Simulation Standard

TCAD Driven CAD

A Journal for CAD/CAE Engineers

Enhanced LVS Reports in *Guardian* and their Inspection

1. Introduction

It is well known that analyzing LVS reports is often a difficult and brain-teasing task because the "global" nature of connectivity. Unlike DRC reports, where geometrical violations are usually restricted to two shapes, in LVS one must take into account many "widely separated" elements. In addition, quite often a small netlist error will propagate itself in an avalanche manner. Therefore many efforts are directed to localization of LVS errors in ***Guardian*** reports, as well as to means of efficient inspection of these reports.

2. Matching Guesses in *Guardian*

When the ***Guardian*** LVS tool finds out that the compared netlists are different, it tries to inform user about possible (potential) matches among unmatched nodes. We will call a record about one potential match a guess record, or matching guess, or simply guess.

When LVS reports about a pair of unmatched nodes as guess? The heuristic behind this is very simple. A pair of unmatched nodes can be matched if there are matched pairs of their neighbors connected to nodes under investigation by terminals of the same type. Let us illustrate this by example. Suppose device MS1 from the schematic netlist is matched to device ML1 from the layout netlist. Their gates are connected to unmatched nets NS001 and NL001 respectively. Then possibly nodes NS001 and NL001 should be matched, and ***Guardian*** reports the pair of NS001 and NL001 as matching guess.

An additional condition to reporting an unmatched pair of nodes as guess is the unique way of deducing unmatched neighbors as candidates for match. Suppose two devices MS1 and ML1 are matched and for each of them both their source/drain terminals (NS002, NS003 from schematic, NL002, NL003 from layout) are unmatched nets. It is impractical to enumerate all possible matches: NS002-NL002, NS002-NL003, NS003-NL002, NS003-NL003 and hide "good" guesses in report among all these "weak" guesses. It is especially true for neighbors of matched nets with a large number of connections e.g. power/ground nets, synchronization signal nets.

Matching guess can be deduced from a number of sources. For example, let MS2 and ML2 be two unmatched devices from schematic and layout respectively, and let their gates be connected to a matched pair of nets NS004 and NL004. One of their source/drain terminals NS005 and NL005 are matched also. So we can come to MS2-ML2 guess from two different sources. For a given guess we will call the number of guess "sources" as guess confirmation number. Note here that not every confirmation of guess can generate guess. Actually, some of them can be "weak" guesses.

When the confirmation number for a guess is very close to the overall number of connections for the given node, it is a strong indication that the guess nodes will be matched after error correction. When the confirmation number is small in relationship to overall number of connections, then it is likely to indicate a reason for the matching error.

Continued on page 2....

INSIDE

SILVACO INTERNATIONAL

3. Guess contradictions

Let us consider different neighboring situations for a given guess pair of nodes A (schematic) and B (layout). Suppose C is a neighbor of A and connected to A by terminal of type T.

1. Node C can be the unmatched node.
2. Node C is matched to some node D in the second netlist and node D is a neighbor of node B via a terminal of type T. This is exactly the guess confirmation case.
3. Another situation is that D is not connected to node B by terminal of type T.

Note here that when C is a neighbor of B, we will have 3 symmetrical situations.

Situation 1 cannot give us much information about our guess. The current version of Guardian does not try to investigate "guesses of second level". Situation 2 is a confirmation of guess under consideration. And if some neighbor of one node of the guess pair gives us situation 3, then we will call it a guess contradiction.

Let us illustrate guess contradiction by a couple of examples.

Below are excerpts from an LVS unmatch report:

```
DISCREPANCIES AND POTENTIAL MATCHES:
  neighboring stats legend:
    overall connections netlist #1/netlist #2 -
    unmatched connections #1/#2 -
    matched connections contradict to guess #1/#2 -
    guess confirmations

L0: N:   top:130  . . . . . . . . . . . >>
top:249
  stats: 288/4 - 0/0 - 284/0 - 4

L0: N:   top:130  . . . . . . . . . . . >>
top:248
  stats: 288/286 - 0/0 - 4/2 - 284
```

The third pair of numbers in this report shows the number of contradictions to the guess. For the first listed guess (top:130 – top:249) there are 4 confirmations and 284/0 contradictions. That means there are 284 matching neighbors of net top:130 which are a contradiction to the reported guess. In this case all these contradictions are confirmations for another guess (top:130-top:248). On the contrary, 4 confirmations for the first guess are contradictions to guess number two (coming from schematic netlist node). There are 2 more contradictions to the second guess. Guardian did not report the corresponding guess record because it is a "weak" guess for some pair (xxx-top:248).

To further investigate the reason of discrepancies, an LVS user possibly needs an exact list of contradictions for more probable guess (top:130-top:248 in the above example). When it is impractical to list all contradictions after each guess, we believe it is a good idea to report contradictions when the number of contradictions is very small in comparison with the overall number of connections. For the example shown above the report continuation will look like:

```
L0: N:    top:130   . . . . . . . . .   >>
top:248
  stats: 388/386 - 0/0 - 4/2 - 384
  contradictions to this guess listed:
#1[g]      top:M30   . . . . . . .. . .  >>
($)top:M793
#2[sd]    top:M31   . . .  . . . . . .  >>
($)top:M258
#2[sd]    top:M30   .. . . . . . . . .  >>
($)top:M793
#1[g]      top:M31   .. . . . . . . . .  >>
($)top:M258
#1[sd]    top:M43   .. . . . . . . . .  >>
($)top:M254
#1[sd]    top:M42   .. . . . . . . . .  >>
($)top:M792
```

Every contradiction record has a prefix with information about the reason of a contradiction. For example, the first contradiction record prefix #1[g] means that the matching pair M30-M793 comes under consideration because device M30 in the first netlist (#1) is connected by gate ([g]) terminal to net 130. It is possible to further investigate the connectivity information for these devices by the Node Walker tool, called by double clicking on a node name in the unmatched report.

4. Smart Editor

This section describes a new ***Guardian SmartEditor*** used for editing and viewing netlists and report files.

It has the following features:

- Editing of Windows/DOS, UNIX and Macintosh text files
- Highlighting syntax of netlists
- Printing support
- Fixed fonts support
- Auto indent
- Dynamic loading and unloading of file information

SmartEditor views and edits Windows/DOS, UNIX or Macintosh text files. The editor has a system of dynamic loading file information.

At the file opening stage ***SmartEditor*** stores only the information about positions of text lines in file.

The text lines are not stored in the memory and are loaded only if you edit or view them.

When you process huge files and the available memory size becomes not enough, ***SmartEditor*** unloads unseen lines of the file and load the ones that you edit or view at the moment.

This system allows you to edit multi-megabyte files.

Main commands performed by ***SmartEditor*** are in "File" and "Edit" menus (see Figure 1).

SmartEditor allows the user to create new documents and open existing ones by means of "File" menu.

"Save" command of "File" menu is used to save the active document to its current name and directory.

When a document is saved for the first time, ***SmartEditor*** will display the "Save As" dialog box, so that you can name your document. Choose the "Save As" command, if you want to change the name of an existing document before saving.

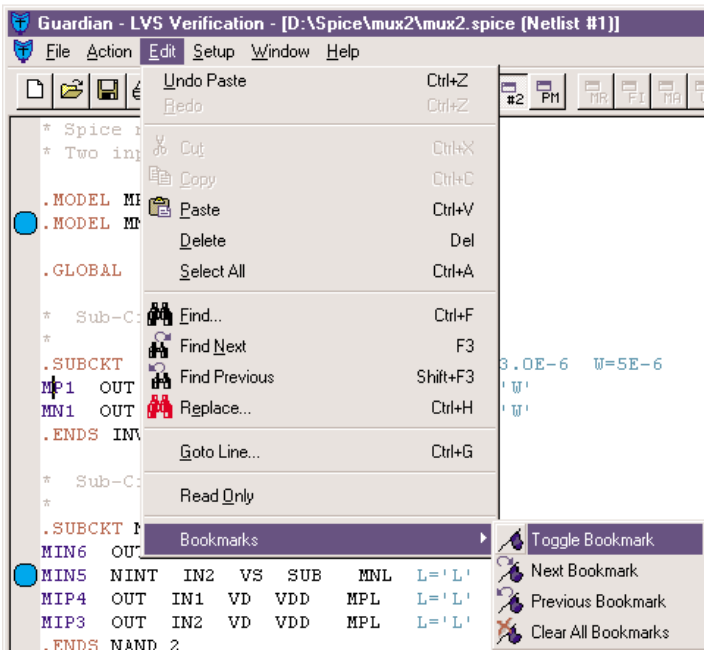

Figure 1. ***SmartEditor*** interface.

Commands in the bottom of "File" menu are used for printing support.

"Page Setup" and "Print Setup" commands allow you to select a printer and specify the size and orientation of pages to be printed, their margins, paper source.

"Print" command prints a document by means of Print dialog box, where the range of pages, the number of copies and other printer setup options can be specified.

"Print Preview" command is used to display the active document as it would appear when printed.

The print preview toolbar in print preview window offers these options:

- View either one or two pages at a time
- Move forward and back through the document
- Zoom in and out of pages
- Initiate a print job

The set of commands in the "Edit" menu allows you to edit the active document.

Operations of editing can be performed in both insert and overstrike modes.

The insert/overstrike mode is changed with the INSERT key and the overstrike mode is indicated in the status bar.

SmartEditor provides automatic line indentation in insert mode, i.e. it automatically indents a line to match the indentation of the previous line. If you select a block of text and press 'Tab' key.

SmartEditor provides block indentation.

"Undo" command is used to reverse the last edit command, including "Redo".

It is unavailable when there is nothing to undo. Command name is changed, depending on what the last action was. The following Undo command types are possible:

- Undo Typing
- Undo Backspace
- Undo Indent
- Undo Drag And Drop
- Undo Replace
- Undo Auto Indent
- Undo Paste
- Undo Delete Selection
- Undo Cut
- Undo Delete

SmartEditor supports multiple steps of undo. Repeated "Undo" commands will attempt to step backwards through the previous commands and undo one at a time.

"Redo" command cancels the last "Undo" command. The same types are allowed for "Redo" command.

The "Undo" and "Redo" commands can be performed by buttons in the ***Guardian*** toolbar.

Next group of commands in the "Edit" menu permit the user to delete data from a document, copy data to the clipboard and paste data from the clipboard.

SmartEditor contains tools for finding a text string in a document and replacing text . The "Find" command from the "Edit" menu is used to specify a search string and the search criteria, such as whether or not to match case.

You can find each occurrence of a combination of any characters, including uppercase and lowercase characters, whole words, or parts of words. You can search the document both above and below the cursor position.

The Commands "Find Next" and "Find Previous" are used to find next and previous occurrence of the text, correspondingly.

"Replace" command allows you to find a given text string in the current window and replaces it with different text. As before, you can replace each occurrence of a combination of any characters, including uppercase and lowercase characters, whole words, or parts of words.

You can replace only an occurrence of the search text or all occurrences at once and can perform replacement in whole document or limit it to the selected area.

The Find and Replace commands are performed by the corresponding buttons in the ***Guardian*** toolbar.

The "Go To" command opens the Goto Line dialog box. This dialog box allows you to jump quickly to a line of the current document. If the line number is greater than the number of lines in the file, the cursor is positioned at the end of the file. You can use "+" or "-" to move relative to the current location. For example, -5 will move backward five lines. The information of cursor position is indicated in the status bar.

To prevent accidental changes in a document you can use the "Read Only" command that toggles the state of the *read only* status of the file. Information about read only status is indicated in the status bar.

SmartEditor allows you to set a bookmark to mark a line in a file.

Once a bookmark has been set at a line, you may use menu or toolbar buttons to move to that line. "Toggle Bookmark" command toggles the bookmark on the line.

"Next Bookmark" and "Previous Bookmark" commands allow users to move to the next and previous bookmarks from current position in a file accordingly.

The "Clear All Bookmarks" command removes all bookmarks.

Bookmark tools are used when you choose the option "Netlist File Highlighting" in "Action upon Double Click" dialog box after you double-click on a name of a device instance or a net in match or unmatch report.

The corresponding Spice file will be open and each line containing this node name will have a bookmark. This node name will be highlighted with a blue color.

You can find the next or previous node name by "Next Bookmark" and "Previous Bookmark" commands in "Edit" menu or corresponding buttons in ***Guardian*** toolbar.

SmartEditor performs parsing and syntax highlighting of netlists; in other words it has an ability to recognize some elements of netlists and display them in different colors. ***SmartEditor*** provides support Spice format and the following groups of elements have different colors:

- Spice commands
- Names of transistors
- Names of diodes, resistors, capacitors, inductors and user-defined elements
- Names of subcircuit instances
- Line comments of two types (line with '*' symbol in the first position and a part of the line starting from ';' symbol)
- Device and subcircuit parameters that have the prefixes 'L=', 'W=', 'R=', 'C=', 'AREA=', 'PARAMS:' and multiplicity factor with the prefix 'M='
- Names of nodes when you perform netlist file highlighting as stated above

Antenna Rule Checks in *Savage*

1. Introduction

Antenna rules is a common name for rules that check ratios of amounts of material in two layers from the same node. They are used to limit the damage of the thin gate oxide during the manufacturing process because of charge accumulation on interconnect layers (metal, polysilicon) on certain fabrication steps.

The name comes from the fact that metal can act as an antenna attracting ions and thus picking up charge during the fabrication process on steps such as plasma etching or ion implantation. This happens when a floating connect layer already connected to gate areas is being fabricated. As a result, the accumulated charge can overstress the thin gate oxide. This leads to the degradation of dielectric breakdown reliability of the device, resulting in degraded performance or even total failure of the device. Antenna effect gets worse in modern technologies, when gate size and thin oxide thickness decrease and at the same time the relative interconnect length increases.

Clearly, antenna rules are based on connectivity information extracted from the layout. This connectivity not necessarily coincides with the one used for LVS netlist extraction. For example, a common way to eliminate antenna effects is to break long interconnects that lead to gates, into segments with safely small area. These segments are then shunted together on subsequent manufacturing steps using short segments of another conductive layer. Therefore antenna checks for each particular interconnect layer necessarily use only partial connectivity information. (Of course, the simplest most conservative approach considers process-induced damage as a cumulative effect, so that one may calculate the total area of poly, metal1, metal2, metal3, ... in a node and relate it to the area of gates connected to this node.)

In a similar way, rules based on relative amounts of material are used to limit the induced capacitance caused by circuit operation, which could degrade device operational characteristics (drive current, transition time).

2. Implementation

To support antenna checks, ***Savage*** provides the following commands:

Select … Stamp, used to transfer connectivity information between overlapping layers

Get_Node_Params, used to calculate specified geometric parameters for shapes in electrical nodes

Check_Node_Params, used to check various constraints for parameters extracted by the preceding Get_Node_Params calls

2.1. Select Operation with Stamp Relation

The command has the following format:

```
SELECT: Relation = STAMP, [Options = (Touch-),]
        Layer1 = <layer1 identifier>,
        Layer2 = <layer2 identifier>,
        LayerR = <result layer identifier>;
```

This operation is used to transfer connectivity information from Layer2 to Layer1 shapes, if possible.

If a polygon from Layer1 overlaps some polygon from Layer2 then the Layer1-polygon is labeled with the same node number as the Layer2-polygon and goes to the output LayerR.

If a polygon from Layer1 overlaps Layer2-polygons belonging to different electrical nodes then the Layer1-polygon is disregarded.

If a polygon from Layer1 does not overlap any Layer2-polygon then the Layer1-polygon is also disregarded.

Options = (Touch-) is used to define exactly what polygon "overlap" means. If "Touch-" option is specified, then shapes that only touch each other are not considered as overlapping shapes. If "Touch-" option is absent, then touching of shapes counts as their overlapping.

Note 1: "Select Stamp" operation does not modify the input Layer1; it generates the output LayerR containing merged Layer1-geometries stamped with connectivity information from Layer2.

Note 2: Output layers of "And" and "Dif" Logical operations automatically inherit electrical node information from the first input layer. Therefore using the "Select… Stamp" operation is not necessary for them.

Example:

```
Select:   Relation=STAMP, Options=(Touch-),
          Layer1=GATE, Layer2=POLY,

          LayerR=&GATE_STM;
```

2.2. Get_Node_Params

The command has the following format:

```
Get_Node_Params: Options = (parameter list),
        Layer = <layer identifier>,
        LayerR = <result file identifier>;
```

This operation is used to extract geometric parameters from input layer Layer for each electrical node and generate the result file specified by LayerR parameter. 'LayerR' parameter is to be used as 'File' input parameter for the Check_Node_Params command.

The 'parameter list' specifies the parameters to extract (Area and Peri (perimeter) parameters are currently supported).

Example:

Get_Node_Params: Options=(Area, Peri), Layer=&GATE_STM,
LayerR=&GATE_PRM;

2.3. Check_Node_Params

The command has the following format:

Check_Node_Params: Formula = (expression),
Value=<value1[:value2]>, Type=<check type>,
Layer = <layer identifier>,
LayerR = <result layer identifier>;

The operation checks geometric parameters extracted by the Get_Node_Params command on each electrical node.

Formula specifies the calculation of a value to be checked for each node.

'Expression' is an arithmetic expression constructed using constants, extracted parameter identifiers, +,-,*,/ binary operators, mathematical functions (***Savage*** currently supports acos, asin, atan, soc, cosh, exp, log, log10, min of two values, max of two values, sin, sinh, sqrt, tan, and tanh) and parentheses.

"Extracted parameter identifier" is in the form: Action.File.Parameter.

"File" is the <result file identifier> from the Get_Node_Params operation used to get this parameter.

"Parameter" is the geometric parameter name (it must be included in "parameter list" of the Get_Node_Params command).

"Action" defines the parameter value to be reported for each electrical node (Min, Max, Sum are currently supported).

"Action" and "Parameter" may be omitted. In this case "Sum" is the default "Action" and "Area" is the default "Parameter".

Layer = <layer identifier> specifies the layer from which the shapes to report are selected.

LayerR = <result layer identifier> specifies the output layer. It will contain the shapes from the input layer "Layer" that belong to the nodes for which the value of Formula satisfies the constraints specified by Value and Type.

Example:

Check_Node_Params:
Formula=(Min.&FPOLY_PRM.Area / Min.&GATE_PRM.Area),
Value=0.25, TYPE=GT, Layer=POLY, LayerR=BAD_POLY;

3. Complete Example

```
// Define input layers
LAYERS:
    POLY (5),
    CNT  (8),
    M1   (9),
    ACT (21);

// Define remaining basic layers
Substrate: LayerR=&BULK;
And: Layer1=ACT, Layer2=POLY, LayerR=&FETGATE;
Dif: Layer1=ACT, Layer2=&FETGATE, LayerR=&DIFF;
And: Layer1=&DIFF, Layer2=&BULK, LayerR=&DFBL;

// Define layer connect sequence from bottom to top
// This command also indicates that connectivity will be reex-
tracted from this point
CONNECT_ORDER: &BULK, &DIFF, POLY, M1;

// Define connections
CONNECT: layer1 = M1,    layer2 = POLY,  layerC = CNT;
CONNECT: layer1 = M1,    layer2 = &DIFF, layerC = CNT;
CONNECT: layer1 = &DIFF, layer2 = &BULK,  layerC = &DFBL;

// Transfer nodal data to gates
Select: Relation=STAMP, Options=(Touch-),
      Layer1=&FETGATE, Layer2=POLY,
      LayerR=&FETGATE_STM;

// Extract gate area per node
Get_Node_Params: Options=(AREA),
Layer=&FETGATE_STM,
            LayerR=GTAREA;

// Extract M1 area per node
Get_Node_Params: Options=(AREA), Layer=M1,
            LayerR=M1AREA;


// Find gates with antenna ratio > 150
Check_Node_Params: Formula=(SUM.M1AREA.Area /
SUM.GTAREA.Area),
              Layer=POLY, Type=GT, Value=150,
              LayerR=&ERM1;
Copy: Layer=&ERM1; // report as violations
```

Expert API

1. Introduction

This article describes ***Expert's Application Programming Interface*** (***Expert API***) based on C/C++ language.

Expert API is intended for customizing and extending the existing software functionality by functions written by the users of ***Expert*** using C/C++.

Expert API is provided in a form of library. It contains numerous API functions that give users very powerful and flexible tools to control, edit and modify ***Expert's*** project data and user interface.

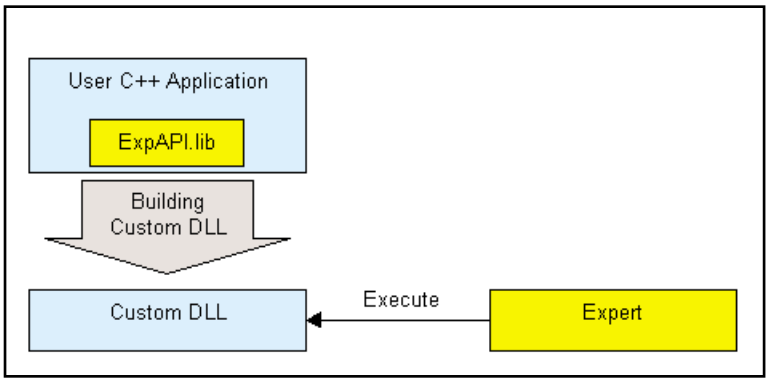

Figure 1. Using Expert API

The library (ExpAPI.lib file) is to be added into a project where users build their plug-in in the form of a Dynamic Linked Library (DLL). The built DLL is to be integrated into ***Expert*** main application.

Good programming experience in C++ is required to use ***Expert API***.

2. Plug-in Creation

The easiest way to start creating plug-ins is to use the example included into the distribution of ***CELEBRITY***. It requires Microsoft Visual C++ v.6.0. The example is complete MS VC++ project for a simple plug-in that creates a new menu item in ***Expert***. You must do the following:

1. Unzip Plugset.zip into any place (PlugSet directory structure will be created there).
2. Run VC++.
3. File >> Open Workspace... select PlugSet\Plug\Plug.dsw
4. Project >> Settings... (Alt-F7)

   Use the following settings for: Win32 Debug and Release:

   Tab "General": Output Files: put \PlugIns subdirectory of the directory where Expert.exe is installed

   (e.g.,. C:\Silvaco\lib\expert\3.4.3.R\x86-nt\PlugIns)

   Tab "Debug": Executable for debug session: Input directory with GuiAppStarter.exe

   (or browse to GuiAppStarter.exe)

   (ex. C:\Silvaco\etc\GuiAppStarter.exe)

   Click OK for close dialog.

5. Builg >> Set Active configuration
   Choose Plug-Win32 Debug (or Release)
6. Build >> Build Plug.dll (F7)
7. Execute (Ctrl-F5)

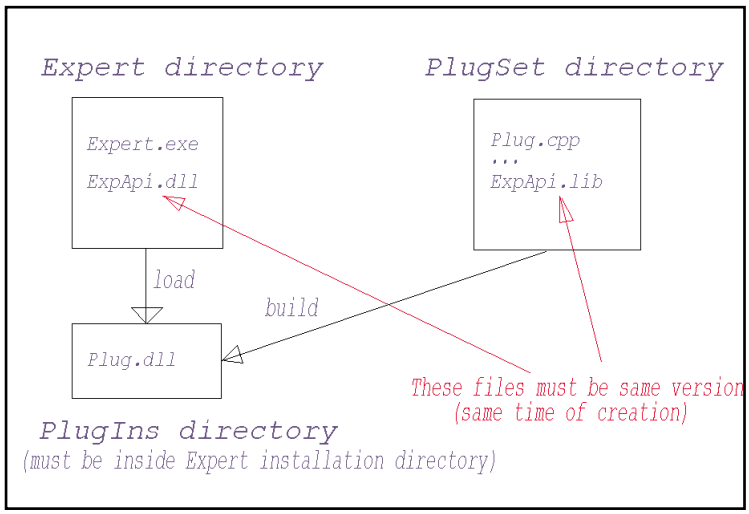

Figure 2. Plug-in Creation

You should see ***Expert*** running, with a new "Plug" menu item.

3. API Description

All ***Expert API*** functions are grouped into the following categories:

3.2 Setup

- 3.2.1: General Setup
- 3.2.2: Setup Technology
- 3.2.3: GDS Setup
- 3.2.4: CIF Setup
- 3.2.5: Applicon Setup

3.3 Project

3.4 User Interface (UI)

3.5 Objects

3.5.1: Objects

3.5.2: Cells

3.5.3: Layers

3.5.4: Primitives

3.6 Tools

3.6.1: Selection Modes

3.6.2: Drawing Modes

3.6.3: Edit Modes

3.7 Support

3.8 Script Language (LISA)

All of these functions have ExpApi prefix.

3.1. Data Types

HOBJECT	– handle to layout object;
HCELL	– handle to cell
HLAYER	– handle to laye
OBJECT_PROPERTY	– structure storing data of layout object;
LAYER_PROPERTY	– structure storing specs of technology layer;
UUNITS	– Enumerated type for measurement units
UNIT_NM	– nanometers (1.0e-3 um)
UNIT_UM	– microns (1.0 um)
UNIT_MIL	– mils (2.54 um)
UNIT_MM	– millimeters (1.0e3 um)
UNIT_CM	– centimeters (1.0e4 um)
UNIT_INCH	– inches (2.54e4 um)
UNIT_M	– meters (1.0e6 um)
UNIT_LAMBDA	– user defined units.
EAPoint	- point class
EARect	– rectangle class
EAPointsArray	– array of points class
GENERAL_SETUP	
TECHNOLOGY_SETUP	
GDS_SETUP	
CIF_SETUP	
APPLICON_SETUP	

3.2. Functions

void	**ExpApiSetUnits**(UUNITS nUnit, double dValue = 0.);
UUNITS	**ExpApiGetUnits**();
Double	**ExpApiGetUnitUMValue**(UUNITS nUnit);
void	**ExpApiXiInit**();
BOOL	**ExpApiXiRun**(LPCTSTR sz);
BOOL	**ExpApiFileSelector**(LPCTSTR szFileName, LPCTSTR lpszFilter = NULL, LPCTSTR lpszDefExt = NULL, BOOL bLoad = TRUE);
void	**ExpApiUpdate**();
void	**ExpApiRedrawView**(CEARect* rect = NULL);
void	**ExpApiRedrawObject**(HOBJECT hObject);
BOOL	**ExpApiProjectNew**(LPCTSTR szProjName, LPCTSTR szTechName = NULL);
BOOL	**ExpApiProjectLoad**(LPCTSTR szProjName, LPCTSTR szOutputProjectName = NULL, LPCTSTR szTechName = NULL, BOOL bReadOnly = FALSE);
BOOL	**ExpApiCloseProject**();
BOOL	**ExpApiSaveProject**();
BOOL	**ExpApiSaveAsProject**(LPCTSTR szProjName);
BOOL	**ExpApiCompressProject**();
CEAPoint	**ExpApiGetMousePoint**(RCEAPoint ptPtev);
CEARect	**ExpApiGetMouseRect**();
BOOL	**ExpApiIsRMousePress**();
void	**ExpApiSetRefPoint**(RCEAPoint pt);
CEAPoint	**ExpApiGetRefPoint**();
BOOL	**ExpApiActivateRefPoint**(BOOL bRefPointActivate = TRUE);
BOOL	**ExpApiSetActiveLayer**(LPCTSTR szLayerName);
LPCTSTR	**ExpApiGetActiveLayer**();
BOOL	**ExpApiSetActiveCell**(LPCTSTR szCellName);

LPCTSTR **ExpApiGetActiveCell**();

void **ExpApiGetObjectProperty**(HOBJECT hObject, OBJECT_PROPERTY& Property);

int **ExpApiSearch**(CEASearchContext& SearchContext, LPCTSTR szCellName = NULL);

HOBJECT **ExpApiGetFoundFirstObject**(LPCTSTR szCellName = NULL);

HOBJECT **ExpApiGetFoundNextObject** (LPCTSTR szCellName = NULL);

HOBJECT **ExpApiGetFoundPrevObject** (LPCTSTR szCellName = NULL);

HOBJECT **ExpApiGetFoundLastObject** (LPCTSTR szCellName = NULL);

BOOL **ExpApiSelectObject**(HOBJECT hObject);

BOOL ExpApiDeleteSelected();

HOBJECT **ExpApiAddObject**(OBJECT_PROPERTY& Property);

HOBJECT **ExpApiModifyObject**(HOBJECT hObject, OBJECT_PROPERTY& Property);

BOOL **ExpApiDeleteObject**(HOBJECT hObject);

HOBJECT **ExpApiGetFirstObject**(LPCTSTR szCellName = NULL, LPCTSTR szLayerName = NULL);

HOBJECT **ExpApiGetNextObject**(LPCTSTR szCellName = NULL, LPCTSTR szLayerName = NULL);

HOBJECT **ExpApiGetFirstInstObject**(LPCTSTR szCellName = NULL, LPCTSTR szInstName = NULL);

HOBJECT **ExpApiGetNextInstObject**(LPCTSTR szCellName = NULL, LPCTSTR szInstName = NULL);

BOOL **ExpApiCreateNewCell**(LPCTSTR szCellName);

BOOL **ExpApiCreateAsNewCell**(LPCTSTR szCellName);

BOOL **ExpApiDeleteCell**(LPCTSTR szCellName = NULL, BOOL bHier = FALSE);

BOOL **ExpApiRenameCell**(LPCTSTR szTargetCell, LPCTSTR szCellName = NULL);

BOOL **ExpApiSaveCell**(LPCTSTR szCellName = NULL, BOOL bCheckInOut = FALSE);

BOOL **ExpApiSaveAsCell**(LPCTSTR szTargetCell, LPCTSTR szCellName = NULL, BOOL bCheckInOut = FALSE);

BOOL **ExpApiDiscardCell**(LPCTSTR szCellName = NULL);

BOOL **ExpApiReadOnlyCell**(LPCTSTR szCellName = NULL);

HOBJECT **ExpApiCreateBox**(double dSx, double dSy, double dEx, double dEy, LPCTSTR szLayerName = NULL);

HOBJECT **ExpApiCreateBox**(RCEAPoint ptPos, RCEAPoint ptSize, LPCTSTR szLayerName = NULL);

HOBJECT **ExpApiCreateBox**(RCEARect rtRect, LPCTSTR szLayerName = NULL);

HOBJECT **ExpApiCreateCircle**(double dSx, double dSy, double dRadius, LPCTSTR szLayerName = NULL);

HOBJECT **ExpApiCreateCircle**(RCEAPoint ptCenter, double dRadius, LPCTSTR szLayerName = NULL);

HOBJECT **ExpApiCreateEllipse**(double dSx, double dSy, double dEx, double dEy, LPCTSTR szLayerName = NULL);

HOBJECT **ExpApiCreateEllipse**(RCEAPoint ptCenter, RCEAPoint ptRadius, LPCTSTR szLayerName = NULL);

HOBJECT **ExpApiCreateEllipse**(RCEARect rtRect, LPCTSTR szLayerName = NULL);

HOBJECT **ExpApiCreateDonut**(RCEAPoint ptCenter, double dRad1, double dRad2, LPCTSTR szLayerName = NULL);

HOBJECT **ExpApiCreateRegion**(CEAPointsArray& Points, LPCTSTR szLayerName = NULL);

HOBJECT **ExpApiCreateLine**(CEAPointsArray& Points, double dWidth, LPCTSTR szLayerName = NULL);

HOBJECT **ExpApiCreateText**(LPCTSTR szText, RCEAPoint ptPos, double dWidth, double dHeight, double dEscapement = 0., BOOL bSlant = FALSE, BOOL bMirroring = FALSE, LPCTSTR szLayerName = NULL);

HOBJECT **ExpApiCreateText**(LPCTSTR szText, RCEARect rtRect, double dEscapement = 0., BOOL bSlant = FALSE, BOOL bMirroring = FALSE, LPCTSTR szLayerName = NULL);

BOOL **ExpApiSetOrigin**(RCEAPoint pt, LPCTSTR szCellName = NULL);

BOOL **ExpApiSetOrigin**(double dx, double dy, LPCTSTR szCellName = NULL);

BOOL **ExpApiSetAngleMode**(DAM daMode);

DAM **ExpApiGetAngleMode**();

BOOL **ExpApiCut**();

BOOL **ExpApiCopy**();

BOOL **ExpApiPaste**(RCEAPoint pt);

BOOL **ExpApiPaste**(double dx, double dy);

BOOL **ExpApiMove**(RCEAPoint pt);

BOOL **ExpApiMove**(double dx, double dy);

BOOL **ExpApiDuplicate**(RCEAPoint pt);

BOOL **ExpApiDuplicate**(double dx, double dy);

BOOL **ExpApiRotate**(double dx, double dy, double dAngle, BOOL bDuplicate = FALSE);

BOOL **ExpApiRotate**(RCEAPoint pt, double dAngle, BOOL bDuplicate = FALSE);

BOOL **ExpApiFlipVertical**(double dCenter);

BOOL **ExpApiFlipHorizontal**(double dCenter);

BOOL **ExpApiMirrorVertical**(double dCenter);

BOOL **ExpApiMirrorHorizontal**(double dCenter);

BOOL **ExpApiSelectNewest**();

BOOL **ExpApiSelectLayer**(LPCTSTR szLayerName = NULL);

BOOL **ExpApiSelectAl**l();

BOOL **ExpApiUnselectAll**();

BOOL **ExpApiInvertSelection**();

BOOL **ExpApiSelReselect**();

BOOL **ExpApiSelSelect**();

BOOL **ExpApiSelDeselect**();

BOOL **ExpApiSelModeInside**();

BOOL **ExpApiSelModeCross**();

BOOL **ExpApiSelModeBoundary**();

BOOL **ExpApiSelModeSolid**();

BOOL **ExpApiSelectByType**(SEARCH_OBJECT_TYPE sType, LPCTSTR szCellName = NULL);

BOOL **ExpApiIgnoreInstances**();

BOOL **ExpApiIgnorePrimitives**();

BOOL **ExpApiMoveToActiveLayer**();

BOOL **ExpApiCopyToActiveLayer**();

BOOL **ExpApiSaveSelectedAsCell**(LPCTSTR szCellName);

BOOL **ExpApiSelectByBox**(double dSx, double dSy, double dEx, double dEy, SELECT_BOOL_MODE selMode = SBM_SELECT_RESELECT);

BOOL **ExpApiSelectByBox**(RCEAPoint ptStart, RCEAPoint ptEnd, SELECT_BOOL_MODE selMode = SBM_SELECT_RESELECT);

BOOL **ExpApiSelectByBox**(RCEARect rtRect, SELECT_BOOL_MODE selMode = SBM_SELECT_RESELECT);

BOOL **ExpApiMergeSelected**();

BOOL ExpApiResizeSelected(double dSize, BOOL bUnderSize = FALSE, BOOL bGrid = FALSE, BOOL bMerge = FALSE);

BOOL **ExpApiMergeWires**();

BOOL **ExpApiGenDLayer**(LPCTSTR szLayerName = NULL);

BOOL **ExpApiGenAllDLayers**();

BOOL **ExpApiCleanCellPrimitivesDLayer**();

BOOL **ExpApiCleanCellHierarchyDLayer**();

BOOL **ExpApiCleanWholeProjectDLayer**();

BOOL **ExpApiCleanCellPrimitivesSLayer**();

BOOL **ExpApiCleanCellHierarchySLayer**();

BOOL **ExpApiCleanWholeProjectSLayer**();

BOOL **ExpApiDeleteTechnologySLayers**();

BOOL **ExpApiCreateNewCellInPlace**(LPCTSTR szCellName, BOOL bInPlace = FALSE);

BOOL **ExpApiExplodeSelected**();

BOOL **ExpApiExplodeAll**();

BOOL **ExpApiFlatten**(LPCTSTR szCellName = NULL);

BOOL **ExpApiReplaceSelectedInstances**(LPCTSTR szCellName = NULL);

BOOL **ExpApiReplaceInstancesInCell**(LPCTSTR szSourceCell, LPCTSTR szTargetCell);

BOOL **ExpApiReplaceInstancesInProject(**LPCTSTR szSourceCell, LPCTSTR szTargetCell);

LPCTSTR **ExpApiGetCellName**(HCELL hCell);

int **ExpApiGetCellNumber**(LPCTSTR szLibName = NULL);

HCELL **ExpApiGetFirstCell**(LPCTSTR szLibName = NULL);

HCELL **ExpApiGetNextCell**(LPCTSTR szLibName = NULL);

LPCTSTR **ExpApiGetLayerName**(HLAYER hLayer);

int **ExpApiGetLayerNumber**(LPCTSTR szCellName = NULL);

HLAYER **ExpApiGetFirstLayer**(LPCTSTR szCellName = NULL);

HLAYER **ExpApiGetNextLayer**(LPCTSTR szCellName = NULL);

LPCTSTR **ExpApiGetCellLayerName**(HCLAYER hLayer);

int **ExpApiGetCellLayerNumber**(LPCTSTR szCellName = NULL);

HCLAYER **ExpApiGetFirstCellLayer**(LPCTSTR szCellName = NULL);

HCLAYER **ExpApiGetNextCellLayer**(LPCTSTR szCellName = NULL);

LPCTSTR **ExpApiGetTechLayerName**(HTLAYER hLayer);

int **ExpApiGetTechLayerNumber**();

HTLAYER **ExpApiGetTechFirstLayer**();

HTLAYER **ExpApiGetTechNextLayer**();

BOOL **ExpApiAddTechLayer**(const LAYER_PROPERTY& Property);

BOOL **ExpApiGetTechLayer**(LPCTSTR szLayerName, LAYER_PROPERTY& Property);

BOOL **ExpDeleteTechLayer**(LPCTSTR szLayerName);

void **ExpApiSetSetupTechnology**(const TECHNOLOGY_SETUP& sTechSetup);

void **ExpApiGetSetupTechnology**(TECHNOLOGY_SETUP& sTechSetup);

void **ExpApiSetSetupGeneral**(const GENERAL_SETUP& sGeneralSetup);

void **ExpApiGetSetupGeneral**(GENERAL_SETUP& sGeneralSetup);

void **ExpApiSetSetupGDS**(const GDS_SETUP& sGDSSetup);

void **ExpApiGetSetupGDS**(GDS_SETUP& sGDSSetup);

void **ExpApiSetSetupCIF**(const CIF_SETUP& sCIFSetup);

void **ExpApiGetSetupCIF**(CIF_SETUP& sCIFSetup);

void **ExpApiSetSetupApplicon**(const APPLICON_SETUP& sAPPLICONSetup);

void **ExpApiGetSetupApplicon**(APPLICON_SETUP& sAPPLICONSetup);

BOOL **ExpApiSaveSettings**(LPCTSTR szFileName);

BOOL **ExpApiLoadSettings**(LPCTSTR szFileName);

HOBJECT **ExpApiCreateBox**(double dSx, double dSy, double dEx, double dEy, LPCTSTR szLayerName = NULL)

HOBJECT **ExpApiCreateBox**(RCEAPoint ptPos, RCEAPoint ptSize, LPCTSTR szLayerName = NULL)

HOBJECT **ExpApiCreateBox**(RCEARect rtRect, LPCTSTR szLayerName = NULL)

Purpose: Creates a Box

3.3 Example Description of Functions for Box Creation

double dSx, double dSy – initial position of Top Left corner;

RCEAPoint ptPos – initial position of Top Left corner;

double dEx, double dEy – width, height;

RCEAPoint ptSize – width, height;

RCEARect rtRect - initial position of Top Left corner, width, height;

LPCTSTR szLayerName – name of the layer. New object will be created in a current layer if layer name is not given.

Example:

HOBJECT hObject = ExpApiCreateBox(0, 0, 10, 100, "Poly");

Calendar of Events

March

1 CLEVER W/S - Scottsdale, AZ
2
3
4 DATE - Paris, France
5 DATE - Paris, France
6 TCAD W/S-Cambridge University
DATE - Paris, France
7 DATE - Paris, France
8 DATE - Paris, France
9
10
11 GOMAC - Monterey, CA
12 TCAD /CAD W/S- Fairfax, VA
TCAD W/S-Cambridge University
GOMAC - Monterey, CA
13 GOMAC - Monterey, CA
14 GOMAC - Monterey, CA
15 ATLAS W/S - Scottsdale, AZ
16
17
18
19
20 Int'l Conf. for Microelectronic Test Structures - Cork, Ireland
21 Int'l Conf. for Microelectronic Test Structures - Cork, Ireland
22 Int'l Conf. for Microelectronic Test Structures - Cork, Ireland
23
24
25
26
27
28 Japan Applied Physics - Aoyama Gakuin University
29 Japan Applied Physics - Aoyama Gakuin University
30 Japan Applied Physics - Aoyama Gakuin University
31 Japan Applied Physics - Aoyama Gakuin University

April

1
2
3
4
5
6 SLIP 2002 - San Diego, CA
7 SLIP 2002 - San Diego, CA
8 GaAs MANTECH -San Diego, CA
9 GaAs MANTECH -San Diego, CA
10 GaAs MANTECH -San Diego, CA
11 GaAs MANTECH -San Diego, CA
12
13
14
15
16
17
18
19
20
21
22 6th Int'l Wrkshp on Fab, Characterization and modeling of Ultra-Shallow Doping Profiles in Semicon - Napa Valley, CA
23
24
25
26
27
28
29
30

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

Bulletin Board

See You in New Orleans

Silvaco International will be showcasing its extensive suite of TCAD tools this year at DAC in New Orleans, Louisiana. Silvaco has become the industry standard for ***TCAD Driven CAD*** tools throughout the world by providing hundreds of leading companies with tools that increase productivity, accuracy, and ease of use. Silvaco is setting the pace in the EDA industry for technology and support. Come see Silvaco at booth #2917 to meet with our seasoned engineers and software developers, and they will demonstrate how Silvaco provides "*the right tools for the job*".

Good Day in Monterey

This year's GOMAC (GOvernment Microcircuit Applications Conference) provided an opportunity to demonstrate our latest solutions for the Military and commercial markets. We also demonstrated on-going partnerships with Dynamics Research Corporation, NAVSEA Crane, and Vanderbilt University in developing a full 3D RadHard TCAD suite for the U.S. Navy's Strategic Systems Programs. For the first time Silvaco publicly demonstrated it's next generation TCAD simulator aptly named ***VICTORY***, running on a Linux cluster using MPI parallelization.

Silvaco Expands European R&D

Following the successful opening of our Grenoble Research Center, Silvaco has decided to continue with its aggressive R&D expansion in Europe. A large site has been selected for constructing a 28,000 sq./ft. facility near Cambridge University. Construction is scheduled for completion in December of 2002. Look for construction updates in *Simulation Standards* to come.

Hints, Tips and Solutions

Mikalai Karneyenka, Applications and Support Engineer

A simple way to add custom commands to ***Expert's*** menu is implemented. It is possible to add menu items that execute xi-commands or procedures.

To do this, you must put a file named 'custom.mnu' with syntax described below into ***Expert's*** executable directory and start (or restart) Expert.

Below is an example custom.nmu file. This file may be found in the "Examples>>Expert" folder of the ***CELEBRITY*** installation.

```
-----------------------------------------------------------------
10 >  Deb&ug
      New &Box           = box 0 0 100 100; zoom;
      -----------
      Get &point      = get_point();
      > &Extra
       &Wordpad      = spawn("C:/WinNT/notepad.exe") /nowait;
       Wordpad &2    = spawn("C:/WinNT/notepad.exe");
       -----------
       &Box big      = box 0 0 1000 1000; zoom;
       > &Point
         Get &point       = get_point();
         &Box3       = box 10 10 100 100; zoom;
       <
      <
      &Two Boxes      = box 0 0 100 100; box 50 50 100 100; zoom;
    <
11 >  C&ustom
    &Box             = box 0 0 100 100; zoom;
    -----------
    Get &point       = get_point();
    <
12 >  Test1
    &Pmos        = layer P-Act; box -2 -1 5 3; layer Poly; box -0.5 -2 2 5;
    Pmos&2       = box -2 -1 5 3 /layer="P-Act"; box -0.5 -2 2 5 /layer="Poly";
    Pmos&3           = box;
    <
-----------------------------------------------------------------
```

Syntax: The syntax will be explained using the previous example.

"10", "11", "12" are the positions in ***Expert's*** main menu for new submenus.

The > character marks the beginning of a submenu.

The < character marks the end of a submenu.

As you can see, submenus may be nested.

"Debu&g" shows how to define the name (Debug) of a submenu and its hotkey (G).

`"New &Box    = box 0 0 100 100; zoom;"` This line defines a submenu command. Here, as above, "New &Box" defines the name and the hotkey for the command. It is separated from the actual command by the equal sign. You may put here several xi-commands separated by semicolon, as in:

```
&Box3 = box 10 10 100 100; zoom;
```

The line

```
&Wordpad    = spawn("C:/WinNT/notepad.exe") /nowait;
```

is an example of calling other Windows programs from ***Expert***.

More advanced customization of ***Expert's*** User Interface and functionality is possible via ***Expert API***.

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