

Print



## PCell Caching in OpenAccess

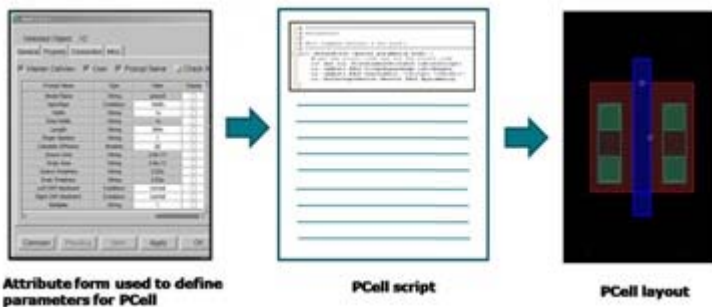
by Courtesy of SpringSoft

EDN

In computer programs, caching is used to store the output from commonly used functions on the disk so that, when executing a repeated instruction, the results may be obtained more quickly without having to reprocess the request. This same mechanism can be used to speed up the display of parameterized cells (PCells) in custom IC design. Some Electronic Design Automation (EDA) tools cache PCells automatically for performance reasons; some require additional licenses; and others offer no caching at all. In addition to the performance benefits, PCell caching can be used to make tool-specific PCells visible in other tools in the design flow.

### Where did my PCell go?

Used in the design of analog and custom digital circuits, PCells are software scripts used to define physical layout in a custom IC layout tool, based upon a prescribed set of variable parameters (Fig. 1.). PCells are the basic building blocks of custom design, offering a single programmable PCell in place of many different versions of a drawn cell. PCells can automate very sophisticated functions, maintain complex relationships and can even interact with their environment.



When a layout containing PCells is opened for viewing or editing by a layout editor, the tool evaluates each PCell script, generates the corresponding layout and holds it in memory. If any parameters are changed – either manually or by changing the parameters in the parameter attribute form – the layout editor must reevaluate the PCell and change the layout appropriately. In many tools when the layout is saved or closed, only the PCell instance and the instance-specific parameters are written to disk, forcing the tool to reevaluate the PCell each time it is opened.

Caching can be used to write the PCell layout to disk to make it available to any tool that is capable of reading the internal database language of the originating tool. Without caching, the PCell layout will disappear when opened by other tools, unless the other tools can also execute the PCell script.



### Proprietary scripting languages, evolving standards

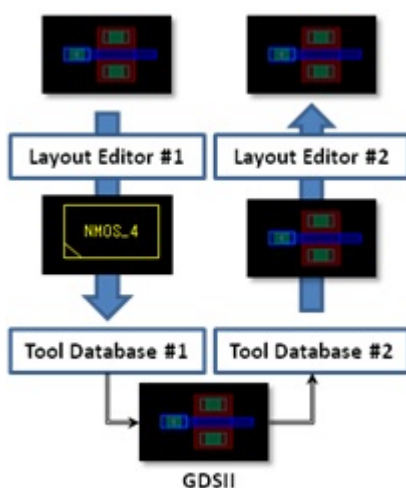
Historically PCells have been written in proprietary scripting languages (such as Cadence SKILL) developed for each layout tool, which means that most PCells in existence today cannot be “seen” by tools from other vendors because other tools do not have the software needed to evaluate the proprietary scripts.

Today this model is changing through the efforts of the Interoperable PDK Library (IPL) Alliance which has initiated a standard that is available to all vendors for executing interoperable PCells (See [www.IPLnow.com](http://www.IPLnow.com)). The PCells used in the IPL Alliance standard are written using the interoperable Python scripting language and are called PyCells by Ciranova, the company that developed them.

For the first time you can create PCells that can be opened and modified by almost any EDA tool.

### An Interoperable Database makes PCell Caching Viable

All EDA tools are built on underlying databases which perform the mundane function of storing and retrieving the semiconductor design data as it is being assembled. Until recently, EDA tools have been built on proprietary databases. If an EDA tool uses a proprietary database, the only way that layout can be “seen” by other tools is if it is converted to a universal semiconductor design format like GDSII. During the conversion process, the original tool evaluates the PCells and their parameters and writes the physical layout to the geometric GDSII format, discarding many of the properties and parameters that are associated with the PCell.

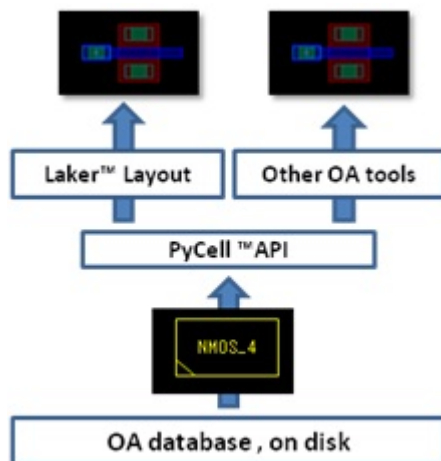


Once converted to GDSII, the layout is just like any other layout cell because in most cases, even the original tool cannot recognize this layout data as a PCell: GDSII conversion is a one-way process for PCells. In order for the user of a tool based on a proprietary database to view or modify their own PCells, they must continue to pay annual licensing fees to the tool vendor for the life of the design! This is a “tool tax” for which no user ever voted.

In recent years, OpenAccess (OA), the interoperable database standard from Silicon Integration Initiative (Si2) has gathered momentum for use in tools for custom IC design of semiconductors. This database allows tools to read and write to the same database and, in some cases, to work from the same memory model, in real time.

PCells cached in an OpenAccess database are visible to every other tool that is OA compliant. Virtually every major layout editor and most custom design tools in the EDA industry are able – or soon will be able – to at least read and write to the OA database. But what do I do if I want to modify a cached proprietary PCell from another vendor? More on this in a moment.

The OA database is also the foundation of PyCells which may be used by any OA-based tool by using the PyCell API and do not require caching to be visible when using other tools. PyCells are completely interoperable when using the Interoperable Component Description Format (iCDF) and Tcl callbacks from the IPL Standard in IPL-complaint tools.



### PCell caching and how it works

The compiled PCell code, ready for use by your custom IC design tool is referred to as the PCell “supermaster.” The supermaster contains no parameter values, just variables that are supplied through the Component Description Format (CDF) - or Interoperable CDF (iCDF) file, depending upon the tool – during evaluation.

When a PCell is manually placed or “instantiated” in the layout, the default parameters are read from the CDF file and the tool creates an instance-specific version of the cell layout in memory. This version is called the PCell “submaster.” Whether the default parameters are retained, or whether they are modified in some way, these parameters are stored as properties of the instance-specific submaster. When the PCell is evaluated, the layout editor will evaluate the submasters, using the instance-specific parameters, write the layout to memory and present the layout for viewing or editing. The next time the layout is opened, the PCells have to be reevaluated just to create the identical layout over again. As you can imagine, in a layout with many thousands of instances, this can take some time.

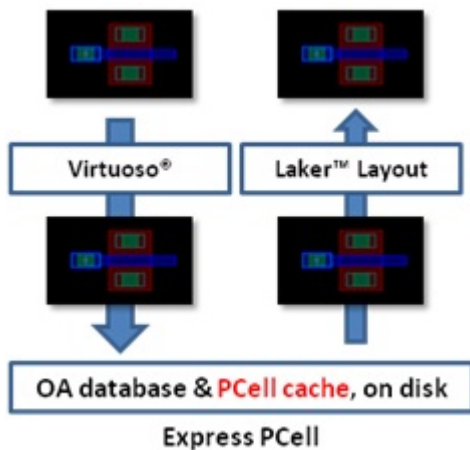
However, if the tool saves the submasters to disk (caching), the reevaluation does not need to take place each time, thus dramatically speeding up the “drawing” time of the layout. In addition, each time a submaster is created, the tool will first look in the cache for an identical submaster before evaluating the PCell and creating a new layout. Even if the same PCell with identical parameters is used thousands of times (a contact cell, for example), only one version of the layout needs to be cached, saving even more time and disk space.

### Caching SKILL-based PCells for interoperability

When interoperable PCells are not an option, or when there is a substantial, existing library of legacy PCells, caching can be very useful for a multi-tool design environment, or during transition to newer tools. There are two major offerings in this area:

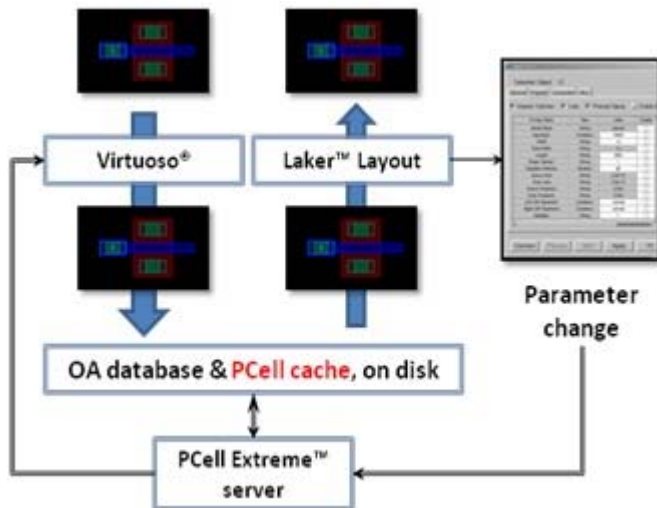
**Express PCells:** Many of today's PCells are written in the proprietary Cadence SKILL scripting language. Even if they have been written for use in the newer Cadence OA versions, SKILL PCells are not natively visible in other tools because they are not automatically cached. In order to "see" them in other tools, Cadence Express PCells may be used to cache the cells in the OA database. This makes the cells visible – read-only – to other tools such as non-Cadence DRC tools.

In some tools it is possible to see the attribute form that contains the instance-specific parameters, but it will not be possible to modify the PCells in other tools without first flattening them into simple layout instances. In addition, stretch handles, auto-abutment, and callbacks will not be available in the "foreign" tool. (For more information about PCell functionality, please see the article *A Silicon-proven Interoperable PDK* located in the technology section of the SpringSoft website.)



For users who use non-Cadence DRC tools, for example, running DRC's directly from OA requires Express PCells. In a multi-tool design flow, where different layout editors have been used to layout different blocks, Express PCells are a viable solution for assembling the different blocks into a single layout database. Further modification to the Cadence-based blocks, of course, would require replacement of the entire block, or at least the modified PCell instances. This is effectively no different than a typical SOC flow where blocks may have been designed in different design centers or even different companies such as an IP supplier.

**PCell Xtreme:** PCell Xtreme from Ciranova also works with the Cadence environment in the same way as Express PCells to cache SKILL PCells and speed up the time to open a design. However, it differs from Express PCells in that the PCell Xtreme server enables other tools to read and modify the PCell parameters in addition to viewing the cached PCell. PCell Xtreme does not translate the PCell code or evaluate it; if there is no cached PCell with the parameters requested by the second tool, PCell Xtreme will spawn a Cadence process and the SKILL PCell evaluator will generate the modified layout which will be cached. The new cached layout is then available to the second tool and will redraw in real time. Thus parameter changes made through the attribute form in the second tool will be honored by both tools. This method requires that the user have at least one licensed set of Cadence tools for evaluation of the SKILL PCells.



However, it should be noted that PCell Xtreme doesn't enable SKILL callbacks. Callbacks will be a critical element of many of your PCell parameters because they are used during the entry of parameters in the parameter attribute form to calculate relative values, out of bounds values, and more. Consequently, simple parameter passing is insufficient. To enable callback functionality in a non-Cadence tool, a duplicate set of interoperable callbacks like the Tcl callbacks defined in the IPL Alliance Reference flow 1.0 must exist for the second tool. Dual callbacks are a requirement for any interoperable PCell to work in the Cadence environment, including the IPL libraries, so it is expected that the Tcl callbacks will exist in most interoperable environments anyway. With both sets of callbacks in place, full parameter interactivity is enabled.

Because stretch handles and auto-abutment are OA properties that are cached along with the layout, it is possible for users to write equivalent stretch handle and auto-abutment procedures in Tcl or other interoperable scripting languages that will match the SKILL functionality. Moving a stretch handle causes the layout editor to update the related parameter, triggering any related callbacks. The parameter changes cause PCell Xtreme to generate and cache the modified layout in the same way as previously described for manually-entered parameter changes. Auto-abutment may be enabled in a similar fashion.

## Summary

PCell caching is a viable way to make use of legacy layout data in OpenAccess-based tools and offers the user a lot more interoperability than offered by a GDS II conversion of parameterized cells.

Although the most interoperable solution is to use interoperable PyCells, callbacks, CDF and stretch handle/auto-abutment technology available from the IPL Alliance, PCell caching is a viable way to make use of legacy PCells in a new OA-based tool, mixed-vendor design environment or just to speed up existing tools.

## Print

© 2010, Reed Business Information, a division of Reed Elsevier Inc. All Rights Reserved.