Simulation Standard

TCAD Driven CAD

A Journal for CAD/CAE Engineers

Advanced Features in Expert Layout Editing Tool: Parameterized Cells

Introduction

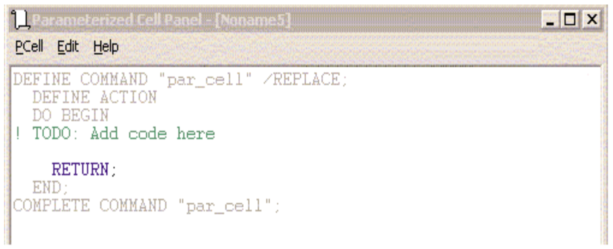
Silvaco ***Expert*** is an extremely powerful layout-editing tool that supports features related to parameterized cells. Parameterized cells, often called P-cells, increase designer productivity by adding enormous flexibility and efficiency to the design process. While standard cells help the designer to avoid repetitive drawing of identical pieces of layout, P-cells extend this functionality to the specific parameters that define the mask geometry. As a result, P-cells assist in the automation of layout design and help speed-up modification through the revision of P-cell parameters instead of wasting valuable resources by repeatedly redrawing the layout geometry.

Main Features

Expert's P-cell capability supports the move from traditional "polygon pushing" to more automated design styles with the following results:

- Increased design flexibility
- Increased productivity
- Easy maintenance of a small set of flexible components rather than a continuous reproducing of all mask geometry.
- Reduce design rule violations and minimize the effort of entry and editing with specific, parameterized data.
- Increased multiple, merged instances that are both DRC and LVS correct.

P-cells are created by writing a xi (***Expert*** Interface) script command in a special format. xi is an extension of the ***LISA*** (Language for Interfacing Silvaco Applications) scripting language that includes commands that execute ***Expert's*** actions. After the P-cell xi-script is created, compiled, and saved, P-cells appear in the cell lists of ***Expert's*** various commands. Here, they are available for viewing, placing their instances, or changing instance parameters through the standard cell-instancing dialog.



```
DEFINE COMMAND "par_cell" /REPLACE;
  DEFINE ACTION
  DO BEGIN
! TODO: Add code here

    RETURN;
  END;
COMPLETE COMMAND "par_cell";
```

Figure 1. New P-cell Panel

Create New Parameterized Cells

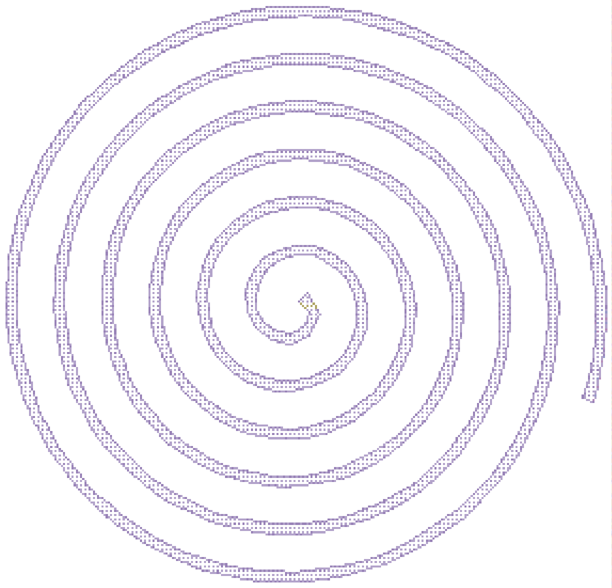
In order to create a P-cell, you must check the Parameterized cell option in the New Cell panel, in addition to specifying a cell name and selecting an associated library. Once the OK button is clicked, the xi-script panel and the empty layout window for the P-cell appears (see Figure 1).

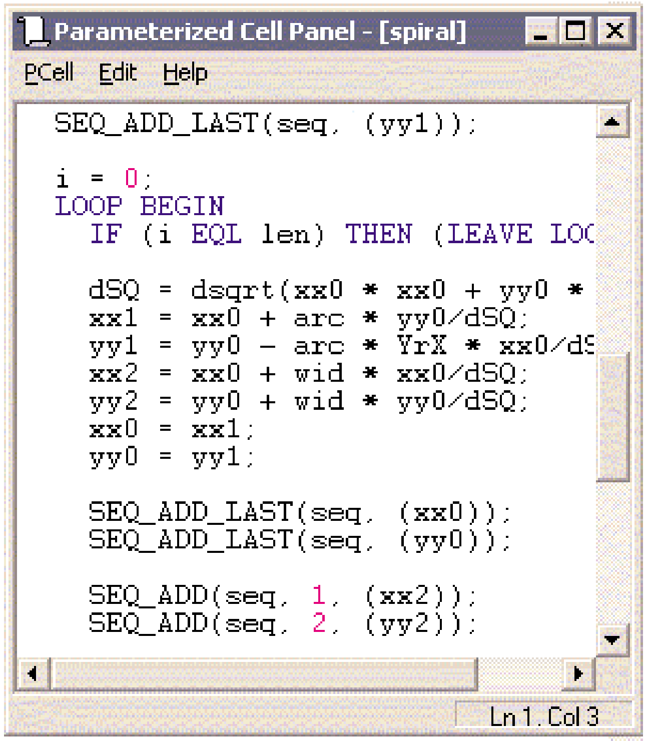
The New P-Cell Panel contains the xi-script that defines a command to generate the P-cell. Unlike the ordinary xi-script panel, the New P-Cell Panel does not support editing the "Define Command" statement syntax, which is automatically generated by EXPERT. Non-editable text is gray in color. All other necessary commands are typed inside the outermost Do Begin…End statement.

Continued on page 2....

INSIDE

SILVACO INTERNATIONAL





```
 SEQ_ADD_LAST(seq, (yy1));

 i = 0;
 LOOP BEGIN
   IF (i EQL len) THEN (LEAVE LOO

   dSQ = dsqrt(xx0 * xx0 + yy0 *
   xx1 = xx0 + arc * yy0/dSQ;
   yy1 = yy0 - arc * YrX * xx0/dS
   xx2 = xx0 + wid * xx0/dSQ;
   yy2 = yy0 + wid * yy0/dSQ;
   xx0 = xx1;
   yy0 = yy1;

   SEQ_ADD_LAST(seq, (xx0));
   SEQ_ADD_LAST(seq, (yy0));

   SEQ_ADD(seq, 1, (xx2));
   SEQ_ADD(seq, 2, (yy2));
```

Figure 2. Parameterized cell - Spiral and its script

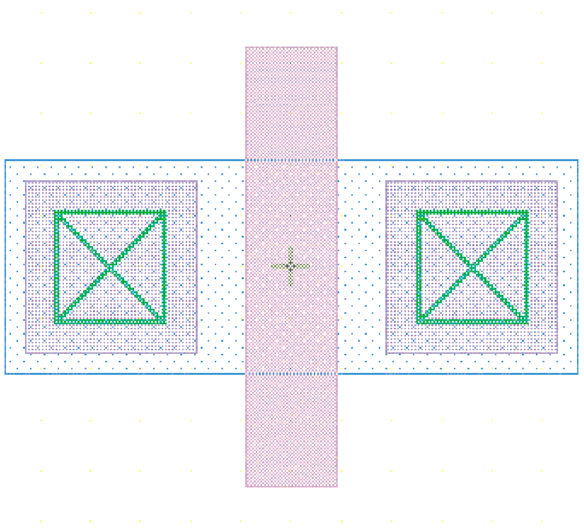
Different Geometry

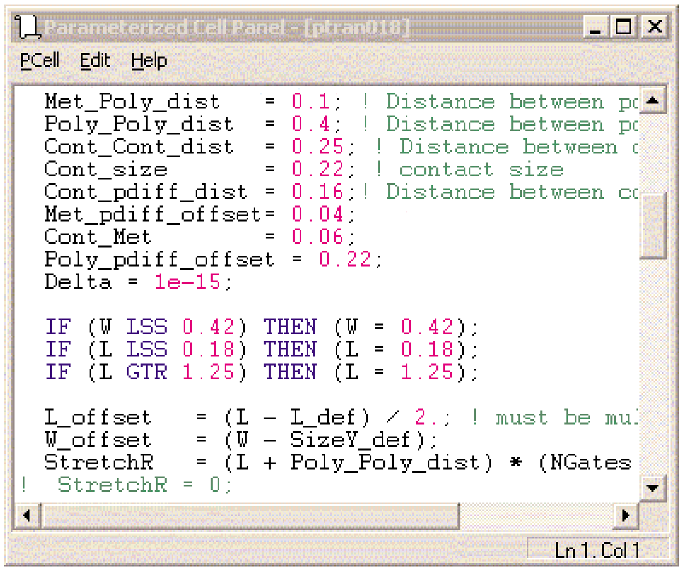
P-cells are used to create different geometry by generating different xi scripts. Figure 2 shows a spiral pattern created with P-cells, along with its xi script.

Parameterized cells (P-cells) are device generators and can generate CMOS single and multiple gate transistors, resistors, and capacitors. In Figure 3, a single P transistor of P-Cell is generated with its script.

Transistor P-cells change geometry length, width, the number of gate segments, and other variables by a simple parameter value adjustment. Figure 4 shows the creation of the P-cell's four P channel transistors, along with the change in the parameter value-gate from 1 to 4.

New P-cell control features help prevent less-than-zero gate numbers during instance creation.





```
 Met_Poly_dist    = 0.1; ! Distance between po
 Poly_Poly_dist   = 0.4; ! Distance between po
 Cont_Cont_dist   = 0.25; ! Distance between c
 Cont_size        = 0.22; ! contact size
 Cont_pdiff_dist  = 0.16;! Distance between co
 Met_pdiff_offset= 0.04;
 Cont_Met         = 0.06;
 Poly_pdiff_offset = 0.22;
 Delta = 1e-15;

 IF (W LSS 0.42) THEN (W = 0.42);
 IF (L LSS 0.18) THEN (L = 0.18);
 IF (L GTR 1.25) THEN (L = 1.25);

 L_offset    = (L - L_def) / 2.; ! must be mul
 W_offset    = (W - SizeY_def);
 StretchR    = (L + Poly_Poly_dist) * (NGates
!  StretchR = 0;
```

Figure 3 Parameterized cell - Single P channel transistor and its script.

Current New Features

1. Control of success

The LISA RETURN() command is used to monitor the computation of a P-cell instances, for controlling P-cell parameters and other areas requiring observation. All P-cell xi scripts now contain at least one RETURN() command at the end. RETURN() is usable in other places as well. Expert checks the value returned by xi script. If the returned value is NIL, then Expert creates the P-cell instance. If the returned value is not NIL, there are two possible results:

- If the p-cell instance is crated / modified by Expert's UI, then a dialog box is displayed:

  PCell 'name'(parameter_values) report: Return value: <ret_value>

  Reject Pcell? Y/N

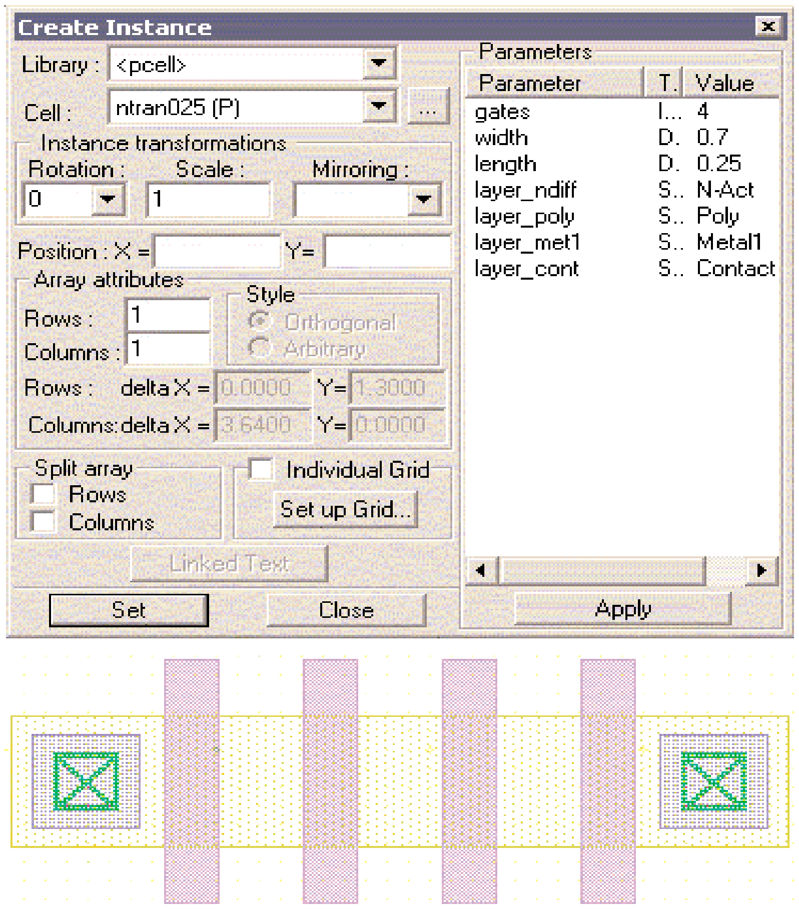
- If the p-cell instance is created / modified through a xi-script, then the return value for the last instance is checkable with the pcell_get_return_value()function.

2. Store the xi scripts and parameters of P-cells without compiling

The Apply command allows the user to save an active xi script and p-cell parameters without compiling the final cell. This is useful if script and parameter entries are incomplete and would compile incorrectly.

Conclusion

Parameterized cells (P-cells) extend traditional geometry cells to include great design flexibility and automation through the use of xi-scripts to process specific, user-defined input parameters. The use of P-cells help to dramatically increase designer productivity. P-cells permit layout changes by simple parameter modification and without the cumbersome process of redrawing the geometry. Changes are made easily through the ***Expert*** UI or through modification of the xi script.



Library : <pcell>
Cell : ntran025 (P)
Instance transformations
Rotation : 0 Scale : 1 Mirroring :
Position : X = Y=
Array attributes
Rows : 1
Columns : 1
Style: Orthogonal / Arbitrary
Rows : delta X = 0.0000 Y= 1.3000
Columns: delta X = 3.6400 Y= 0.0000
Split array: Rows / Columns
Individual Grid

Parameters

Parameter	T.	Value
gates	I...	4
width	D.	0.7
length	D.	0.25
layer_ndiff	S..	N-Act
layer_poly	S..	Poly
layer_met1	S..	Metal1
layer_cont	S..	Contact

Figure 4. Parameterized cell - Four P channel transistors and its script

Expert Wiring Tool: Router at Hand

Introduction

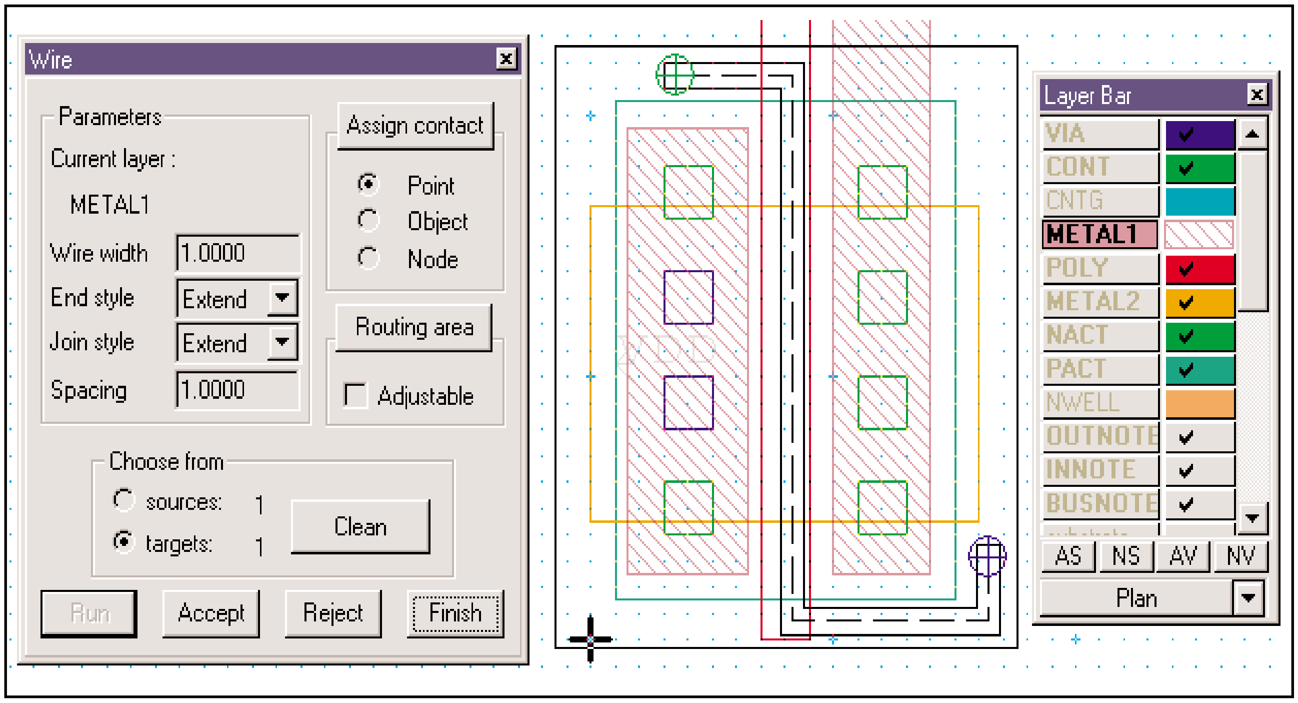
Modern complex circuit designs require automatic router functionality in circuit layout editors. In multi-layer hierarchical design, it is often necessary to build a single wire between two objects or arbitrary points. It is unpractical to call a heavyweight router that is hardly aware of the design settings and technology parameters. ***Expert's*** built-in Wiring Tool helps to bypass these time consuming operations, since all necessary information for routing is made available during ***Expert*** session.

The Purpose and Features of Wiring Tool

The purpose of ***Expert*** Wiring tool is to draw a wire in the active layer between one of the selected source contacts and one of selected target contacts (Figure 1). These source and target contacts are chosen by a routing algorithm in order trace the shortest route between them. Each of the contacts is a point, an object, or an electrical node. The resultant wire connects them electrically and avoids obstacles in the active layer in the whole hierarchy of the current cell.

To run a wire, select the **Tools > Wiring**... menu command. This launches the **Wire** panel. ***Expert*** Wiring Tool has the following features:

- Select an active layer to run a wire by clicking **Tools > Pick Layer,** or by choosing a layer from the **Layer Bar**. The **Current layer** name is then displayed in the **Wire** panel and all wires are built in this layer until altered.
- Customize wiring parameters from wire to obstacles in the **Wire** panel with the **Wire width, End** and **Join styles**, and **Spacing** options. Every time an active layer is changed, the wiring parameters are also changed in accordance with the technology parameters or customized settings.
- Select the **Routing area** by the corresponding button on the **Wire** panel or choose to make it automatically **Adjustable** by clicking on the correspondent checkbox. If a fixed routing area is chosen, a frame appears around it. In this case, the routing is allowed only in this area. This saves time and memory if its certain the area is routable.


Wire

Parameters

Current layer : METAL1

Wire width 1.0000

End style Extend

Join style Extend

Spacing 1.0000

Assign contact: [✓ Point] Object Node

Routing area: ☐ Adjustable

Choose from: sources: 1 [✓ targets: 1]

Layer Bar

VIA	✓
CONT	✓
CNTG	
METAL1	▨
POLY	✓
METAL2	✓
NACT	✓
PACT	✓
NWELL	
OUTNOTE	✓
INNOTE	✓
BUSNOTE	✓

Figure 1. Wiring between two points in fixed routing area.

- ***Expert*** divides all contacts by sources and targets. This indicates the direction of wiring. **Choose from sources** or **targets** in **Wire** panel.
- Contacts are points, objects, or electrical nodes. Select as many contacts in the layout as necessary. The numbers of source and target contacts are displayed in the corresponding strings. Click the same contact twice to deselect it. Click the **Assign contact** button to choose the type of wiring source or target contacts: **Point / Object / Node** (electrical).
- Click **Clean** to deselect all contacts.
- After the routing is defined, click **Run**. The wait time depends on routing area and design complexity. It often takes several minutes to build a draft wire for middle size design cell. The tool displays either the shortest possible draft wire between the selected contacts, or one of the warning messages, if wiring is impossible.
- **Accept** or **Reject** the draft wire.
- Hide the **Wire** panel by pressing **<Esc>**. The contacts and routing area stay untouched and ready for wiring. To drop all assignments, just click **Finish** button.
- Along with an easy to use interface, ***Expert*** implements advanced functionality in Wiring algorithm.

Technology Insight

Expert's Wiring Tool is based on a wave propagation router that employs a uniform routing grid for all source and target contacts. The Routing grid contains routing cells small enough to provide path between obstacles, but large enough to significantly save memory.

Expert Wiring tool encounters the traditional problem of trade-off between computational speed and memory usage. Addressing to machine word is considerably more speed effective. Taking this into account, the parameters of routing grid cell is described by the following data structure:

int: empty, source, target, obstacle

int: wave propagation queue number

bool: routed / un-routed

The ineffective memory usage is clear. The same parameters are described by the following compact bit-field structure:

bit 0-1: empty, source, target, obstacle

bit 2-3: un-routed, cyclic wave propagation queue #1, cyclic queue #2, cyclic queue #3

This compact cell allocation drastically saves memory. The wave propagation queue is cyclic and has depth=3. This is enough for flat wave propagation. Estimated memory usage gain for Win32 model multiplied twenty times when compared to the first data structure.

At first sight, access to the compact grid cell data may seem somewhat slow. This is because the data requires bit operations. However, compact data requires fewer CPU cache updates, and importantly prevents addressing to virtual memory at larger routes (larger routing grids). Addressing to virtual memory negates most advantages of the Wiring Tool.

A set of contacts is mix of points, objects, and electrical nodes in both sources and targets. Wave propagation from multiple sources to multiple targets locates the closest pair of these contacts in order to build the shortest possible route between them. Efficient connectivity between electrical nodes provides a low-cost solution for electrically connected circuit construction.

The routing algorithm recognizes hierarchical obstacles all over the project cell. Running wire avoids encountered obstacles whether in a simple, flat design or in a deeply encapsulated cell hierarchy.

Conclusion

Silvaco ***Expert* Wiring** tool:

- Reduces memory usage
- Connects multiple sources to multiple targets with one wire
- Connects wires to and from electrical Nodes;
- Avoids hierarchical obstacles.

CELEBRITY_C++, C++ Interface for *Expert*

CELEBRITY_C++, the C++ interface for Silvaco's ***Expert*** layout tool, is now an exclusive customization language that expands the tool's handling of high-level customization, such as a design application working with ***Expert's*** database. Users of the interface develop dynamic link libraries (DLLs) with Microsoft Visual C++ that are stored in the Silvaco install directory. ***Expert*** loads these DLL files and implements the user-defined commands in the menu bar.

Expert is already customizable with Silvaco's LISA/xi scripting language. However, ***CELEBRITY_C++*** extends these features. The following are some benefits of ***CELEBRITY_C++***:

- Enables functional access to the ***Expert*** layout database and commands from Visual C++.
- Permits high-level customization through the use of functions and classes that are prepared in Visual C++
- Easy implementation of graphic-user-interfaces (GUIs) such as dialog boxes, form windows, and toolbars.
- Compiles the source code to a fast-running dynamic link library (DLL) file.
- DLLs are easy to distribute and install.

Using *CELEBRITY_C++*

1) Create a new Visual C++ project

Take the following steps to create a new Visual C++ project:

1. Click File > New from Visual C++ menu bar to launch the "Create a new file/ project" dialog.
2. Click the "MFC AppWizard (dll)" icon and specify the project name.

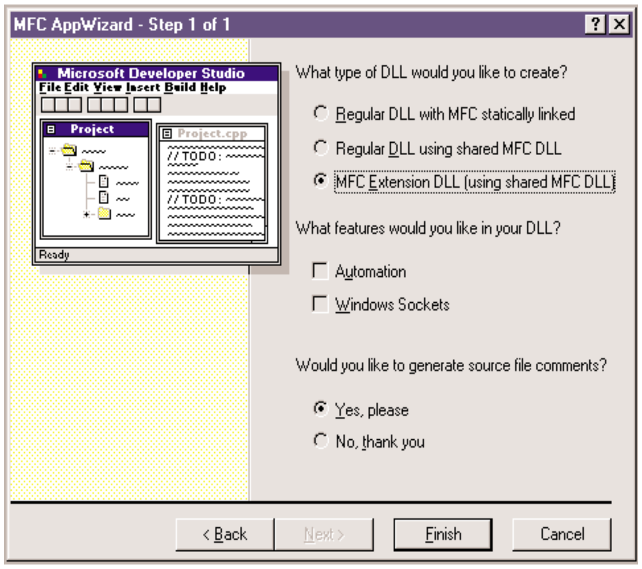

MFC AppWizard - Step 1 of 1
Microsoft Developer Studio
File Edit View Insert Build Help
Project
Project.cpp
//TODO:
//TODO:
Ready
What type of DLL would you like to create?
Regular DLL with MFC statically linked
Regular DLL using shared MFC DLL
MFC Extension DLL (using shared MFC DLL)
What features would you like in your DLL?
Automation
Windows Sockets
Would you like to generate source file comments?
Yes, please
No, thank you
< Back
Next >
Finish
Cancel

Figure 2. MFC AppWizard.

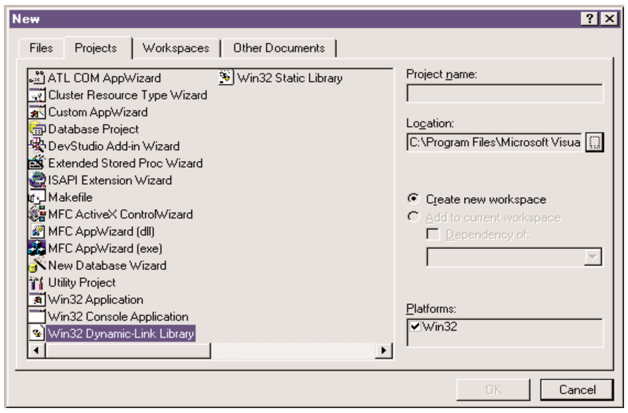


Figure 1. Create a new file/project dialog.

3. Choose "MFC expanded DLL (using MFC common DLLs)" radio button
4. Click "Finish" to complete the operation and open a new Visual C++ project.

2) Project Setting

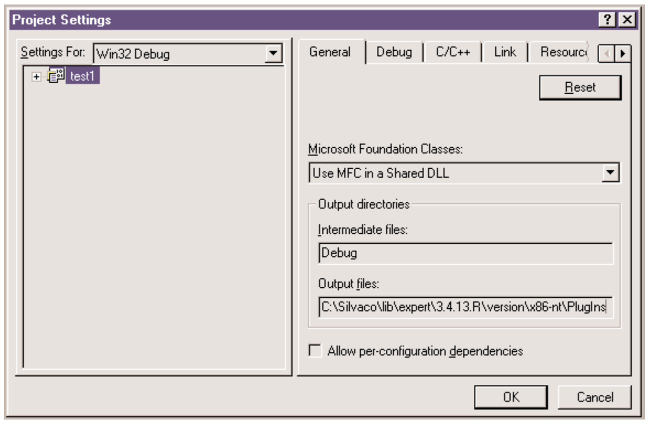
Certain library and header files are required to access Expert database and commands. Take the following steps to specify these files:

1. Click Build > Setting Active from the menu bar and select "<project name> - Win32 Release".
2. Expand the compressed file <SILVACO_INST_DIR>\examples\expert\plugset.zip to the ExpApi and Include folders.
3. Click Tools > Options from the menu bar and specify the path to the header files in the "Directories" page.
4. Click Project > Setting from the menu bar to set the following:

```
Output directory:
<SILVACO_INST_DIR>\lib\expert\<version>\x86
-nt\PlugIns

Executable for debug session:
<SILVACO_INST_DIR>\etc\GuiAppStarter.exe

Parameters for Executable: -lib-dir-name
Expert -exe-name Expert

Object / Library module:
<SILVACO_INST_DIR>\lib\expert\<version>\x86-nt\
```

5. Add the following lines in the Visual C++ project file StdAfx.h:

```
#include <afxtempl.h>

#include <afxwin.h>
```



Settings For: Win32 Debug

test1

Microsoft Foundation Classes: Use MFC in a Shared DLL

Output directories

Intermediate files: Debug

Output files: C:\Silvaco\lib\expert\3.4.13.R\version\x86-nt\PlugIns

☐ Allow per-configuration dependencies

Figure 3. Project Setting.

3) Writing source code

It is necessary to write commands that add the menu items to the DLL file. First, a new C++ source code file is necessary in order to write the program.

You can add menu items by generating CCustomMenu constructor, as illustrated in the following example:

```
void File1Out();
void File1OutAll();
void File2Out();
void FileIn();


CUSTOMENU_ITEM Items1[] =
{
  CUSTOMENU_ITEM("Current Cell(&L)",
&File1Out),
  CUSTOMENU_ITEM("All Child Cells(&L)",
&File2OutAll)
};


CUSTOMENU_ITEM Items2[] =
{
  CUSTOMENU_ITEM("Current Cell(&L)",
&File2Out)
};
CUSTOMENU_ITEM Items[] =
{
  CUSTOMENU_ITEM("Output File1(&L)",
Items1, EA_countof(Items1)),
  CUSTOMENU_ITEM("Output File2(&D)",
Items2, EA_countof(Items2)),
  CUSTOMENU_ITEM(),
  CUSTOMENU_ITEM("Load Files(&L)", &FileIn)
};
CCustomMenu cm(10, "Output Files", Items,
EA_countof(Items));
```

Menu items are defined as arrays of CUSTOMENU_ITEM. Add extended menus by specifying the name of another CUSTOMENU_ITEM and the number of items instead of the name of function. The function EA_countof()counts the number of items in an CUSTOMENU_ITEM array. Commands used in CUSTOMENU_ITEM should be defined in advance.

Finally, generate a constructor of CCustomMenu class by using the CUSTOMENU_ITEM array. The parameters of CCustomMenu::CCustomeMenu() are the position in the menu bar, the name of menu, the name of CUSTOMENU_ITEM array, and the number of items in the CUSTOMENU_ITEM array.

4) Compiling the program

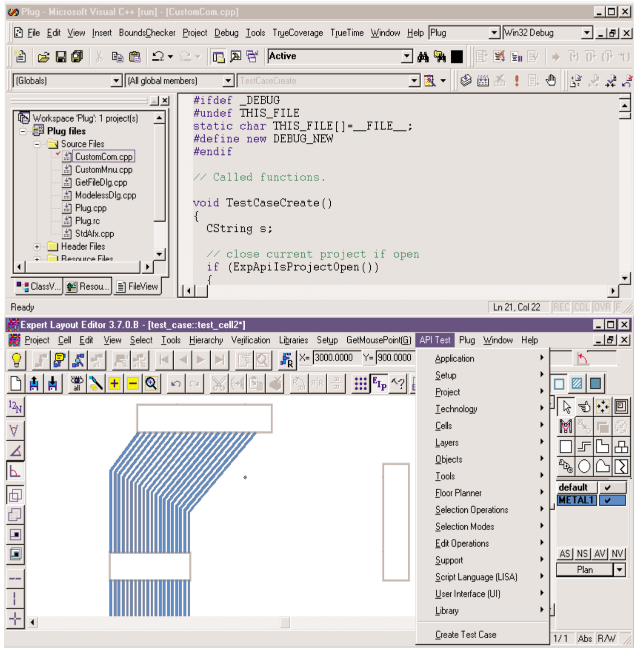
When the source code is complete, run Build > Build to compile the program into a DLL. When compiling is complete, start ***Expert*** to check if the program is working as intended. Place ready-to-deliver DLL files in the <SILVACO>\ lib\expert\<version>\x86-nt\PlugIns directory.

Conclusion: Comparing CELEBRITY_C++ and LISA / xi Scripts

LISA / xi scripts are also used to customize ***Expert***. xi is an extension of the ***LISA*** (Language for Interfacing Silvaco Applications) scripting language. ***CELEBRITY_C++*** performs most of the same features as a ***LISA*** / xi script. The option is dependent on the project's objective and scale.

LISA / xi scripts are easy to develop but are proprietary to ***Expert***. This makes them ideally suited for simple or small utility development. Despite its simplicity, ***LISA*** / xi is very flexible.

CELEBRITY_C++ is better suited for the development of larger, more complex applications. ***CELEBRITY_C++*** makes more efficient use of the processor, and is not modifiable in its compiled form.



```
#ifdef _DEBUG
#undef THIS_FILE
static char THIS_FILE[]=__FILE__;
#define new DEBUG_NEW
#endif

// Called functions.

void TestCaseCreate()
{
  CString s;

  // close current project if open
  if (ExpApiIsProjectOpen())
  {
```

Figure 4. Using users DLL in ***Expert***.

Calendar of Events

September	October
1	1 BCTM - Monterey, CA
2	2 Non-Stoichiometric IIIV Compounds - Monterey, CA
3	3 Non-Stoichiometric IIIV Compounds - Monterey, CA
4	4 Non-Stoichiometric IIIV Compounds - Monterey, CA
5	5
6	6
7	7 IEEE SOI Conf. - Williamsburg, VA
8	8 IEEE SOI Conf. - Williamsburg, VA
9	9 IEEE SOI Conf. - Williamsburg, VA
10	10 IEEE SOI Conf. - Williamsburg, VA
11	11
12	12
13	13
14	14
15	15
16	16
17	17
18	18
19 RADECS - Italy	19
20 RADECS - Italy	20
21	21 GaAs IC Symposium Monterey, CA
22 IIT2002 - Taos, New Mexico	22 GaAs IC Symposium Monterey, CA
23 IIT2002 - Taos, New Mexico	23 GaAs IC Symposium Monterey, CA
24 IIT2002 - Taos, New Mexico ESSDERC - Italy	24
25 IIT2002 - Taos, New Mexico ESSDERC - Italy NUSOD - Switzerland	25
26 IIT2002 - Taos, New Mexico ESSDERC - Italy NUSOD - Switzerland	26
27 IIT2002 - Taos, New Mexico NUSOD - Switzerland	27
28	28
29 BCTM - Monterey, CA	29
30 BCTM - Monterey, CA	30 LCD/PDP 2002 - Japan
31 BCTM - Monterey, CA	31 LCD/PDP 2002 - Japan

Bulletin Board

BCTM 2000

Monterey, CA is host to the 2002 Bipolar/BiCMOS Circuits and Technology Meeting September 29th through October 1st. This year's conference emphasizes the strong future of Bipolar/BiCMOS circuits and technology. Silvaco has exhibited at BCTM for many years and looks forward to many more successful conferences with them.

Visit us at IEEE SOI Conference

Silvaco is the world's leader in SOI modeling from the TCAD as well as the SPICE view. See our Virtual Wafer Fab design flow for SOI for a comprehensive process and device modeling solution. Visit us at the IEEE SOI Conference in historic Williamsburg VA in October 7-10.

Silvaco Returns to GaAs Conference

As a sequel to last year's blockbuster performance, Silvaco once again exhibits at the GaAs Conference. Held in Monterey from October 21-23, Developers, Salesmen, and Application Engineers will be present to bring you the future of GaAs design. Highlights included the ultrafast physical simulator ***FastBlaze*** and new HBT models in ***SmartSpice*** and ***UTMOST***.

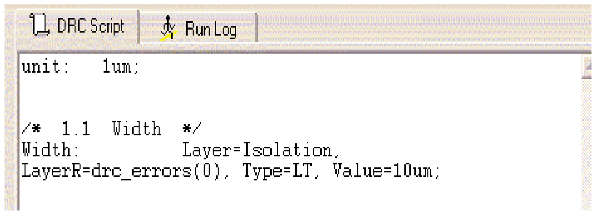
For more information on any of our workshops, please check our web site at ***http://www.silvaco.com***

The Simulation Standard, circulation 17,000 Vol. 12, No. 9, September 2002 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) 341-7220 (Japan), or your nearest Silvaco distributor.

Hints, Tips and Solutions

Sherry Rauen, Applications and Support Engineer

Q: After I create my round curve and run a DRC check on my design, it comes up with hundreds of errors associated with curves. For example, I have the layer named Isolation in the shape of a rectangle with rounded corners; the curves have been designed such that the outer and inner curves are concentric. The DRC then needs to check that the width of the feature is not less than 10um all the way around (shown in Figure 1). After the DRC script is run, it finds errors around the curves as the curves are approximated with straight lines (shown Figure 2). How can I solve this problem?



```
unit:   1um;


/*  1.1  Width  */
Width:          Layer=Isolation,
LayerR=drc_errors(0), Type=LT, Value=10um;
```

Figure 1. DRC Script.

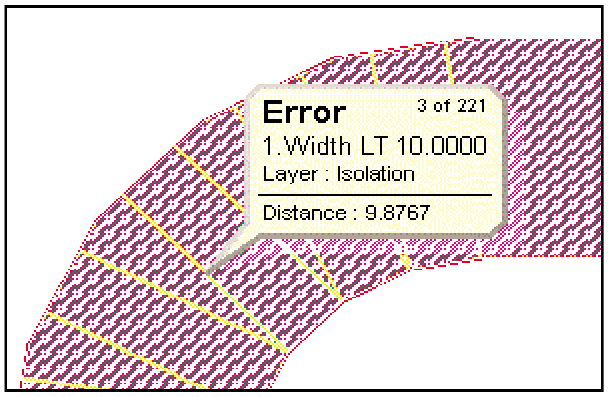
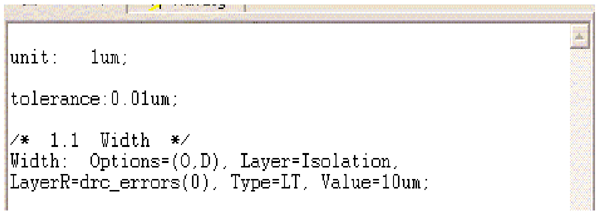


Figure 2. Curve Shape with DRC Error.

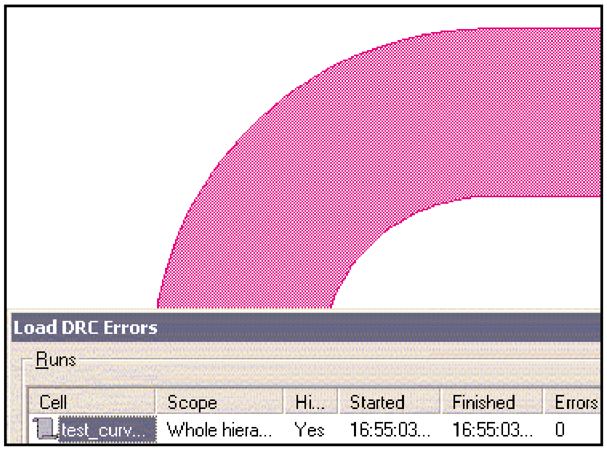
A: First of all, you need to create the better curve with larger vertex value, small grid, and polygonized object. If you go to Setup->Technology->General->Vertex Value = 100, the round corner will be drawn in much more smooth shape. Then if you setup grid as small as possible, select the object, and go to Tools -> Polygonize, all verities will snap to the grid, and the round corner is converted into polygon.

Second, before we run DRC script on round curve, we need to add the command on DRC script. Any curve with concentric outer and inner curves will not get perfectly width 10um with any layout tools. In order to avoid to get some faulted errors, we have to set the tolerance command into DRC script to pass DRC. As shown in Figure 3, I add tolerance:0.01um into DRC script. After running the new script, we get DRC error free on curved object as shown in Figure 4.



```
unit:   1um;

tolerance:0.01um;

/*  1.1  Width  */
Width:  Options=(0,D), Layer=Isolation,
LayerR=drc_errors(0), Type=LT, Value=10um;
```

Figure 3. The new DRC script with added tolerance command.



Cell	Scope	Hi...	Started	Finished	Errors
test_curv...	Whole hiera...	Yes	16:55:03...	16:55:03...	0

Figure 4 The new round corner curve with DRC error free.

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

Your Investment is Safe

18 Years and Growing

Financially Rock-Solid

Fiercely Independent

Analog EDA Design Leader

We are NOT For Sale

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

E*ECAD PARTNER

Products Licensed through Silvaco or e*ECAD