

Print



A Silicon-proven Interoperable PDK

by By Rich Morse, SpringSoft, and Tom Quan, Taiwan Semiconductor Manufacturing Co.

EDN

On July 21, 2009, TSMC announced the availability of the industry's first interoperable process design kit (iPDK). The kit was fully validated on TSMC's 65 nanometer (nm) MS/RF process and all major EDA vendors announced their support, including Cadence, Ciranova, Magma, Mentor, SpringSoft, Synopsys, and others. On March 24, 2010, the Interoperable PDK Libraries (IPL) Alliance of which TSMC is a member, released the IPL 1.0 standard, and made the underlying technology for the TSMC iPDK available to the entire industry.

THE PROBLEM WITH PDKS

The original use of the term Process Design Kit (PDK) referred to a PCell library supplied by a foundry. Because there were always questions about which design rule checking (DRC) deck or simulation model should be used with each PDK, PDKs eventually grew to include all of the documents and design infrastructure elements necessary for custom IC design. Key PDK elements now include: electrical design rules; simulation models of various kinds; manufacturing design rules, DRC and LVS run set files for tools that verify physical and electrical layout; and schematic symbol libraries, parameterized cells (PCells), a layer-map and technology (tech) file for layout tools.

Until recently all PDKs were EDA vendor specific in addition to being dedicated to a specific foundry, technology node and process variant. Today, all major providers of custom IC design tools including SpringSoft, Cadence and Mentor Graphics collaborate with TSMC on joint development and validation of PDKs. However, while proprietary PDKs have been available for many years, today the number of custom IC design tool suppliers is increasing, the number of supported technologies continues to grow, and the PDKs for advanced technologies have grown increasingly complex (Fig. 1). As more tools and more complex processes become available, supporting and qualifying PDKs for all the various combinations has become a difficult problem

for large foundries like TSMC which released 2500 new and modified PDKs in 2007 alone!

The solution to this problem for TSMC was to explore the possibility of interoperable PDKs (i.e. PDKs that can be directly used by multiple vendor tools with identical results) to reduce the support issues and to offer more choices to their customers. One of these solutions was to create an open PCell infrastructure.

Increased Complexity

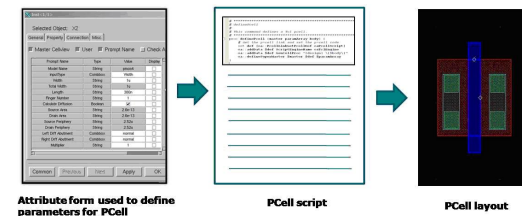
Comparison of 0.18µm and 65nm PDK

Items	0.18µm RF	65nm RF
Device Number	125	590
Utility Number	0	16
MOS p-cell code CDF Option	10	68
QA Time	15hrs	90 hrs
MOS Call-back Function(lines)	275	4000
DRC (lines)	4016	23464
LVS (lines)	3867	25574

Over 2,500 design kits released in 2007!

WHAT IS A PCELL?

Used in the design of analog and custom digital circuits, PCells are software scripts used to define physical layout based upon a prescribed set of variable parameters (Fig. 2). PCells are the basic building blocks of custom design, offering a single programmable PCell in place of many different versions of a drawn cell. By substituting different values for specified dimensional variables (parameters), layout engineers can create an almost unlimited number of variations. For example, a single PCell for an NMOS transistor can be used to create many different NMOS devices simply by changing the gate length parameter for each particular placement or “instance.” The EDA tool will then automatically generate the correctly sized PCell layout based upon this new parameter, from a single PCell and without the user having to “draw” a single shape by hand.



Custom circuits are designed in schematic form using symbols provided in the PDK that correspond to specific PCells which have specified default values associated with them. Before layout begins, the values for each parameter of each PCell instance can be modified by the circuit designer based upon simulations run using simulation models provided in the PDK. When the corresponding PCells are placed in layout, the layout generated by the layout system will

automatically reflect the adjusted values and are expected to yield silicon results that match the simulations.

Like the PDK itself, PCells require a number of infrastructure sub-components that enable more advanced functions. These include embedded functions like stretch handles and auto-abutment and separate files for “callbacks” and the “CDF database” which will be described later.

While complexity varies by company and application, PCell functionality is defined by and limited to however much code a PCell developer wants to write. PCells can automate very sophisticated functions, maintain complex relationships and can even interact with their environment. In advanced processes like those developed by TSMC, design rules are becoming increasingly complex and intra-device relationships so sensitive that users increasingly depend on the TSMC PDK to enable world-class yields. Advanced PCell algorithms provide differentiated value for TSMC PDKs.

THE BEGINNINGS OF A SOLUTION: A COMMON DATABASE

Fortunately a lot of work has already been done to help enable interoperable PDKs. The first element of the solution is the OpenAccess (OA) database, created by Cadence and donated to the Silicon Integration Initiative (Si2, www.si2.org) standards organization around 2001. Now OA is becoming the de facto database for custom IC design tools.

In the past, EDA companies developed proprietary databases which had to be translated – usually with some loss of functionality – to be read by tools from other vendors. Today, there are over 30 major EDA vendor companies that belong to the Si2 OpenAccess Coalition who, along with major users, are collaborating to develop and maintain this database standard. EDA tools that are OA-complaint are able to interoperate without data conversion and, in some cases, in real time.

INTEROPERABLE PCELLS

The second element necessary for an open PCell infrastructure is an interoperable scripting language. The OpenAccess standard features plug-ins for many open standard scripting languages like Tcl, C++, and Python. With enough code, any of them could be used to create PCells that are readable by OA-based tools. However Ciranova – one of the founding members of the IPL – has already laid the groundwork for interoperable, OpenAccess-based Python PCells which they call PyCells.

When the IPL was founded, the founders agreed to take advantage of these emerging

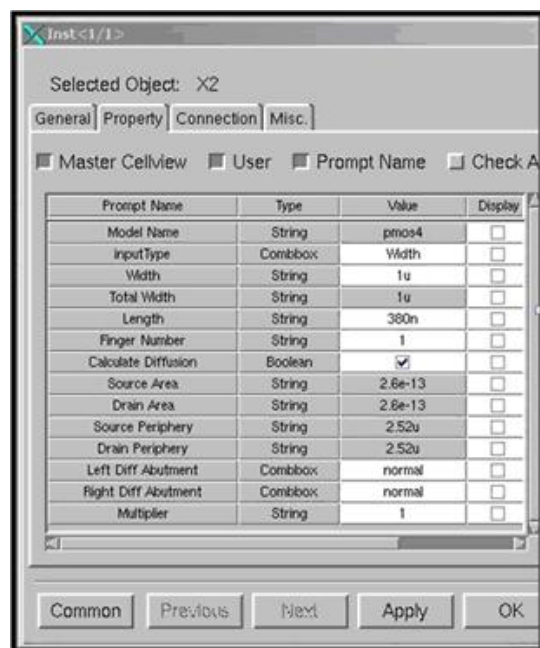
technologies to develop and release a proof-of-concept PCell library that was soon operating in eight different tools from five different EDA companies. This was a persuasive demonstration showing that it was possible to create interoperable PCells with existing technology. Recognizing the potential of OA and PyCells, TSMC agreed to partner with a group of IPL members to create a prototype project.

PYCELLS

Ciranova chose to use the Python scripting language instead of Tcl because of its modern capabilities like object-oriented programming, the availability of Python programmers, and faster run times. PyCells not only have significantly fewer lines of code than typical PCells, but also provide performance improvement when compared to legacy PCells. PyCells also support advanced features such as abutment, stretch handles and DFM rules. The free PyCell Studio download available from Ciranova provides an interactive environment for PyCell development and efficient PyCell debugging thereby improving PCell development productivity and shortening cycle times. In addition, PyCell Studio supports dozens of PCell-specific Python extensions that enable customization, and support advanced PDK features.

WHAT IS CDF?

Component Description Format (CDF) essentially describes the contents of the Parameter attribute form which is where PCell parameters are displayed and made available for modification by the user in commercial Schematic and Layout editing systems (Fig. 3). It is critical for all tools use a common syntax so that this very basic information can be communicated, regardless of the source.



This CDF file stores the following items:

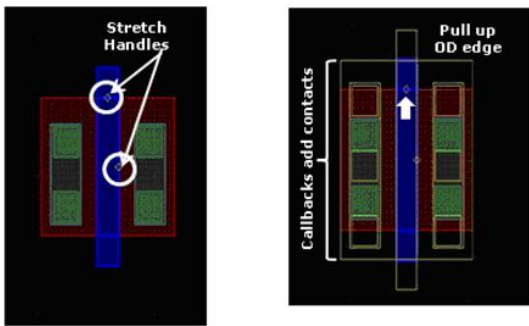
- Parameter names like length, width, space etc.
- How parameters are displayed in the parameter attribute form; which parameters are displayed; and which are editable by the user
- Default values for the parameters in the PCell

An original compiled PyCell in the iPDK includes the CDF, stored in the OA database as a property of the PyCell. However, each time a symbol is placed in the schematic or a PCell is manually instantiated in layout, a parameter attribute form is automatically opened which displays this CDF information. Once a symbol or PCell is placed (instantiated), all of the CDF information plus any changes made to the default values are stored in the OA database as a property of the instance. Later changes to the values in the CDF will not be automatically reflected in placed instances and vice versa.

The contents and functions of CDF are fairly generic in that all tools provide the same or similar functions but, in order to make the CDF interoperable, all of the IPL members had to agree to a specific syntax and definition of each term. As with other elements of the IPL standard, Synopsys had already done some work to develop a CDF-like syntax called “iparams” that was subsequently donated to the IPL and is now called “iCDF” (interoperable CDF) to avoid confusion by users.

CALLBACKS

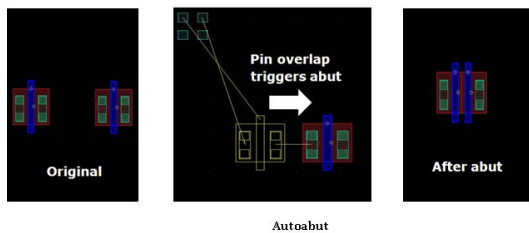
Advanced PCells offer the ability to substitute formulas or functions (“callbacks”) for some variables so that necessary relationships between the shapes are maintained. At a simple level, the callbacks can not only control the input of legal values (like $X > 0$) but can define what the tools should do if an illegal value is entered in the parameter attribute form: should it be a warning message, or should it automatically supply the nearest legal value, or both? In addition, callbacks are used to maintain dependent parameters: In this example (Fig 4.), when the poly gate width is increased by stretching the OD (diffusion) layer, the callbacks maintain the poly endcap dimension, makes the change symmetrical to the center of the transistor and adds contacts when there is enough space.



Callback function names are usually stored in the CDF (called “form field callbacks”) but the callbacks themselves, like PCells, are written in a scripting language like Tcl or Python. While it was tempting to use Python for the callbacks, too many of the IPL partners already had substantive investments in Tcl callback technology and as before, the alliance went with existing, proven technology as the path of least resistance and most likely success.

STRETCH HANDLES AND AUTOABUT

Stretch handles appear in PCell layout as small diamonds or squares which mark the edge of a shape as being stretchable. Stretchable edges are tied to a specific parameter like gate length or width as in Figure 4. Stretch handles allow the user to manually adjust the marked edge in the layout editor. Doing so changes the value of the associated parameter in the parameter attribute form so that any applicable callbacks can be executed – such as the number of contacts in the example – and the layout is dynamically adjusted in real time to meet all of the dependent criteria.



Autoabut refers to the ability to automatically merge two PCells that have designated connectivity. This is often done to share diffusion areas of transistors to reduce the size of the layout. The cells may also be “un-abutted” to split them back into two separate transistors again.

In order to stretch or abut a PCell, the behavior is described in a Tcl file and called from the PCell script.

KICKING OFF THE PROJECT

When TSMC teamed with its EDA partners to develop interoperable PDKs, much of the basic

foundation for PCells had been established, but it was essential for this iPDK team to have real-world PDK specifications to work with, ideally an established, production-proven PDK on an advanced process with a complex PCell library. TSMC's 65nm Mixed-Signal RF single-vendor PDK, developed with proprietary PCells fit these requirements perfectly. There were 480 PCells in this PDK; the most complex MOS transistor had 68 CDF options; and the callback functions included more than 4,000 lines of code.

THE IPDK PROJECT

In the early stages of the iPDK project, all of these PDK pieces were tested and agreed to: OpenAccess, PyCells, PyCell Studio, iCDF and Tcl callbacks. Weekly meetings were held to evaluate progress while multiple companies in multiple offices around the world set to work to create and validate a TSMC 65nm MS/RF iPDK. In the background, entire tool sets had to be configured to meet the evolving standard and to overcome the real world challenges that cropped up during development.

As an initial proof of concept, TSMC chose two MOSfet devices which represented the most difficult challenges in the library. Together the two PCells had 89 CDF options and a relatively large amount of callback code. To validate the results, layout was generated that fully exercised all of the parameters and callbacks and compared to the existing production-qualified PDK. During this process, it became obvious that there is more than one legal way to create each version of the PCells and the only practical way to validate them – and eventually the entire library – was to force the PyCells to match the legacy PCells even if it required more lines of code.

Once the proof-of-concept cells were qualified, TSMC selected a set of 93 PCells which represented at least one of every discrete type of PCell in the library. During development a few previously unavailable features were added to the PyCell library in addition to matching the existing PDK. Once the 93 PCells were validated, the decision was made to complete the entire library of PyCells and ready a production-worthy iPDK.

About this time the TSMC joined the IPL as the first foundry member. At the 2008 Design Automation Conference (DAC) the first multi-vendor interoperable PDK was demonstrated using a partial TSMC iPDK PyCell library and a small low noise amplifier (LNA) device. Eight tools from five different EDA vendors were shown modifying and passing design and layout data back and forth in real time without any data translations.

Qualifying the first iPDK represented some unique challenges. Since it was an entirely new methodology, the testing needed to be much more rigorous than for a previously qualified single-

vendor PDK. It was necessary to test all of the possible different combinations of parameters and callbacks in order to verify that the iPDK could work in every situation.

The TSMC iPDK validation was extremely rigorous. The final qualification test suite included 480 PyCells, varying from 200 instances (inductors) up to 200,000 instances (MOSfets) per cell which created output data sizes of over 300GB. Each discrepancy, no matter how small, had to be analyzed and accepted or rejected. If rejected, the PyCell, CDF, or callback had to be modified, and the validation re-run. Eventually the entire library was validated.

Once the iPDK was fully qualified on one tool, the same tests were run on other EDA Layout tools, comparing them to the qualified output from the first tool. When run from the same iPDK, XOR error outputs (a DRC-like check that flags any differences in layout) have been consistently clean.

In addition, TSMC was also able to test the iPDK to verify that the iPDK worked properly in the Cadence Virtuoso platform when the CDF and SKILL callbacks from the existing PDK were mated with the PyCells from the iPDK.

THE WORLD'S FIRST IPDK

The fully qualified TSMC 65nm Mixed-Signal RF iPDK was announced at DAC in 2009, and support for it was announced by every major vendor. Multi-vendor interoperability demonstrations were available from many vendors throughout the show as well as the TSMC Open Innovation Platform booth.

TSMC's unified iPDK works across multiple OpenAccess-based EDA design environments, eliminating the need for multiple proprietary PDKs, and enabling full reuse of design data between different custom IC design toolsets.

The TSMC iPDK is based on and was instrumental in certifying, the IPL standard. It employs the OpenAccess database and data model from Si2. It features open standard languages, Tcl and Python, for parameterized layout cells, and callbacks. The iPDK also includes SKILL callbacks and component description format (CDF) files to provide compatibility with current OpenAccess-based PDKs and Cadence IC6.1 environment.

The TSMC 65nm iPDK is in beta testing as of February 2010 and is expected to go into general release in Q2 2010. A number of IP blocks developed using the iPDK have been taped out and silicon-proven. 28nm and 40nm iPDKs are currently under development by members of the iPDK development teams and TSMC.

An interoperable PDK benefits the entire TSMC design chain. TSMC customers will be able to use one unified interoperable PDK to provide advanced functionality across multiple EDA vendor tools, improve design accuracy, shorten design cycle times, and promote design reuse. EDA vendors will be able to offer TSMC customers innovative new custom IC design tools. TSMC is able to reduce their PDK development, validation, support and distribution costs while expanding the number of tools they support. Combined, these benefits improve the return on design investment for their customers.

[SpringSoft](#)

[TSMC](#)

[Print](#)

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