Simulation Standard

TCAD Driven CAD

A Journal for CAD/CAE Engineers

Inter-Tool Communication within *CAD CELEBRITY*

Introduction

One of the main features of modern complex Computer-Aided Design Systems is the ability to support the communication between concurrently executing applications within the design process. This feature allows applications, intended for different steps in a design flow, to exchange design data in an integrated manner and therefore to solve design tasks more accurately, easily and quickly. At the same time these applications keep a high degree of modularity and functional independence.

Using Inter-Tool Communication CAD Systems can solve such tasks as:

- Cross-Probing, sometimes named Cross-Highlighting, - the possibility to highlight images of a selected object in different applications. The CAD system may highlight several aspects of design object description, including:
  - textual description in HDLs;
  - schematic view;
  - layout view;
  - wave form diagrams, etc.
- Static Back-Annotation - that an application at some time in the design process may request some design information (e.g. transistor parameters) from another application, receive this information, and display it by its own methods.
- Dynamic Back Annotation - two or more applications establish a direct connection between their sub-processes through Inter-Tool Communication and dynamically (during a long period of time) exchange design data. An application which receives data uses it only for displaying in its windows.
- Driving Design Process - manipulations in one application cause actions in another application. An example of such communication is schematic driven design of a layout.
- Managing Design Process - requests to do something or to send some information. This means that in the CAD System there is a special application which manages the process of design development, i.e. it controls the sequence of design procedures and the presence of files needed for successful completion of procedures.

All these tasks will be solved within CAD ***CELEBRITY***. This paper will cover only the basic principles of Inter-Tool Communication and Cross-Probing.

Main Concepts

The main concept of Inter-Tool Communication implemented within CAD ***CELEBRITY*** is a client-server approach based on Socket of message exchange. Sockets were selected as a basic mechanism due to their advantages over other mechanisms of Inter-Process Communication (pipes, shared memory, signals, etc.). Socket's Main features are:

- Providing a two-way communication path.
- Existing within communication domains.
- Providing access to communications networks such as the Internet.
- Enabling communication between unrelated processes residing locally on a single host computer and residing remotely on multiple host machines.

Continued on page 2....

INSIDE

SILVACO INTERNATIONAL

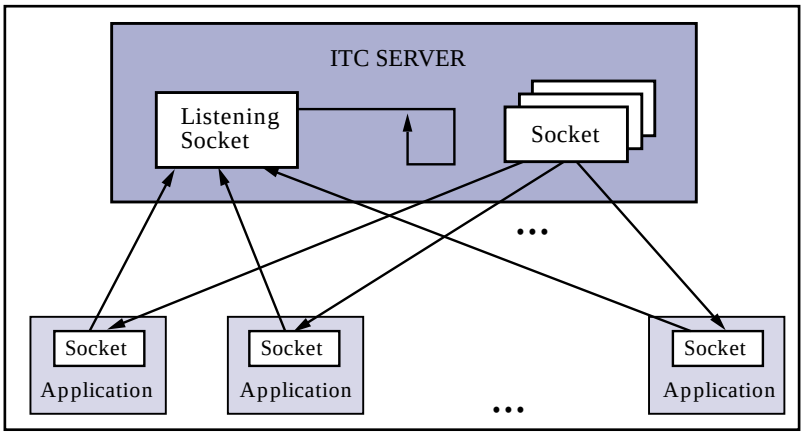

Figure 1. Client-Server approach based on socket mechanism

The implemented client-server approach is schematically shown in Figure 1. In this Figure ITC Server has one listening socket, which is intended for permanent reading from the communication channel. Also it has a list of sending sockets, each of which supports the connection to the appropriate application. Sending sockets are created and added to the list when any application sends a request to connect to the ITC Server.

Each application has a universal socket which provides listening, as well as receiving information, from an ITC Server. These sockets are created before establishing a connection with the ITC Server.

Message Structure. Messages used for Inter-Tool Communication have several attributes. The most important are shown below:

- MessageType - messages may have different types, e.g. Request, Notification, Broadcasting, etc.
- MessageCode - messages may have different codes, e.g. Connect to Server, Command, Information, etc.
- CommandType - if message code is command, CommandType attribute has to be set into one of settled values. Each application has its own set of commands.
- FromApplicationId - the identifier of an application from which this message was sent. This attribute is used by the Design Manager for understanding identifier which sends the application.
- ToApplicationId - the identifier of application to which this message was sent to. This attribute is used by ITC Server for understanding the destination of a message. The Message would be sent only to the a specified application.
- FromApplicationType - the type of application from which this message was sent. This attribute is used by Design Manager and ITC Server for understanding the type of sending application.
- ToApplicationType - the type of application to which this message was sent to. This attribute is used by ITC Server for understanding the destination of message. If ToApplicationId is not set the message would be sent to all applications of the given type.
- Text - any textual information.

Message processing contains two parts: sending and receiving. They are shown schematically in Figure 2.

Sending. The user may make some action, for example clicking the left mouse button on some object in the schematic drawing. The program processes this action, i.e. it finds an object whose bounding box contains the point with cursor coordinates. Then it displays the results, e.g. highlighting the selected object. If the cross-probing mode is established for the current window, the program will form the appropriate message and send it to ITC Server.

Receiving. Application receives messages from ITC Server through socket. The message is decoded, i.e. the application analyzes all of its attributes: message type, message code, application, etc. (see above, section describing message structure). After decoding, the application makes the decision and processes the appropriate action(s), e.g. finding objects which it has to highlight and then displaying the results.

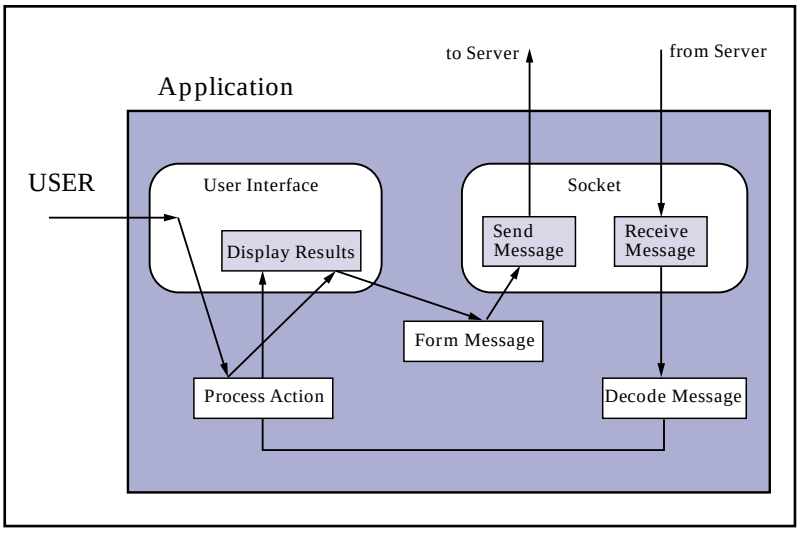

Figure 2. The mechanism of data sending/receiving

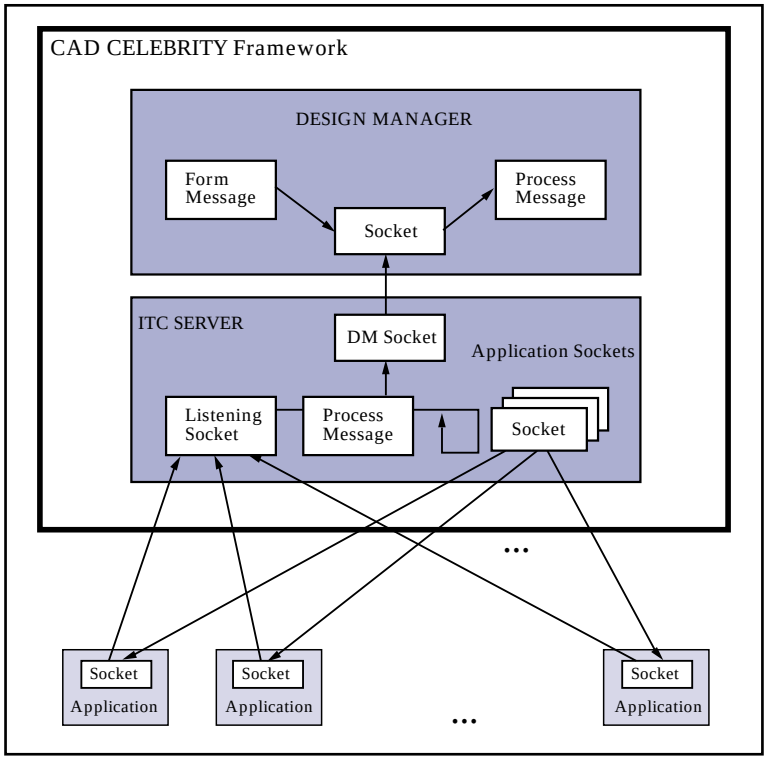

Figure 3. ITC structure within CAD CELEBRITY Framework

Cross-Probing

In this section we will consider cross-probing as one of the tasks solved with the use of Inter-Tool Communication. Cross-Probing within ***CELEBRITY*** is implemented between three applications:

- ***Scholar*** - schematic editor;
- ***Expert*** - layout editor;
- ***SmartSpice*** - simulation tool.

Before any communication can start, the user should run the ITC Server. If the user did not put the channel number as the starting parameter, the program will display the dialog window, shown in Figure 4. All applications which plan to communicate through the appropriate ITC Server should use this channel number.

After running, ITC Server minimized and its icon (see below) appears in Window's status bar.

The architecture in Figure 2 shows the general approach for Inter-Tool communication. Since ***CAD CELEBRITY*** has a framework intended for managing the design process, the general structure was improved (see Figure 3). In this structure ITC Server plays the key role for delivery of messages (requests and notifications) to appropriate applications. The main controlling module within framework, named Design Manager, from the ITC point of view, acts as all other applications it sends and receives messages. But according to its special controlling role Design Manager has a special place within the ITC architecture. In Figure the socket, intended for supporting communication between ITC Server and Design Manager, is separated from sockets of other applications.

All applications within ***CAD CELEBRITY*** can be automatically connected to the ITC Server. For this purpose there is a file, named crossprobing.ini, which contains all necessary information including: host name, where ITC Server was run channel number, flags for automatic run etc. If the flag for the source application is set (=1), it will be automatically attached to the ITC Server.

After running applications, the user should open the required files (e.g; <name>.schlr in ***Scholar***) and in the appropriate window set the cross-probing mode.

In order to set the cross-probing mode within ***Scholar*** the user should select sub-item "Cross-Probing" from menu item "Show" (see Figure 5) or press the button with a bulb picture (see Figure 6).

Figure 5. Sub-items of Show menu item.

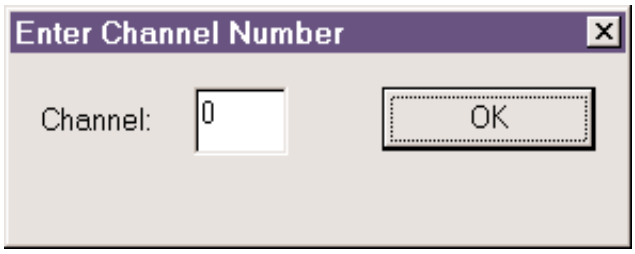

Figure 4. Dialog for Channel number setting

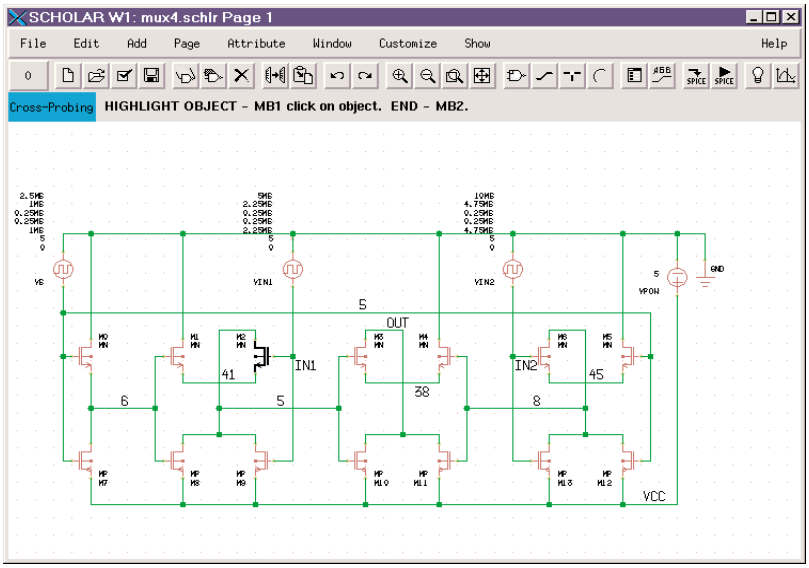

Figure 6. Cross-Probing mode within Scholar's Drawing window

Within the ***Expert*** layout editor, the user can set the cross-probing mode for the whole cell, and thus all of its views can send and receive messages concerning selecting and highlighting of objects. To set the cell into the cross-probing mode, the user should push the button with a white bulb picture. After that, the bulb on the button will change its color to green. This is the signal for the user that a window was set to cross-probing mode. If the color of the bulb was already green, and the user pushes this button, the program will reset the cross-probing mode for this cell. To reset cross-probing mode for all cells in the current project, the user should push the button with three bulbs. If bulbs on this button are white, there are no cells in the cross-probing mode.

There is one restriction for cross-probing with the layout editor. It can find and highlight objects only on layouts which were processed by the procedure of the netlist extraction with checked "Annotate Layout" flag. This procedure flattens hierarchical layout and generates names for transistors and nets.

When ***Scholar's*** drawing window enters into Cross-Probing mode, it changes the shape of cursor to "hand" image. The hint line under the toolbar describes possible operations of this mode. An exit from this mode can be made by pushing the <Escape> button or by pushing any button on toolbar, by selecting any other menu item, or by pressing the middle mouse button.

Within this mode the user can select objects by clicking left mouse button on the required image, and a message with appropriate information about the selected object will be sent to the ITC Server. ITC Server retranslates this message to all applications involved in cross-probing. Therefore all applications of the predefined types connected to the ITC Server will receive requests to highlight objects with the given type and name. The application which receives such a message finds out the appropriate object and highlights it on all child windows, which are set into cross-probing mode.

For example, when the ***Expert*** layout editor receives a request to highlight the transistor, selected in the ***Scholar*** schematic editor, it finds an appropriate object and highlights all polygons constructing this transistor, see Figure 7. In the figure transistor M2 corresponds to an orange rectangle. For better visibility it was filled with inclined lines. In real application the color and pattern of highlighting objects are up to the user's preferences.

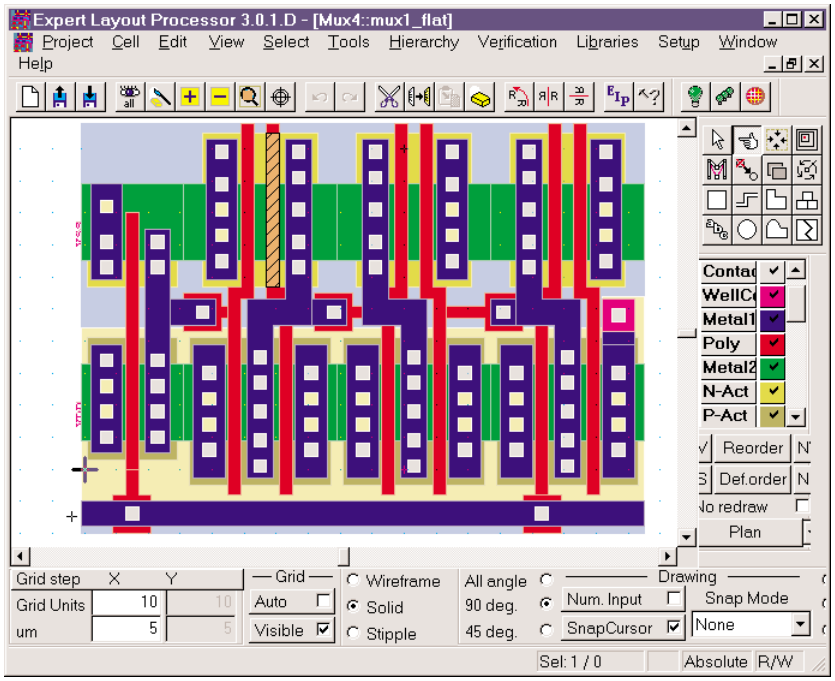

Figure 7. Cross-Probing mode within ***Experts***' Cell layout window.

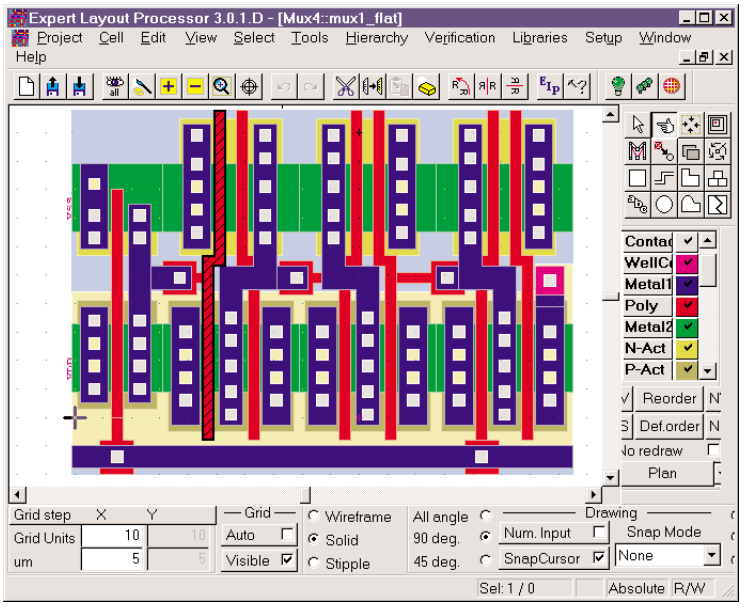

Figure 8. Selecting polygon for cross-probing

one net belongs to several pages of one drawing, all of its parts will be highlighted. The user can go through all the pages of analyzed drawings and see all parts of the selected net.

If ***SmartSpice*** is also connected to an ITC Server the user can see highlighted curves on plot windows for objects selected in ***Scholar's*** drawing or in ***Expert's*** cell view. Figure 10 shows the result of selection transistor "M2" in ***Scholar*** (see Figure 6) within the cross-probing mode. Also it is useful to see highlighted objects in schematics and / or in layout by clicking on a curve in ***SmartSpice's*** chart.

The same mechanism works in the opposite direction. So, if the user selects any polygon in the layout window, which was previously set to cross-probing mode, ***Expert*** will determine the name of an object to which the geometry of a selected polygon belongs, and sends the request to highlight the object. For example, if the user selects a polygon marked by a black border and filled with inclined lines (see Figure below). ***Expert*** will determine that this polygon belongs to net (node) with name the "6", form the appropriate message, and send it to the ITC Server, which will retranslate it to ***Scholar***.

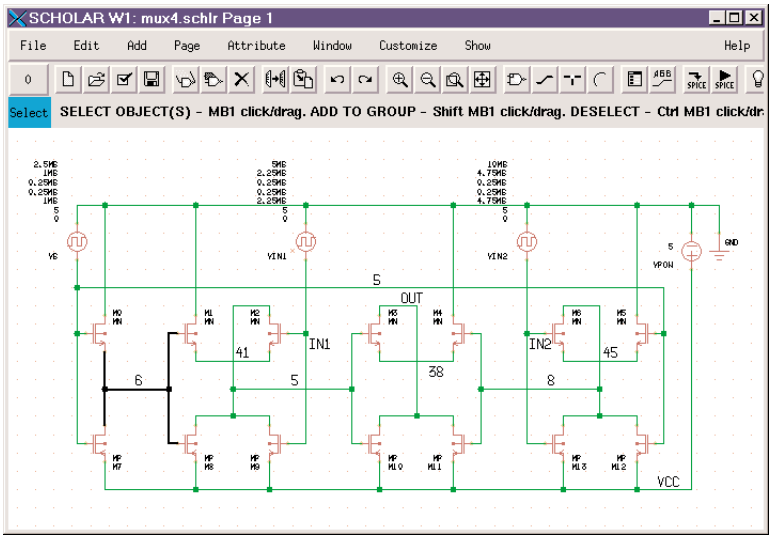

Figure 9. Highlighting nets in Scholar drawing.

Scholar analyzes all drawings, set in the cross-probing mode, and highlight the net with the given name (see Figure 9). If there are several nets with the given name, all of them will be highlighted. Also if

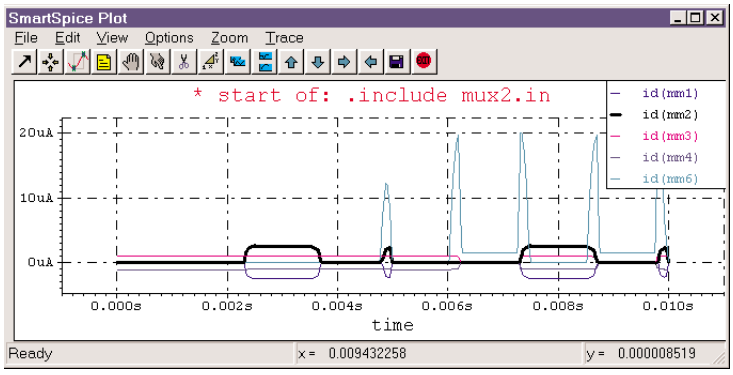

Figure 10. The result of cross-probing in SmartSpice

Conclusion

Inter-Tool Communication within ***CAD CELEBRITY*** is based on server / client architecture and socket mechanism of message exchange. This approach is in conjunction with the innovative structure of communication messages allowing for the possibility to solve all inter-tool communication tasks mentioned in the introduction. Some of them are as follows: cross-probing, dynamic back annotation and design management which has been already implemented in the beta version of the ***CAD CELEBRITY*** Framework.

Also the basic principles of ***CAD CELEBRITY*** Inter-Tool Communication allow to integrate applications concurrently executed on different computers with different platforms.

Scholar - SmartSpice Interface

One of the most useful features of the ***Scholar*** schematic capture is tight integration with the ***SmartSpice*** simulator. In effect, ***Scholar*** makes it possible to automatically generate ***SmartSpice*** netlist on the basis of the schematic. Such a possibility covers both analog and digital designs.

The main features of the interface are:

- ***Scholar's*** library of symbols which makes it possible to generate netlist
- mechanisms of creating a netlist which is oriented especially for the ***SmartSpice*** simulation process

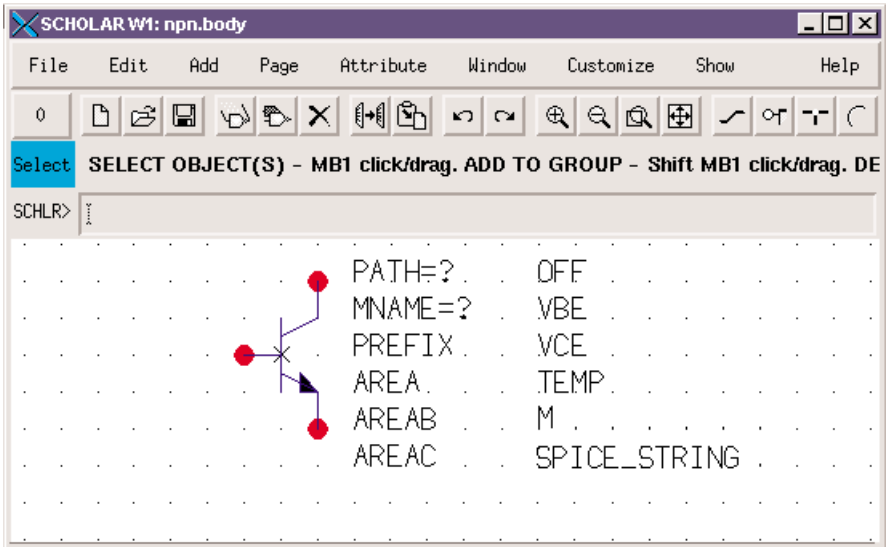

Figure 1. Symbol of npn transistor.

Library of symbols

Scholar's library of symbols consists of three major sets of elements.

- Analog devices
- Digital devices
- Basic ***Scholar*** schematic symbols

The main advantage of ***Scholar's*** library is the complete set of elements that correspond to ***SmartSpice*** devices. They are the following:

- Current sources - idc.body, cccs.body, cccspoly.body, cccspwl.body, ccs.body, ccvs.body, ccvspoly.body, ccvspwl.body;
- Voltage sources - vccs.body, vccsbhv.body, vccspoly.body, vccspwl.body, vccstbl.body, vcs.body, vcvs.body, vcvsbhv.body, vcvspoly.body, vcvspwl.body, vcvstbl.body, vdc.body, vexp.body, vpl.body, vpulse.body, vpwl.body, vpwlfile.body, vpwlfiledesc.body, vpwlfiledesc2.body, vsffm.body, vsin.body,
- Transistors - njfet.body, njfet4.body, njfet_b.body, nmes.body, nmes4.body, nmes_b.body, nmos.body, nmos4.body, nmos_b.body, npn.body, npn4.body, npn_b.body, pjfet.body, pjfet4.body, pjfet_b.body, pmos.body, pmos4.body, pmos_b.body, pnp.body, pnp4.body, pnp_b.body,
- Other devices - cap.body, diode.body, ind.body, res.body, bcs.body, bvs.body, ltl.body, tline.body, txl.body;

Additionally, the library includes:

- logical symbols – and2.body, inv.body, nand2.body, nor2.body, or2.body;
- special schematic symbols – attributes.body, biflag.body, frame.body, frame_s.body, frame_b.body, gnd.body, gnd_2.body, inflag.body, outflag.body, pageflag.body, parameters.body, pwr.body.

All elements have clear graphical representations that are similar to those used in popular schematic capture tools. An example of a library element is shown in Figure 1.

Generation of Netlist Statements

The mechanism of the ***SmartSpice*** netlist generation is based on the use of Scholar's symbols' attributes.

As is described in [1] ***Scholar*** makes it possible to attach attributes to symbols. The attributes of the terminal library elements have the same names and meanings as corresponding components of ***SmartSpice*** statements.

The algorithm of the ***SmartSpice*** statement generation are embedded into a library element. It is based on the use of these attributes. In particular, the values of attributes are placed in ***SmartSpice*** statements. Therefore, in order to create a correct netlist a user need only to set the attribute values.

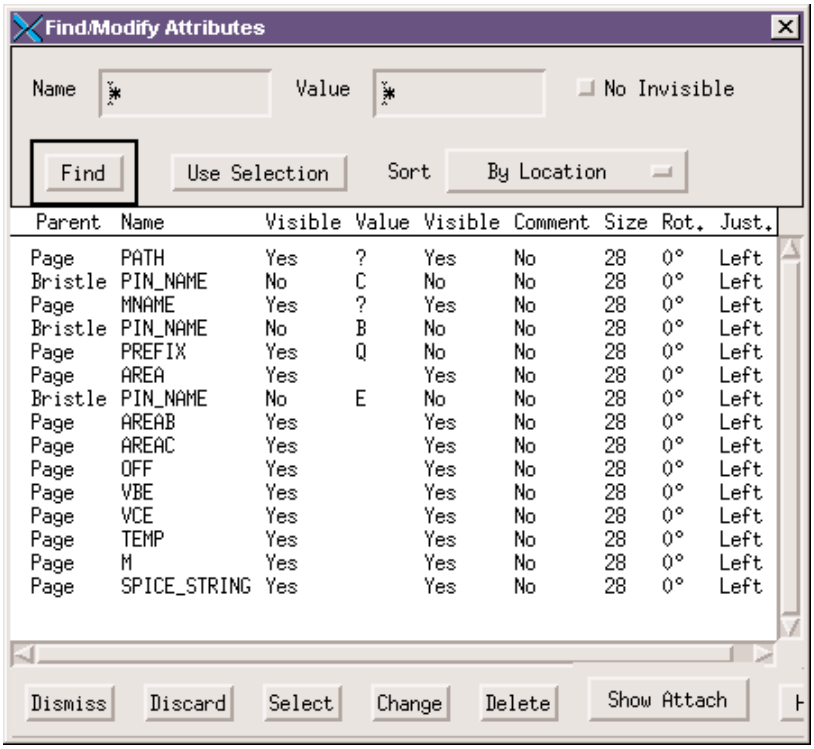

Figure 2 Find/Modify dialog.

The resulting name of the element is formed using ***SmartSpice*** prefix, just as described in the ***SmartSpice*** *User's Manual*, and the original name is taken from the drawing.

Scholar provides the user with a number of possibilities concerning attribute modification. For example, a user can modify the values of attributes using ***Scholar*** Find / Modify dialog. An example of it is shown in Figure 2.

A user has the ability to change not only the values of attributes, but also their visibility and placement on a drawing. It can be useful in order to make a drawing more clear and readable.

According to the ***SmartSpice*** statement description some attributes are mandatory but others are optional. ***Scholar*** sets the values of mandatory attributes to "?". If the user forgets change the values of attributes ***Scholar*** will remember to set such values. It writes an error message to the session window during the checking drawing operation.

We just described the way that ***SmartSpice*** statements are created automatically. Another way is the use of the user defined statements attached to a symbol. Each element of the ***Scholar*** library has an additional attribute, named SPICE_STRING. It's value appears in the end of spice statement, if it is not empty. This attribute is reserved for special users' purposes. The main one is passing a user-defined statement into a netlist. It also can be used to add a comment at the end of a statement.

Let's consider an example of library element.

The library has the number of elements that cover BJT devices. They all have the same ***SmartSpice*** statement, which is the following:

```
Qxxx nc nb ne <ns> mname <area>
<OFF> <IC=vbe, vce>
+ <TEMP=val> <M=val> <DTEMP=val>
```

Three of them (npn.body, npn4.body, npn_b.body) are n-p-n transistors, and another three elements (pnp.body, pnp4.body, pnp_b.body) are p-n-p transistors.

One of them is a n-p-n transistor that has three pins (shown on Figure 1). A second kind of transistor has four pins. One pin is a bulk node, which can be manually connected to any net on the drawing. Another kind of n-p-n transistor can be useful when a user wants to set the bulk node by default. There is ***Scholar's*** environment variable NBULK_NODE, which contains the name of the default net. During the writing of the netlist ***Scholar*** sets all such pins to the default bulk node for n-p-n transistors:

```
set NBULK_NODE=GND
```

This is an example of setting default bulk node from the operating system shell, which the user can include in the ***Scholar*** configuration script.

The resulting ***SmartSpice*** string, for instance of npn_b transistor is the following:

QQ2 NET4 NET5 NET6 GND QNL 1.5 IC=0.6, 5.0

This transistor has the default bulk node, model with the name QNL, emitter area factor 1.5, and initial conditions are "vbe" 0.6 and "vce" 5.0.

Using *Scholar-SmartSpice* Interface

The process of using ***Scholar-SmartSpice*** interface consists of the following.

A user creates a schematic drawing of circuits and subcircuits, consisting of symbols of resistors, inductors, current and voltage sources, other semiconductor devices and other ***Scholar*** schematic symbols such as input / output flags, page connectors and so on. An example of the schematic is shown in Figure 3.

The resulting netlist for the circuit in Figure 3 is:

```
**
* Scholar Spice Netlist Generator
**
* Section name: SCHLR
* Section timestamp: 16:34:24.00
*
* Structdef name: EX1
*
QQ2 NET4 NET5 NET6 QNL
QQ1 NET3 NET2 NET6 QNL
VVEE NET9 GND DC -12
VVCC VDD GND DC 10
CCLOAD NET3 NET4  5PF
QQ3 NET6 NET7 NET9 QNL
QQ4 NET7 NET7 NET9 QNL
RRC1 VDD NET3  10K
RRC2 VDD NET4  10K
RRS1 NET1 NET2  1K
RRS2 NET5 GND  1K
RRBIAS VDD NET7  20K
VVIN NET1 GND DC 0  SIN(0 0.1 5MEG ) AC 1
* End of the netlist
```

Scholar-SmartSpice Interface provides the ability to operate Input Deck, which usually consist of three main parts:

- Netlist;
- Control statements;
- Model descriptions;

Then the user modifies the values of attributes for devices. If a circuit is hierarchical, a user needs to create a generic wire list (gwl) file for all sub-circuits in the design. Also the user needs to prepare a model and control file to finish the creation of the input deck for simulation. Then the user is ready to run ***SmartSpice***, using menu command or the button on the ***Scholar*** toolbar.

Example of Input deck is the following:

```
.include ex1.in
.include ex1.mod
.include ex1.ctrl
.END
```

Also the user can only write a netlist file and use it in their own environment for their own purposes. It can be done from the "Write SmartSpice" Scholar menu command.

Reference

1. Scholar: An Enhanced Multi-Platform Schematic Capture. Simulation Standard, Volume 10, Number 9, September 1999

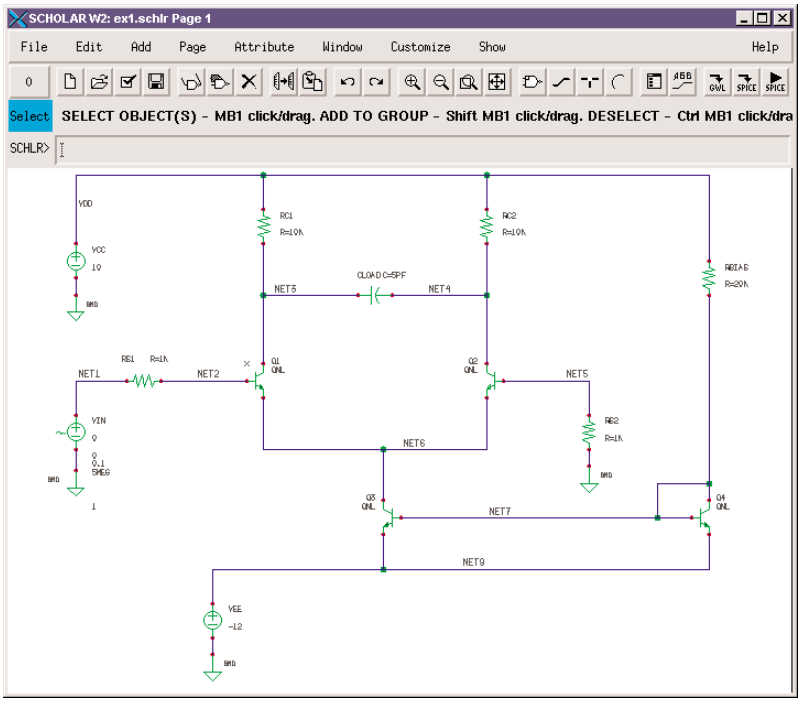

Figure 3. Example of ***Scholar*** Drawing

Calendar of Events

March	April
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8 SLIP 2000 - San Diego, CA
9	9 SLIP 2000 - San Diego, CA
10	10 David Warren's Birthday
11	11
12	12
13	13
14 ICMTS - Monterey, CA	14
15 ICMTS - Monterey, CA	15
16	16
17	17
18	18
19	19
20 GOMAC - Anaheim, CA	20
21 GOMAC - Anaheim, CA	21
22 GOMAC - Anaheim, CA	22
23	23
24	24
25	25
26	26
27 DATE 2000 - Paris, France	27
28 DATE 2000 - Paris, France	28
29 DATE 2000 - Paris, France	29
30 DATE 2000 - Paris, France	30
31	

Bulletin Board

GOMAC

This year's GOMAC re-emphasizes our relevance to military EDA needs. Numerous positive responses from the attendees reaffirms that our reputation for powerful, accurate, and reliable tools precedes us.

DATE 2000

At the largest EDA conference in Europe, Silvaco's Euoropean staff is proud to provide attendees with a choice in TCAD software. Our continued innovation and leadership in the EDA industry is proven by the overwhelming interest in our products.

DAC Comes June 5-7

Coming this June is the largest EDA conference in the US. Visit us at booth #2519, and see our innovative tools. Inquiries will be answered by one of our many engineers and developers attending the Design Automation Conference.

For more information on any of our workshops, please check our web site at ***http://www.silvaco.com***

Hints, Tips and Solutions

Mikalai Karneyenka, Applications and Support Engineer

Q: Usually, when I want to change the display of a drawing in the Drawing window, I select the Window pull-down menu in order to zoom in/out, pan, and fit. Sometimes, if I need to do these operations, this way is not convenient. What is a more effective way to control displaying?

A: ***Scholar*** supports a number of effective ways to control displaying.

The simplest way is to use the toolbar which in the current ***Scholar*** version, is included in both schematic drawing and symbol drawing windows. Zoom In/Out and Fit functions are placed in toolbar (Figure 1). You can press these buttons as many times as needed to call zoom in/out and fit.

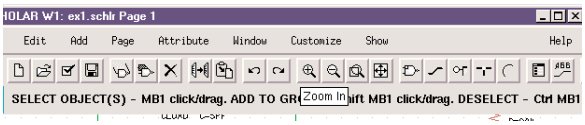

Figure 1. Toolbar.

The disadvantage of these functions is that zooming is carried out according to a zoom factor .

If you prefer to use the mouse to zoom in/out of a particular part of your drawing, the best way is to use Change View functionality. Change View button is also present in the toolbar (Figure 2).

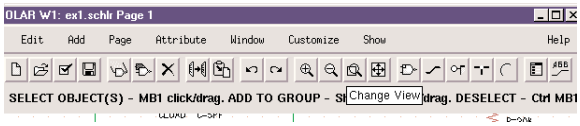

Figure 2. Change View button.

Changing View functions makes it possible to perform complex window commands using a combination of the Shift key and mouse button.

The default mouse buttons while in the Change View command are:

Zoom In - MB1 drag

Zoom Out - MB3 drag

Fit - Shift/MB1 click

Pan - Shift/MB3 click

End Change View mode - MB2 click

It is recommended to use Change View mode commands while performing some other commands such as wiring. If Change View mode is set up while in the wire mode, any windowing operation can be performed. Control is returned to the Wire mode when the window operation is completed by an MB2 click.

If you are in the Select mode, it is also convenient to select Change View using the pop-up menu that is shown in Figure 3. This menu is opened by pressing MB3 in the empty part of a drawing window. Change View button changes mode from Select to change views. So, you can use Zoom, Pan, Fit, and other commands as many times as needed.

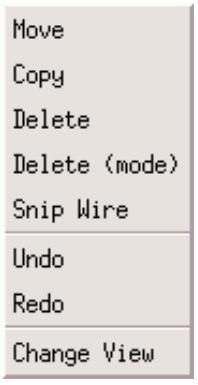

Figure 3. Pop-up menu.

Q: How to create a nonrectangular array of symbols and macroboxes?

Scholar makes it possible to create and represent an array as the real array of symbols. That may be either a rectangular array or not. You can use Edit=>Arrayed Copy menu item in the following scenarios:

1. Place the symbol to be arrayed. It is considered as a starting element of the symbol instances matrix. Suppose the AND2 symbol is placed in the starting position of an array.
2. Select the Edit=>Arrayed Copy menu item. Arrayed Copy dialog box appears (Figure 4). MB1 click on Constraint button to set up it OFF.
3. In the Arrayed Copy dialog box, enter the desired number of copies and press Enter.
4. Select the starting instance and MB1 click in it. Then, move the copy up and right, and MB1 click. The copies are placed with the same Y and X offsets as specified between the first copy and the original symbol (Figure 5).

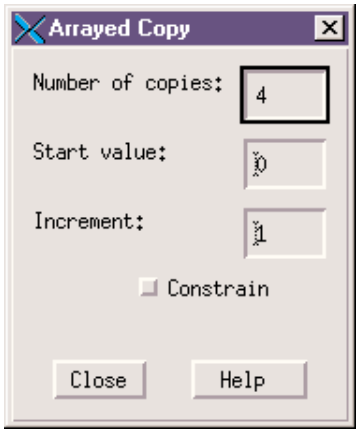

Figure 4. Arrayed Copy dialog box.

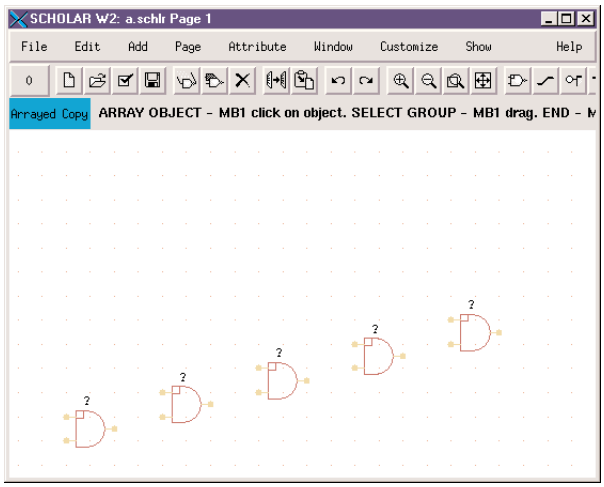

Figure 5. Creation of a non-rectagular array of symbol.

5. Select the entire set of these symbol instances. Enter the desired number of copies in the other direction.
6. MB1 click on the selected set of symbol instances. Their copies appear attached to the pointer.
7. Move these copies and click MB1 to place them. The set of copies will be placed with the same offset specified between the first copy set and the next one (Figure 6).
8. Press MB2 to end the Arrayed Copy command.

The following scenarios can be used to create an array of macroboxes:

1. Create the macrobox definition or definition/instance to be arrayed.
2. Create macrobox instance. Specify pound signs as part of the instance name (for example, MB;INST##). The current macrobox instance is considered as the starting position of the macrobox instances array.
3. Select Edit=>Arrayed Copy menu item. An Arrayed Copy dialog box appears. Enter the desired number of copies. Enter the desired start value that will be formed automatically in the name of first macrobox. Enter the desired increment value that will be used for automatic forming of macrobox names.
4. Select the starting macrobox instance, MB1 click in it. Then, move the copy, and MB1 click. The copies are placed with the same offsets as specified between the first copy and the original macrobox. The first pound sign in each of the copies is replaced with the appropriate number.
5. Select this entire set of macrobox instances. Enter the desired number of copies in the other array direction. Enter the desired number of copies, start value, and increment value.
6. MB1 click on the selected set of macrobox instances. Their copies appear attached to the pointer.
7. Move the copies and click MB1 to place them. The copies will be placed with the same offset specified between the first copy set and the first set of macrobox instances. The second pound sign in each of the copies is replaced with the appropriate number.
8. Press MB2 to end the Arrayed Copy command.

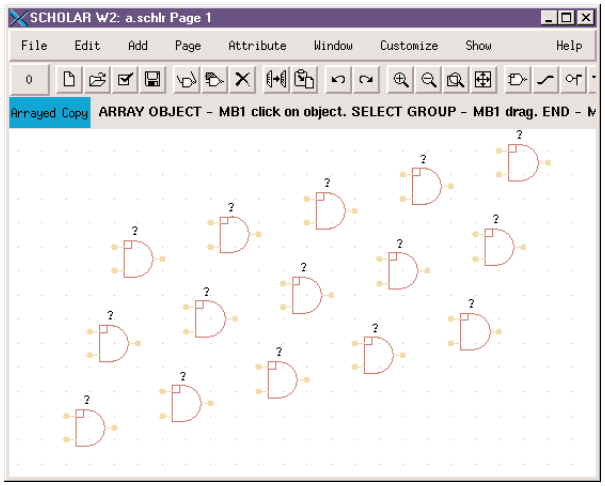

Figure 6. Creation of a set of a non-rectangular array of symbol.

Call for Questions

If you have hints, tips, solutions or questions to contribute, please contact our Applications and Support Department

Phone: (408) 567-1000 Fax: (408) 496-6080

e-mail: *support@silvaco.com*

Hints, Tips and Solutions Archive

Check our our Web Page to see more details of this example plus an archive of previous Hints, Tips, and Solutions

www.silvaco.com

Join the Winning Team!

Silvaco Wants You!

- Process and Device Application Engineers
- SPICE Application Engineers
- CAD Applications Engineers
- Software Developers

fax your resume to:
408-496-6080, or
e-mail to:
jobs@*silvaco.com*

Opportunities worldwide for apps engineers: Santa Clara, Phoenix, Austin, Boston, Tokyo, Guildford, Munich, Grenoble, Seoul, Hsinchu. Opportunities for developers at our California headquarters.

SILVACO
INTERNATIONAL

USA HEADQUARTERS

Silvaco International
4701 Patrick Henry Drive
Building 2
Santa Clara, CA 95054
USA

Phone: 408-567-1000
Fax: 408-496-6080

sales@silvaco.com
www.silvaco.com

CONTACTS:

Silvaco Japan
jpsales@silvaco.com

Silvaco Korea
krsales@silvaco.com

Silvaco Taiwan
twsales@silvaco.com

Silvaco Singapore
sgsales@silvaco.com

Silvaco UK
uksales@silvaco.com

Silvaco France
frsales@silvaco.com

Silvaco Germany
desales@silvaco.com

Products Licensed through Silvaco or e*ECAD

e*ECAD
Vendor Partner