TCAD Driven CAD

A Journal for CAD/CAE Engineers

Introducing Expert ULSI Layout Processor for PC-based Platforms

Expert is a state-of-the art hierarchical full-custom ULSI layout editor including advanced design functions. Running on PCs under Windows NT or Windows 95, it easily handles multimillion object ULSI designs. **Expert** includes special navigation and inspection tools useful for designers of physical layout of integrated circuits. It is also recommended to all those who are interested in exploring and reusing large hierarchical integrated circuit layout designs.

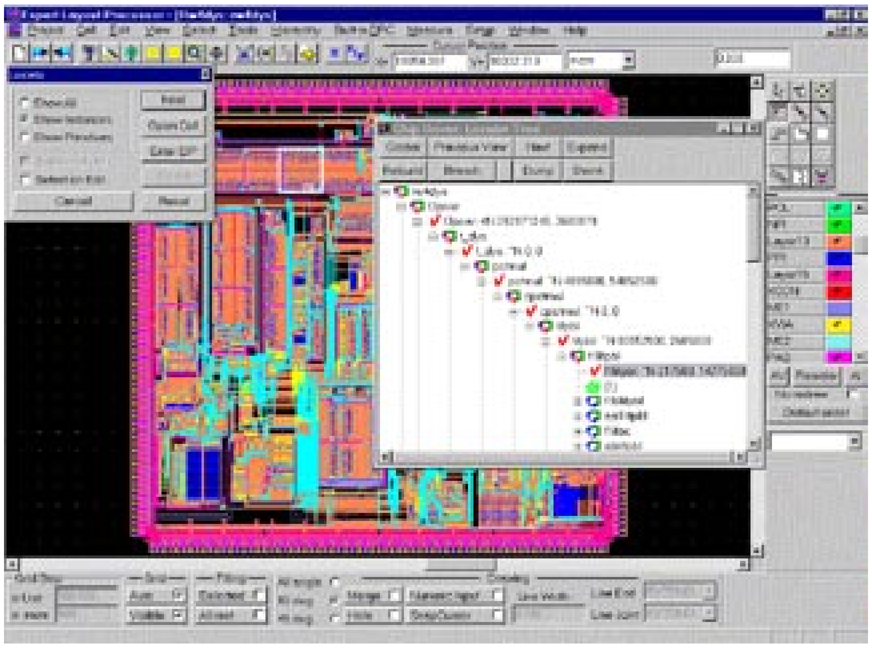


Figure 1. Clip out operation: A piece of layout is saved as new cell with hierarchy preserved whenever possible.

Expert provides high-speed editorial and verification functions and advanced graphical tools. It has no restrictions on layout complexity neither on the number of IC components or on the type of geometrical limitations set for specific layouts. A proprietary high speed data base allows efficient processing of ULSI designs of any size, density, complexity of hierarchy, or number of cells. The dimensions of processed layouts are limited only by the amount of available disk space.

The chip navigation tool "Chip Rover" presents the layout hierarchy in the form of level-by-level expanded hierarchy tree of instances/ cells. It gives easy-to-use ways for exploring complex hierarchical VLSI designs. Chip Rover allows fast loading of a required cell instanced at any level of hierarchy, marking arbitrary instances on the background of the whole chip, and displaying selection and search results. Chip Rover is a key component of Edit-in-Place, selection operations, and Multiview Navigator functions.

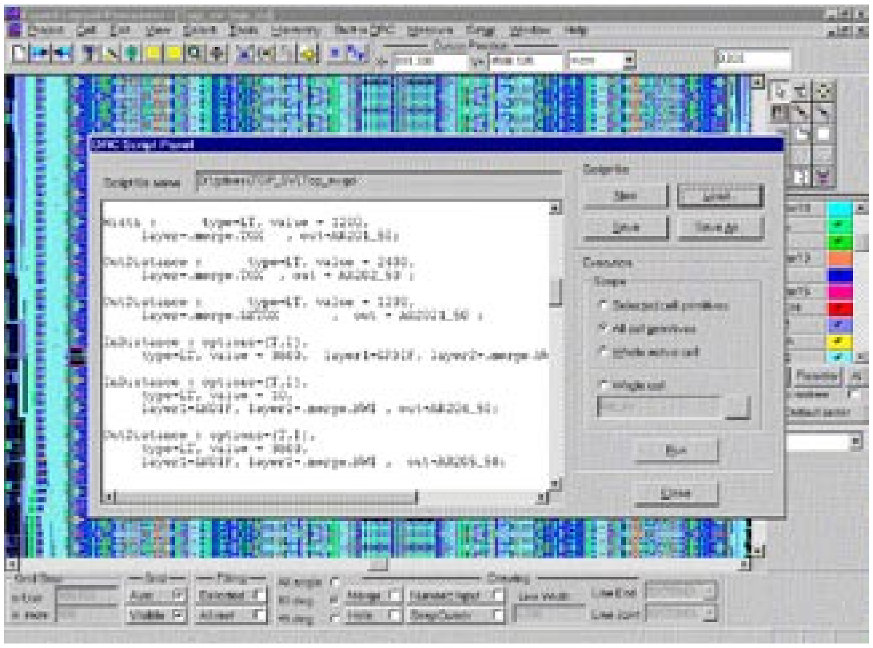
Expert delivers an advanced environment and set of features to support hierarchical design and editing of VLSI layout. These functions are provided by a number of specific Cell/Hierarchy operations. They make possible a user-defined logical subdivision of layout and full control of hierarchical structure. The key operations are recursive Edit-in-Place, hierarchical view settings, flattening and grouping into cells. Edit-in-Place gives an easy and convenient way to edit objects of a cell within the surroundings of any its instance, while the group

Continued on page 2....

INSIDE

Volume 8, Number 9, September 1997

SILVACO International



```
Width :        type=LT, value = 1200,
     layer=.merge.DOX    , out=AR201_50;

OutDistance :        type=LT, value = 2400,
     layer=.merge.DOX   , out = AR202_50 ;

OutDistance :        type=LT, value = 1200,
     layer=.merge.[illegible]       ,  out = AR203_50 ;

InDistance : options=(T,1),
     type=LT, value = 8000,  layer1=[illegible], layer2=.merge.[illegible]

InDistance : options=(T,1),
     type=LT, value = 10,
     layer1=[illegible], layer2=.merge.DM1 , out=AR204_50;

OutDistance : options=(T,1),
     type=LT, value = 8000,
     layer1=[illegible], layer2=.merge.DM1 ,  out=AR205_50;
```

Figure 2. Built-in DRC: Script running.

of hierarchical view operations creates a flexible environment for the execution of basic editorial functions, which allows various levels of detail, ranging from showing the floorplan level to presenting all-level objects together with instance frames.

Expert supports polygon level entry, as well as component level layout entry. A comprehensive set of objects is available for editing complex layouts: cell instances, arrays, polygonal regions, rectangular boxes, text for labeling of cells/ primitives, multiple-style wires of any width, ellipses, and rolled regions. All-angle, 90 and 45-degree geometries, arcs and rolled drawings provide advanced editing capabilities for high-frequency and mixed-signal designs.

The editor is supplied with a highly efficient Hierarchical Locator inspecting any desired cell instance or primitive at any level of hierarchy. It serves as a convenient tool for fast browsing and editing of IC components located inside the specified area.

Expert is equipped with a flexible grid setup providing both user-definable and automatic (zoom-adjustable) grids and subgrids in several representations. There are convenient means for fast on-layout measurements: automatically graduated axes, flexible polyline rulers, distance meter and cursor locator. Layer settings, arrangement, and visibility are easily controlled from Layer bar and Layer Setup dialog.

The advanced Multiview Navigator uses Map View for controlling all opened layout windows. Multiview Navigator is supplied with convenient tools for resizing/ focusing of dependent windows. It allows users to view the layout at different resolutions and to globally monitor the corrections made in a locally viewed work area.

A key feature of **Expert** is the built-in verification subsystem and set of automatic operations:

- Highly efficient and flexible Design Rule Checker
- Clipping Out of arbitrary layout areas with the capability of saving the obtained pieces as separate cells. This clip-out feature preserves the hierarchy of instances that lie completely in the clipped area
- Merge, Hollowing, Resizing and Cutting by arbitrary polylyne of layout regions
- Fast automatic Extraction of Nodes as sets of electrically connected primitives from several layers

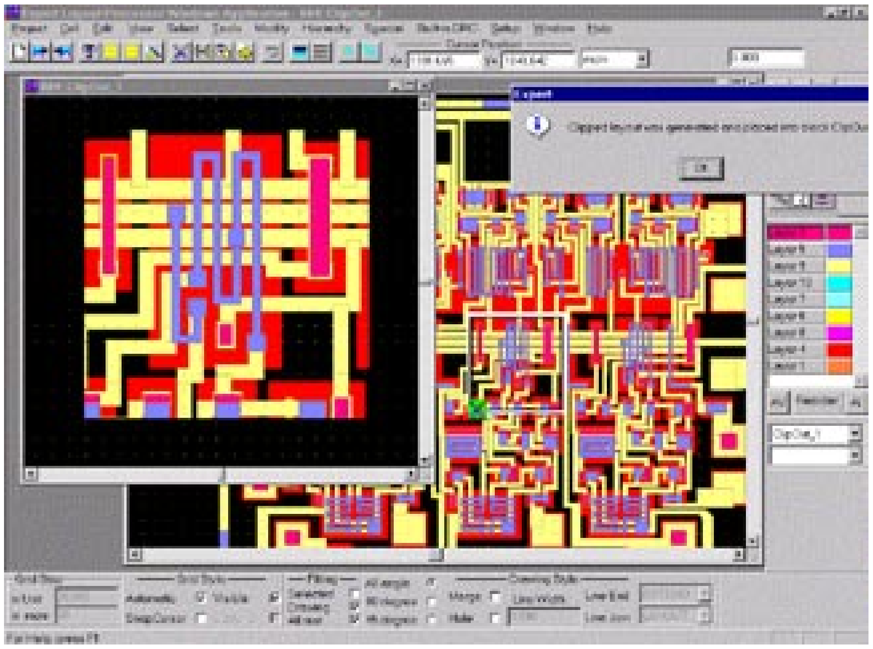


Figure 3. Expert's hierarchical Locator Tool.

- The Routing tool providing fast automatic construction of wires (routes) between pairs of objects or points

Note, the all mentioned operations (especially DRC) require intelligent geometry processing implemented on the based of proprietary extensions of scan-line technique, developed in SILVACO's CAD Research Center.

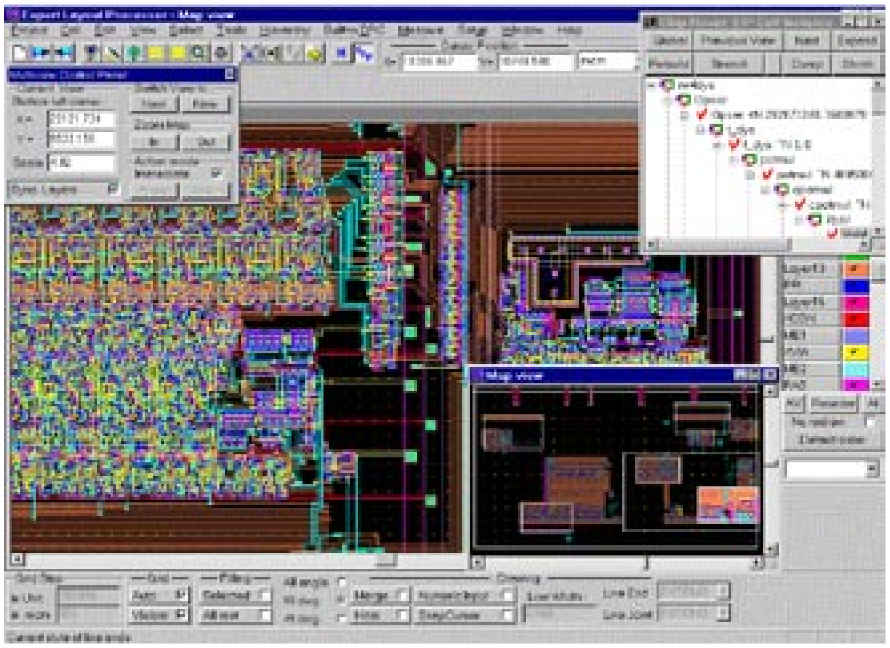


Figure 4. Multiview Navigator: Map view shows positions of all opened windows.

Chip Navigation in Expert

Introduction

With chip density increasing, device size shrinking into the sub-quarter micron range, geometries growing 3D, it is increasingly important to recognize and establish a proper usability tradeoff between automated tools for physical IC design (automated floorplanning, placement, routing) and "polygon" layout editors that combine manual input/editing with built-in verification.

An observable shift from full-custom to semicustom design methodologies, burdened with problems of reusability and scalability, places additional requirements on a layout editor.

A modern layout editor capable of working with multi-million transistor designs, must among other things, provide fast and easy access to any parts of chip's layout for local inspection, simulation, modification, etc.

Expert combines traditional features of polygon layout editors with enhanced means of navigation throughout the chip, both down its hierarchy and across the design plane, by means of Chip Rover, Multiview Navigator, and Locator.

Chip Rover: Navigation through Hierarchy

Chip Rover is a tool for navigating through the hierarchy of the chip design represented in the form of a tree. In addition to hierarchy tree navigation, Chip Rover provides a convenient visual supplement for other layout editing and navigation operations that are somehow related to chip hierarchy: It highlights selected cell instances, visualizes the results of the Locator operation, shows the actual on-tree place of Edit-in-Place, and allows users to specify and show instances placed anywhere in chip's hierarchy.

Depending on Chip Rover's actions it shows chip's hierarchy in two different forms: Basic Tree View and Branch Tree View. For a simple example layout from Figure 1 these two different tree views are illustrated in Figure 2 and Figure 3, respectively.

To understand the difference, let us clarify what chip hierarchy actually is. While it is often loosely referred

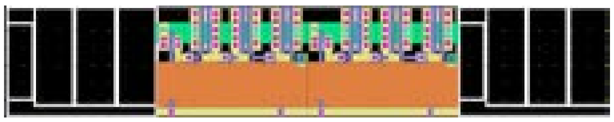
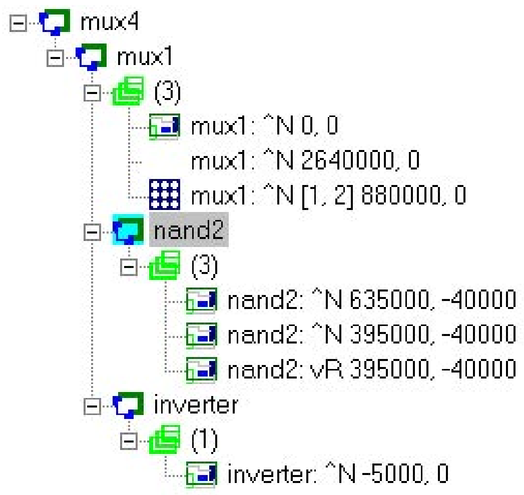
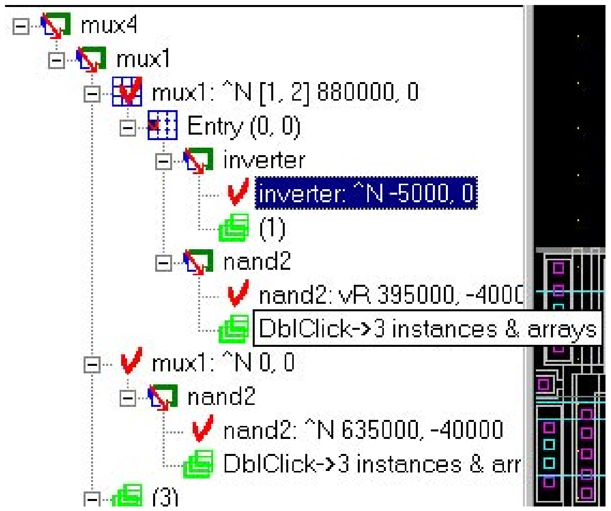


Figure 1. Sample layout to demonstrate Chip Rover's tree.



```
mux4
  mux1
    (3)
      mux1: ^N 0, 0
      mux1: ^N 2640000, 0
      mux1: ^N [1, 2] 880000, 0
    nand2
      (3)
        nand2: ^N 635000, -40000
        nand2: ^N 395000, -40000
        nand2: vR 395000, -40000
    inverter
      (1)
        inverter: ^N -5000, 0
```

Figure 2. Basic tree view for sample layout from Figure 1.



```
mux4
  mux1
    mux1: ^N [1, 2] 880000, 0
      Entry (0, 0)
        inverter
          inverter: ^N -5000, 0
          (1)
        nand2
          nand2: vR 395000, -400C
          DblClick->3 instances & arrays
    mux1: ^N 0, 0
      nand2
        nand2: ^N 635000, -40000
        DblClick->3 instances & arr
    (3)
```

Figure 3. Branch tree view for sample layout from Figure 1.

to as "cell hierarchy", a chip in fact consists of hierarchically nested cell instances. If one tried to display this cell instance hierarchy as a tree, it would look something like the tree in Figure 4.

Now imagine that the hierarchy becomes deeper and more instances of the same cell appear. It is evident that such "natural" tree quickly becomes enormous and incomprehensible.

Therefore our Basic Tree View branches out only along cells. The instances of the same cell in a common parent cell are grouped together and do not branch further. At Figure 2 one can see that mux4 cell contains mux1 cell instanced three times: one time as an array and two more times as separate instances. Further, mux1 contains two instances of nand2 and one of inverter.

Sometimes, nevertheless, it is necessary to view the place of an instance in its "natural" instance nesting hierarchy. This is the case, during a hierarchy-transparent Locator operation and in other situations. It is not necessary however to see all instances everywhere along the tree. We are interested only in specific ones. The Branch Tree View displays these views. In Figure 3 the following selected instance chains can be seen:

```
-inverter instance
    in entry (0,0)
        of 1x2-array of mux1 instances
           in top-level mux4;

- nand2 instance
    in the same entry (0,0)
        of 1x2-array of mux1 instances
           in top-level mux4;

- another nand2 instance
    in a "standalone" mux1 instance
           in top-level mux4.
```

Chip Rover provides a two-way communication between hierarchy tree and chip's layout.

From Tree into Layout:

- Double clicking on a cell item you switch to the window for the cell layout.
- Double clicking on a cell instance/array you switch to a layout window for the father of the cell instance. This instance is seen marked in the layout.
- If you can specify a chain of nested instances by its names and positions, you may input it into Chip Rover's Branch Tree (see below), and these instances (deep in the hierarchy) will be marked in the layout.
- you may enter into Edit-in-Place for this instance from Branch Tree by clicking the corresponding button.

From Layout to Tree:

The following actions in layout are immediately reflected on tree:

- All selection/deselection operations (in Basic Tree view);
- Edited-in-Place cell's position in the hierarchy (in Branch Tree view);
- Locator operation, see below.

All operations that modify cell hierarchy (cell/instance creation/modification) are reflected on the tree after pressing the Refresh button.

Chip Rover may show several trees:

Active Cell Tree shows the hierarchy of the cell in the current active window.

Project Tree shows the hierarchy of the whole project.

Branch Tree allows Chip Rover to access cell instances laying deep in hierarchy of the active cell.

Locator (Hierarchical Selection)

Locator is an advanced version of the selection operation. It allows the user to identify any object of the layout, no matter how deep in the hierarchy it is situated. Note that Select operations works only with objects of the edited cell, i.e., at the top level of the cell's hierarchy. You just draw a selection box and right mouse clicks produce you with layout objects one by one. Optionally, Chip Rover may be set to show the located instance on Branch Tree with the whole nested chain of its ancestor instances.

The user may choose whether the Locator will:

- pick all types of objects;
- pick only cell instances and arrays;
- pick only primitives, i.e., regions, boxes, lines
- pick objects that belong to the active cell only, not go deep into hierarchy.

The following actions are possible with the currently located object:

- It may be selected from its parent cell.
- Its parent cell may be opened for editing.
- Its parent instance may be opened for edit-in-place.

Multiview Navigator: Across the Design Plane

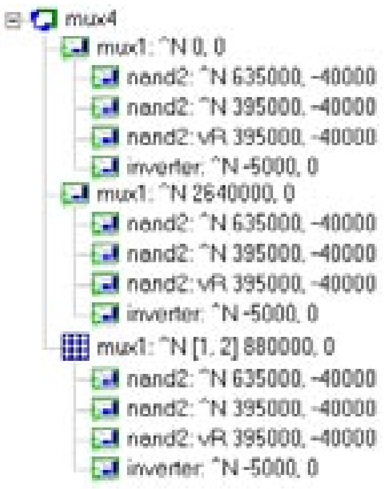
This tool allows you to conveniently manage multiple views of a cell's layout. There is a special window (Map view) that allows the user:

- to see all work windows (views) as they are overlaid on the whole cell layout.
- to switch from one view of the cell to another.
- to change view position and zoom by dragging its image on the Map view.
- to create new views at selected areas of layout.

When you choose this tool, The Multiview Control Panel and the Map View opens. Initially, the Map View zooms to the view contained in the active window, however, it contains frames for all opened views (windows) of the active cell. To see them all, press the Out button on the Multiview Control Panel.

In the Map View you see the cell layout and a number of frames, assigned to windows opened for the active cell. Each frame shows which piece of layout is visible in the corresponding window, and it reflects the view in that window. Actually a frame exactly matches the corresponding view all the time: any view operations (zoom, pan, scrolling) influences the corresponding frame and vice versa: any changes in size and position of a frame lead to changes of the corresponding view.

This arrangement is very convenient when you explore several places of a large chip simultaneously: you can



```
mux4
  mux1: ^N 0, 0
    nand2: ^N 635000, -40000
    nand2: ^N 395000, -40000
    nand2: vR 395000, -40000
    inverter: ^N -5000, 0
  mux1: ^N 2640000, 0
    nand2: ^N 635000, -40000
    nand2: ^N 395000, -40000
    nand2: vR 395000, -40000
    inverter: ^N -5000, 0
  mux1: ^N [1, 2] 880000, 0
    nand2: ^N 635000, -40000
    nand2: ^N 395000, -40000
    nand2: vR 395000, -40000
    inverter: ^N -5000, 0
```

Figure 4. All-instance tree fro sample layout in Figure 1.

readily see where you are on the whole layout, to which window you must switch next, how you must zoom/pan the current view to include an interesting nearby feature, etc.

Some frames may be invisible or only partially visible, if the corresponding window shows the piece of layout that lies outside the part of layout shown in Map View. You may zoom out in Map view by ordinary view controls to see more frames. Alternatively, you may click the Out button on the Multiview panel.

The active frame corresponds to the active window (and the active view in it). The remaining frames are inactive. Inactive frames are just white boxes. The insides of the active frame are made brighter than the rest of layout in the map view: its brightness is the same as in an ordinary window. If the active frame is too small, its bottom left corner is supplied with cross-hairs, so that this frame may be easily detected.

If you click the left mouse button with this cursor pointing at the border of an Inactive frame, it becomes active and it activates the corresponding window.

If you click the left mouse button with this cursor pointing at the border of an active frame, the cursor assumes the Border Grab shape. The user can then modify the frame as an ordinary box. You may grab a side or a vertex of the active frame and then drag it, resizing the frame. The view in the active window zooms accordingly, i.e., it will show the chip area falling into the frame. As a result, if the active frame is made larger, the active view zooms out; if the frame shrinks, the active view zooms in.

When the cursor is within the active frame, it assumes the Frame Grab shape. The active frame may be grabbed by pressing and holding the left mouse button, dragging it into a new position and releasing the button.

Introducing Savage
Efficient Design Rule Checker for PC-based Platforms

Savage is a powerful ULSI Layout Verification System with Programmable Geometric Design Rules. It performs design rule checking on IC layout without any restrictions on geometry shape and size. **Savage** does checking as a combination of logical and resizing operations, and conventional spacing checks. It processes layouts of digital, analog, UHF, as well as mixed signal ICs for any fabrication technology. **Savage** runs on PC / Workstations under Microsoft Windows NT / 95.

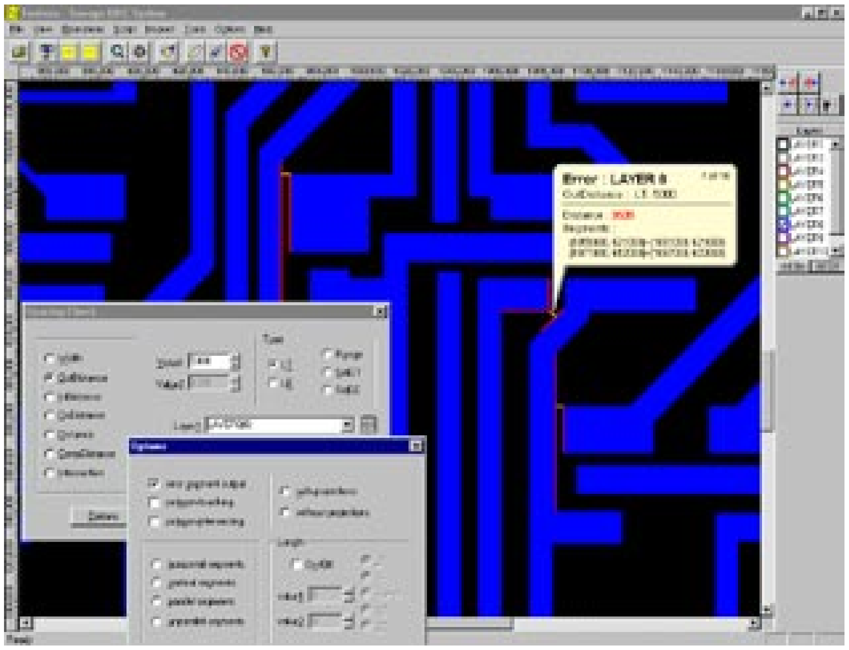


Figure 1. Graphical DRC error reporting in Savage.

The advanced mathematical methods implemented in **Savage** provide incomparable speed of DRC execution. It works faster than all other DRC tools available on the market of CAD systems on PC platform. Proprietary geometry data base allows to process ICs of arbitrary size without special requirements to main memory size. Two modes of operation are possible: interactive and batch. So, code for design rule checks and layout operations may be specified by means of **Savage** scripts, DRACULA command files, or input interactively. In addition, interactive command input may be captured into script for future re-use. Check reports are output in text or in graphical form with exact identification of the detected errors. **Savage** is compatible with the DRACULA system for input and output the command set. Layout input/output is possible via GDSII format or **Savage** own LLD format.

Efficiency

The cornerstone of **Savage's** efficiency is systematic approach to the DRC operations. From point of view of computational geometry, all of them are based on restricted set of abstract operations:

- finding segment intersections
- determining of subset of the given point set enclosed by contour or set of contours
- construction of the tree of visibility for arbitrary set of plane contours
- expansion and compression of regions
- execution of boolean operations on regions
- evaluation of distance between segments and regions

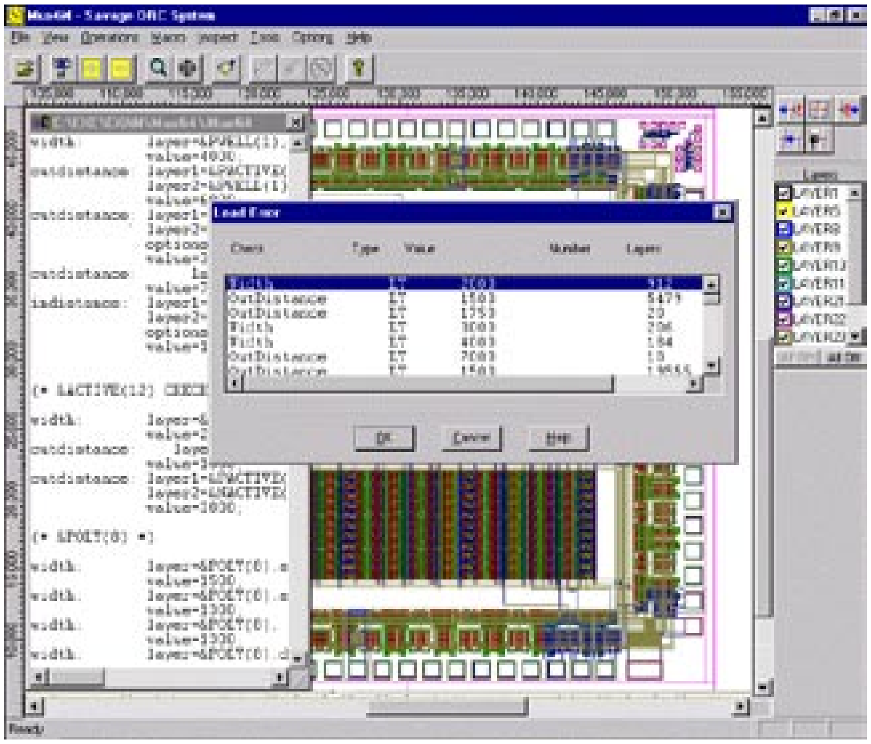
It is well known that the unified approach to the solution of these problems is the scan-line technique. However the key point is acceleration of the base scan-line method by applying some heuristics, based on specifics of regions from real layout. The accumulation of experience collected by SILVACO experts in this area have lead to speed-up of 3-5 times in comparison with common sweep algorithms for huge layout processing.

System Components

1. Structure of DRC Subsystem

The DRC subsystem actually performs geometric spacing checks on an IC layout. It is programmed tailoring to various user-defined checks. System's operation is easily tuned to any chip fabrication technology. The DRC subsystem has the following command blocks:

- project management
- layer manipulation
- layout region selection
- logical operations
- size/spacing checks
- layout object resizing


MicroGR - Savage DRC System
File View Operations Macro Inspect Tools Options Help
Load Error
OK
Cancel
Help
Ready

```
width:          layer=&PWELL(1).
                value=4000;
outdistance:    layer1=&PACTIVE(
                layer2=&PWELL(1)
                value=6000;
outdistance:    layer1=
                layer2=
                options
                value=
outdistance:         la
                value=
indistance:     layer1=
                layer2=
                options
                value=

(* &ACTIVE(12) CHECK

width:          layer=&
                value=
outdistance:       laye
                value=
outdistance:    layer1=&PACTIVE(
                layer2=&NACTIVE(
                value=1000;

(* &POLY(0) *)

width:          layer=&POLY(0).
                value=1500;
width:          layer=&POLY(0).
                value=1300;
width:          layer=&POLY(0).
                value=1300;
width:          layer=&POLY(0).
```

```
Check          Type   Value                Number   Layer
Width          LT     2000                 512
OutDistance    LT     1500                 5479
OutDistance    LT     1750                 20
Width          LT     3000                 206
Width          LT     4000                 154
OutDistance    LT     7000                 15
OutDistance    LT     1500                 19555
```

Figure 2. Savage: DRC script and error logs are shown.

The convenient viewing subsystem is used to visualize the layout and examine the errors detected by the checking functions.

2. Optimizer

When **Savage** is run in batch mode, the input script is first processed by the Optimizer. It generates a new script and controls its execution. The optimized script takes into account the specifics of the processed layout, eliminates redundant duplicate checks, selects an optimal sequence of checks, etc. This effectively boosts the processing speed for an order of magnitude in comparison with line-by-line execution of the original input script. The heart of the Optimizer is an ingenious algorithm based on modern combinatorial optimization techniques.

3. Basic Operations

Savage operations may be invoked during an interactive session or processed in batch mode from user-supplied scripts.

* SELECT Operations are used:

- to select regions which are in a specified relation with the regions of another layer,
- to generate new layers from the selected regions for submitting them to checking operations.

The possible relations for Select operations are :

“inside”, “outside”, “cut”, “hole”, “touch”, “enclose”, “overlap”, “vertex”, “area”

* LOGICAL operations with layout layer supplemented in **Savage** are the following:

- intersection (AND),
- union (OR),
- difference (DIF),
- symmetric difference (XOR), and
- negation (NOT).

This set is both functionally complete and includes most natural logical operations.An important feature of Savage is an immediate calculation of complex logical formulas in terms of layout layers, which is performed many times faster than direct execution of the basic logical operations from the formula one by one.

* SPACING CHECKS

Savage implements the conventional basic checks:

- angle checks (Angle),
- intersection (Intersection),
- admissible internal dimension (Width),
- admissible internal distance (inDistance),
- admissible outer distance (outDistance),
- admissible overlap distance (ovDistance), etc.

More than one check may be run at a time.

In real applications most checks are performed on derived layer produced from the original ones by logical, selection, resizing, and other operations. Savage allows the user to perform these checks immediately, without creation of new layers, significantly decreasing the run time. Savage has two ways of representing errors: segment mode and region mode. In segment mode, the errors as segments are saved in an error file for further processing, viewing, or printing. In region mode, the error regions are saved in a new “error” layer. The way a region is treated as “error” is specified in the corresponding check command. Any error may be reported in either or both of the above modes.

* RESIZING operations allow to undersize or oversize a region by moving its edges inside or outside. They may produce forbidden configurations: self-intersections, overlaps, etc. Savage detects these violations and treats them as follows:

- It automatically corrects them whenever possible.

- Optionally, new layers are created containing all violations that might occur. The user may examine them together with the result of resizing and perform further corrections if he is dissatisfied with the automatic ones.
- Simultaneously with resizing, it is possible to perform a merge operation or some spacing checks for the created layers.

In contrast with some other PC-based DRC systems **Savage** processes all- angle geometries of any number of segments. It is flexibly tunable to various conventions on geometric representation of the layout. In particular, multiply connected regions and nested boundary polygons may be allowed. It is efficient for handling of both regular and irregular IC layouts and adaptable to circuit fabrication technologies.

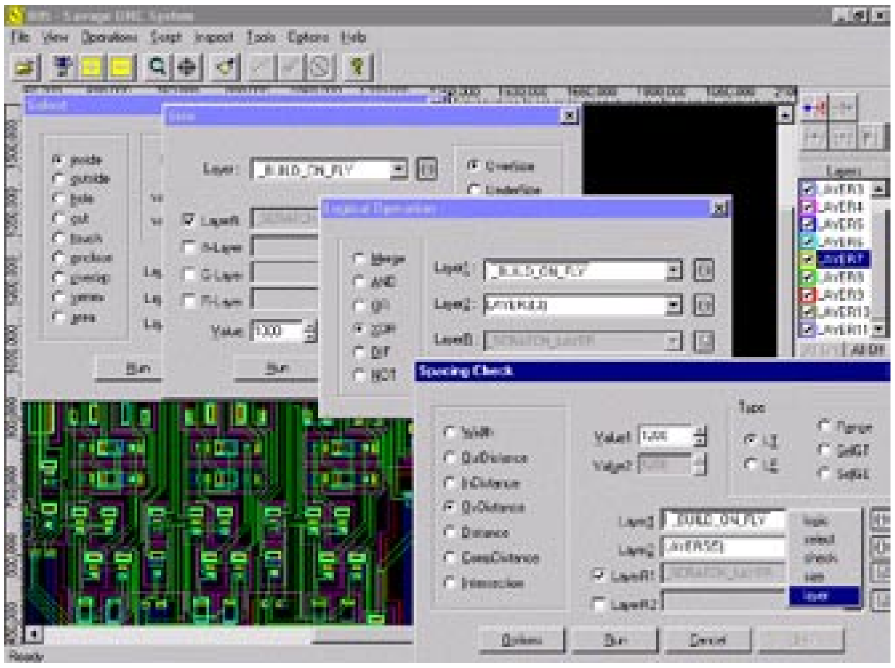


Figure 3. On-line creation of a complex DRC check operation.

Generalized Convexity Approach to Cell Boundary Shape Approximation

Abstract

Using generalizations of the convexity concept, an approach is described for approximating polygonal domain by simplifying its geometric shape. The problem arises in IC layout design as part of the hierarchical approach to design rule checking and device/ subcircuit extraction implemented in Silvaco's **Expert** and **Savage** CAD tools. Appropriate functions are based on algorithm of polynomial time complexity obtained for computing a generalized- convex approximation with a prior constraints on the number of vertices of the resulting domain. The quality of approximation is evaluated as area increase with respect to the original domain. This increase is minimized by solving a certain extremum combinatorial problem.

Problem Statement and Model Description

With the increase of the complexity of modern ICs it is impossible to achieve a speed-up of chip layout verification without taking cell hierarchy into consideration while designing. This is the case, e.g., for design of a DRC subsystem. It is a nice idea to localize checks within a cell or between a cell instance and its parent cell instance in chip's hierarchy, both to speed-up the checks and to avoid error report multiplication. However this idea works well only for ideal hierarchies with clearly separated cells. Unfortunately, the latter is not the case in real designs. Quite often cells from the same hierarchy level overlap, (e.g., to ensure connectivity). These overlap areas deliver pathological cases for DRC checks, for device extraction, and for any other hierarchy- based analysis tools.

A possible way to eliminate, or at least, minimize pathologies is to minimize these cell overlaps. A possible approach is to redefine the notion of the cell shape. Usually the shape of cell is considered to be a minimal rectangular box enclosing all objects of the cell. However quite often such box is not densely populated by cell's objects near its boundary. Therefore one might want to consider the cell shape to be the boundary of the union of regions constituting the cell. With this approach cell overlap will be reduced to an absolute minimum. A drawback of such definition is increased complexity as compared with box shape that would lead to slowing down various search-based operations. Therefore the goal is to find a trade-off between the complexity of the shape and the degree to which this cell shape approximates the actual area occupied by the cell.

The idea of approximating complicated geometric objects by those that a are of less complicated shape and easier to describe is the basis of many algorithms of computational geometry finding their multiple applications in image processing, pattern recognition, computer graphics, and, of course, VLSI layout design automation. The approximation of a set X of n-dimensional space R_n by a set H is called an outer approximation, if X is contained in H. The sets X and H are assumed to be measurable, and the quality of approximation is judged from the value of the *approximation density meas*(X)/ *meas*(H), called , where *"meas"* is the Lebesgue measure.

The classical convex hull is an example of an outer approximation which simplifies the shape of an object and performs better than the minimal bounding box in terms of approximation density. However for many classes of geometric objects important in practice the usage of classical convexity for this purpose gives poor results. Moreover, VLSI geometries are often characterized by so-called isothetic domains, the boundary of which is formed by segments parallel to the coordinate axes. the use of a conventional convex hull for domains of this kind violates the isotheticity condition.

These factors make classical convexity very difficult to use and force us to turn to wider classes of outer approximations considered in literature. A suitable class for the approximation of isothetic objects is orthoconvex objects, where orthoconvexity is convexity with respect to coordinate axes [1].

The widest and, therefore, most preferable class in the sense of density of approximation, S-convexity [2] was investigated.

Let us consider some formal definitions.

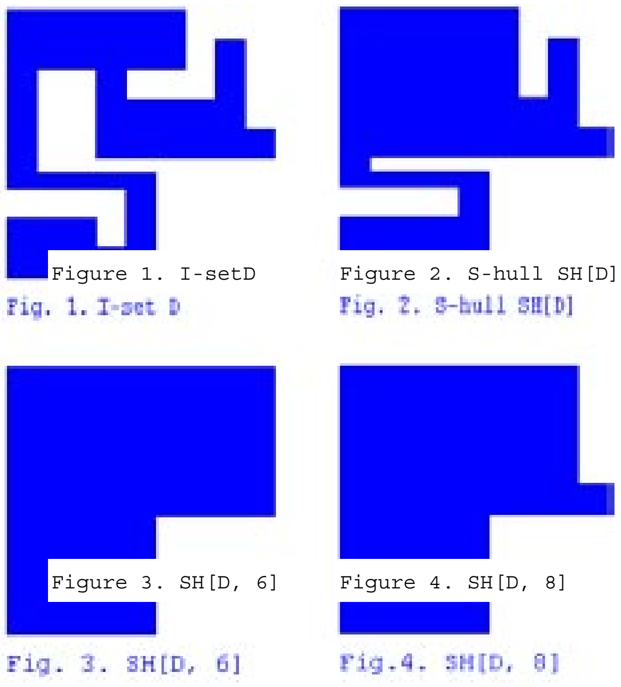
D1. An isotetic set (I-set) is a subset of Euclidian plane E^2 that can be represented as a finite union of axi-oriented rectangles, which might be degenerate (Figure 1.)

D2. An isotetic domain (I-domain) D is an I-set that can be represented in the form of a finite union of non-degenerate rectangles.

D3. The R-hull (denoted by RH[D]) of the I-domain D is the minimum axi-oriented rectangle (with respect to inclusion) containing D.

D4. The sider U of the I-domain D is the maximum rectangle contained in RH[D]-interior[D] having a non-empty intersection with the boundary of RH[D]

D5. S-hull (denoted by SH[D]) of the I-domain D is the isotetic domain obtained by removing all its siders from the R-hull of the I-domain D (Figure 2).



Figures 1-4. Non-rectangular cell shapes generated by Expert.

It is readily seen that the S-hull satisfies the convexity axioms. Therefore, S-convex hulls are defined with help of equation SH[D]=D.

The problem of optimal simplification of cell boundary shapes is in computing the S-convex approximation of highest density with number of vertices bounded by or reasonable constant r. Let us denote this approximation by SH[D,r] (Figures 3 and 4).

It was discovered that this original geometric problem is reduced to a particular discrete extremal problem, for which an algorithm based on dynamic programming was constructed.

The estimation of the complexity of the algorithm takes O(rN^2) operations what is acceptable for practiced designs.

References

[1] Rawlins G.J.E. and Wood D. Ortho-convexity and its generalizations, in: Computational Morphology, 137-152. Elsevier, 1988.

[2] Azarenok A., Martynchik V., Metelskii N. Computation of Generalized Convex Approximations, in: Comp. Maths. Phys., Vol. 33, No. 12, pp. 1641-1651. Elsevier, 1994.

Calendar of Events

September

1	
2	
3	
4	
5	
6	
7	
8	ISCS / SISPAD
9	ISCS / SISPAD
10	ISCS / SISPAD
11	
12	
13	
14	
15	RADECS ‘97
16	
17	
18	
19	
20	
21	
22	ESSDERC ‘97
23	ESSDERC ‘97
24	ESSDERC ‘97
25	
26	
27	
28	
29	BCTM - Minneapolis, MN
30	BCTM - Minneapolis, MN

October

1	
2	
3	
4	
5	
6	Intl. SOI Conference
7	Intl. SOI Conference
	European Symposium on Reliability
8	Intl. SOI Conference
	European Symposium on Reliability
9	Intl. SOI Conference
	European Symposium on Reliability
10	
11	
12	IEEE Intl Rel. W/S
13	IEEE Intl Rel. W/S
14	IEEE Intl Rel. W/S
15	IEEE Intl Rel. W/S
16	IEEE Intl Rel. W/S
17	
18	
19	
20	
21	
22	
23	
24	SmartSpice W/S - Tokyo
25	
26	
27	
28	
29	
30	
31	

Bulletin Board

Silvaco Expands R&D Operations!

To accommodate our rapidly increasing development staff Silvaco has acquired the fifth, new building at same Santa Clara location. Building 8 will house our CAD development staff. This expansion is necessary as our CELEBRITY product is gaining tremendous interest.

Silvaco To Exhibit Modeling Capabilities at IEEE International SOI Conference!

Silvaco International will be presenting the Complete Solution for SOI Modeling at the 23rd Annual IEEE International SOI Conference October 6th - 9th in Yosemite, CA. Silvaco application engineers will demonstrate our unique SOI modeling capability including process modeling and development, SOI device design, a full suite of SOI models implemented in SmartSpice (Berkeley, Honeywell, University of Southampton, Fully and Non-Fully depleted Fossum models), and Silvaco’s industry standard UTMOST parameter extraction and SPICE modeling software.

Visit the Silvaco International Booth in the Grand Tenaya Ballroom.

News from the TCAD World!

The recent acquisition of TMA by AVANT! leaves Silvaco as the only significant independent TCAD vendor committed and focused on process and device simulation development. We will continue to stay independent and we invite you all to join our expanding satisfied customer base.

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

The Simulation Standard, circulation 15,000, Vol. 8, No. 9, September 1997 is copyrighted by Silvaco International. If you, or someone you know wants a subscription to this free publication, please call (408) 567-1000 (USA), (44) (1483) 401-800 (UK), (81)(45) 820-3000(Japan), or your nearest Silvaco distributor.

Simulation Standard, TCAD Driven CAD, Virtual Wafer Fab, Analog Alliance, Legacy, ATHENA, ATLAS, FastATLAS, ODIN, VYPER, TSUNAMI, RESILIENCE, TEMPEST, CELEBRITY, Manufacturing Tools, Automation Tools, Interactive Tools, TonyPlot, DeckBuild, DevEdit, Interpreter, ATHENA Interpreter, ATLAS Interpreter, Circuit Optimizer, MaskViews, PSTATS, SSuprem3, SSuprem4, Elite, Optolith, Flash, Silicides, SPDB, CMP, MC Deposit, MC Implant, Process Adaptive Meshing, S-Pisces, Blaze, Device 3D, Thermal 3D, Interconnect 3D, TFT, Luminous, Giga, MixedMode, ESD, Laser, DGEM, FastBlaze, FastMixedMode, FastGiga, FastNoise, UTMOST, UTMOST II, UTMOST III, UTMOST IV, PROMOST, SPAYN, SmartSpice, MixSim, Twister, FastSpice, SmartLib, SDDL, EXACT, CLEVER, STELLAR, HIPEX, Scholar, SIREN, ESCORT, STARLET, Expert, Savage, Scout, Guardian and Envoy are trademarks of Silvaco International.

Hints, Tips and Solutions

Mikalai Karneyenka, Applications and Support Engineer

Q: How can I create the same regions in two layers simultaneously?

A: Consider a simple tutorial example. Suppose we want to create identical objects in two layers ME1 and ME2 (e.g., when creating bonding pads). A possible plan of actions is:

- Create objects in one layer, ME1;
- Select the objects to be duplicated in layer ME2;
- Save the selected objects into ME2.

The detailed actions are as follows.

- Select from menu: Cell>>New, and enter cell name"pad".
- At the filling bar (see Figure 1), we uncheck "selected" and check "all rest", because it will be convenient to see selected objects in wireframe mode and nonselected in filled mode.
- Click the layer name strip for layer ME1 at the Layer bar (Figure 1). This strip becomes colored indicating that layer ME1 becomes active, i.e., primitive objects may be created in it.

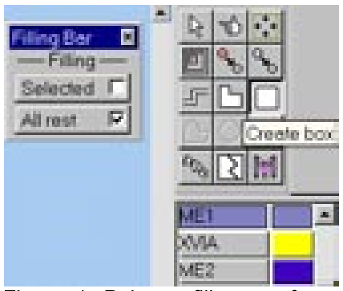


Figure 1. Polygon fill menu from the Tool Bar.

- Pick "Create Box" tool at the Tools bar (Figure 1) and draw some boxes, see Figure 2. You will see that the color strip for layer ME1 becomes checked, indicating that now layer ME1 contains some objects.
- Pick "Select" tool at the Tools bar (the second tool in the first row, see Figure 1) and click at the required objects to select them. They will change their color to white and filling mode to wireframe, see Figure 3.

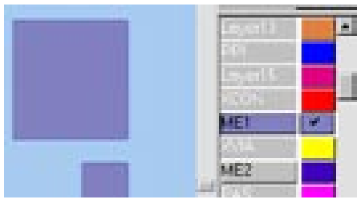


Figure 2. Objects drawn in layer ME1.

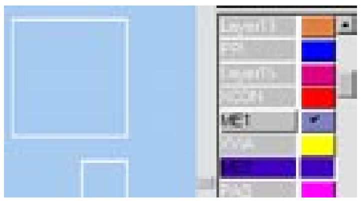


Figure 3. Wireframe display of selected polygons.

- Click the layer name strip for layer ME2 at the Layer bar to make it active.
- Select from menu: Select>>Save Selected>>To Layer. This command copies the selected object into the active layer. You will see that the color strip of layer ME2 at the Layer bar will become checked (see Figure 4), indicating that now this layer contains objects. Moreover, you will see that the selected boxes change their color (because the active layer overpaints all the remaining layers on the screen).

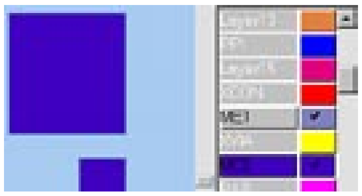


Figure 4. Objects copied to layer ME2.

You may wish to verify whether there are really two boxes lying one under another. The simplest way is to click AI (all invisible) button at the layer bar. You will see only the objects of the active layer (ME2). Then you make ME2 active and click AI button again. You will see only the objects of ME1. However if you wish to verify whether there are EXACTLY two boxes, you better use the Locator tool (the first one in the second row of the Tools bar, Figure 1).

You will easily see that there are only the required boxes at this place indeed, see Figure 5.

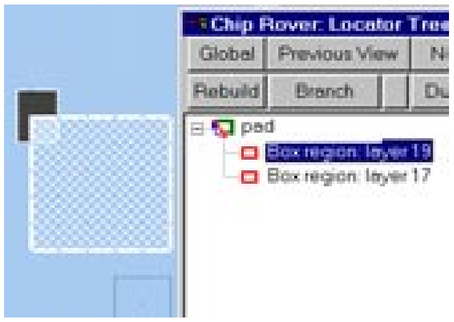


Figure 5. Using locator tool to verify copy operation.

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
http:://www.silvaco.com

SILVACO WANTS YOU!!

The last significant independent TCAD and SPICE developer invites you to join our team . Find out what our engineers are so excited about!

- TCAD Developers
- TCAD Application Engineers
- Software Developers
- SPICE Application Engineers

fax your resume to:
408-496-6080, or
e-mail to:
personnel@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

4701 Patrick Henry Drive, Building 2
Santa Clara, CA 95054

Telephone:	**(408) 567-1000**
Fax:	**(408) 496-6080**
URL:	**http://www.silvaco.com**