: explicit selection and implicit selection.

Explicit Selection

There are several ways to make an explicit selection. You must select objects explicitly before performing the following actions:

- **Cut**, **Copy**, **Clear**, or **Duplicate** operations.
- Operations on a set of objects.
- Operations on one of several close or overlapping objects.

Selection by Clicking

Position the cursor over the desired object and click the **Select** button. S-Edit will deselect any previously selected object or objects.

Selection by Enclosing (Selection Box or type Select - Enclose)

Drag the cursor with the **Select** button held, forming a *selection box* around the objects. Draw the selection box so that it completely encloses all the objects you want to select but no others. (To select a port, you must enclose the starting point, but not the port text.) S-Edit will deselect any previously selected objects. (See Selection Behavior Options on page 26.)

Selection by Intersection (type Select - Intersect)

When you type select-intersect in the Command window, objects that are intersected by the selection box (i.e. partially enclosed) will be selected as well as fully enclosed objects. (See Selection Behavior Options on page 26.)

Extend Selection

To add an object to a set of selected objects, perform a click- or drag-selection with the Extend Select (**Shift + Select**) button held. Previously selected objects are *not* deselected.

Cycle Selection

You can also select separate objects in succession by clicking repeatedly near them. This *cycle selection* technique lets quickly cycle through several overlapping or adjacent objects to select the one of your choice.

In cycle-selection, S-Edit first selects the closest object within selection range. When you click again with the cursor in the same spot, S-Edit deselects the first object and selects the next closest object. You may continue clicking to select successive objects until there are no more within the selection range. At that point, all objects are deselected. If you click again with the cursor in the same spot the cycle will resume with the closest object.

The actual ordering of candidate objects for cycle-selection is as follows: (1) objects the cursor is *inside*, ordered according to the closest edge; (2) objects the cursor is *outside* (yet still within the selection range), ordered according to the closest edge. Objects that do not enclose the cursor can only be selected if they are within the selection range.

Select All (Edit > Select All) and Deselect All (Edit > Deselect All)

Use **Ctrl + A** to select all objects in the active window and **Alt + A** to deselect all objects in the active window.

To remove objects from a group of selections, hold the **Deselect** (**Alt + Select**) button while performing a click- or drag-selection.

Implicit Selection

You can **move**, **edit**, or **copy** an object without explicitly selecting it, if it is within the selection range set in Selection Behavior Options.

Clicking the MOVE-EDIT or COPY button with the cursor near such an object implicitly selects it for these operations. The object is deselected after the operation.

Deselection

To deselect an object or remove objects from a set of selected objects, hold the **Deselect** (**Alt + Select**) button while performing a click- or drag-selection.

Deselecting an object that was selected has no effect on the object.

Automatic Deselection

When you begin a move, edit, or copy operation, S-Edit checks to see how far the cursor is from the bounding box of all currently selected objects. If this distance is greater than the deselection range, then S-Edit deselects all current selections and performs the implicit selection to begin the operation.

Moving and Editing Objects

To move an object, select it (being careful not to select just an edge or vertex) and drag the object to the new position while holding the MOVE-EDIT button. To move multiple objects you must explicitly select them. S-Edit surrounds the objects with a *selection box* and maintains their relative positions during the move.

The function of the MOVE-EDIT button will change depending on the position of the cursor. If you click on or very near an object's edge or vertex with this button, you will move only that edge or vertex and thus change the object's size or shape.

Use Selection Behavior Options to specify the distance from a vertex or edge at which S-Edit will perform an edit rather than a move.

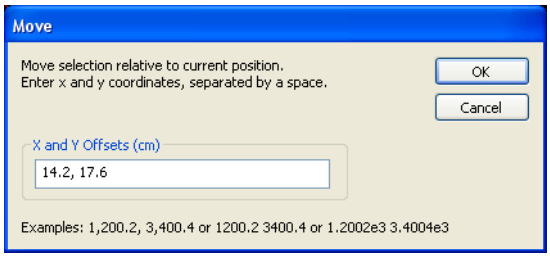
Maintaining Connectivity while Objects are Moved—"Rubberbanding"

Wires in S-Edit "rubberband"—that is, when you move an object, the wires remain connected (unless the force move operation is used. See Forcing a Move Operation Instead of an Edit (Draw > Force Move or Alt + M) on page 64.)

Moving Operations

Moving by a Specific Amount (Draw > Move By)

Use **Draw > Move By** to enter an amount by which the selected objects will be moved in the x and y direction.



Forcing a Move Operation Instead of an Edit (Draw > Force Move or Alt + M)

In normal drawing mode, the **edit** command is active when the cursor is within the edit range of an object's edge (see Selection Behavior Options on page 26), and the **move** command is active when the cursor is outside that range.

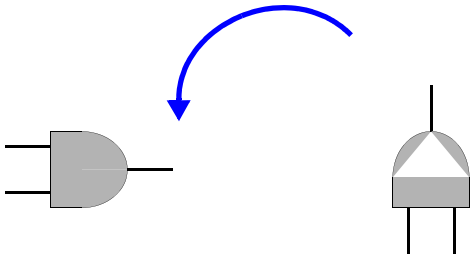
Draw > Force Move forces a move operation regardless of the cursor position. After the move operation is finished, S-Edit will revert to normal drawing mode, or you can cancel the force move operation by pressing **ESC**.

Note that a forced move does *not* preserve an object's electrical connections; that is, it does not rubberband the object.

Rotating Objects (Draw > Rotate 90 degrees or R)

Draw > Rotate 90 degrees (shortcut key **R**) rotates the selected objects by 90° counterclockwise about their starting point. For a single object the center of rotation is the object's origin. For multiple objects the center of rotation is the center of their MBB.

Note that you can use the **R** shortcut key to change an object's orientation before clicking to place it in the design area.

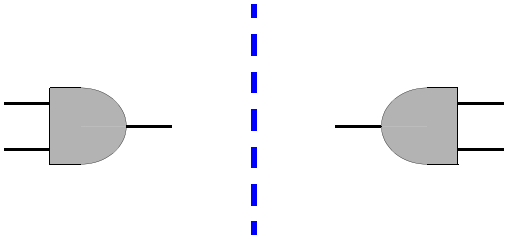


Alternately, the **Draw > Rotate** command allows you to rotate the selected objects counterclockwise by any degree around the starting point, with up to six decimal points accuracy. Use a negative value for clockwise rotation.

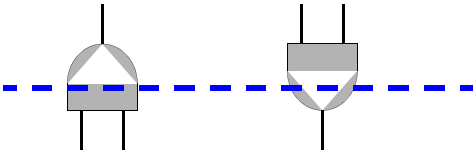
Flipping Objects (Draw > Flip)

These commands flip the selected objects through the center of their MBB, either horizontally (**Draw > Flip > Horizontal**, shortcut key **H**) or vertically (**Draw > Flip > Vertical**, shortcut key **V**).

Note that you can use the **H** and **V** shortcut keys to change the orientation of an object before you click to place it in the design area.



Draw > Flip > Horizontal (shortcut key **H)**



Draw > Flip > Vertical (shortcut key **V)**

Copying and Duplicating Objects

To copy objects, select the object(s) and do one of the following:

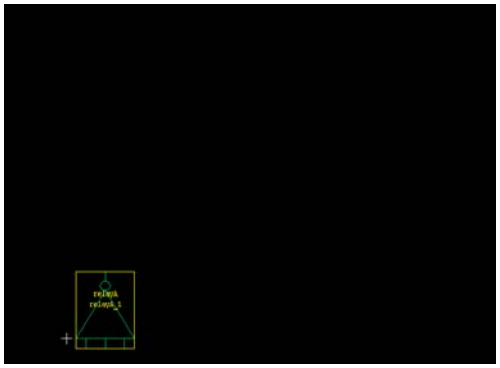
- Use the **Edit > Copy** (**Ctrl + C**) command and then the **Edit > Paste** (**Ctrl + V**) command.
- Use the **Edit > Duplicate** (**Ctrl + D**) command.

The **Copy** command puts the copy of the selected object(s) in the internal clipboard. The copy does not appear in the design area until it is placed with the **Paste** command. The copy remains selected after this operation.

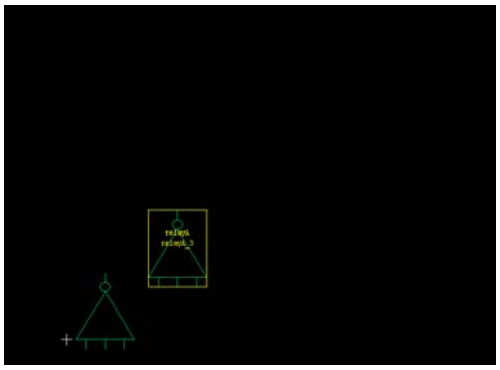
Creating Arrays Using the Duplicate Command

The **Duplicate** operation creates a duplicate of the selected object(s) and places it in the active cell, offset from the original by one grid point horizontally and vertically. The new objects stay selected and can be moved to a new offset.

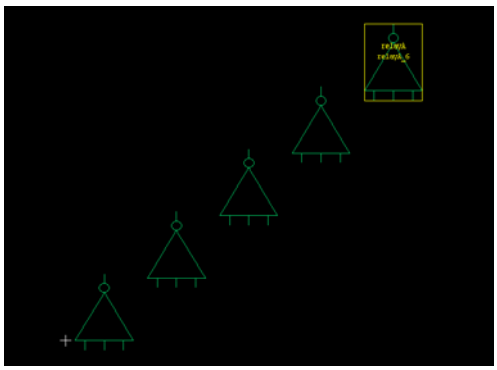
When you use the mouse-drag or **Duplicate** operation, S-Edit stores the offset of that operation. Subsequent use of the **Duplicate** command replicates both the object and the displacement, aiding in the rapid creation of regular structures like arrays. (See Creating an Array on page 78.)



Select an object and use **Edit > Duplicate** (**Ctrl + D**) to initiate the duplicate operation.



Click to place the first copy and set the offset.



Use **Ctrl + D** to continue to place copies with the same offset.

Pasting Objects

S-Edit stores cut or copied objects to an internal clipboard. To paste the contents of this clipboard, click in the design area, use the **Edit > Paste** (**Ctrl + V**) command, and click again.

When the paste command is active the clipboard objects are visible and move with the cursor (attached at the lower left corner of their MBB) until the second click places them.

Pasting Objects to Other Applications

You can use **Edit > Capture Window** in an S-Edit design file and to copy and paste the visible design window to another application in bitmap format.

Using Undo & Redo

Edit > Undo (Ctrl + Z)

You can use the **Edit > Undo** command (**Ctrl** + **Z**) to reverse the most recent edit operation and restore the S-Edit file to the state it was in before that operation. Specifically, **Undo** reverses a move; any resize, reorient or reshape operation; the draw, copy, or duplicate operations; deletions, and changes made using **Edit > Edit Object**.

Edit > Redo (Ctrl + Y)

Similarly, you can use the **Edit > Redo** command (**Ctrl** + **Y**) to restore changes reversed with a previous **Edit > Undo** operation.

Deleting Objects

Use one of the following commands to remove selected object(s) from a view:

- **Edit > Cut** (**Ctrl + X**)

The **Cut** command puts the deleted objects in the internal clipboard. From there they can be restored to the current cell or placed into another cell.

- **Edit > Clear** (**Del**)

The **Clear** command does not put the deleted objects into the internal clipboard.

5 Creating a Schematic

A schematic for a component is made by placing a number of symbols on the schematic page and then connecting the symbols together with wires at their defined connection points. A set of connection points connected together is called a net. You can make a symbol for a new schematic and then use it in other schematics, thus allowing for the construction of hierarchical designs.

Elements of a Schematic View

A schematic view contains any of the following elements:

Instances of Symbols

Instances of symbols refer to a particular symbol in a cell. A schematic may contain many instances of the same symbol or of different symbols. Instances contain graphics which provide the illustration of the symbol, and ports

Ports

Ports provide connection points for attaching nets. Ports on the symbol and ports on the instance are different in that ports on the symbol are shown as their port name, whereas ports on the instance do not. Ports on the schematic define how connections made to ports on symbol instances connect to nets on the schematic for that symbol.

Nets

A net is a wiring connection between two or more instance ports. A net can be a single wire, or a collection of wires called a bus or a bundle.

Properties

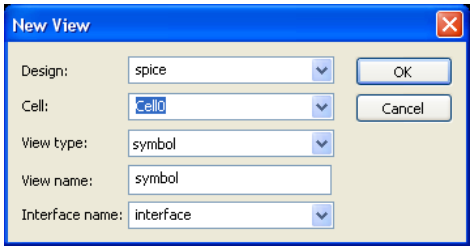
Properties are name-value pairs that are usually used to describe some characteristic of a device, such as a transistor length, width, or Source/Drain areas and perimeters. Properties can be put on an instance to override the value defined in the symbol, or new properties can be created on an instance.

Annotation Graphics

Annotation graphics are non-electrical objects such as boxes, polygons, paths, and labels used to add comment or illustrations to the schematic.

Creating a Schematic View

To create a new schematic view, use **Cell > New View** (shortcut **N**). Simply select from the drop-down menus in each field. You can also use **Cell > New View** to create an entirely new cell.



Creating a New View (Cell > New View or "N")

Design	Select a design file or library file from the drop-down list of those open where the cell will be saved.
Cell	Enter a cell name.
View type	Select the view type, i.e. schematic.
View name	Specify a view name.
Interface name	Specify which interface the new view belongs with.

Placing and Naming Instances

Instances in a cell are references to other cells. These cells might be common library cells (such as basic circuit elements) or larger, custom-designed cells. Instances are dynamically linked to their source cell so that any change you make to a source cell is reflected in each higher-level instance of that cell.

Any option you set in the originating cell will be the default for all of its instances. However, you can override this setting in any of the instances, and the change will affect that instance only.

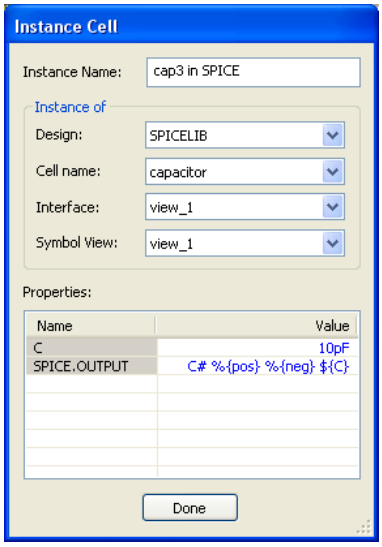
Instancing a Cell (Cell > Instance or "I")

[1] Highlight the cell in the **Libraries** cell list.

[2] Select a cell to instance by:

- using the **Instance** button in the Libraries navigator,
- using the **instance** icon on the toolbar  , or
- using the menu command **Cell > Instance (**shortcut **"I")**

[3] Property values inherited from the symbol can be overridden on a per instance basis by changing their values in the **Instance Cell** dialog before placing the instance. Modified values in the instance cell dialog will persist for all subsequent instances until modified again.



[4] Drag the instance with the mouse and click in the design area to place it.

You will continue to instance the same cell until you either click **Done** in the **Instance Cell** dialog, choose a different cell to instance, select a new tool, or choose the **select** arrow.

Instances can be flipped horizontally or vertically prior to placement by pressing the "**H**" or "**V**" key, and can be rotated prior to placement by pressing the "**R**" key. Properties for each instance can be modified in the **Instance Cell** dialog prior to each placement.

You can change the cell being instanced by selecting a new **Cell name** in the **Instance Cell** drop-down.

Instance Name — Enter a name for the instance. If no name is entered, S-Edit will by default use the *cellname_x*, where x is an integer that increments automatically.

Instance of:

- **Design—** Select a design or library file from the list of those open.
- **Cell name**— Select a cell from the the list of those loaded to the design.
- **Interface**— Select an interface from those defined for the cell.
- **Symbol**— View the list of those open

Properties — You can change the properties of the cell you are about to instance. This will be local override over the symbol default values.

Done — Click **Done** to stop instancing.

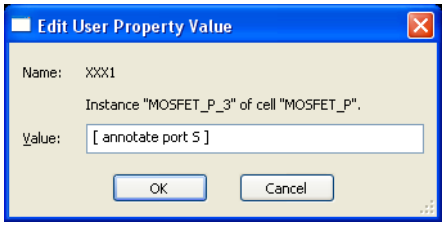
Editing Instance Properties

To add a property, see Adding User Properties on page 89.

Editing Instance Properties from the Work Area

Use **Ctrl + click** to select an instance property directly from the design area. You will see a selection box when it is active in the Properties window. If multiple instances are selected, all of them will be affected by changes made in the Properties window (see Editing with the Properties Window on page 90).

Once a property is selected you can also use **Ctrl + double-click** to open the **Edit User Property Values** dialog, where you can change just the value of the property. When you change the value of a property in a single instance, the change affects only that instance.



Moving Instance Properties

Properties can also be moved in the design area. Use **Ctrl + click** to select a single property then use the middle mouse button to move it. Or, **right-click** on one property (it will not appear to be selected) then use **Ctrl + right-drag** to select multiple properties. Once selected, you can use the middle button to move them.

Selecting All Instances of a Cell

To quickly select all instances of a particular cell, select that cell name in the Libraries list and press the **Find** button. S-Edit will find and select each instance in the active work area.

Editing Properties on Multiple Instances

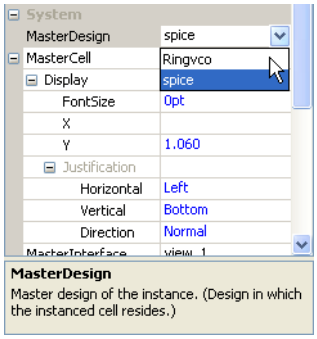
You can change the properties on multiple instances at the same time. To do this, simply select all instances you want to change, then modify the desired values in the Properties window. You may modify any property value in this manner.

To select multiple properties, **right-click** on one property (it will not appear to be selected), then **Ctrl + right-drag** to select multiple properties (selection boxes will be visible at this point).

Changing Instances of a Cell to Instances of a Different Cell

To change all instances of one cell to instances of another cell, select the cells you want to change, then select a new cell from the **Master Cell** property under System properties in the Properties window. If the

new new cell you want to use is in a different library, you should select the library first from the MasterDesign property.



Setting the Visibility of Properties on Instances

Using the Display sub-property, you can also chose to show a property's name and value, its value only, or to hide the property altogether.

Note that property values are only hidden on instances in schematic view; they will always be visible in a symbol view.

Making and Labeling Connections

Connectivity is defined in terms of nets. In the most general sense, a net is created when one or more ports, labels, or wires are connected.

Nets derive their names from the ports or net labels to which they are attached. Connections formed by net labels exist only within a particular cell. Connections formed by ports can extend outside the cell.

Drawing Wires

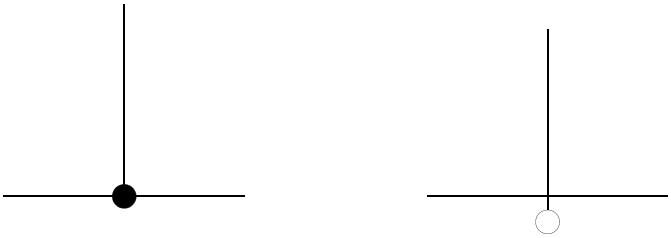
To draw a wire, select the "Wire" drawing button on the toolbar and click the left mouse button to place the first vertex. Subsequent clicks with the left mouse button will place additional vertices. Clicking the right mouse button ends the wire without placing a vertex, and double clicking the left mouse button ends the wire by placing a vertex at the double click location.

Connections Points

A *connection point* is defined as an end of a wire or a bus, an instance of a port, a global or local port, a net label or a net cap.

Connection points that coincide are considered connected and are shown as filled-in circles unless they occur at the edge of an instance. Connections are usually explicit—for example, a wire end connected to a port, or two ports or two wires connecting directly to each other. Note that it is possible to form an implicit connection. For example, if you instance a cell that has two or more symbol ports named A, anything you connect in the instance to any port corresponding to A will also be implicitly connected to other pins corresponding to A.

An open connection point is one that does not coincide with another connection point. Unconnected ports and unconnected wire ends are shown as unfilled circles.



Note that if you click repeatedly on a connection point you will cycle selection through the wire segments that attach to it.

Creating a Connection where Wires Intersect

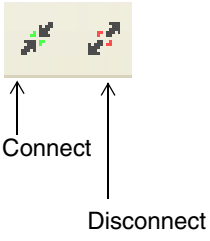
Intersecting wires in S-Edit are not connected unless you explicity add a connection point, using the solder point tool. Note that solder points must coincide with snap grid points.

Rubberbanding

Rubberbanding is the characteristic in S-Edit that preserves connectivity in a schematic view when an object is moved or edited. It is the default behavior. You can detach rubberbanding before moving an object by using **Draw > Force Move** (shortcut key **Alt + M**).

If an object or wire with an open connection point is created, moved or edited, S-Edit will occasionally split a wire to create a new connection. This can occur if pre-existing connection points fall on each other or when a selected wire is split due to the edit.

You can use the **Draw > Electrical > Connect** and **Draw > Electrical > Disconnect** commands to attach or detach wires from pins.

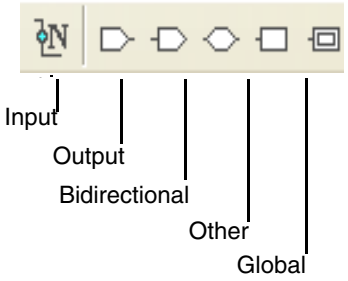


Adding Ports

Ports define the connection points that can be made to a symbol when the symbol is instanced. When instanced, ports do not display their name, but appear as an open *connectivity circle* for connecting a wire. When connected to two or more items, they appear as filled circles; when connected to one item, they disappear. Ports exist in both symbols and schematics, and you can create and modify ports in both views.

Types of Ports

Five types of ports are available: Input, Output, Bidirectional, Other and Global, as shown in the toolbar below.



The first four port types are functionally identical, and S-Edit treats them in the same way. They only differ in appearance for the sake of design readability. Global ports, however, have a different function—they are used to make global nets, which create connectivity throughout a design.

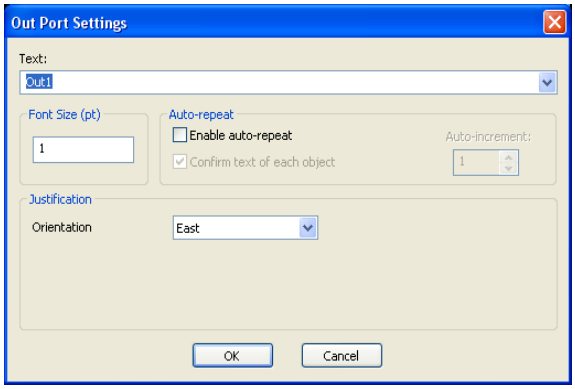
Drawing and Labeling Ports

To draw a port, select one of the port tools, and click in the design area. S-Edit will open the corresponding **Port Settings** dialog.

Note that you can use the **R**, **H** and **V** shortcut keys (for rotate 90°, flip horizontally and flip vertically) to change the orientation of a port before clicking to place it in the work area. This, rather than using the Properties window, is the suggested method.

Use the right mouse button or press the **ESC** key to exit port placement mode.

Draw > Electrical > {port type} Port



Text Enter the port label text in this field.

Font Size Enter the text size.

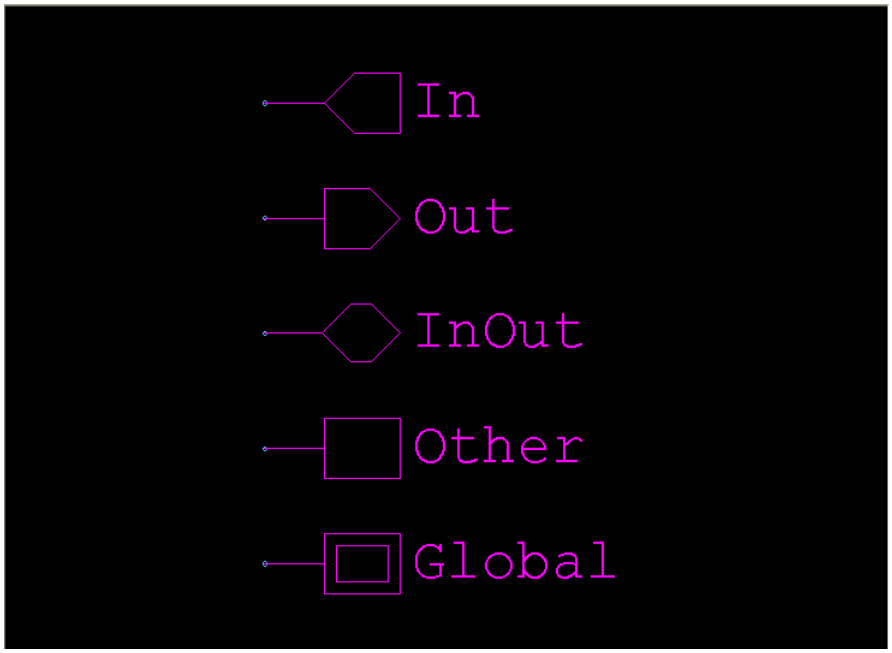
Auto-repeat If **Enable auto-repeat** is not checked, S-Edit opens the **Port Settings** dialog just once, places the port when you click **OK**, and reverts to **Select** mode.

With **Enable auto-repeat** checked, you can place ports repeatedly by clicking in the work area, without having to re-open the port settings dialog each time. S-Edit remains in port mode until you select a different mode or press **ESC**. The following controls may be added:

- **Confirm text of each object**—when checked, re-opens the **Port Settings** dialog each time a port is placed so you can enter changes to the next port placed.
- **Auto-increment**—when **auto-repeat** is active and the last character in the **Text** field is a numeral, S-Edit automatically increments the port label by the value entered here.

Justification Select **North**, **South**, **East** or **West,** where the cardinal directions describe the orientation of the port text with respect to the center of the connection point.

Each port type has its own graphic so it can be easily identified in the work area.



Naming Ports

In most cases, the port names on schematic views of a cell must match the port names on symbol views of that cell. Specifically, a cell is correctly defined when the arrangement of ports is such that:

- For every non-global port on the symbol view, there exists a corresponding port of the same name and type on at least one of the cell's schematic views.
- For every non-global port in a schematic view, there exists a corresponding port of the same name and type on the cell symbol.

Attaching ports or net labels with the same name to differently named objects forms an implicit connection, even if those objects are not directly connected by wire. If you rename a port in one net, you break that implicit connection. Similarly, when a cell contains more than one net with the same name, those nets are connected, even if they are on cells.

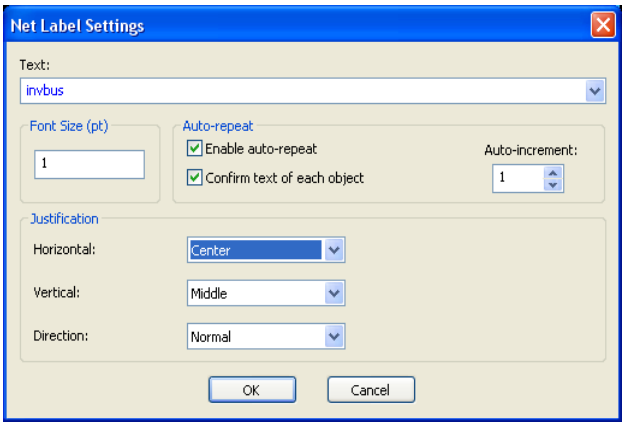
Labeling Nets

You can label nets to make your design more readable and to indicate connections within a cell. Connections formed by net labels exist only within a particular cell. When different cells contain nets with the same name, those nets are generally unconnected, with the important exception of global nets (see Global Nets on page 81).

Example of a net label with West orientation.

Click on the **Net Label** tool  or use **Draw > Electrical > Net Label**. S-Edit will open the **Net Label Settings** dialog.

Note that you can use the **R**, **H** and **V** shortcut keys (for rotate 90°, flip horizontally and flip vertically) to change the orientation of a net label before clicking to place it in the work area.



Text Enter your net label text in this field.

Font Size Enter the text size.

Auto-repeat If **Enable auto-repeat** is not checked, S-Edit opens the **Net Label Settings** dialog just once, places the net label when you click **OK**, and reverts to **Select** mode.

With **Enable auto-repeat** checked you can place net labels repeatedly by clicking in the work area, without having to re-open the net label dialog each time. S-Edit remains in net label mode until you select a different mode or press **ESC**. The following controls may be added:

- **Confirm text of each object**—when checked, re-opens the **Net Label Settings** dialog each time a net label is placed so you can enter changes to the next label placed.
- **Auto-increment**—when **auto-repeat** is active and the last character in the **Text** field is a numeral, S-Edit automatically increments the net label text by the value entered here.

Justification All text orientations are with respect to the origin point of the label, indicated by a cross.

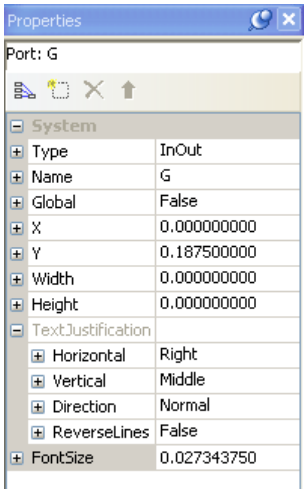
Horizontal: select **Left**, **Center** or **Right** horizontal text alignment.

Vertical: select **Top**, **Middle** or **Bottom** vertical text alignment.

Direction: select **Normal**, **Upside down**, **Down** or **Up** for the text direction.

Editing Port Properties

Ports, net labels, and comment labels may be edited in the property browser. Simply select the desired objects, and change their values in the property browser. Sub-property values such as text size or justification may also be changed. Multiple objects may also be edited at the same time.



Buses, Bundles and Arrays

S-Edit supports buses, arrays and net bundles. A *net* is the fundamental single unit of connection. A *bus* is a set of connections with the same name, plus a numerical identifier and increment which is the syntax that defines it as such. Similarly, an *array* is a set of ports within an instance with the same name plus a numerical identifier and increment that defines related multiple connection points.

A *bundle* is collection of both nets and buses, where the nets do not need to share a name with the bus. A *port bundle* is a port defined to accept more than one wire, defined as such by the same numerical identifier and increment syntax that defines a bus.

Creating a Bus

You create a bus with the net label tool by using a special naming syntax that specifies how many nets are in the bus, and how their number increments. The step increment is one by default and can be omitted.

For example, crosstown<0:7:2> creates a bus eight bits wide, with nets crosstown<0>, crosstown<2>, crosstown<4>, ... crosstown<6>.

You can create two-dimensional buses by naming a wire *busname* <n1:n2:stepa><n3:n4:stepb>. As with arrays (see below), the second range increments first.

Creating an Array

You create an array by applying array syntax when you name an instance. So, an instance named *array_name*<n1:n2:*step*> creates an array of instances named array_name<n>, where n starts at n1, ends at n2, and increments by the *step* value. The step increment is one by default and can be omitted.

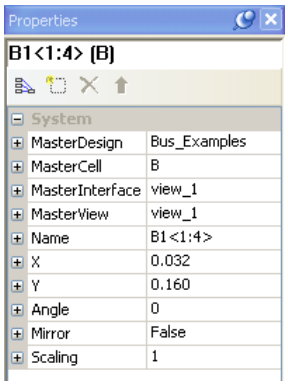
Two dimensional arrays may be created by naming an instance U<n1:n2:step1><n3:n4:step2>, where the second range increments first.

For example, the instance name U<0:7><0:3> creates an array of instances named:

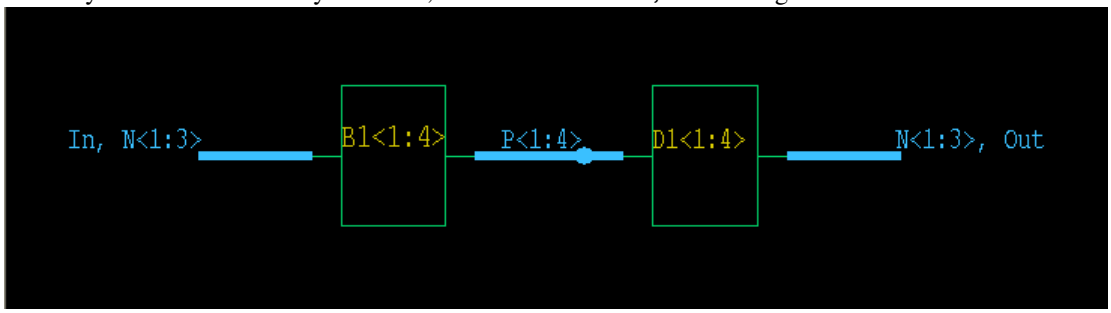
```
U<0><0>, U<0><1>, U<0><2>, U<0><3>,
U<1><0>, U<1><1>, U<1><2>,  U<1><3>,
U<2><0>, U<2><1>, U<2><2>,  U<2><3>,
...
U<7><0>, U<7><1>, U<7><2>,  U<7><3>
```

Example 1

In the following example, we have created an array of four cells by giving an instance of cell B the name **B1<1:4>**.

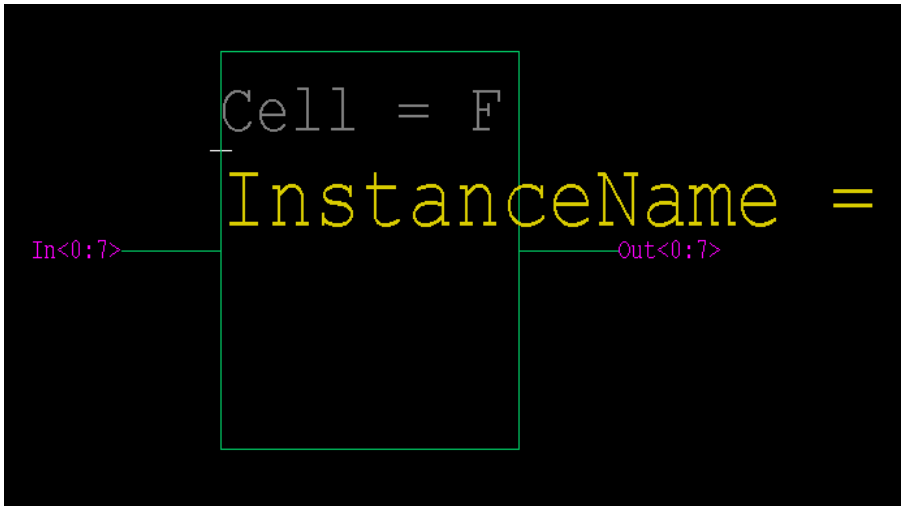


The bundle **In, N<1:3>** connects four wires **In, N<1>, N<2>, and N<3>** to array **B1,** and bus **P<1:4>** similarly connects to the array **D1<1:4>**, with bundle **N<1:3>, Out** exiting.

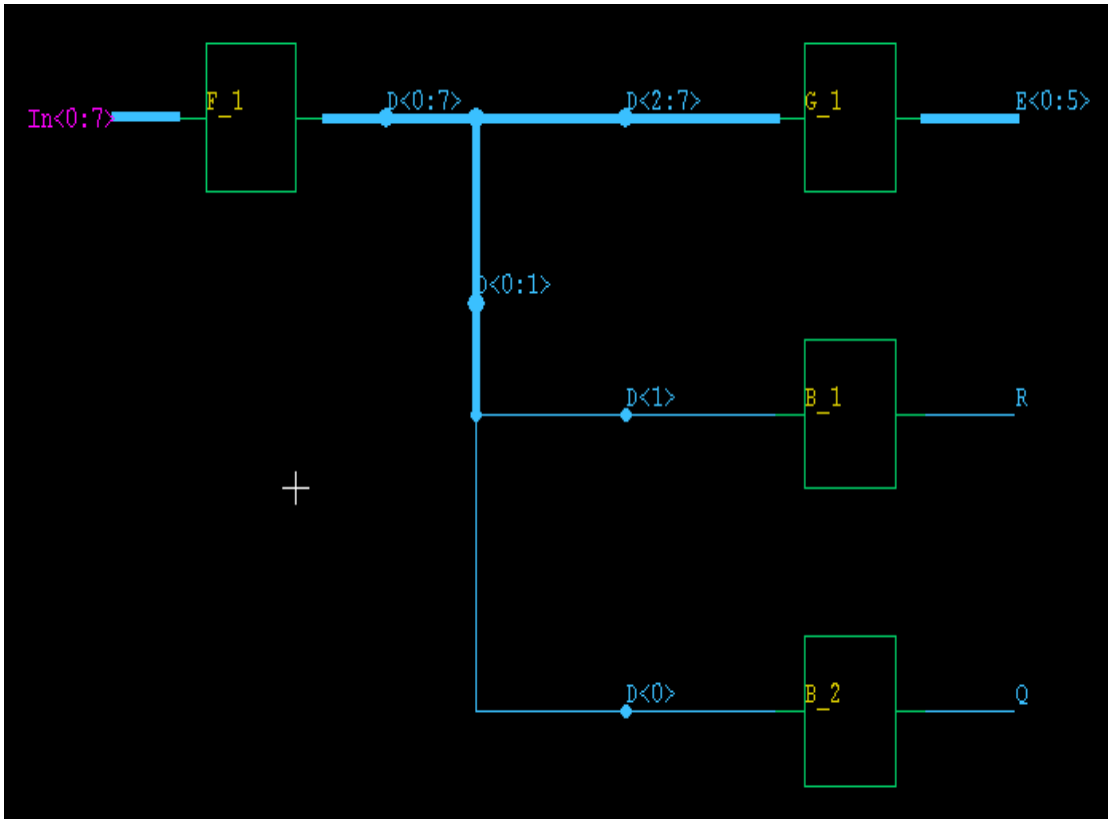


Example 2

In this example, cell **F** contains port bundles **IN<0:7>** and **Out <0:7>** so that it can connect to eight wires.



Buses **In<0:7>** and **D<0:7>** form an eight bit wide bus, respectively, in and out of instance **F_1**, which splits so that wires **D<2>**, **D<3>**, **D<4>**, **D<5>**, **D<6>**, and **D<7>** go into instance **G_1** with five-bit port bundles in and out to accommodate **D<2:7>** and **E<0:5>.** The single wire **D<1>** goes into instance **B_1** and out as net **R**, and single wire **D<0>** goes into instance **B_2** and out as net **Q**.



Global Nets

Global nets simplify the drawing and maintenance of electrical schematics. When a net is global, you can connect or disconnect it throughout a design without drawing or deleting wires. Global nets are especially useful for power, ground, clock, reset, and other circuitwide nets that require routing throughout a design.

Like standard nets, global nets connect across all of the schematic views of a cell. Unlike standard nets (which connect to nets outside the cell only through ports), global nets automatically connect between all cells in a design (subject to certain scoping rules; see Naming Global Nets on page 82). Any changes made to a global net's name or state propagate throughout the design.

There are two ways to create a global net. You can add a global port to a schematic page, or you can instance a "global symbol"—that is, a symbol which has a global port or has one on it's corresponding schematic page.

Global Ports

Global ports indicate connection points for global nets. Unlike regular ports, the names of global ports are significant outside of the cell where they are placed. The name of a global port is the default name of the global net associated with it. For example, if you want to create a global net with the default name Gnd, name the global port Gnd.

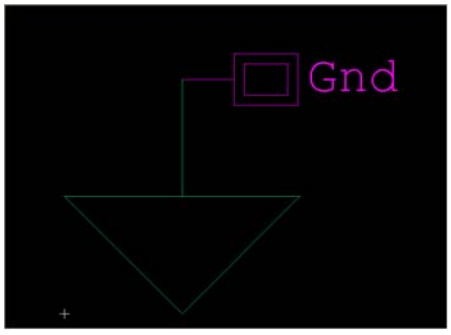
To place a global port, use the **Global Port** tool (See Adding Ports on page 74.)

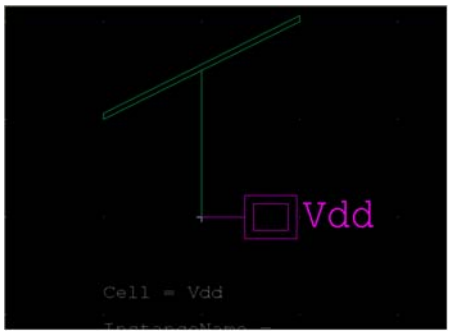
Global Symbols

Global symbols are special cells that function as wireless connectors. They are defined as such when they contain one or more global ports.

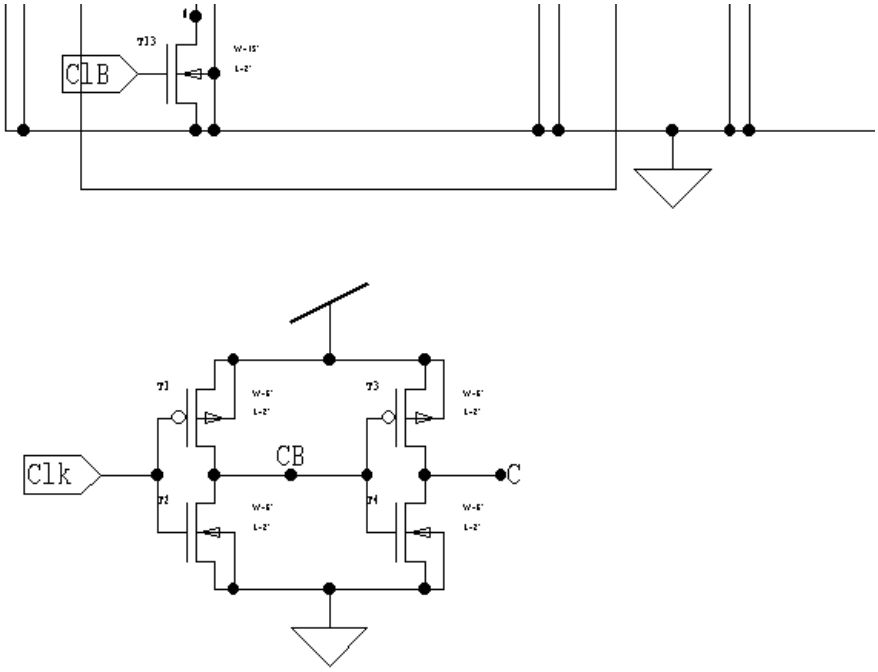
When you attach a global symbol to a net, you connect that net to all other nets on every view and cell in the design file that are attached to a like global symbol. Such nets then become global nets. Conversely, a net ceases to be global when all global symbols are detached from it. You can add any number of new global symbols to a design.

Sample files shipped with S-Edit contain symbols for **Gnd** and **Vdd** as shown below.





In the figure below, two Gnd symbols are attached to two different wires in a cell. Since Gnd is a global symbol, the two wires are thus connected to each other. In addition, any wires in any other cell which has the Gnd symbol attached is also connected to these nets.



Naming Global Nets

The default name of a global net is the name of the global port in the symbol that defines the global net. An uncapped global net can never be renamed, and always uses the default net name; if you place a port or net label on an uncapped global net, it will automatically acquire the name of the global net.

You can rename a capped global net using a port or net label. The new name affects the current cell only. When you export a netlist, however, the new name will affect any cells containing uncapped global nets which are instanced in the current cell. By renaming capped global nets, you can use the same global symbol to represent different nets with different names.

propagated A global port or an instance containing one appears on the cell, with no net caps on the associated global net. The net propagates up and down the cell hierarchy.

capped A global port or an instance containing one appears on the cell, with a net cap named so it is associated with the global net. The net does not propagate up the cell hierarchy.

hidden A global port or an instance containing one does not appear on this cell, but global net propagation “passes through” this level of the hierarchy because the global port appears, uncapped, in one of the cells instanced within the current cell. In SPICE output, these nets will actually appear in the cell as “placeholder” nets. You can connect to a hidden net by placing a net label (or port) in the cell with the same name as the hidden net (or by instancing the global symbol).

When global symbols collide, S-Edit will arbitrarily choose one of their names as the new name for the net. There is no guarantee that this net name will be stable; it may change when further edits are performed.

If a global symbol collision causes two separate global nets to be connected, S-Edit will use the common new name to designate the net and all nets connected to all instances of all colliding symbols. Until the collision is resolved, all of these nets continue to be electrically connected. Avoid such situations wherever possible.

One way to avoid collisions between global symbols on the same net is to use a voltage source cell to set the voltage between the nets. Let a property of the cell be the voltage between the nets. Then edit the property to set any desired voltage between the nets, even 0 volts. You can find voltage source cells in the S-Edit sample libraries.

Effective Design with Global Nets

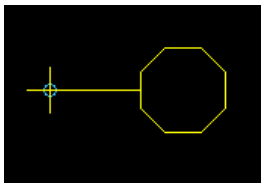
Here are some tips for the effective use of global nets in your design.

- Avoid using several similar global symbols, such as **Vdd1**, **Vdd2**, and so on, for the purpose of separating similar global nets. Instead, use a single Vdd global symbol everywhere. Then place net caps at appropriate places to separate the different nets from each other. Finally, simply use net labels at the capped level to rename the nets. By using only one global port, you can instantly change the scope of these global nets without having to search for and replace different ports.
- Avoid placing global ports or net caps on a net that is connected via a port to higher levels of the design. The presence of a global port on such a net will automatically make its higher-level connections global, regardless of the designer's intention. Moreover, net caps will not prevent such a net from propagating. Be careful when naming different grounds.
- T-Spice considers all nets named Gnd, GND, gnd, and 0 to be connected.

Capping Global Nets

Global nets automatically propagate up the design hierarchy. For example, if the cell in the previous example is instanced in another cell, Gnd will exist in that upper cell.

Net caps halt, or “cap,” the propagation of a global net from subcircuit definitions that are lower in the hierarchy.



Graphic for a net cap.

You place a net cap with the **Net Cap** tool.

When a net cap is placed in a cell, the global net will not propagate to any cell that instances that cell. The net’s scope is limited to the current cell and any cells instanced on its schematic views. Any global nets with the same global symbol that exist outside the scope of the capped net may have the same name as the capped net, but are unconnected.

One important exception to this rule occurs when you export SPICE files from S-Edit for use in T-Spice and some other SPICE simulators, which treat all ground nets—in T-Spice, nets with any of the names Gnd, GND, gnd, or 0—as connected to 0.0 volts, regardless of S-Edit's scoping rules.

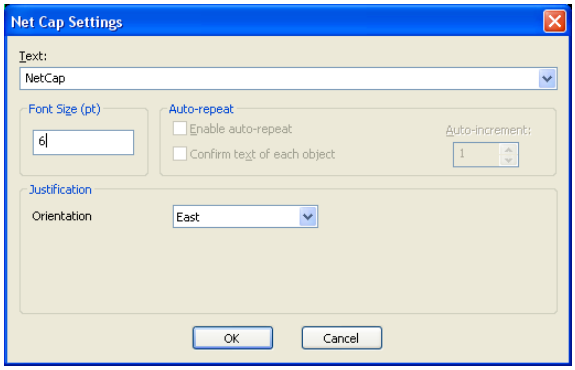
Note that the net cap does not have to be attached to a net—it can be placed anywhere in the work area. However, the name is critical as it defines the net cap.

Naming Net Caps

A net cap must be given the name of the net you wish to cap.

Click on the **Net Cap**  toolbar button or use **Draw > Electrical > Net > Cap** and place the net cap anywhere on the schematic page where you want the cap to occur.

If a port is connected to a global net, the net propagates to the higher levels through the cell port connection, just as a regular net does. This happens even if the global net is capped. This is the only way a capped net can propagate to a higher cell.



Text — Enter the name of the net cap in this field.

Font Size — Enter the text size in points.

(Enable auto-repeat is not functional for this operation.)

Justification — Select **North**, **South**, **East** or **West,** where the cardinal directions describe the orientation of the text with respect to the center of the connection point of the cap.

Checking a Design for Errors

The S-Edit design checker searches for many of the mistakes commonly made during the schematic creation process. These are categorized into *errors*, which will prevent the proper connectivity from being formed, and *warnings*, which do not prevent S-Edit from extracting connectivity but are likely to be unintended.

Some of the common items the design checker will check for include:

- Dangling wires—wires with unconnected ends.
- Dangling ports—ports on instances with no connection.
- Instances with no name or a non-unique name.
- Nets with at most one output port reference connected to them.
- Ports with the same name having the same type.

Schematic Checks

Use **Tools > Design Checks** or click on the  icon to run the design checker.

Checks on Names

Design Check checks the names of these objects based on the validation constraints in **Setup > Technology > Validation**. Each violation triggers a warning.

- Cell names
- View names
- Instance names
- Port text
- Net labels

Checks on Instances

Design Check examines instances to determine whether the following conditions are met and issues either an error or a warning if not.

- All instances must refer to a master that can be resolved (error).
- All instances must have a unique name (error).
- There cannot be any cyclic dependencies (error).
- All instances must have a name (warning).
- All properties can be evaluated—i.e. correct syntax, legal references, etc. (warning).
- Dangling port references (those that are not connected) do not exist (warning).

Checks on Nets

Design Check examines nets to determine whether the following conditions are met and issues either an error or a warning if not. Note that although it will not result in either an error or a warning, we recommended that you do not give a net the same base name as a bus. For example, a net named N and a bus named N<0:7> could cause confusion.

- Nets ripped from a netbundle must actually exist in the netbundle (error).
- Bus dimensions are compatible with rippers, portbundles, subscripts, etc. For example, you're not trying to rip a 5th wire from a 4-wire bus (error).
- Nets have at most one OUTPUT portref connected to them (error).
- Nets have at least one portref that is NOT of type INPUT connected to them, unless they are connected to an interface port of type INPUT (error).

- Dangling nets, i.e. those with a wire end that is not connected, do not exist (error).

Checks on Interface Integrity

Design Check examines the interface to determine whether the following conditions are met and issues either an error or a warning if not.

- Ports with the same name must have the same type (error).
- All instance portrefs must reference an existing port on the interface (warning).
- Do not allow ports that exist on the interface but do not exist on the symbol (warning).

6 Creating a Symbol

The process of schematic capture first involves the creation of symbols. Symbols are a pictorial representation of an electrical component together with a definition of the electrical connections that can be made to that component. Symbols may also contain properties that define the electrical characteristics of the component.

In many design situations, a library of symbols already exists, usually of basic standardized components which a designer can use to create their schematic.

Elements of a Symbol View

A symbol view contains the following elements:

Symbol Graphics

Symbol graphics comprise the graphical image of the symbol. This is the image that is seen when the symbol is instanced. Symbol graphics can be boxes, polygons, paths, or circles.

Labels

Text labels can be added to a symbol, and are visible when the symbol is instanced.

Ports

Ports define the connection points that can be made to the symbol when the symbol is instanced. When instanced, ports do not show their text, but appear as an open circle connection point for a wire.

Properties

Properties are name-value pairs that are usually used to describe some characteristic of a device, such as a transistor length, width, or Source/Drain areas and perimeters. Properties can also be used for other purposes, such as controlling the SPICE statements written for a device. Properties on a symbol provide the default values when that symbol is instanced, but may be overridden on an instance-by-instance basis.

Creating and Updating Symbols Automatically

S-Edit can automatically create a symbol from the ports on a schematic view using **Cell > Update Symbol**.

If the symbol view is empty, **Update Symbol** creates simple graphics and places ports corresponding to those on the schematic, with input ports on the left and output ports on the right. It will also place text labels corresponding to each port.

If a symbol already exists with graphics or ports, **Update Symbol** will add the new ports from the schematic to the symbol view, but will not modify the graphics or remove any existing ports.

Creating a Symbol

Follow these steps to create a new symbol:

[1] **Create a View**—Create a new cell with a new symbol view, or create a new symbol view of an existing cell using **Cell > New View**. Give the symbol a **View name**, and select the interface it is to be associated with. (See Creating a New View (Cell > New View or "N") on page 41.)

[2] **Draw and label graphics**—Draw a graphical representation of the symbol using the object drawing tools. You can add text with the Label drawing tool. The graphical representation has no electrical meaning, but provides a recognizable way to identify instances of the cell. (See Drawing Tools for Annotation Graphics on page 56.)

[3] **Add ports**— Ports define the connection points that can be made to a symbol when the symbol is instanced. They can be of type **In**, **Out**, **In/Out**, **Other** or **Global**. (See Adding Ports on page 74.)

[4] **Add Properties**—If needed, add properties to the symbol. Properties on a symbol provide the default values when that symbol is instanced, but may be overridden on a per instance basis. (See Symbol Properties, below.)

Symbol Properties

Properties are used to store parametric descriptions of design elements. Properties can characterize a cell's physical parameters, such as length, width, and perimeter; its nonphysical parameters, such as device type and comments, and its output strings, which specify how S-Edit will write a cell to a netlist.

Properties are name-value pairs. Properties can be text or a numerical value, or they can be expressions that require evaluation, in particular during instancing or exporting.

System Properties and User Properties

User properties such as Drain Area (AD), Source Area (AS), Length (L) etc. are shown in the **User** area of the Properties window. Properties such as the design or library from which the symbol comes or the cell name are shown in the **System** area of the Properties window. In general, the system properties should not be edited.

S-Edit uses property information in two ways. When you instance a cell with properties, S-Edit interprets the properties (as long as they are not hidden) and displays their interpreted values in the instancing cell. If a property has a numeric value, S-Edit displays the value of the number. If a property has a string value, S-Edit displays literal text and the interpreted value of expressions, if any are present. When you export a netlist, S-Edit writes out the values of output properties appropriate to the netlist format.

You can use SPICE output properties to include simulation commands in the netlist. For more information on output properties, see Exporting SPICE Files on page 102.

Default Properties

The properties on a symbol are default properties that are used when the symbol is instanced. When you change a property in the symbol view of a cell, that property will change in all the instances of that cell.

Properties for a symbol are visible on the symbol itself as well as in the Properties window. The Properties window will contain different fields depending on the type of object selected.

You can add, delete, or edit properties at any time, and you can override properties assigned in the original cell with properties assigned to the instance. You can also show or hide a property's name or value, for most but not all properties.

Displaying Properties

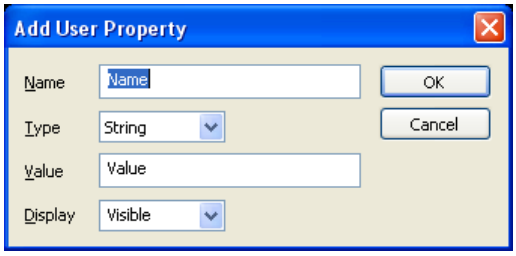
When a symbol is instanced, you can choose to make its properties visible, hidden, or "value only." The default colors for hidden properties and visible or value only properties are slightly different shades of gray so they can be distinguished in the work area.

Adding User Properties

You can add user-defined properties with the **Property** icon, the **Add Property** button in the Properties window, or using **Draw > Electrical > Property**.

User properties are used, for example, to count cell instances, to reference ports or other properties, or to format property outputs. S-Edit interprets user properties in instances (when shown in schematic mode) or during export to a netlist. S-Edit parses all other text without expansion.

S-Edit opens the **Add User Property** dialog when you click in the work area after invoking **Property**.

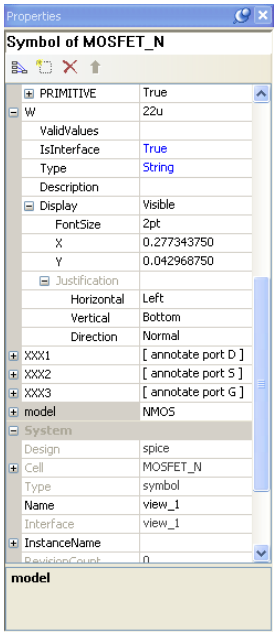


Evaluated Properties

A property can be an explicit value or an expression. For example, in the MOSFET_P example below, the value of **L** is 2u and the value of **W** is 22u. Properties **AD**, **AS**, **PD** and **PS** are expressions which reference the values of other properties using the "$."

When used on a symbol, the "$" references a value on the symbol, and when used on an instance, the "$" references a value on the same instance. In this symbol, the Drain area (**AD**) is 3u times the gate

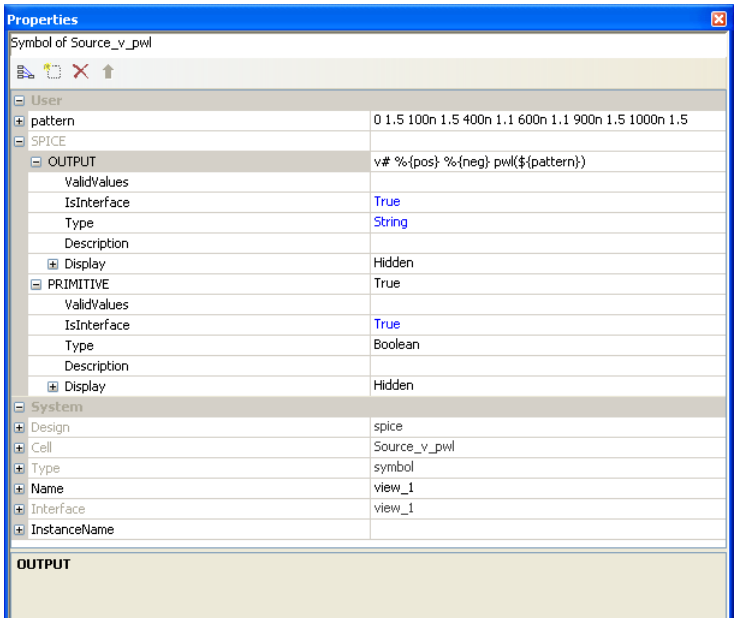
Width (**W**), which we have expressed in the property browser as **AD** = $W*3u. This will be evaluated to yield **AD** = 22u*3u = 66p. See the chapter Evaluated Properties on page 93 as well.



Editing with the Properties Window

You can use the Properties window to edit any attribute of a user property (e.g. visibility, font size, or text positioning) on any number of properties that are selected in the active view.

Use **Ctrl + click** to select a property and make it active in the Properties window, or **right-click** on one property (it will not appear to be selected), then **Ctrl + right-drag** to select multiple properties (selection boxes will be visible at this point). Any edits made in the Properties window affect all the selected objects.



Note that edits made in the Properties window are unrestricted and are not validated by S-Edit. For example, it is possible to draw a 90° line and then convert it to an all-angle line using the value field of the related property. S-Edit does not give a warning if you make such a change.

Properties Window Toolbar

Collapse all—collapses the display to property name level.

Add property—opens a dialog so you can add a user property.

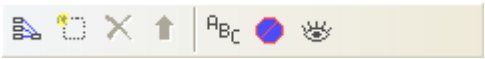
Delete property—deletes the highlighted property. Use **undo** to restore.

Reset—resets a property value on an instance to that of its source cell.

Sort by name—sorts properties ialphabetically separately by category.

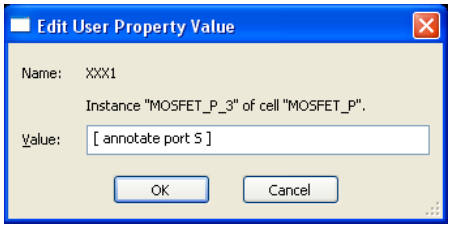
Don't show inherited—hides properties inherited from the symbol primitive.

Show visible only—hides property values that are set to hidden in the design.



Editing Property Values from the Work Area

You can edit a property from the work area by using **Ctrl + click** to select it and then **Ctrl + double-click** to open the **Edit User Property Value** dialog. Only one object may be edited at a time with this dialog.



Port Placement

S-Edit is more flexible than many other schematic editors with respect to port placement. S-Edit allows you to place a port at any location on the symbol, whereas other schematic editors require that you place the port on the symbol boundary. If you plan to export your design to another schematic editor, you will probably achieve better results by following this convention and placing your ports on the symbol boundary.

When you instance a cell, S-Edit translates the ports on the symbol view into ports on the cell instance. Ports thus provide connection points between lower-level cells and their instances. When you connect

objects to a port on a higher-level instance, you also connect them to any object connected to the corresponding port on the schematic of the originating cell.

For example, when you connect an object to the input port **In** of an instancing inverter, (corresponding to a port named **In** on the symbol view of the inverter cell), you have connected it to the net named **In** on the schematic of the instanced inverter cell.

7 Evaluated Properties

S-Edit supports expressions as property values; in fact, every property value is implicitly an expression.

Expressions as Property Values

A property can be an explicit value, or an expression which references the values of other properties.

S-Edit expressions support standard mathematical operators -, *, /, **, as well as standard functions such as sin(), cos(), etc.

The prefixes %, $, ?, and @ on a property value are used to reference other properties. Braces are optional, and should be used when the property name has a space, or when abutting something immediately after property name. Expressions can also reference the vales of other properties using the following prefix operators.

Reference	*Interpretation*
%port or %{port}	References the name of the node connected by the terminal T. In the "in context" view, this name is the hierarchical name of the net.
$P or ${P}	Refers explicitly to another property on the same instance, or in symbol view to another property on the symbol. For example, in the expression AD=$W*6u, the $W refers to the value of a property W on the same device. The reference for a property is first looked for as a User property, and then as a System property. If $P does not exist as either a user or a system sibling property, $P is looked for as a TCL variable. This option corresponds to the Cadence **iPar()** function.
?P or ?{P}	References the parent cell's property, but only looks up one level of the hierarchy. As usual, overridden properties on the instance of the parent have higher priority than the default value on the symbol. In particular, a property value **?P** on an instance looks for the property P first on the instance of the parent cell, then on the schematic page containing the instance, then on the symbol of that schematic, and then for a TCL variable. As with the **$** reference, at each level, the property is first looked for as a User property, and then as a System property.

(continued)

Reference	*Interpretation*
?P or ?{P} (continued)	For example, consider a symbol of cell BOT instanced in a schematic of cell MID. S-Edit will resolve a property with value **?P** on an instance of BOT by looking for a property P as follows: 1 Look for value of property P on the instance of cell MID. This assumes you have navigated in context through an instance of MID and are now looking at a symbol of BOT, or, similarly, are exporting SPICE from a higher level schematic. (The value on an instance of MID is first looked for as an override on the particular instance of MID, and then looked for on the symbol of MID.) 2 Look for a value of property P on the schematic of MID. 3 Look for a value of property P on the symbol of cell MID. (This is not needed when BOT is looked at in a particular context of MID, but provides a default value when MID is opened without context, or as a toplevel cell) 4 Look for a value of property P defined as a TCl variable. The **?** can be used to iteratively look up the hierarchy. In the above example, cell MID can be instanced in a schematic of cell TOP. The value of property P on the instance of cell MID can be ?Q, and Q could have its value defined on an instance of TOP. This option corresponds to the Cadence **pPar()** function.
@P or @{P}	References the highest-level definition of P. If we have a cell TOP which contains an instance of MID, which contains an instance of BOT, then the priority order of a property P "inside" BOT is then (high to low): global, MID_{INST}.P, MID_{SYM}.P, BOT_{INST}.P, BOT_{SYM}.P.

TCL Commands in Expressions

In the expression for a property value, a string may be passed to the TCL interpreter for evaluation. Any substring in a property value contained in brackets **[ ]** is passed to the TCL interpreter; if the result is successful, it is included verbatim, otherwise the TCL error string is included.

For example, consider that the following TCL process is defined in S-Edit:

```
proc size { val } {
   if {[string compare $val small ]== 0} {return 5}
   if {[string compare $val large ]== 0} {return 20}
   return 10
}
```

An instance (or a symbol) of a cell could have a property A = [size $B]. If there is a property B = small on that instance, then property A after substitution of B becomes A = [size small], and after TCL evaluation becomes A = 5. Similarly, one could have A = [size ?B] or A = size @B].

Built in TCL functions

S-Edit makes use of some of the built-in TCL functions to evaluate expressions.

Selective Evaluation: se

S-Edit uses the built-in TCL command "se" to selectively evaluate properties and format the result depending on whether the property is defined or not.

The syntax of the command is:

`se` ***property true-clause { false-clause }***

If the property exists, the true-clause is returned; otherwise, the (optional) false-clause is returned. Because property substitution precedes TCL evaluation in property evaluation, the se function is needed to identify the case where a nonexistent property returns an empty string.

For example:

```
[se $L {L=$L} {L=2u}]
```

will either evaluate to L=2u or L=<the value of the property L>.

Annotate Port: [annotate port T]

You can use the built in TCL command "annotate port" to view display certain values of interest on the ports of an instance. These include:

- Port name
- Net name
- DC Voltage
- DC Current
- DC Charge

The first two of these are always available to display, the next three depend on values being present from a DC simulation run. To place a property near the desired port use:

```
Z = [annotate port T]
```

The key elements of these properties are i) only the property value is important, the property name can be anything, and, ii) the property value contains the string **annotate port *portname*** in square brackets, where *portname* is the name of a port on the symbol. In the preceding example, the square brackets indicate that "annotate port T" is a TCL function being evaluated.

You can set which of the annotate property values to display in the work area by selecting from the drop-down menu under the **Display Evaluated Properties** button  on the Spice Simulation toolbar.

Example:

Place the following three properties on a symbol of a MOSFET P, near the corresponding source, drain, and gate:

```
Z.1 = [annotate port S]
Z.2 = [annotate port D]
Z.3 = [annotate port G]
```

Viewing Property Values In Context

You can use the **Push into Context**  button when an instance is selected to open a specific instance of a cell. Depending on the object and type of analysis, when you push to deeper levels of the design hierarchy you can see, for example, small-signal parameter values, property values derived from expressions, or operating-point voltages.

While thus editing "in context," you can only select or edit objects contained in the instance. However, you can continue to push down to lower levels of instances within an instance until you reach a SPICE primitive.

Use the **Pop Context** icon  to "pop" out of the last instance you pushed into until you return to the top level of the cell hierarchy.

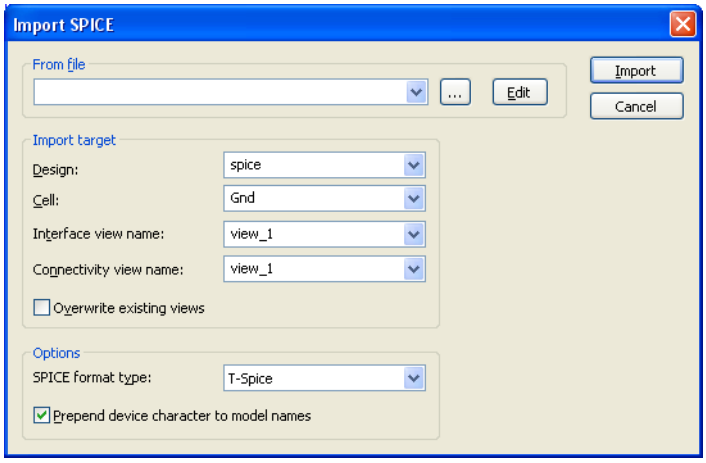
8 Importing and Exporting Netlists and Schematics

Importing a Design

You can import EDIF and SPICE files into S-Edit, and export EDIF, SPICE, Verilog and VHDL files from S-Edit.

Importing SPICE Files

You must have a design open before you can import. The import operation creates a complete project directory for the design.



From file	Specifies the SPICE file to import.
Import target	Specifies the **design**, **cell**, **interface view**, and **connectivity view** into which the SPICE file will be imported.
Overwrite existing views	If cells of the same name exist in the SPICE file and the design it is being imported to, a check in this box causes the SPICE file to overwrite the views in the existing cell.
SPICE format type	Choose from **T-Spice**, **HSPICE**, **PSpice** and **CDL**.
Prepend device character to model names	When checked, SPICE device characters will be prepended to model names in the netlist.

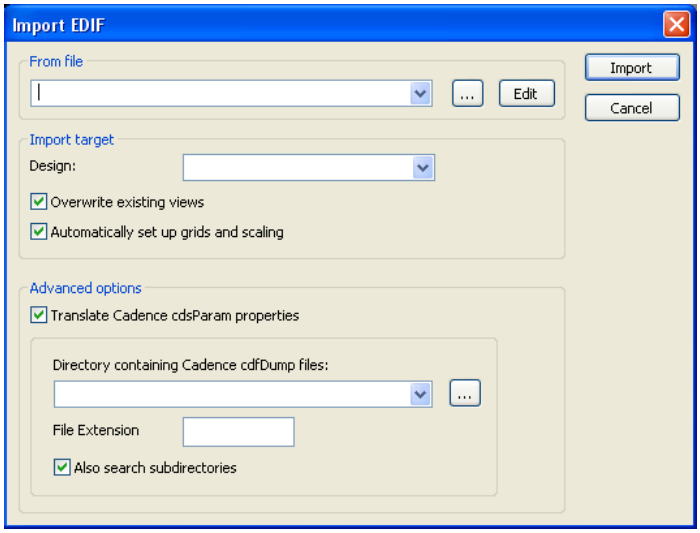
Importing EDIF Files

When importing EDIF from third party vendors such as Cadence (Virtuoso) and Mentor Graphics (ViewDraw), S-Edit will automatically translate properties and other entities in the EDIF file to be

compatible with S-Edit as described below. Note that Cadence applications usually export some properties to files other than the EDIF file. When this is the case, you must prompt S-Edit to search for those files and place limits on the directories that will be searched.

You must have a design open before you can import. The import operation creates a complete project directory for the design.

Note: Note that standard cell place and route (SPR) in L-Edit requires either a flattened EDIF netlist or one with only one level of hierarchy.



From file Specifies the EDIF file to import.

Import Target Specifies the design into which the EDIF file will be imported. If a library exists in the EDIF file with the same name as the design or one of the libraries of the design, then the contents of that EDIF library will be imported into the design or library of the same name. Libraries in the EDIF file that do not match the design name will be created as new libraries of the selected design.

Overwrite existing views If cells of the same name exist in the EDIF file and the design it is being imported to, a check in this box causes the EDIF file to overwrite the views in the existing cell.

Automatically set up grids and scaling Causes S-Edit to analyze the contents of the EDIF file for pin spacing and calculate the best grid spacing and scaling.

Translate Cadence cdsParam properties

Cadence EDIF export places the **cdsParam s**ymbol properties in a separate file from the EDIF files called a **cdfDump** file.

When this box is checked, S-Edit searches the directory specified for **cdfDump** files of the **File Extension** specified, translates the **cdsParam** properties and creates SPICE.OUTPUT when an EDIF file is imported from Cadence. Checking **Also search subdirectories** will do just that for the directory you have entered.

The imported properties will be displayed in the standard S-Edit Properties window.

Directory containing Cadence cdfDump files

Specifies the location of the Cadence ATT file to be used for translating **cdsParam** properties.

This field is only used for translation of Cadence cdsParam labels and to create SPICE.OUTPUT properties, and may be left blank for EDIF files from vendors other than Cadence, or if a Cadence database is not available. See EDIF Translations for Composer, below.)

An example of the cdsParam file for a NMOS is shown below.

```
/*****************************************************/
 LIBRARY = "tsmc18rf"
 CELL    = "pmos2v"
/*****************************************************/
let( ( libId cellId cdfId )
    unless( cellId = ddGetObj( LIBRARY CELL )
        error( "Could not get cell %s." CELL )
    )
    when( cdfId = cdfGetBaseCellCDF( cellId )
        cdfDeleteCDF( cdfId )
    )
    cdfId  = cdfCreateBaseCellCDF( cellId )

;;; Parameters

    cdfCreateParam( cdfId

        ?name           "model"

        ?prompt         "Model name"

    ?defValue     "pch"

    ?type        "string"
    ?editable     "nil"
    ?parseAsCEL    "yes"
  )
  cdfCreateParam( cdfId
    ?name         "macro"
    ?prompt       "Subcircuit name"
    ?defValue     "iPar(\"model\")"
    ?type        "string"
    ?display      "nil"
    ?parseAsCEL    "yes"
```

SPICE.OUTPUT Property for Composer

Properties in S-Edit are created from the namePrefix, termOrder, instParameters, and otherParameters properties referenced by Cadence CDF. Given the following properties in the ATT database;

```
namePrefix "M"
termOrder (D G S)
instParameters (m w l ad as pd ps)
otherParameters (bn)
```

the SPICE.OUTPUT property is created as follows:

```
SPICE.OUTPUT = M${INSTANCE} %D %G %S bn m=!m w=!w l=!l ad=!ad as=!as pd=!pd
   ps=!p
```

EDIF Translations for Composer

These translations are performed on all symbols and instances. Electrical ports with names ending with "!" are made global.

cdsTerm(name) For every cdsTerm label, a property with name "~cds.NNN" is created, with value "[annotate port name]". Here NNN is a unique integer in the symbol scope. The newly created property is placed at the same location as the original label, with the same orientation and justification.

cdsName For every cdsName label, set the location and text size of the system property "Name" to the values of this label.

cdsParam(n) If **Translate Cadence cdsParam properties** is checked, and a Cadence database directory containing ATT files is provided, S-Edit properties are created from the parameter name and value that "**n**" references in the appropriate ATT file in the Cadence database.

NLP label Labels containing "**[@**" are converted to properties that reference the corresponding S-Edit properties.

Property values using the Cadence formats pPar, iPar and atPar that refer to other properties are translated to the S-Edit reference format as follows:

pPar("name") ?name

iPar("name") $name

atPar("name") @name

EDIF Translations for ViewDraw

When S-Edit imports EDIF from ViewDraw it performs these translation:

[1] All instances of a cell called 'SPLITTER' are removed, and cell SPLITTER itself is removed. This is a ripper cell that S-Edit does not need.

[2] Buses and bus components are renamed to adhere to S-Edit syntax. For example, D<0:7> for a bus, D<1> for a single bit from the bus.

[3] The leading '@' is removed from property names.

[4] The leading '$' in instance names is replaced with '_'.

[5] A SPICE.OUTPUT property is created from the ViewDraw properties PREFIX, PINORDER, and ORDER.

[6] A SPICE.PRIMITIVE property is created on a symbol and its value is set to 'true' only if there is a property 'PREFIX' on the symbol.

[7] The 'IsInterface' attribute is set to false for 'PINORDER', 'FLATORDER', 'LEVEL' and 'PARNAME'

SPICE.OUTPUT Property for ViewDraw

Properties ORDER, PINORDER and PREFIX in ViewDraw are translated to the S- Edit SPICE.OUTPUT property. Typical properties and values in ViewDraw are:

```
MODEL = PCH
ORDER = MODEL$ L$ W$ AD= AS= PD= PS=
PINORDER = D G S B
PREFIX = M
```

These are translated to the S- Edit SPICE.OUTPUT format, as follows:

```
SPICE.OUTPUT = M${Name} %{D} %{G} %{S} %{B}  $MODEL $L $W [se $AD
   {AD='$AD'}] [se $AS {AS='$AS'}] [se $PD {PD='$PD'}] [se $PS {PS='$PS'}]
```

Tokens in the ViewDraw ORDER property are translated to the SPICE output property as follows:

```
name$     translates to  $name
name=    translates to [se $name {name='$name'}]
```

Tips for Creating a New Design from an EDIF File

If you wish to create a new design from an EDIF file, you should create a design with the same name as the top level or root library in the EDIF file, and then import the EDIF file into that design. The root library is often written near the bottom of the EDIF file with a “design ROOT” entity as:

```
(design ROOT
            (cellRef rootcell
                        (libraryRef rootlibrary)))
```

Where *rootlibrary* is the name of the root library. The root library is the one nearest the bottom of the EDIF file, so you can also identify the root library by locating the entity “library *libraryname*” that is nearest the bottom of the file.

Another way to create a new design from an EDIF file is to create a design with any name, import the EDIF file, and save the design and its libraries. Use the Top-level filter in the Libraries navigator to assist in finding the toplevel library. You can then open the toplevel library directly (in this context it becomes the design) and you can delete from disk the placeholder design that you initially created.

Exporting a Design

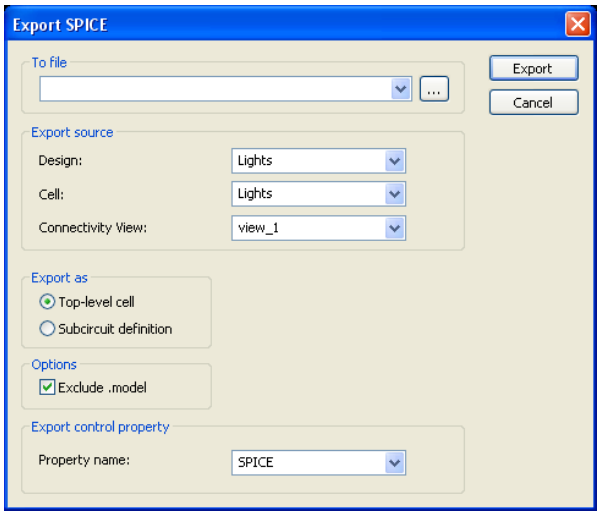
S-Edit can export your design or any portion of it to a variety of netlist formats for simulation or placement and routing. Once you have exported a netlist, you can use T-Spice to simulate it and W-Edit to view the waveform output, directly from S-Edit.

S-Edit writes a netlist for the current cell and all of its instances, unless any of these instances contain output properties that describe them as primitive cells. The following sections provide details on S-Edit's precise export behavior according to the type of netlist being exported.

Exporting SPICE Files

When you export a SPICE netlist file, you can choose several options related to waveform probing. You can also choose whether to suppress the **.END** command in SPICE output. If you plan to include the netlist file in other SPICE netlist files, you should check **Exclude .model**.

To export a SPICE netlist, use the **File > Export > Export SPICE** command.



To file — Enter or browse to the filename you wish to export.

Export Source — Enter the **Design**, **Cell** and **Connectivity View** you want to export.

Export As — **Top Level Cell**—identify this cell as the top level of your design in the netlist output.

Subcircuit Definition—identify the netlist as a subcircuit using the following format:

```
.SUBCKT name pin1 [pin2 ...] [par1=val1 par2=val2 ...]
<subcircuit definition>
.ENDS [name]
```

Exclude .model	When checked, excludes the **.model** statement, if any is defined, from the netlist output. This option only applies to designs with cells having connectivity defined by importing a SPICE netlist. When a netlist is imported, a model definition is created internally for primitive devices. To suppress that model definition from being exported as a .model statement, check this box. It is often desirable to exclude the .model statement in the case where such models are included from external library files. (See also SPICE.OUTPUT property on page 104.)
Export control property	Property name containing **.OUTPUT**, **.PRIMITIVE**, **.ORDER**, and **.DEFINITION** sub-properties. (See SPICE Export Properties, below.)

SPICE Export Properties

SPICE export traverses the design hierarchy from the specified cell, and writes a device instantiation or subcircuit instantiation for every instance in the schematic. For each unique subcircuit instanced, it writes a subcircuit definition. The subcircuit definition consists of a subcircuit header, a device or subcircuit instantiation for every instance in the schematic of that subcircuit, and a **.ends** statement.

SPICE netlisting is controlled by the contents of several properties. Typically, these are placed on a symbol, and provide the default netlisting. They might also be overwritten on a per-instance basis, to allow for custom exports. The properties that control SPICE output are:

- **.OUTPUT**
- **.PRIMITIVE**
- **.ORDER**
- **.DEFINITION**

These properties must be created as a sub-property of a property. For example, a symbol can have properties, where SPICE1.OUTPUT specifies one set of output parameters and SPICE2.OUTPUT can specify a different set of output parameters. Similarly, SPICE1.PRIMITIVE and SPICE2.PRIMITIVE can express different levels at which traversal of the hierarchy should proceed. You would then enter either SPICE1 or SPICE2 in the **Export control property** to specify which .OUTPUT and .PRIMITIVE properties to use when you export SPICE. (In most cases there will be only one set of .OUTPUT and .PRIMITIVE properties.)

Note: The default Tanner sort order for output properties is first numeric then alphabetic, to yield the sequence 1, 2, 3, 10, 20, 25, 30, 40 as opposed to 1, 10, 2, 20, 25, 3, 30, 40.

S-Edit will export a hierarchical SPICE netlist to the specified file according to the following rules:

- If a cell contains a **SPICE.OUTPUT** property on its symbol view, S-Edit will interpret the property value and write its interpreted value to the file, followed by a new line. S-Edit will not search the cell's schematic views for additional instances.

If the cell's symbol view does not contain a **SPICE.OUTPUT** property, S-Edit:

- Writes a subcircuit call with the name of the instanced cell.
- Writes a subcircuit definition by examining the cells instanced on the schematic views in alphabetic order and writing out each instance it encounters.

- Generates the correct **.subckt** and **.ends** lines bracketing each subcircuit. You cannot customize the **.subckt** or **.ends** lines; however, you can change the order in which non-global ports are written.
- If a cell contains instances but no ports or propagated global nets, S-Edit will generate a subcircuit call without ports.
- If a cell contains no instances but contains ports or propagated global nets, S-Edit will write that cell to the netlist as an empty subcircuit.
- If a cell contains no instances, ports, or propagated global nets, S-Edit will ignore the instance.
- S-Edit writes the top-level cell as the main circuit. It is the last block of circuitry S-Edit writes to the file.

S-Edit will append any properties on a subcircuit symbol except for output properties to the subcircuit definition block as subcircuit parameters.

SPICE.OUTPUT property

The SPICE.OUTPUT property is used to control how instances are written to the SPICE netlist. In particular, the SPICE output property is used to control how terminals, model names, and properties are written in device or subcircuit instantiations. The value of the SPICE output property may contain expressions that reference ports, TCL expressions or other properties. The values of these expressions are then substituted and the results are exported to SPICE.

Evaluation is a two-step process where first properties are substituted then inline TCL code is executed.

A language equivalent to the general property language is used for SPICE output. Braces are optional, and should be used when the property name has a space, or when abutting something immediately after the token. S-Edit searches user properties first, then system properties. The following may be used:

%{N}	Substitute the netname connected to pin N.
${P}	Substitute the value of property P on the same instance as the SPICE.OUTPUT property.
?{P}	Becomes just P, and references the property P directly above in the hierarchy.
@{P}	Becomes just P, and references a property P above in the hierarchy, searching from the top of the hierarchy down until the value is reached.
%	The percent sign by itself is a shortcut for all nets connected to the ports of the device in Tanner-standard order.
$	The dollar sign by itself is a shortcut for all the “interface” properties of the device.
[string]	Pass the string contained in brackets to the TCl interpreter for evaluation.

If the SPICE.OUTPUT property is not required, and not present, then a SPICE subcircuit call is emitted, and all interface properties are appended. If SPICE.OUTPUT is present, but expands to an empty string, no output is written.

SPICE.PRIMITIVE property

A property SPICE.PRIMITIVE property with value TRUE on an instance causes the traversal to stop at this level of the hierarchy (i.e., the schematic page is NOT exported, nor are any subcircuits instanced in it), and causes no definition to be written for the symbol in question.

SPICE.ORDER property

A property SPICE.ORDER with integer value on an instance controls the relative order in which SPICE statements are emitted. Instances are written in increasing SPICE.ORDER value.

SPICE.DEFINITION property

The SPICE.DEFINITION property is used in the definition of a subcircuit. Typically, this is a ".subckt", with some subset of parameters as shown below.

```
.SUBCKT name pin1 [pin2 ...] [par1=val1 par2=val2 ...]
<subcircuit definition>
.ENDS [name]
```

SPICE Output Examples

Example 1: MOSFET

A MOSFET symbol will typically have the following properties:

```
SPICE.OUTPUT = M{$Name} %D %G %S %B $MODEL W=$W L=$L AS=$AS AD=$AD PS=$PS
    PD=$PD
SPICE.PRIMITIVE = true
```

The symbol will also usually have properties for MODEL, W, L, AS, AD, PS, PD. Consider an instance of a MOSFET with he following properties:

```
MODEL=PMOS
W='24*l'
L='2*l'
AS='114*l*l'
AD='72*l*l'
PS='60*l'
PD='30*l'
```

The name of the instance is P4, and the drain, gate, source and bulk pins of the instance are connected to QB, Q, Vdd, and Vdd respectively.

When the SPICE output statement above is evaluated, the following steps occur:

[1] For $Name, substitute the value of the property "Name" that is on the same instance as the SPICE.OUTPUT property. The property "Name" typically does not exist as a user property, but does exist as a system property, and is the name of the instance, in this case P4.

[2] For %D %G %S %B, substitute the names of the nets connected to ports D, G, S, and B of the instance, in this case QB, Q, Vdd, Vdd.

[3] For $MODEL W=$W L=$L AS=$AS AD=$AD PS=$PS PD=$PD, substitute the values of these properties.

The SPICE device statement written for this instance will then be:

```
MP4 QB Q Vdd Vdd PMOS W='24*l' L='2*l' AS='114*l*l' AD='72*l*l' PS='60*l'
    PD='30*l'
```

The SPICE.PRIMITIVE = true property prevents a subcircuit definition for the MOSFET from being written.

Example 2: MOSFET with Property Substitution

Consider again the SPICE.OUTPUT property for a MOSFET:

```
SPICE.OUTPUT = M{$Name} %D %G %S  %B $MODEL W=$W L=$L AS=$AS AD=$AD PS=$PS
    PD=$PD
```

In the above example, the properties MODEL, W, L, AS, AD, PS, PD themselves can use the general property language. For example, the following properties exist on an instance

```
MODEL = PMOS
W = '?Width'
L = '2u'
AS = '$W*3u'
AD = '$W*3u'
PS = '2*$W + 6u'
PD ='2*$W + 6u'
```

And a property Width =20u exists on the instance of the cell in which the MOSFET is located. The SPICE device statement written for this instance will then be:

```
MP_4 QB Q Vdd Vdd PMOS W='20u' L='2u' AS='20u*3u' AD='20u*3u' PS='2*20u +
    6u' PD='2*20u + 6u'
```

Example 3: Conditional Output

In the above example, the properties were assumed to always exist. If a property does not exist, the substitution will result in an empty string. This can produce unwanted results in the SPICE output, such as “AD=” if no value for AD exists.

TCL commands can be inserted into the SPICE output string by placing the TCL command in square brackets. The build in TCL command “se” performs selective evaluation, and can be used to perform conditional output, as in the SPICE.OUTPUT property below:

```
SPICE.OUTPUT = M{$Name} %D %G %S %B $MODEL W=$W L=$L [ se $AS {AS=$AS} ] [
    se $AD {AD=$AD} ] [ se $PS {PS=$PS} ] [ se $PD {PD=$PD} ]
```

Here, if a property $AS is resolved with value AS_value, then [se $AS {AS=$AS}] will return AS=AS_value. If $AS does not resolve to anything, because the property AS does not exist, then [se $AS {AS=$AS}] will return an empty string.

Example 4: Controlling Subcircuit Output

The SPICE.OUTPUT property can also be used to control the output of subcircuits, as well as of primitive devices. If no SPICE output statement exists on an instance, then the instance is written as a

subcircuit instantiation, with the "X" prefix, and all properties marked as interface properties (sub-property IsInterface=true) are written out.

If needed, the output can be explicitly specified:

```
SPICE.OUTPUT=X${Name} % ${MasterCell} P1=$P1 P2=$P2
```

Here *Name* and *MasterCell* are system properties referring to the instance name and cell name respectively. % by itself outputs all the nets connected to the subcircuit ports.

Example 5: Title Block

A title block is an instance that is used to display information about a cell. It references properties and displays information about the cell in which it is instance. A title block will typically have properties:

```
Cell = ?{Cell}
Info = ?{Info}
Author = ?{Author}
SPICE.OUTPUT=[ ]
SPICE.PRIMITIVE = true
```

The title block uses properties with the ? reference to refer to the parent cells properties. The SPICE.OUTPUT property value [ ] is used to return an empty string, because we don't want any SPICE output for this instance.

Passing Subcircuit Parameters to the Originating Cell

When you export a SPICE netlist, S-Edit can pass the values of designated properties in high-level cells down the design hierarchy to the subcircuit definition block of the instanced, or originating, cell.

A property is eligible for subcircuit parameter passing when it meets the following conditions:

- It does not appear on the symbol of a primitive cell—that is, a symbol containing a **SPICE.OUTPUT** property.
- It does not appear on the symbol of the top-level cell.
- Its name is listed in the value of a **SPICE.PARAMETER** property.

Steps in the Subcircuit Parameter Passing Process

[1] Open a high-level cell and switch to a symbol view, if necessary. Add a property whose value contains the parameter you want to pass down the hierarchy.

[2] Add a **SPICE.PARAMETER** property whose value contains the name or names of the properties you wish to pass to the subcircuit.

[3] Switch to a schematic view, select an instance of a cell to which you want to pass the parameter, and replace the value of the originating cell's property with the name of the property in the instancing, or high-level, cell, whose value you want to pass down the hierarchy.

[4] Finally, export a SPICE netlist. S-Edit will include the values of high-level cell properties in the subcircuit definition block of the instanced cell. It will also record property value overrides in *instances.*

Exporting Global Node Connections

When S-Edit exports a SPICE netlist, it writes out global node connections by adding "hidden" ports to each cell's symbol and instances of that symbol. Global connections are thus compatible with any SPICE simulator, without the use of complex node aliasing commands.

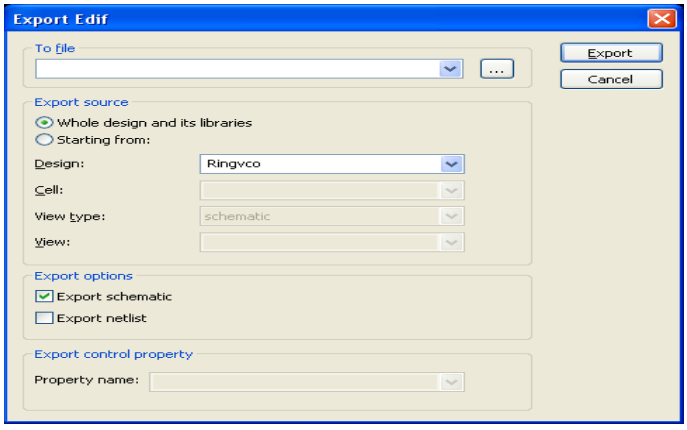
The following is an excerpt from an exported SPICE file containing global nodes. Omitted text is indicated by ellipses (**...**). The subcircuit definition for **OR2** involves user-defined nodes **A**, **B**, and **Y**, and two additional nodes, **Gnd** and **Vdd**, created by S-Edit to propagate the corresponding global nodes into the calls to subcircuit **OR2** inside the definition of **core**.

```
.SUBCKT OR2 A B Gnd Vdd Y
M54 Y N8 Gnd Gnd NMOS W=22u L=2u AS=66p AD=66p PS=24u PD=24u
M55 Y N8 Vdd Vdd PMOS W=22u L=2u AS=66p AD=66p PS=24u PD=24u
M56 N8 B Gnd Gnd NMOS W=22u L=2u AS=66p AD=66p PS=24u PD=24u
M57 N8 A N11 Vdd PMOS W=22u L=2u AS=66p AD=66p PS=24u PD=24u
M58 N8 A Gnd Gnd NMOS W=22u L=2u AS=66p AD=66p PS=24u PD=24u
M59 N11 B Vdd Vdd PMOS W=22u L=2u AS=66p AD=66p PS=24u PD=24u
.ENDS

.SUBCKT core CLOCK DONT_EW ... YELLOW_EW YELLOW_NS
XAND2_1 N4 RED_NS Gnd Vdd GREEN_EW AND2
XOR2_1 TEST_POINT N5 Gnd Vdd N4 OR2
XAND2_2 N4 RED_EW Gnd Vdd GREEN_NS AND2
...
XDFFC1_7 RESETB CLOCK N66 Gnd N65 N66 Vdd DFFC1
.ENDS
```

Exporting EDIF Files

S-Edit can export EDIF schematics retaining the hierarchy, properties and all text. Use the **File > Export > Export EDIF** command.



To file — Enter or browse for the name of the EDIF file to be output.

Export source — Select the **Whole design and its libraries** you want to export. Or, select **Starting From** and then specify the **Design**, **Cell**, **View type** and **View** of the top level view you want to export.

Export options	**Export schematic**—this option outputs the graphical elements of the design, including all symbols and schematic views, as well as interfaces. **Export netlist**—this option outputs the connectivity of the design. Use this option to export a netlist for place and route. If neither option is checked, then a list of cells with interfaces is exported.
Export control property	Enter the name of the property containing the .PRIMITIVE sub-property used to stop traversal of the design hierarchy.

EDIF export will normally write out an entire design to full extent of the hierarchy. To stop export at a particular instance, place a property with the sub-property .PRIMITIVE having value “true” on the symbol of the cell, or on each instance of that cell.

If a cell's symbol contains an EDIF **.PRIMITIVE = true** property, S-Edit:

- Writes an EDIF cell definition for the cell without examining the cell's schematic pages for additional instances. The cell definition will contain ports and global ports.
- Treats the cell as if it were instanced in the top-level cell.

If a cell does not contain an EDIF **PRIMITIVE = true** property on its symbol page, S-Edit examines all of the cell's schematic and writes out each instance to the netlist.

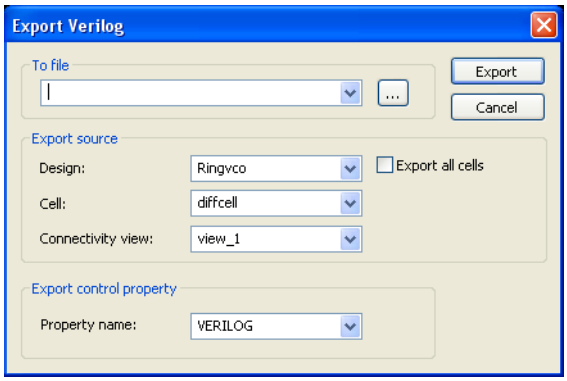
- If the instanced cell contains no ports or propagated global nets, S-Edit ignores the instance.
- If the instanced cell contains ports but no instances, it is an error. S-Edit writes a message to the netlist file identifying the cell with the error and stating that it “requires an EDIF definition.” S-Edit also displays a dialog box to warn you of the error.
- After writing all primitive cell definitions, S-Edit writes a top-level cell containing instances of all primitives in the design and the nets that connect these primitives.
- S-Edit generates scoped node names, which uniquely identify a node by including its hierarchical position in the node name-e.g., *inst1/inst2/.../node_name*, where *inst1*, *inst2*, etc. are the names of the instances in descending order from the top level of the design to the level of the node, and *node_name* is the name of the node. Local node names are scoped to show the hierarchy of instances from the top level to the level that contains the node. Global node names are scoped to show the hierarchy of instances from the cell containing the global node symbol to the level at which the global node is capped. Uncapped global nodes will be written without a scope-that is, **Gnd** will simply appear as “Gnd” in the output file.
- S-Edit will automatically convert any names that are incompatible with EDIF naming requirements to a legal EDIF name using the **rename** construct.

Example

A netlist-only EDIF file that terminates at standard cells, rather than going to transistor level, is desirable when you export EDIF for standard cell place and route. To do this, create a property EDIF.PRIMITIVE = true (the period indicates PRIMITIVE is a sub-property of EDIF) on each standard cell symbol, and then enter EDIF as the **Export control property**.

Usuallyan EDIF schematic that traverses all the way down the hierarchy is the desired output when you export EDIF to view in another tool. In this case, leave the **Export control** property blank.

Exporting Verilog Files



To file Enter or browse for the name of the netlist file to be output.

Export source Enter the **Design**, **Cell** and **Connectivity View** you want to export.

Export all cells Check this box to export all cells in a design.

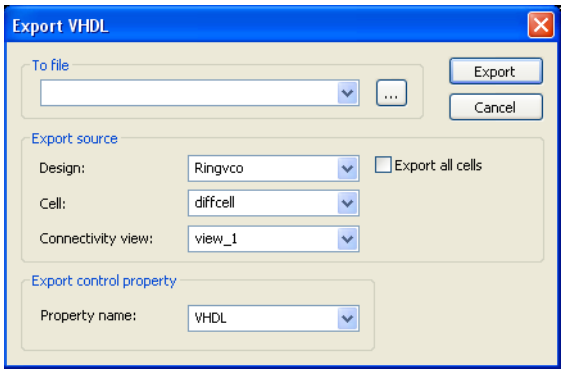
Export control property Enter the **Property name** containing the .PRIMITIVE sub-property used to stop traversal of the design hierarchy.

The Verilog output option in S-Edit creates a subcircuit call for every level in the design hierarchy. S-Edit cells that have symbols but no schematics are considered primitive elements, and no definition of these elements is written to the Verilog output file. It is assumed that these primitive elements are contained in user-supplied Verilog libraries, if they are required at all.

In some cases, you might want to identify an element which does contain a schematic view as a primitive element. This situation most typically arises with standard-cell libraries that contain cells such as NAND, which have transistor-level schematics attached.

To identify a cell that contains a schematic as a primitive, such as a NAND cell, place a property on the symbol of the cell with a subproperty named PRIMITIVE and value true, and identify that property name as the export control property in the **Export Verilog** dialog. Typically one would create a property called VERILOG, with subproperty PRIMITIVE and value true, but any name can be used in place of "VERILOG."

Exporting VHDL Files



To file	Enter or browse for the name of the netlist file to be output.
Export source	Enter the **Design**, **Cell** and **Connectivity View** you want to export.
Export all cells	Check this box to export all cells in a design.
Export control property	Enter the **Property name** containing the .PRIMITIVE sub-property used to stop traversal of the design hierarchy.

S-Edit will write the connectivity information in a design in VHDL format according to the following rules:

- If an instanced cell contains a VHDL.PRIMITIVE property on its symbol view, S-Edit will write a VHDL entity with an empty behavioral architecture, which you can edit in the VHDL code to provide a behavioral definition. S-Edit will not search the cell's schematic views for additional instances.

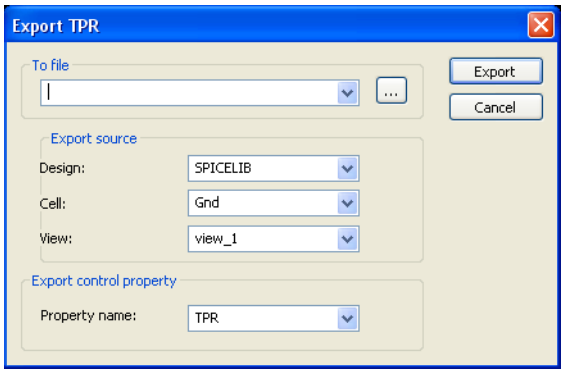
If the instanced cell's symbol view does *not* contain a VHDL.PRIMITIVE property, S-Edit:

- Writes a structural architecture with the name of the instanced cell.
- Writes a structural architecture by examining the cells instanced on the schematic views and writing out each instance it encounters, plus the connectivity of the instance, so that you can conveniently provide a description in VHDL.
- Defines the primitive as a VHDL entity with ports defined as std_logic signal ports.
- If an instanced cell contains no instances, S-Edit will ignore the instance
- If an instanced cell contains ports but no instances, S-Edit will signal an error and write the cell to the netlist as an entity with empty structural architecture.
- S-Edit writes the top-level cell as the top-level entity in the VHDL file.

Note: Unlike other netlist formats, VHDL netlists require that the top-level cell have a valid symbol view, with ports that correspond to its schematic ports. This information is used to define the top-level entity.

Exporting TPR Files

TPR is a flat netlist format you can use to place and route your design in L-Edit.



To file Enter or browse for the name of the netlist file to be output.

Export source Enter the **Design**, **Cell** and **View** you want to export.

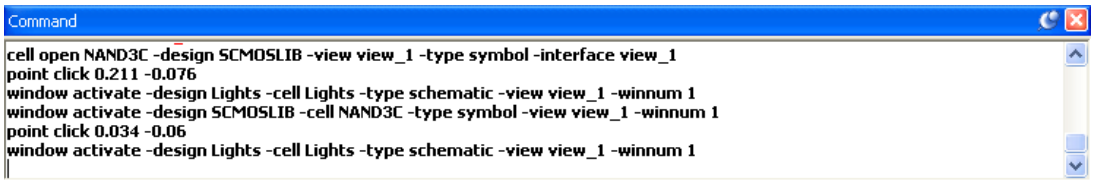
Export control property Enter the name of the output property, typically simply TPR.

If an instanced cell contains a TPR.OUTPUT property, S-Edit will generate a netlist according to the following rules. It will not examine the schematic for additional instances.

- S-Edit generates two lines for the cell and writes them to the file. The first line indicates formal parameters for the instanced cell, and the second line indicates the mapping of formal parameters to node names in the instance.
- S-Edit generates no global node ports for the cell.
- The property TPR.OUTPUT can have any name, which is then entered in the Export control property field.
- S-Edit generates scoped names which uniquely identify an instance by including its hierarchical position. For example, top/cell1/cell2/.../instance_name, where cell1, cell2, etc. are the names of the cells in descending order from the top level of the design to the level containing the instance. Global symbol names are scoped to show the hierarchy of instances from the cell containing the global symbol to the level at which the global net is capped. Uncapped global nets will be written without a scope—that is, Gnd will simply appear as “Gnd” in the output file.
- If an instanced cell does not contain a TPR.OUTPUT property on its symbol view, S-Edit examines all cells in alphabetic order and writes out each instance to the netlist.
- If the instanced cell contains no ports or propagated global nets, S-Edit ignores the instance.
- If the instanced cell contains no instances but contains ports, it is an error. S-Edit writes a message to the netlist file identifying the cell with the error and stating that it “requires a TPR definition.” S-Edit also displays a dialog box warning of the error.

9 Scripting with TCL

S-Edit supports the TCL scripting language as a method to enter commands in the Command window, and also to run scripts. When a command in entered in S-Edit through the graphic interface, the corresponding TCL command is issued in the Command window. It is actually the TCL command in the Command window that causes S-Edit to execute the command; text entered into the Command window functions as a script to instantly execute operations.



Text can be written in directly, copied from executed operations and then pasted back into the Command window, or invoked from a saved TCL file. Scripts written in the TCL language can also be dragged and dropped from a browser into the Command window to execute immediately.

TCL Commands Available in S-Edit

A full list of S-Edit TCL commands is available by typing “help” in the Command window. The available S-Edit commands are listed below. Help on any specific command, as well as a list of subcommands and options, can be obtained by entering the command followed by “-help”.

cell	export	point
copy	find	port
cut	help	property
database	import	redo
delete	instance	setup
design	label	texteditor
document	mode	tsource
draw	netcap	undo
duplicate	paste	window
exit	path	workspace

Running TCL Scripts

To run a TCL script in S-Edit, you can:

[1] Drag and drop the script into the Command window from a browser

[2] Invoke **File > Open > Execute Script**

[3] Invoke **File > Recent Scripts** and select a previously run script.

[4] Type “source” followed by the path and filename for the script, as described in Source Scripts, below.

You can also have a script run automatically when S-Edit is opened, when its closed, or when a design is loaded, simply by saving the TCL file in the appropriate directory. (See Launching Scripts Automatically on page 114.)

Source Scripts

Source scripts take the form :

```
source filename | foldername [-subfolders] [-relativeto user|design]
   [-mru false|true]
```

filename	Source runs the specified script file. The filename can contain an absolute or relative path.
foldername	Source runs all scripts in the specified folder in alphabetical order. The foldername can contain an absolute or relative path. Does not recurse into subfolders by default.
-relativeto user | design	**-relativeto user** If a relative path is given, and **-relativeto user** is specified, the root is **C:\Documents and Settings\<*username*>\Application Data\Tanner EDA\scripts**. **-relativeto design** If a relative path is given, and **-relativeto design** is specified, the root is the design folder. If no **-relativeto** option is specified, then **-relativeto** *user* is assumed.
-subfolders	When a foldername is specified, source recurses breadth first into subfolders. (All scripts in the specified folder are run first, in alphabetical order, then subfolders are recurred into in alphabetical order).
-mru false | true	By default, “source filename” puts that script into the mru, and “source foldername” does not put the scripts in that folder into the mru. Specifying “-mru false” or “-mru true” allows you to override that behavior.
-help	Displays an explanation of command options and syntax.

Launching Scripts Automatically

To Run a Script when S-Edit Opens

To run a script automatically when S-Edit is started, place the script in the folder:

C:\Documents and Settings\<*username*>\Application Data\Tanner EDA\scripts\startup

To Run a Script when S-Edit Closes

To run a script automatically when S-Edit is shut down, place the script in the folder:

C:\Documents and Settings\<*username*>\Application Data\Tanner EDA\scripts\shutdown

To Run a Script when a Design Opens

To run a script when a design is opened, place it in the folder:

C:\Documents and Settings\<*username*>\Application Data\Tanner EDA\scripts\open.design

S-Edit creates the **open.design** folder during installation. Whenever a design is opened in S-Edit, all the TCL scripts in this directory will be executed. File naming is unrestricted, but note that S-Edit reads files in alphabetical order. Thus if you have a required sequence, use file names to control the order in which files in the the **open.design** folder are executed.

If it is just setup definitions that you want to apply locally, you can alternately save your **Setup** dialog values to the **{user preferences folder}**, from which S-Edit populates the **open.design** folder. (See Saving and Loading Setup Options on page 17.)

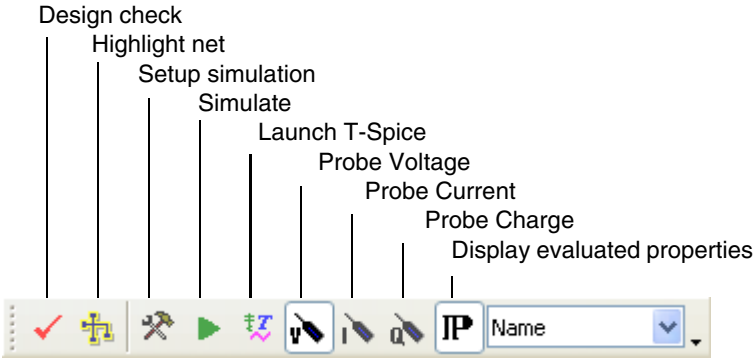
For example, if you want to view wires in red in in each design, regardless of the default setting for the file, you would use the **Wires** field in **Setup > Technology > Schematic Colors** to select red, make sure that the corresponding checkbox is checked, and then save to your **{user preferences folder}**. This will save a **colors.tcl** script to the **open.design** folder so that every time you load a design, S-Edit will run **colors.tcl** and set the wires to red.

10 Simulation and Waveform Probing

Tanner's schematic and simulation tools are fully integrated to allow AC, DC, or transient analysis of your design, with interactive setup, simulation, and post-processing. The three components of this process are S-Edit, T-Spice, and W-Edit, and there are three primary stages to the simulation flow:

- In the *setup* stage, the S-Edit user enters commands and information which describe the type of simulation (DC, AC, transient, etc.), and establish the simulator options and outputs.
- In the *design export* and *simulation* stage, S-Edit generates a SPICE file (a netlist) from the design. Then, T-Spice simulates the SPICE file to create a probing data file with voltage and current values for each node and device in the design, and for each analysis specified in the SPICE file.
- In the *probing* stage, W-Edit displays traces from the probe data file corresponding to an analysis type and a specific net or device selected in S-Edit. S-Edit can also annotate your schematic with node voltages and device terminal currents and charges.

The SPICE simulation toolbar provides quick access to key functions.

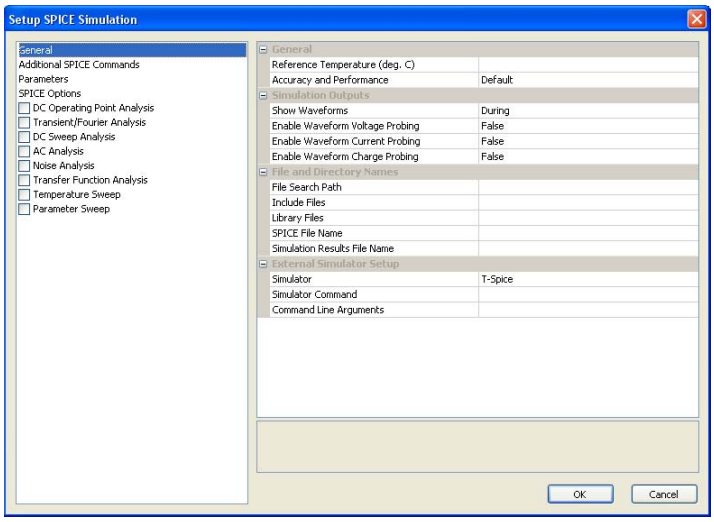


SPICE Simulation Settings

Before running the T-Spice simulation you must enter some basic settings which will define the type of simulation to be performed and the outputs from the simulation. You enter these values in the SPICE simulation setup dialog.

General SPICE Settings

The **General SPICE** simulation settings are described below.



Reference Temperature

Tnom - the nominal or reference temperature at which device model parameters were measured; value may be overridden in individual model definitions using the **tnom/tref** model parameter. Tnom will also represent the default operating temperature of the circuit. (degrees Celsius)
Related T-Spice command: **.option tnom=*temp***

Accuracy and Performance

Allows the user to set macro-level switches to control the tradeoff of accuracy versus performance. Valid selections are **default**, **accurate**, and **fast**. The **accurate** setting will enhance accuracy at some expense of performance. The **fast** setting will provide faster simulations with some compromise of accuracy. The **default** setting, which is generally the best setting, balances accuracy and speed.
Related T-Spice commands: **.option fast** and **.option accurate**

Additional SPICE Commands

A list of additional SPICE commands which will be entered directly in the netlist. Multiple commands should be separated by semicolons.

Show Waveforms

Select when W-Edit should display the simulation results.
Choices:
During - Display solutions during simulation with realtime updating
After - Display solutions at the end of the simulation
Don't Show - W-Edit will not be activated at all

Enable Waveform Voltage Probing

Indicate whether node voltage values should be included in the probe data output file.
Related T-Spice command: **.option probev=true | false**

Enable Waveform Current Probing

Indicate whether device terminal current values should be included in the probe data output file.
Related T-Spice command: **.option probei=true | false**
Warning: May cause excessively large output files.

Enable Waveform Charge Probing

Indicate whether device terminal charge values should be included in the probe data output file.
Related T-Spice command: **.option probeq=true | false**
Warning: May cause excessively large output files.

Enable Waveform Noise Probing

Indicate whether device noise values should be included in the probe data output file.
Related T-Spice command: **.option probei=true|false**

File Search Path

A list of directories containing the library and include files. You must separate multiple list entries with semicolons.
Related T-Spice command: **.options search=*path***

Include Files

A list of semicolon separated files whose contents should be included in the netlist.

Include files can be configured in three different ways:

1. To specify an exact path and filename for an include file, enter the full path and filename in the include file field.

2. To specify an include file in or relative to the project folder, enter the relative path and filename without quotes, and the full path with filename will be automatically constructed and placed in a .include statement in the SPICE file.

3. To specify an include file in or relative to the folder specified by the search path, enter the relative path and filename in quotes, and relative path and filename will be placed directly in a .include statement in the SPICE file.

Multiple include files should be separated by semicolons.

Related T-Spice command: **.include *filename***

Library Files

A list of semicolon separated model library files and optional section names. Note - *libraryname* should be quoted if *sectionname* is included.

Library files may be specified in the same three ways as include files. However, with library files there is also the name of the library inside the file that must be specified. The name of the library is placed immediately after the library name.

“LibfileA” LibnameA; “LibfileB” libnameB; ...

This generates SPICE .lib statements that looks like:

.lib “LibfileA” libnameA
.lib “LibfileB” libnameB

Related T-Spice commands:
.lib *libraryname*
.lib *libraryname sectionname*

SPICE File Name

The name of the SPICE file which will be created and simulated when the **Simulate** command is executed.

Simulation Results File Name	The name of the output file which will contain T-Spice simulation results, and can be plotted by W-Edit.
Simulator	Allows selection of different simulators. *Choices*: **T-Spice, other**
T-Spice Engine	The name of the T-Spice engine DLL. The default will be the most recently executed T-Spice version.
Simulator Command	The command or executable name for running the SPICE simulator
Command Line Arguments	Additional arguments to be included in the simulator execution command
Post-Simulation Command	The command or executable name for running the post-processing waveform viewer.

Note:

When the SPICE netlist is generated, all directory and filenames entered in the **General Settings** dialog will be expanded to refer to fully qualified path names. If this is not the desired behavior, then the name should be entered within quotes.
For example, an include file identified as either ./mosis2u.md or mosis2u.md will be written to the SPICE file as *designpath*\mosis2u.md, where *designpath* is the directory containing the schematic design. If the filename had been entered as "mosis2u.md", then it would not have any pathname expansion performed.

SPICE Parameters and SPICE Options

The **Parameter** setup allows you to enter an arbitrarily long list of parameter assignments. Parameters are ***name=value*** pairs which define numeric variables that can be used throughout the SPICE input files in equations or as model and device parameter values.

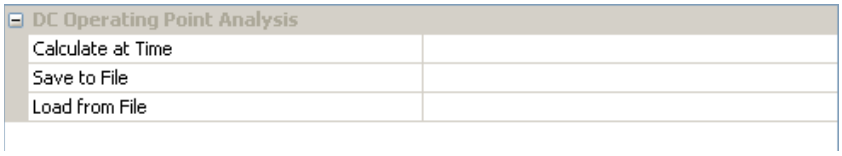
Related T-Spice *command*: **.param *name=value***

SPICE options are also entered as name and value assignment pairs. The name can be any valid T-Spice option name, and the value is the numeric or string value of the option.

Related T-Spice command: **.option *optionname=setting***

For both these dialogs, use the **create new** button to enter a parameter and the **delete** button to remove one.

DC Operating Point Analysis



Print DC Operating Point	Indicate whether the DC Operating Point bias and AC small-signal information should be written to the output file. *Related T-Spice command*: **.op**

Calculate at Time
A list of transient timepoints at which the bias information and AC small-signal data should be printed.
Related T-Spice command: **.op *T0 T1 T2***

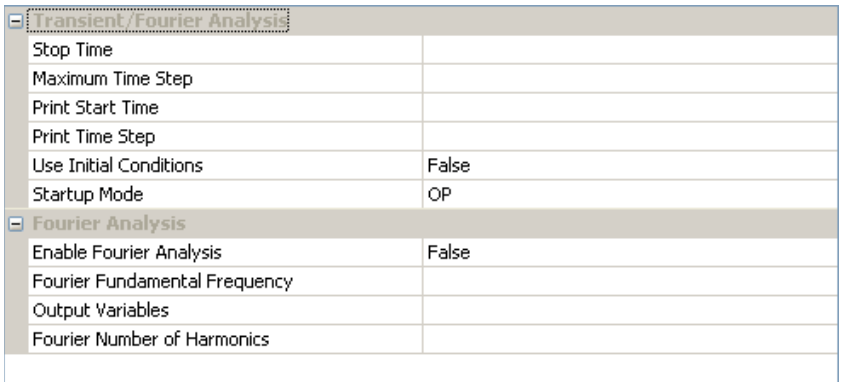
Save to File
Save the node voltage values for the DC operating point to a file in a format which can be loaded in subsequent simulations. Computational time may thus be reduced by providing an improved initial guess for voltages in the subsequent runs.
Related T-Spice command: **.save file=*filename***

Load from File
Load the bias point information from a previous simulation, and use the node voltage values as the initial guess in the DC operating point computation.
Related T-Spice command: **.load file=*filename***

Transient/Fourier Analysis

Fourier analysis can be performed in W-Edit, as well as in T-Spice. There are a number of advantages to performing the Fourier analysis in W-Edit:

- Sampling density can be adjusted
- Start and stop times can be varied
- Windows can be applied in order to reduce the error effects of finite time sampling
- Results can be computed and plotted interactively, without rerunning the simulation



Stop Time
The transient simulation stop time, i.e. the total simulation time.
Related T-Spice command: **.tran *Tstep Tstop***

Maximum Time Step
The maximum timestep to be taken during the transient simulation.
Related T-Spice command: **.tran *Tstep Tstop***

Print Start Time
The time when simulation results printout will begin. If not set, then the results printout will begin immediately, at time = 0.
Related T-Spice command: **.tran *Tstep Tstop* START=*Tstart***

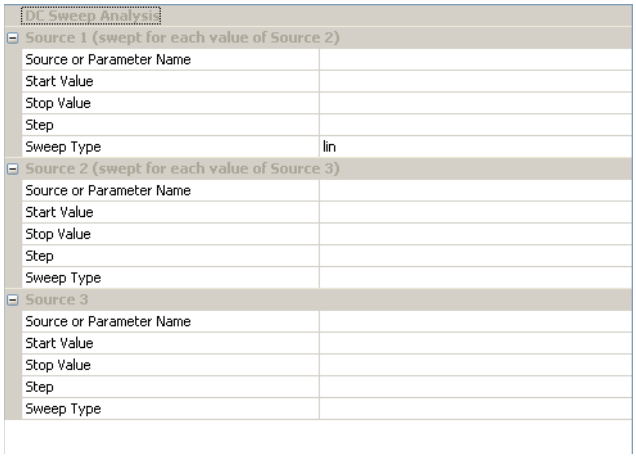
Print Time Step
The time increment at which solutions are printed.
Related T-Spice command: **.option prtdel=*step***

Use Initial Conditions	Instructs the simulator to use node voltage initial condition values at startup time 0, rather than computing the DC operating point. *Related T-Spice command*: **.tran *Tstep Tstop* UIC**
Startup Mode	Select the transient simulation startup mode. Choices: **OP** - standard DC operating point computation (default) **Powerup** - voltage and current sources are ramped up **Preview** - Does not perform an actual transient analysis; for previewing source values. *Related T-Spice command*: **.tran/powerup** and **.tran/preview**
Enable Fourier Analysis	True or False.
Fourier Fundamental Frequency	The fundamental frequency about which the harmonic components are computed. *Related T-Spice command:* **.four *freq outputs***
Output Variables	List of voltage and current values for computation of spectral components. *Related T-Spice command:* **.four *freq outputs***
Fourier Number of Harmonics	The number of harmonic frequencies for which Fourier components are computed. *Related T-Spice command:* **.four *freq outputs* nfreqs=*N***

DC Sweep Analysis (or DC Transfer)

The **DC Sweep Analysis** dialog permits the user to define up to three levels of sweeping for DC analysis. DC sweeping is also referred to as DC transfer analysis. This analysis computes the DC states of a circuit while a voltage or current source is swept over a given interval. T-Spice allows three or more levels of source sweeping, in which the first source sweep is the innermost loop, and is swept for each value of the second source, which in turn is swept for each value of the third source.

In addition to sweeping current and voltage source values, parameter values may be swept in order to yield a DC curve as a function of the parameter value. Parameters are defined in the **Parameters** section of the setup dialog, and also include the intrinsic **temp** temperature value, which is the operating temperature of the circuit.



Source 1 (Swept for each value of Source 2):

Source or Parameter Name Voltage source, current source, or parameter name to sweep.

Start Value The beginning value of the sweep variable.

Stop Value The final value of the sweep variable.

Step The value step size for linear sweeps, or the number of points per decade/octave for decade and octave sweeps.

Sweep Type The type of sweep.
Choices and *Associated T-Spice commands*:
linear (lin) - **.DC *variable* linear *start stop increment***
decade (dec) - **.DC *variable* dec *start stop increment***
linear (oct) - **.DC *variable* oct *start stop increment***

Source 2 (Swept for each value of source 3):

Source or Parameter Name Voltage source, current source, or a parameter name to sweep.

Start Value The beginning value of the sweep variable.

Stop Value The final value of the sweep variable.

Step The value step size for linear sweeps, or the number of points per decade/octave for decade and octave sweeps.

Sweep Type The type of sweep.

Source 3:

Source or Parameter Name Voltage source, current source, or a parameter name to sweep.

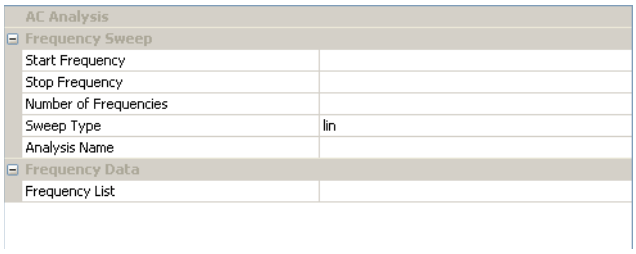
Start Value The beginning value of the sweep variable.

Stop Value The final value of the sweep variable.

Step The value step size for linear sweeps, or the number of points per decade/octave for decade and octave sweeps.

Sweep Type The type of sweep.

AC Analysis

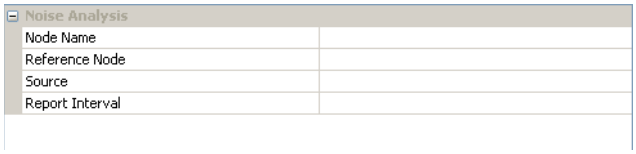


Start Frequency The beginning frequency.

Stop Frequency The final frequency.

Number of Frequencies	The number of frequency steps - total steps for linear sweeps, or steps per decade/octave for logarithmic sweeps.
Sweep Data	A list of data
Sweep Type	The type of sweep. Choices and *Associated T-Spice commands*: linear (lin) - **.AC linear *Fstart Fstop Fstep*** decade (dec) - **.AC dec *NF Fstart Fstop*** linear (lin) - **.AC oct *NF Fstart Fstop***

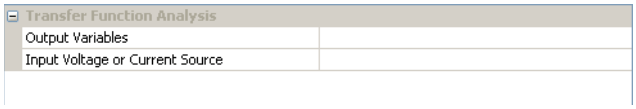
Noise Analysis



Node Name	Node at which the total noise is to be computed..
Reference Node	Reference node for the nodal voltage (default: GND).
Source	Voltage or current source to which input noise is referred.
Report Interval	Interval for printing noise report summaries; i.e. print a report every **N** frequencies.

Related T-Spice command: **.noise V(*node*, *ref*) *source interval***

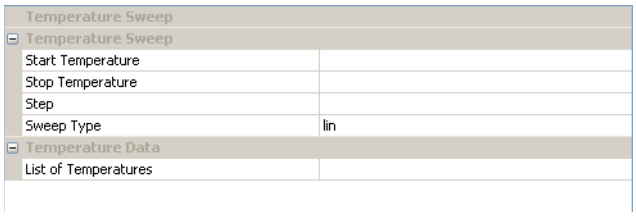
Transfer Function Analysis



Output Variables	The output small-signal variables. This is a list of any valid **.print** commands; e.g. v(out) i1(mos2)
Input Voltage or Current Source	The small-signal input source; i.e. the name of a current or voltage source in the schematic.

Related T-Spice command: **.tf *outvar invar***

Temperature Sweep



Start Temperature — The beginning operating temperature sweep value (degrees Celsius)

Stop Temperature — The final operating termperature value

Step Size or # of Steps — The temperature step size for linear sweeps, or the number of points per decade/octave for logarithmic sweeps

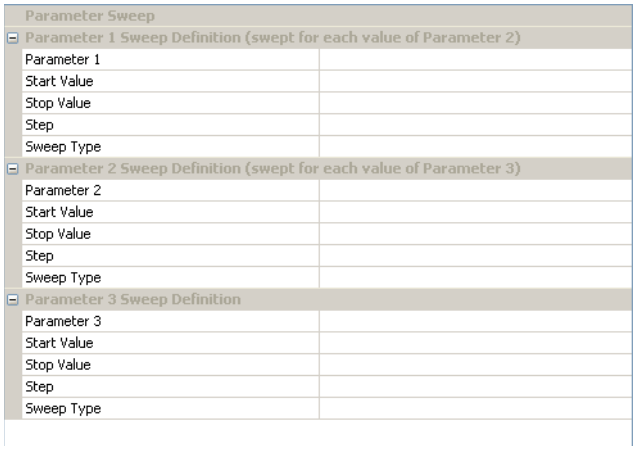
Sweep Type — The type of sweep.
Choices and *Associated T-Spice commands*:
linear (lin) - **.step temp linear *Tstart Tstop Tstep***
decade (dec) - **.step temp dec *NT Tstart Tstop***
linear (lin) - **.step temp oct *NT Tstart Tstop***

List of Temperatures — A list of circuit operating temperatures. All analyses will be rerun for each temperature.
(degrees Celsius)
Related T-Spice command: **.temp *T0 T1 T2 ...***

Note: The temperature sweep as defined by the four other variables and the **List of Temperatures** entry will generate two separate temperature variation commands. Typically only one or the other type of temperature variation will be used, not both.
Since every type of analysis in the SPICE netlist will be rerun for each temperature point, temperature sweeps should be used judiciously.

Parameter Sweep

The **Parameter Sweep** analysis dialog permits the user to define up to three levels of parametric sweeping. For each parameter value of the sweep, *every analysis command* which has been defined will be performed - i.e. for a sweep of length N, there will be N total transient, DC, and AC simulations performed.



Parameter 1 Sweep Definition (Swept for each value of Parameter 2):

Parameter Name The first sweep parameter.

Start Value The beginning value of the sweep variable.

Stop Value The final value of the sweep variable.

Step The value step size for linear sweeps, or the number of points per decade/octave for decade and octave sweeps.

Sweep Type The type of sweep.
Choices and *Associated T-Spice commands*:
linear (lin) - **.step *variable* linear *start stop increment***
decade (dec) - **.step *variable* dec *start stop increment***
linear (lin) - **.step *variable* oct *start stop increment***

Parameter 2 Sweep Definition (Swept for each value of Parameter 3):

Parameter 2 The second parameter to sweep.

Start Value The beginning value of the sweep variable.

Stop Value The final value of the sweep variable.

Step Size or # of Steps The value step size for linear sweeps, or the number of points per decade/octave for decade and octave sweeps.

Sweep Type The type of sweep.

Parameter 3 Sweep Definition:

Parameter 3 The third parameter to sweep.

Start Value The beginning value of the sweep variable.

Stop Value The final value of the sweep variable.

Step Size or # of Steps	The value step size for linear sweeps, or the number of points per decade/octave for decade and octave sweeps.
Sweep Type	The type of sweep.

Running Simulations

Once you have defined the simulations you want to run, use **Tools > Start Simulation**, or click on the  icon, to start SPICE simulation using Tanner's T-Spice application.

Use the General page in **Setup SPICE Simulation** to set whether W-Edit should display the results during simulation, after simulation, or not at all.

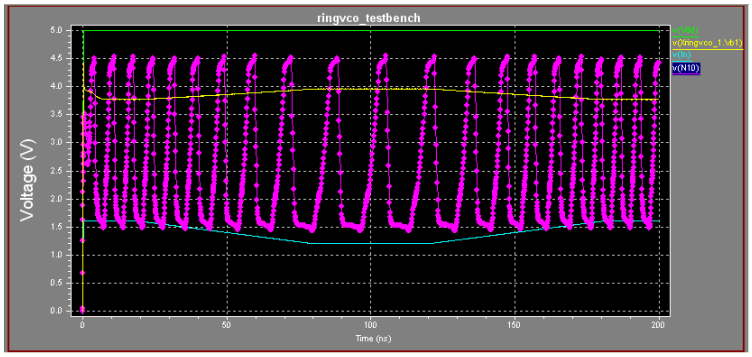
Probing Waveforms

Waveform probing allows you to probe nodes and devices in an S-Edit design to examine their circuit simulation results. The process takes place in three stages:

[1] In the *export* stage, you export a SPICE netlist from your S-Edit design file with at least one type of waveform probing enabled in **Setup > SPICE Simulation > General** so that S-Edit will write **.probe** commands to the netlist.

[2] In the *simulation* stage, you use T-Spice to simulate the exported netlist. When T-Spice encounters a **.probe** command during simulation, it outputs data to the probe data file, a binary file.

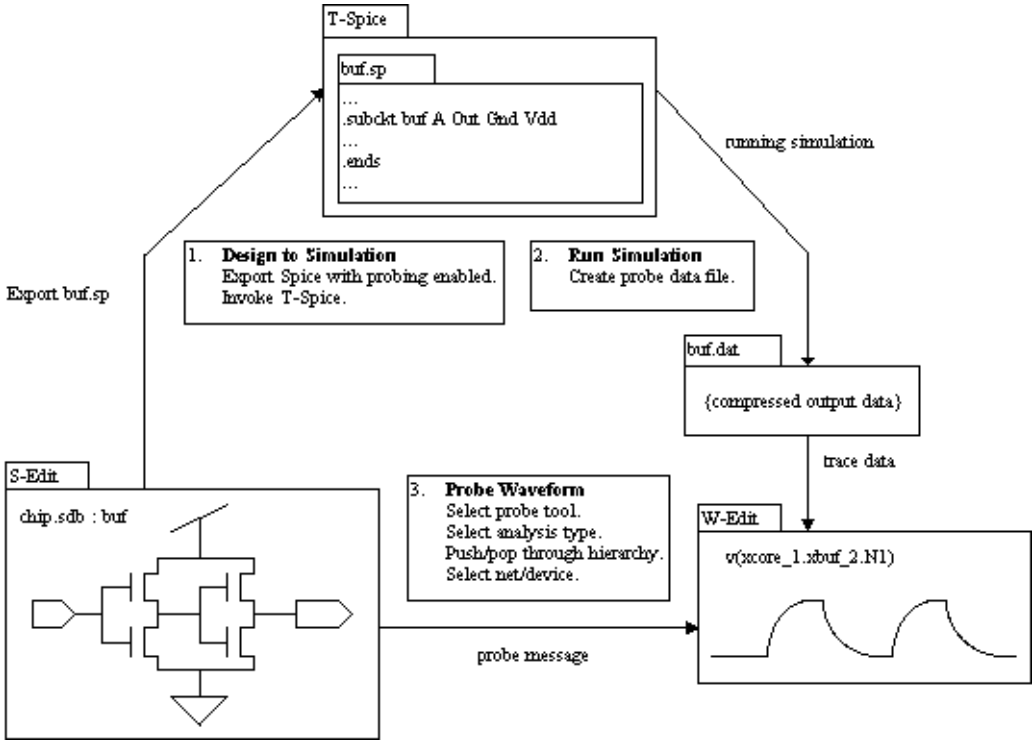
[3] In the *probing* stage, you select a net or device for analysis using a probe tool. When you probe the schematic design, S-Edit invokes W-Edit, which automatically displays the waveforms corresponding to the simulation results for the nodes or devices you probed. .



Using the .probe Command

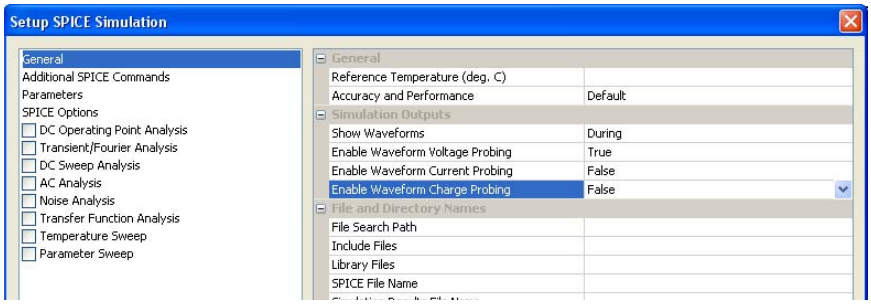
For additional control of waveform probing, you can manually enter a **.probe** command in the SPICE netlist. Several T-Spice options pertaining to waveform probing may also be set with the **.options** command.

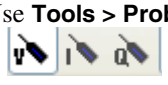
For more information on the **.options** and **.probe** commands, refer to "Simulation Commands" in the *T-Spice User Guide and Reference*



Probing Setup

You must enable at least one of the waveform probing types—voltage, current or charge—for S-Edit to collect probing data during simulation, by setting the appropriate field to "true" on the **General** page of **Setup SPICE Simulation**. (You can also use the Properties window to set these values.)



The behavior of the probe tool depends on the data included in the simulation and the object being probed. Use **Tools > Probe V** to probe nets, wires, net labels and ports. Use **Tools > Probe I** or **Prove Q** to probe device terminals and subcircuits, or click on one of the probe  icons to start probing.

Probing for Properties Unique to an Instance—"In Context" Values

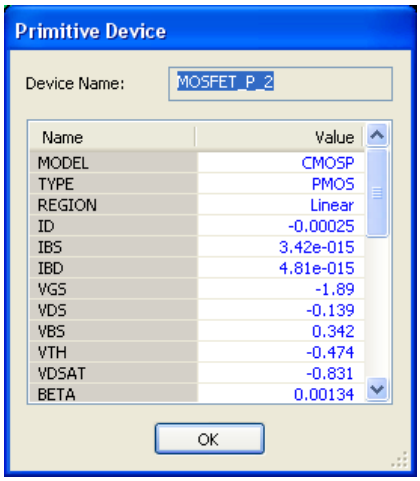
You can use the **Push into Context**  button, or double-click using a **Probe** tool, to open a specific instance. (This operation will yield no results if a cell is not an instance.)

Depending on the object and type of analysis, when you push to deeper levels of the design hierarchy to probe within the subcircuit you can see, for example, small-signal parameter values, property values derived from expressions, or operating-point voltages.

When you reach a primitive cell

Use the **Pop Context** icon  to “pop” out of the last instance you pushed into until you return to the top level of the cell hierarchy.

While editing “in context” you can only select or edit objects contained in the top-level instance. You can continue to push down to lower levels within an instance until you reach a SPICE primitive.



Index

SYMBOLS

A

B

C

D

E

F

G

H

I

K

L

M

N

O

P

R

S

T

U

V

W

Z

Credits

Software Development

David Cardoze
Radoslav Getov
Nikita Jorniak
Massimo Sivilotti
Barry Dyne
Alexander Ivanov
Dan'l Leviton
Nicolas Williams

Quality Assurance

Luba Gromova
Lena Neo

Documentation

Judy Bergstresser
Ken Van de Houten
Barry Dyne