Simulation Standard

TCAD Driven CAD

A Journal for CAD/CAE Engineers

Design Version Control System for *Expert*

Introduction

The DVCS subsystem of ***CELEBRITY*** CAD suite provides significant improvement in management of complex designs, design reuse and teamwork management. The most important features of DVCS are the means for version tracking and team design coordination. Projects within DVCS follow the client-server architecture. The shared project files are saved into the Repository. The Repository is a special DVCS database controlled by the DVCS server process. When sharing a project between two or more developers, the Repository enables the task to be accomplished quickly and efficiently. When adding a file to DVCS, the file is backed up in the Repository and made available to other people. Changes that have been made to the file are saved, however, old versions can be recovered at any time. Members of a team can access the latest version of any cell, make changes, and save a new version of the cell in the database. DVCS project organization makes team coordination easy and intuitive. When a project is connected to DVCS, the DVCS server makes it easy to share and secure different versions of a selected set of cells.

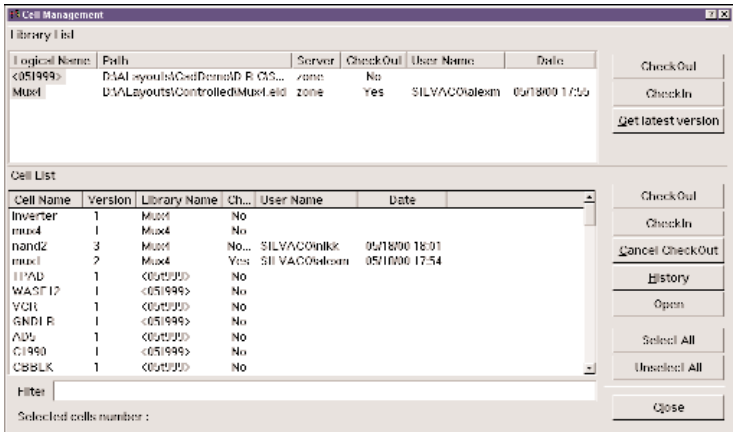

Figure 1. Design version control interface.

Before users can add projects to the Repository, they must create one or more projects in the working directory. After creating a project, the user can add projects from their computer to the Repository.

DVCS can maintain multiple versions of cells, including a record of the changes to the file from version to version. Version control addresses the following areas:

- Team coordination — making sure that only one person at a time is modifying a cell. This prevents cells from accidentally being replaced by another user's changes
- Version tracking — tracking old versions of cells , which can be retrieved for change tracking and other purposes

The working area is the directory where the user actually works with their design. When "Check Out Project" or "Get Project" is selected, DVCS copies the item into the working area. After making changes to the project and checking it in, DVCS copies it from the working area back into the DVCS database. From then on, DVCS manages the working project by creating cells as needed on the computer when projects are checked out.

The "Cell Management" dialog displays important status information, such as current connected projects, search criteria, status of cells, and so on. Some of this information is shown in columns, and other information is shown in the additional "Details" and "History" dialog boxes.

Continued on page 2....

INSIDE

SILVACO INTERNATIONAL

When a cell is checked in, this dialog shows the corresponding timestamp and user name.

Version Control and History

DVCS provides version control and history services, to ensure that each version of a cell is recoverable. The unique version of a cell is registered each time when a changed cell is checked in. The unique internal version of a project is maintained by DVCS during any changes made in the project. The user has no control over these numbers. Every version of every cell and project in DVCS has a version number. The version number is always an integer number and it never decreases with time.

Version number:

- Assigned automatically by DVCS,
- Always an integer numeric value
- Always increases to next integer number
- Increases each time an action that affects storage is taken on a cell or project, such as adding or checking in
- Displayed in history and cell properties dialog boxes
- Cannot be edited or changed by user

User Management, Security, Access Rights

DVCS uses the operation system user maintenance means to control the rights of access to DVCS. DVCS Administrator can add and delete users, assign access rights to projects, and change their access to DVCS. When new users are added, by default they have full access to a project.

When you install DVCS, the default security system is enabled. The administrator can, however, customize security in the installation to allow only specific users to have access to certain projects and certain commands.

Default Security

Default security in DVCS installation is simple. The administrator has only two levels of access rights to choose from when adding new users to the DVCS installation:

- Read-only rights: The user can see everything in DVCS but can change nothing.
- Read / write rights: The user can see and change anything in the DVCS database.

Each time a user is added to the installation, the level of access rights for the new user must be determined. The "Add User" dialog box contains a check box labeled "Read only". When selecting this check box, the new user can read files but not change them. If these levels of access rights are adequate nothing further needs to be done to to enhance security.

Setting Security for DVCS Administrator

All DVCS security is governed by DVCS Administrator.

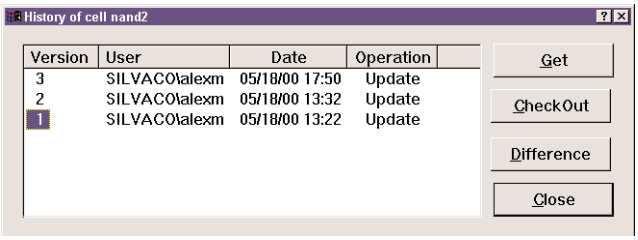

Figure 2. Cell history dialog box.

Any user can be assigned DVCS Administrator, so access to the Administrator program is protected. To grant Administrator rights to users and setup Repository, an existing Administrator must use Server Administrator utility.

Project Security / User Access Rights

Project security in DVCS is based on user access rights. Each project is accessible only to those users who have the appropriate rights. Each command can be used only by those users who have the rights associated with that command.

There are four user access rights, described in the following table.

Note: Each access level includes all the levels that precede it. For example, the Change rights includes the Read-Only rights.

Rights	Description
Administrator	Add new project and delete existing project by using such commands as **Add Project To DVCS, Delete Project from DVCS.**
Full access	Add new cells and delete existing cells by using such commands as **Add, Delete,** and **Rename**. Change technology by using **Check Out Project, Check In Project.**
Change	Modify cells by commands **Check Out, Check In,** and **Undo Check Out.**
Read-Only	View, but not change, files by using such commands as **View, Get, Get Version.**
No access	No access to project.

Conclusion

Design Version Control System used together with ***Expert*** Layout Editor and other ***CELEBRITY*** CAD tools significantly enhances productivity for large projects in multiuser design environment. This is achieved by its advanced capabilities, such as concurrent access to shared design files, version control and security enforcement.

At the same time, a single designer will benefit from better project manageability delivered by DVCS. For a single-user mode, the installation and running of DVCS is extremely simple and transparent, and it does not carry any setup and training overhead usually associated with complex systems.

Maverick and Guardian - Enhancements

Introduction

The latest release of Layout versus Schematic tools from ***CELEBRITY*** CAD suite (***Maverick*** full-chip parametric netlist extractor and ***Guardian*** hierarchical netlist comparator) delivers a number of significant advances. The engines of both tools were tuned up to achieve essential reduction of running time while processing huge designs. ***Maverick*** and ***Guardian*** became integrated more with other ***CELEBRITY*** systems. They provide extended setups that are easily customized, and offer new functions that serve to achieve better precision for parameter extraction. ***Maverick*** and ***Guardian*** reduce the number of verification / modification iterations.

LVS CROSS-VIEW NAVIGATOR

LVS Cross-View Navigator provides easy and convenient way to analyze the results of the LVS run and to make necessary corrections in the design. Using this tool (Figure 1) makes it possible to view all nodes mentioned in reports directly on layout and schematic graphical views from the perspective of both netlists (extracted from layout and derived from schematic) involved in comparison.

Utilizing the capabilities of Silvaco Intertool Communication Server (Figure 2), LVS Navigator provides drive among basic LVS results, points out the relevant node names on target report and netlists (Figure 3) and highlights images of devices (Figure 4) or nets (Figure 5) simultaneously in ***Expert*** Layout Processor and ***Scholar*** Schematic Capture.

SHOW SCOPE: The user of Cross-View Navigator is offered with the possibility to trace any subset or full scope of the related views:

- Post-run LVS report from one of the predefined categories (see the following)

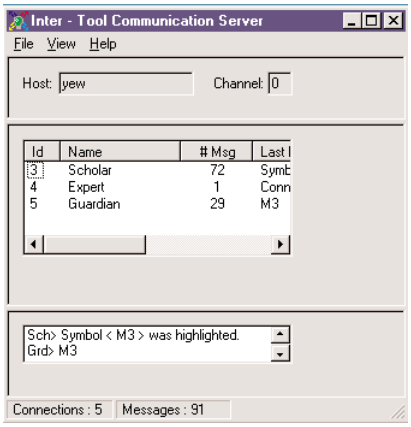

Figure 2. Reference Panel of Inter-Tool Communication Server.

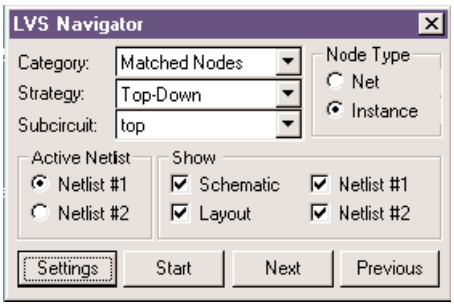

Figure 1. Workbench of LVS Navigator.

- SPICE Netlist extracted from layout cell
- SPICE Netlist produced by schematic capture
- Layout presented in the active window of ***Expert*** layout processor
- Schematic drawing loaded in ***Scholar*** schematic capture graphical window

The flexible scope gives the user the convenience of having a small number of views necessary for the particular verification activity on the screen.

NODE TYPE: As long as all basic LVS reports (matches, discrepancies, etc.) are split between relations involving nets and devices, LVS navigator works in one of the two corresponding modes - to show nets or instances (devices) inside all views included into the show scope.

REPORT CATEGORY: The set of design elements to be browsed during particular navigation session is defined by the nature of the report under consideration. LVS Cross-View Navigator supports (Figure 6) four categories of reports for browsing control:

- Matched Nodes (based on LVS report which is contained in .MTC file)
- Unmatched Nodes (based on .UNM file)
- Discrepancies (based on .UNM file)
- Parameter Errors (based on .PAR file)

Note the difference between discrepancies and unmatched nodes. Both of them reference nets and instances (devices) that had not been matched by ***Guardian***. For discrepancies, potential matches are generated by the tool (in most cases it means that if some local errors were fixed, these nodes would match), while for unmatched nodes they are not produced. For example, if one of the netlists contains some extra device which cannot be merged with other ones and has no equivalent in another netlist, the mentioned device should be classified as an unmatched node.

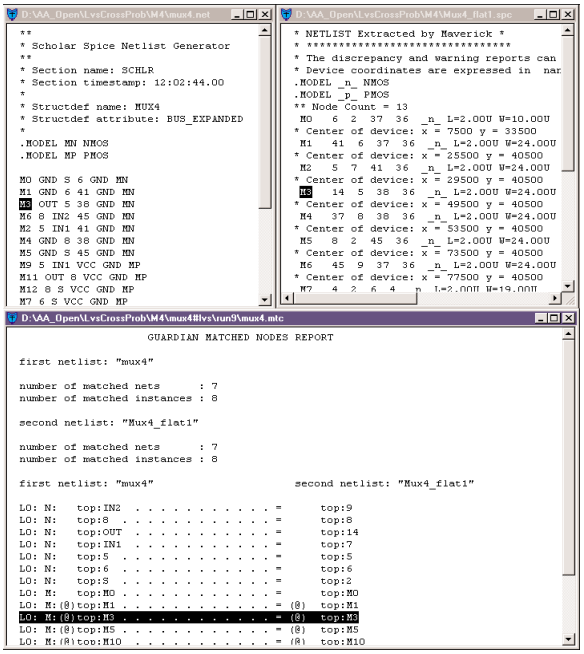

Figure 3. Combined View of Compared Netlists and Active LVS Report

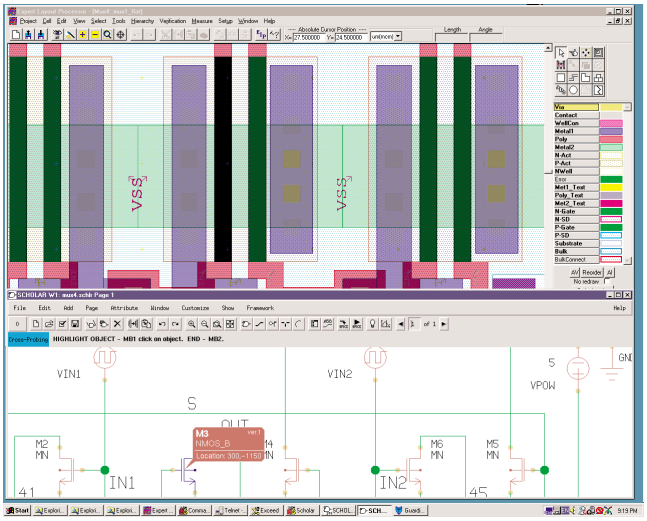

Figure 4. Device Cross-View initiated by LVS Navigator

ACTIVE NETLIST / BROWSE STRATEGY: These settings affect the order of LVS report processing in the case of hierarchical netlists. ACTIVE NETLIST determines the netlist whose hierarchy will be taken into account while applying the selected strategy. STRATEGY, in turn, can be Top-Down or Down-up. The first choice means that references from the active netlist that are taken from the root (top) subcircuit are processed first, then the references are taken from primary instances from the root, and so on down to the leaf subcircuits (which do not contain instances of the other subcircuits).

SUBCIRCUIT: The choice of a subcircuit from the active netlist is provided to reduce navigation to those nodes that are contained in this subcircuit.

NAVIGATION COMMANDS: The "Start" command serves to activate the browsing list after changing any options (category, strategy etc.). The "Next" and "Previous" commands provide the initiation (highlighting) of nodes which are mentioned in the browsing list after or before the currently processed ones.

Extended Specifications for Resistor Parameters in *Maverick*

Maverick extracts geometry-dependent resistance value for resistors. The efficient numerical procedure implemented in ***Maverick*** provides high accuracy of extracted resistance values for resistors of arbitrary configuration. Theoretically, those values like head and contact resistance traditionally prescribed to some resistors are not needed, because the head shape is taken into account automatically. For contacts a separate type of resistor could be defined. However, to meet common approaches, some extensions in definition of resistors were made (Figure 7). This provides more flexibility in definition of device recognition layers for various types of resistors.

To extract the value of the resistance, the user has to specify non-zero value for at least sheet resistance. Contact resistance is treated as the resistance of a single

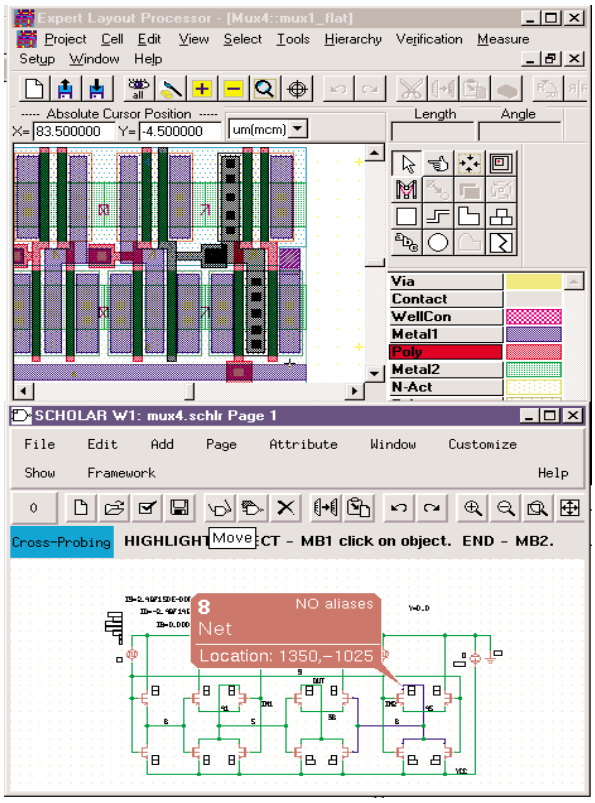

Figure5. Highlighting of nets in ***Expert*** and ***Scholar*** drawings.

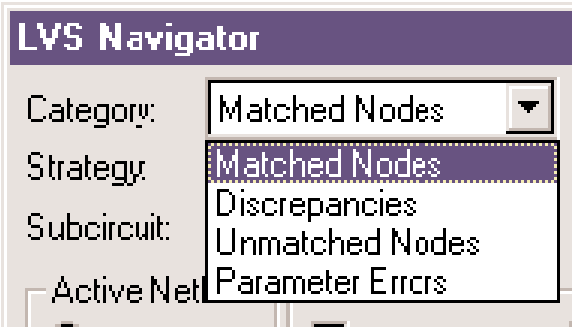

Figure 6. Choice of Report Categories in LVS Cross-View Navigator

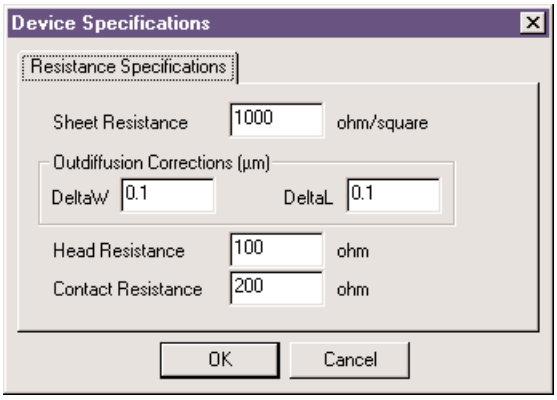

Figure 7. Extended resistor setup

contact shape. If actual connection of a resistor head has been designed using a group of contact shapes, resistances of all of those shapes are summed as usual parallel resistors.

If resistor heads are parts of resistor recognition layers (see, for instance, grey area in Figure 8), then the head resistance is calculated from geometrical information and sheet resistance value is specified. The value of head resistance shown in Figure 7 is ignored in this case.

If the resistor recognition shape (see the hatched area in Figure 9) does not cover the resistor head's areas, then the head resistance is not calculated. The extractor utilizes the value of head resistance shown in Figure 7 instead.

Resistance extraction routines need sheet resistance value (in Ohms/square) to be specified for the resistor recognition layer in the technology file. This can be done in the way as the following fragment of ***Expert*** technology file shows.

```
Layer
  {
  Name = "N-Act"
  ...
  Material
    {
    MaterialName = ""
    Resistivity = 555
    Permittivity = -1.00
    Thickness = 1.00
    }
  }
```

Figure 8. Resistor heads are parts of recognition layers: automatic calculation of head resistance.

The layer N-Act is the resistor recognition layer with sheet resistance equal to 555 Ohm/square. The whole set of resistor parameters is specified using the set of the following constructions within Device statement.

```
DevParam
  {
   NameP = "XXXXXXXXX"
  ValueP = 100
  }
```

NameP can be one of the following predefined strings:

"SheetRes"	for sheet resistance
"DeltaW"	for width outdiffusion parameter
"DeltaL"	for length outdiffusion parameter
"HeadRes"	for head resistance
"ContRes"	for contact resistance

ValueP indicates the actual numerical value of the parameter

Conclusion

Silvaco's verification suite (***Maverick, Guardian, Savage, Dragon***) is a dynamically developing system, permanently adjusting to the diversity of customer needs. The evolution of verification tools is followed by their tight integration with layout processor and schematic capture. This allows the designers to run complicated projects much faster and more accurately.

Figure 9. Resistor heads are not in recognition layers: head resistance should be prescribed.

Dragon DRC:
Performance Improvement Techniques

Introduction

Dragon DRC is a new advanced hierarchical DRC system. The design principles this system is based on are carefully selected to ensure that ***Dragon*** DRC will deliver maximum performance in different execution environments. Designed as a highly adaptive and truly hierarchical DRC system, ***Dragon*** is able to execute DRC scripts much faster than any flat DRC system in most real-life cases. An important feature of ***Dragon*** is its multithreaded parallel processing capabilities implemented at several system levels including parallel processing of layout, parallel execution of DRC commands and a number of internal parallel algorithms. Even though a well-thought-out implementation of the parallel hierarchical approach to layout DRC alone results in substantial performance boost over flat DRC system, ***Dragon*** is also able to carry out various preliminary and run-time optimizations of DRC script. The main purpose of the above optimizations is to generate the optimal schedule of execution of DRC commands, so that all components of the system will be used with maximum efficiency throughout the whole execution process, both in single processor environments and in multiprocessor ones.

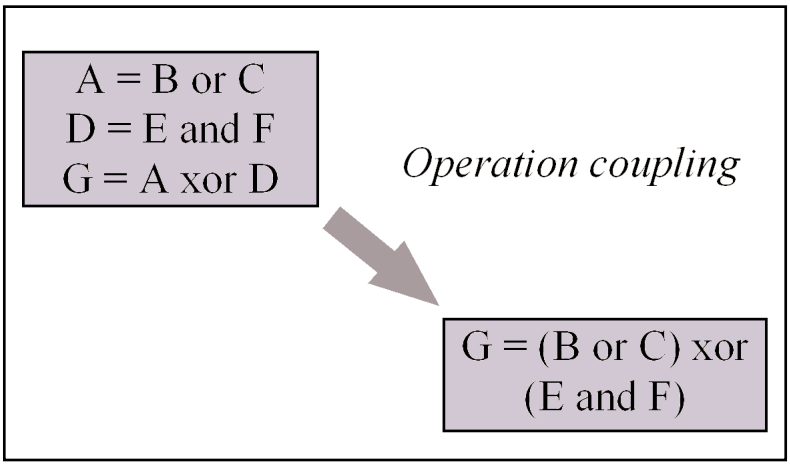

Figure 1. Simple example of operation coupling

Operation Coupling

To improve DRC performance on single processor systems, ***Dragon*** implements a separate scheduling algorithm called operation coupling. This technique is based on the fact that an atomic operation of the internal DRC engine of ***Dragon*** can be much more complex than an average single operation of a typical DRC script. Several operations from the script can be combined into one internal DRC operation and executed simultaneously, which is considerably faster than executing them one by one.

A straightforward example of such optimization can be easily built from several Boolean operations (see Figure 1).

However ***Dragon's*** script optimization capabilities are not limited to merging simple Boolean operations into Boolean formulae [1]. Implementation of almost all supported DRC operations in ***Dragon*** is based on various modifications of the widely known scanline technique [2]. The fact that internal algorithmic engines for operations from different groups, such as Boolean, Select and Spacing Check operations, are based on the same foundation opens great opportunities for effective operation coupling. The separate component of ***Dragon*** DRC called DRC script execution scheduler is responsible for detecting and realizing all reasonable operation coupling opportunities in a given DRC script. On multiprocessor systems the same component of ***Dragon*** DRC also takes care of generating parallel execution schedule for the script (see below).

Parallel Processing at Script Level

Even though the operation coupling technique described above possesses a substantial potential for optimization of DRC script execution process, ***Dragon*** takes additional steps in order to achieve even higher DRC performance on multiprocessor systems. As was mentioned above, there is a number of relatively independent parallel DRC algorithms implemented in ***Dragon***.

The DRC script execution scheduler introduced above is also responsible for parallel DRC script execution. Both parallel execution schedule and operation coupling schedule for DRC script are created by analyzing the informational dependencies of operations within the script. The primary target of the scheduler is to split the graph of informational dependencies of the given DRC script info separate operation clusters, each consisting of a bunch of easily coupled and highly dependent operations. After that it can generate one or

more coupled DRC operations for each cluster and, finally, produce a schedule of parallel execution such that independent operations from different clusters are carried out simultaneously.

It is easy to see that in many cases the same set of DRC operations can be optimized in accordance with either operation coupling or parallel execution approach (see Figure 2).

Because of that it is often possible to generate several execution schedules for the same DRC script. Either one might be optimal depending on different external parameters, i.e. parameters of DRC execution environment (including the number of available processors, amount of available virtual and physical memory and other system parameters). For this reason ***Dragon*** is equipped with a number of well-balanced heuristic criteria which are used to analyze the execution environment so as to properly adjust the DRC script scheduler priorities and fine-tune its internal algorithms. As a result, it is expected to generate the execution schedule optimal for a given DRC script in a given execution environment.

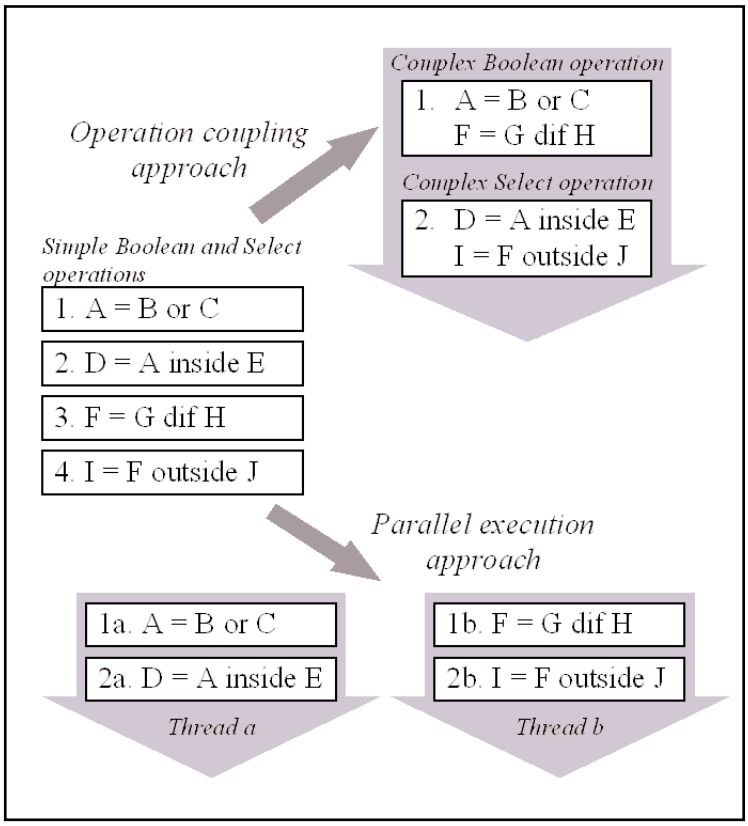

Figure 2. Operation coupling and parallel execution - two alternative ways of DRC performance improvement

Parallel Processing at Command level

Still another important parallel processing technique is employed by ***Dragon*** during the execution of one DRC command. The technique is layout-related, not DRC script-related as the one above. Working with a hierarchical layout, ***Dragon*** executes each DRC operation in cell-by-cell manner. In multiprocessor environment groups of cells can be processed simultaneously, i.e., in parallel mode. Of course, in most practical cases a hierarchical layout cannot be treated as just a collection of independent cells. On the contrary, internal geometry of one cell may have many interactions - overlaps - with geometry of other cells in many places in the layout. This can substantially complicate its hierarchical processing during DRC. In order to obtain correct and exhaustive DRC results and at the same time increase system's performance, ***Dragon*** uses a number of algorithms to preprocess the existing layout hierarchy. It dose this by rebuilding it in order to decrease the amount of cell interaction in the layout (resolving overlaps). That includes selective flattening of overlapping cell instances, overlapping instance merging, array decomposition and other techniques of cell interaction resolving. It is important to mention here, that whatever changes were made to the initial layout hierarchy by internal ***Dragon*** algorithms, the DRC results can always be reported in terms of the original hierarchy [3].

By preliminary resolution of cell interactions in the layout, ***Dragon*** can significantly increase the efficiency of parallel cell processing during DRC. The special component of the program called cell processing scheduler keeps track of all known or potential cell interactions and makes sure that all cells that can be processed by parallel threads are processed that way. The growth of the number of parallel threads created by Dragon at all supported levels of parallel processing is limited by available system resources.

Conclusion

The powerful performance improvement approaches described above are built into other proven by time, carefully tuned and fit together components of ***Dragon*** DRC system to guarantee its top performance.

References

[1] Application of Scan Line Methodology to Perform Metric Operations in DRC. Simulation Standard. Volume 8, Number 12, December 1997, pp. 7-9.

[2] V. Feinberg. Geometrical problems of VLSI computer graphics, Radio i Sviaz, Moscow, 1987.

[3] Savage Enhanced with Recognition and Reporting of Hierarchical Structure of Errors. Simulation Standard. Volume 9, Number 3, March 1999, pp. 1-4.

Parametric-Cells Implementation in *Expert*

Introduction

A parameterized cell (P-Cell) is a cell with user-specified parameters. It is possible to create customized p-cell instances with different composition according to parameter values.

Using P-Cells gives a considerable memory reduction in the case of multiple instancing of a cell. ***Expert*** layout editor creates temporary ordinary cells for each set of parameter values. All instances of a P-Cell with the same parameter values refer to the same ordinary cell. P-Cells are stored in the Expert database in the form of ***LISA*** procedure. Temporary cells are not stored in the database file.

Moreover, P-Cells add enormous flexibility to design and increase designer productivity. The notion of cells allows the designer to avoid repetitive drawing of identical pieces of layout. P-cells provide an additional benefit: a user can quickly introduce changes into the layout by modeling the parameters of an instance of P-Cell rather than redrawing the geometry.

Creating and using P-Cells in the Expert environment is extremely easy.

Creation of P-Cell involves three actions described below:

- write the XI script that actually generate the P-cell
- specify parameters
- compile the definition of P-cell.

There are two ways to create and modify P-Cell XI script: automatically using menu commands "PCell >> Parameters ..." and "PCell >> Compile" or manually through menu command "Cell >> New ...". The first method allows the user to create simple P-Cells. To create complex P-Cells the user can use powerful features of ***LISA/XI*** language that are impossible to generate automatically.

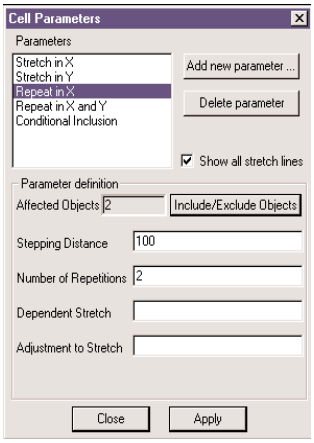

Figure 2. Repetition parameters.

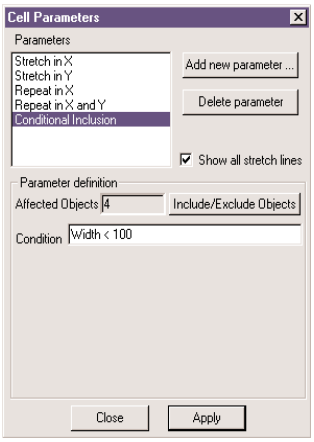

Figure 3. Conditional inclusion

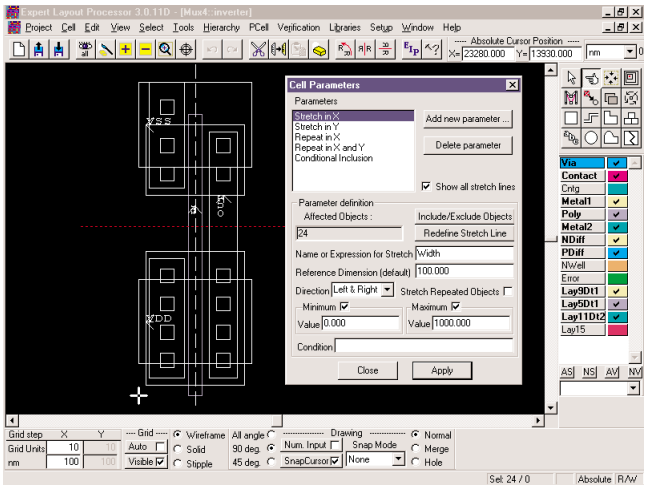

Figure 1. Main dialog for parameter insertion into the master cell: Stretch parameter.

Automatic Generation of P-Cell

For automatic generation of P-Cell the user must open any ordinary cell from the processed project or activated libraries and select "PCell >> Parameters ..." menu command. The "Cell Parameters" panel appears, see Figure 1.

The button "Add new parameter" selects and adds new parameter to the P-Cell. Various parameter definitions are associated with specific parameters (stretch, repetition, conditional inclusion, and so forth). "Parameter definition" group of "Cell Parameters" panel is to specify parameter definitions for the selected parameter. As a rule, parameter definition has a name and default value which are used in the instancing dialog when you want to place an instance of the P-Cell.

Stretch parameters change the size and position of layout objects that are included in the specific stretch group. The name of the stretch group must be either an ordinary name or XI expression. Stretch line is used to select objects and determine the stretch direction. The horizontal stretch line controls up and down direction and the vertical stretch line controls left and right direction. Directions are indicate using "Direction" combo box. Default dimension will be used in the instancing dialog as default parameter value. Minimum and maximum value is used to control value in stretching the cell.If the "Stretch Repeated Object" box is checked, then objects included into stretch and repetition groups will be stretched.

Repetition parameters are to repeat objects from repetition group in x direction, y direction, or both directions, see Figure 2. User's can add or remove objects from the repetition group by using the "Include/Exclude" button. The main repetition parame-

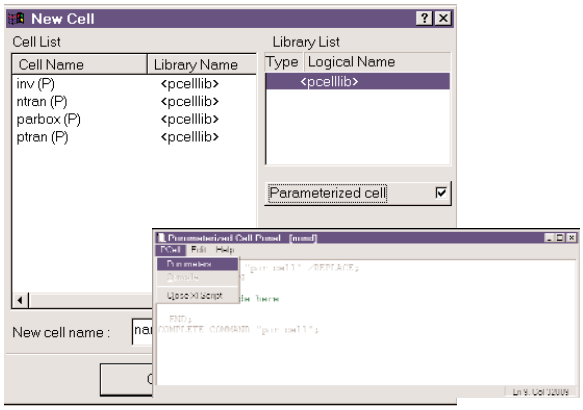

Figure 4. P-cell xi-script panel.

ters are "Stepping Distance" (distance between repeated objects) and "Number of Repetition". Any name or valid XI expression can be used to specify this parameters. If the "Dependent Stretch" edit field name of the previously defined stretch group is typed in then objects from repetition group will be stretched after repetition.

Conditional inclusion parameters define the group of objects that are included or excluded from the cell depending on the condition, see Figure 3. The name of the condition group must be any name or valid XI expression.

When parameter definitions are complete, the P-Cell must be complied before the user can place instances of this cell.

Textual creation of P-Cell

To create a P-cell by coding its xi-procedure, the "Parameterized cell" option in the "New Cell" panel must be checked, in addition to cell name and library selection. After the "OK" button is clicked, the user will see the "Parameterized Cell Panel" and the empty layout window for this cell, see Figure 4. This panel has its own menu.

To define or modify parameters, select the P-cell Panel menu command "Parameters", see Figure 5. For each parameter the user must type its name, select type of

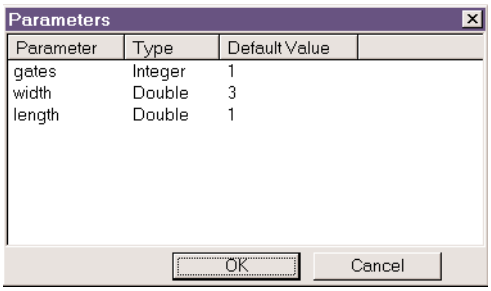

Figure 5. Parameter specification window.

parameter and set a default value. There are four types for parameters: double integer, boolean and string.

The P-Cell Panel has a large editing field for typing a P-Cell XI script. This editor has the same standard features as a typical Windows editor. The following common editing commands are available for editing in the P-Cell Panel from its Edit menu: cut (Ctrl-X), copy (Ctrl-C), paste (Ctrl-D), clear selection (<Del> key), delete all, find, replace. Pieces may be cut from other scripts and pasted into the P-Cell Panel. The difference from the ordinary XI-script panel is that of the "Define Command" statement, can not be edited i.e., the beginning and the end of the script. It is generated automatically by ***Expert***. The non-editable text is gray in color. All other necessary commands may be typed inside the outermost Do Begin...end statement.

The "PCell>>Compile" command of the P-Cell Panel menu compiles the definition of the P-cell into ***Expert*** database. If compilation passed without errors, the compiled P-cell is stored in the library and is ready to use. The layout window shows the drawing of the P-cells with default values of its parameters, see Figure 6.

Instancing of P-Cell

If a parameterized cell is activated for instancing, then the Create Instance dialog assumes the form shown in Figure 7. The values of the parameters may be edited by after clicking them with the mouse.

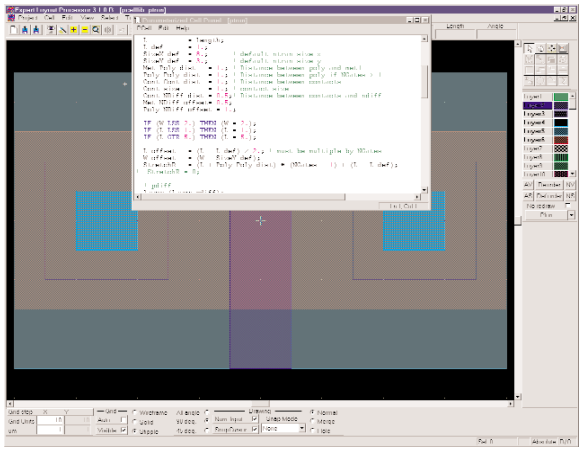

Figure 6. Completed P-cell definition

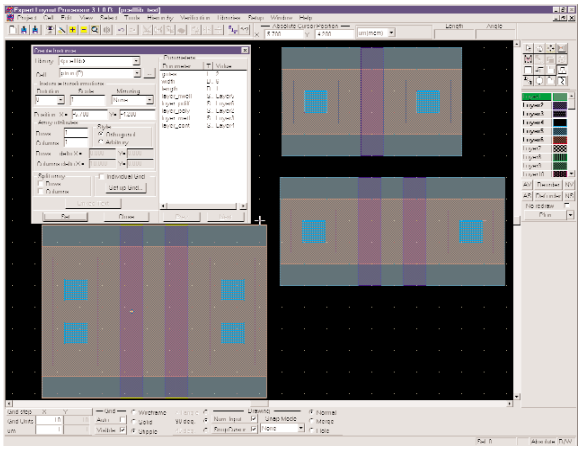

Figure 7. Instancing of a P-cel

Calendar of Events

June

1	
2	
3	
4	
5	IITC - San Francisco, CA DAC - Los Angeles, CA
6	IITC - San Francisco, CA DAC - Los Angeles, CA
7	IITC - San Francisco, CA DAC - Los Angeles, CA
8	
9	
10	
11	
12	HiTEC -Albuquerque, NM
13	HiTEC -Albuquerque, NM
14	HiTEC -Albuquerque, NM
15	
16	
17	
18	
19	
20	
21	EWLTE Noordwijk, - The Netherlands
22	EWLTE Noordwijk, - The Netherlands
23	EWLTE Noordwijk, - The Netherlands
24	
25	
26	
27	
28	
29	
30	

July

1	
2	
3	
4	
5	10th Fine Process Technology - Tokyo, Japan
6	10th Fine Process Technology - Tokyo, Japan
7	10th Fine Process Technology - Tokyo, Japan
8	
9	
10	
11	
12	IWAM - Tokyo, Japan
13	IWAM - Tokyo, Japan
14	IWAM - Tokyo, Japan
15	
16	
17	
18	
19	
20	
21	
22	
23	
24	NSREC - Reno, Nevada
25	NSREC - Reno, Nevada
26	NSREC - Reno, Nevada
27	NSREC - Reno, Nevada
28	NSREC - Reno, Nevada
29	
30	
31	

Bulletin Board

EWLTE

Silvaco travels to The Netherlands for this year's 4th European Workshop on Low Temperature Electronics from June 21st through the 23rd. We predict further excitement about our tools, which continue to revolutionize ***TCAD Driven CAD***.

Japan

Silvaco's Japan offices are leading the way in the semiconductor development hub of Japan. Our Japanese team will be at 2 important conferences in Tokyo. Don't miss the 10th Fine Process Technology Conference and 7th International Workshop on Active Matrix & TFT Technology.

NSREC

The last big conference of the summer is the 2000 IEEE Nuclear and Space Radiation Effects Conference. Held in Reno, Nevada, you can gamble on this year's NSREC being a lot of fun. In the next issue we will outline our ambitious schedule for the fall.

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: To simplify netlist extraction, I specify one kind of diffusion resistors in my layout by means of a special resistor definition layer. To my surprise, the extracted value for one such resistor always exactly twice the value I expected, no matter how I stretch this resistor. Other resistors (of the same type) are extracted by *Maverick* exactly as expected. Moreover, in my previous version of technology which defined diffusion resistors by means of intricate DRC operations all values were good.

A: The most probable reason is that you have duplicated shapes exactly overlapping each other, so that you do not see the duplication. Such shapes might result from erroneous copying. However quite often they are produced by overlapped cell instances.

When you defined resistors by means of DRC operations, the resulting device recognition layer could be merged, e.g., during logical operations, and hence could not contain overlaps. However when device definition layers are drawn manually, a common mistake to leave such layers "as is" for device extraction. Even for such "simple" device (and pin!) layers you must always perform merge operation, to avoid the problem discussed here.

If you don't want to add merge operation (sure, it slows down the LVS), a good idea is to use the Cell>>Info command and compare the number of shapes in the device layer and the number of the corresponding extracted devices.

The puzzle with stretch resolves easily as well: most probably, you pick object to stretch by box, so both overlapping shapes stretch exactly in the same way.

Q: Are there any means to compare two layouts in *Expert*?

A: You may perform layout comparison tailored exactly according to your needs by means of xi-scripts. An example of such scripts is given in ***Expert's*** distribution. The first script (compare.xis) defines the comparison command. The second one (compproj.xis) is for data input. These scripts are included into ***Expert's*** distribution. They are easy to understand, and you may easily modify them, adding more flexibility and fine tuning to your needs.

You need to run the first script only once per ***Expert's*** session, e.g., from the autostart xi-script (see Setup>>General>>Auto run scripts).

```
!! script: COMPARE.xis
! This script allows you to compare two eld,
gds or apl files,
! even of different formats
! The comparison is performed cellwise and
layerwise by means of XOR
! bFlat parameter selects whether to perform
comparison of flattened cells
!       or of cell's own shapes only.
```

Q: After each upgrading *Expert* to a new version I keep receiving messages, like: "Polygonal font file "C:\Silvaco\lib\expert\3.0.9.R\x86-NT\polgfont.gds not found. Please update your settings". How can I get rid of them once and for all?

A: This file contains drawings for characters used when you are creating "manufacturable" texts (To do this, use Edit>>Create>>Text command with Edit>>Numeric Input mode on and "Polygonal text" option checked on the Text panel). A polygonal font file is an ordinary gds file, and you may replace font drawings by your favorite ones. You may even have several font files and switch between them.

The mentioned message appears when you de-install the previous version of ***Expert*** and this default file is deleted. To get rid of it, put your polygonal font file into a permanent place on your disk and select it into ***Expert's*** Setup>>General.

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