



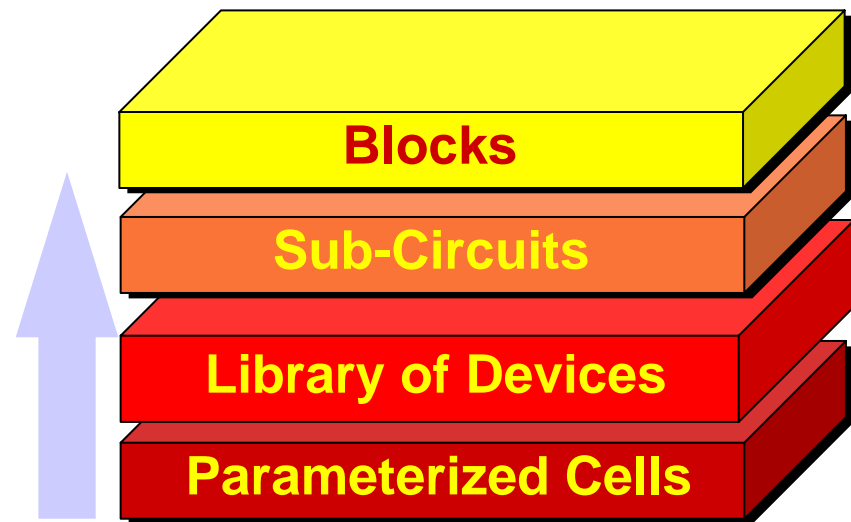
CIRANOVA®

OpenAccess PCell Interoperability

**Ed Petrus
VP Engineering**

About CiraNova

- ▶ CiraNova is developing an OpenAccess based framework for layout generation: *Santana*TM
- ▶ Using Python as the extension language for the framework
- ▶ OpenAccess PCells are the building blocks: we call ours *PyCells*TM



Please see demo later in the evening

What We Mean by PCell Interoperability

- ▶ **Create a library of PCells as part of a PDK**
- ▶ **Access the PCells - as live PCells - in design tools from different EDA companies (or custom in-house tools)**
 - PCell geometries can be translated out as polygons through standard design formats (e.g. GDSII) but lose their flexibility
- ▶ **Today this is mostly possible within the silos of individual EDA companies**
 - We are aware of at least different 5 different silos where PCell capabilities are offered

Create the library once and deploy it everywhere

The Business Case

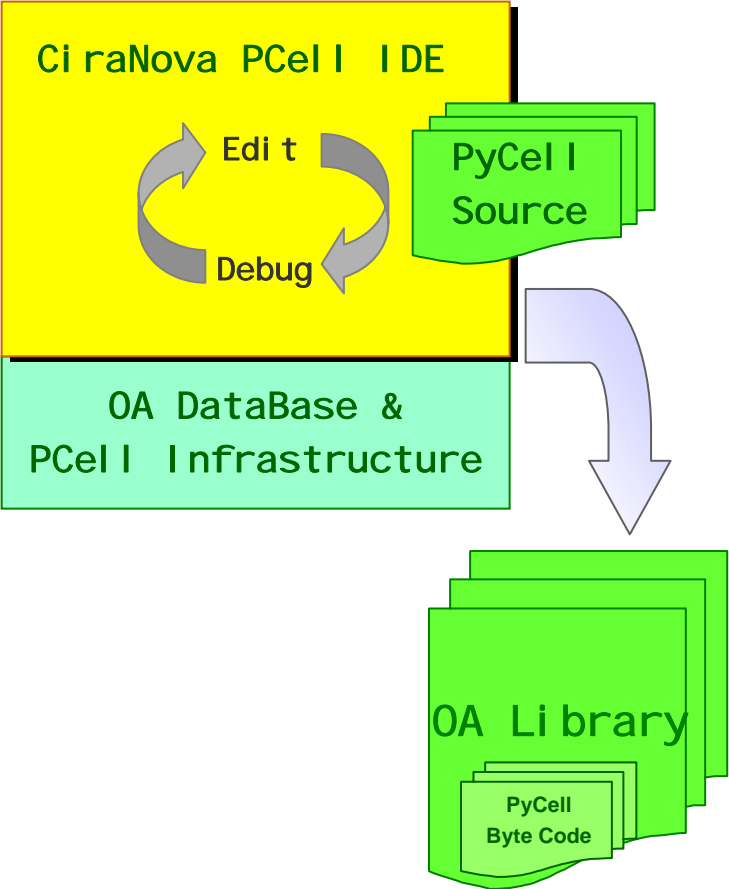
- ▶ **Efficiency is the main driver**
- ▶ **Minimize tool integration overhead so the customer can focus on their core business**
- ▶ **Allow the customer to easily build flows using tools from multiples of EDA vendors**
- ▶ **The process of moving design data through such a flow must be as efficient as doing it through a flow based on one company's tools**
- ▶ **OpenAccess can begin to make this possible**
- ▶ **PCell interoperability is just one piece of the puzzle that must be in place for OpenAccess to achieve its full potential**

How Will PCell Interoperability Work ?

- ▶ **There are two main parts to the problem and its solution(s)**
- ▶ **First - The mechanics of authoring PCells, packaging and deploying libraries and inserting PCell instances into designs and maintaining design data**
- ▶ **Second - The large body of conventions that instances of PCells must abide by in order to work in various tools and flows**

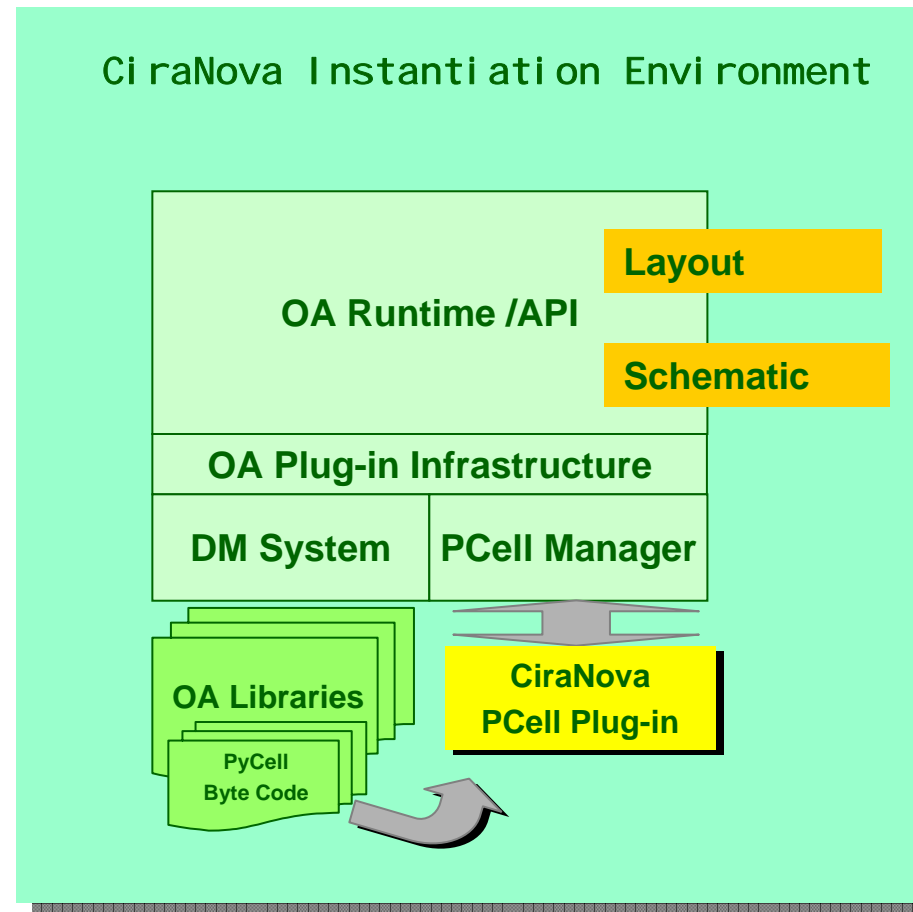
Mechanics of Authoring and Deploying PCells

- ▶ **OpenAccess infrastructure - plug-in, callback mechanisms**
– make it possible to author and instantiate PCells transparently into multiple tools
- ▶ **CiraNova has created a state-of-the-art authoring environment for PCells and layout generators entirely built around OpenAccess**



Mechanics of Instantiating PCells

- ▶ At CiraNova we have demonstrated that we can write powerful *PyCells* and have them work in tools from multiple EDA vendors (Cadence, Silicon Navigator)
- ▶ We consider this problem solved
- ▶ Many thanks to the OpenAccess team



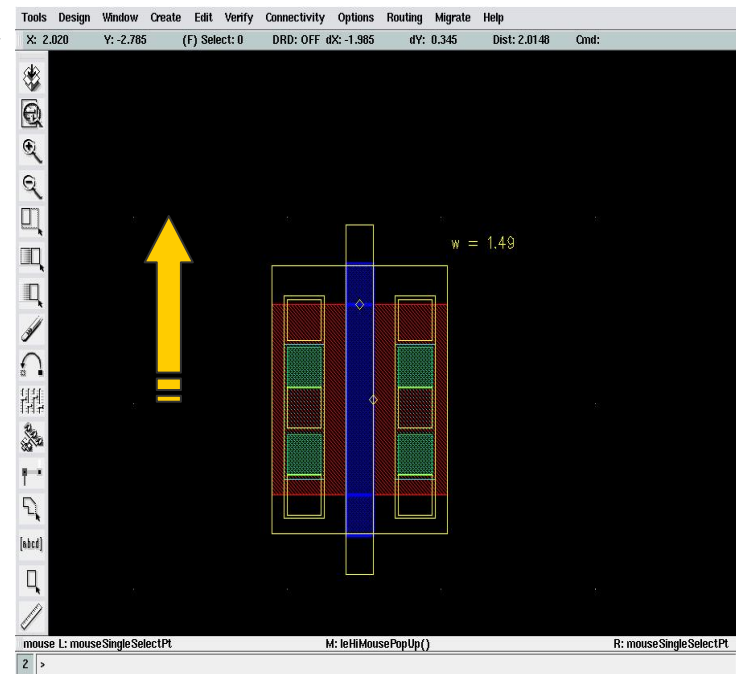
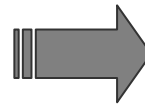
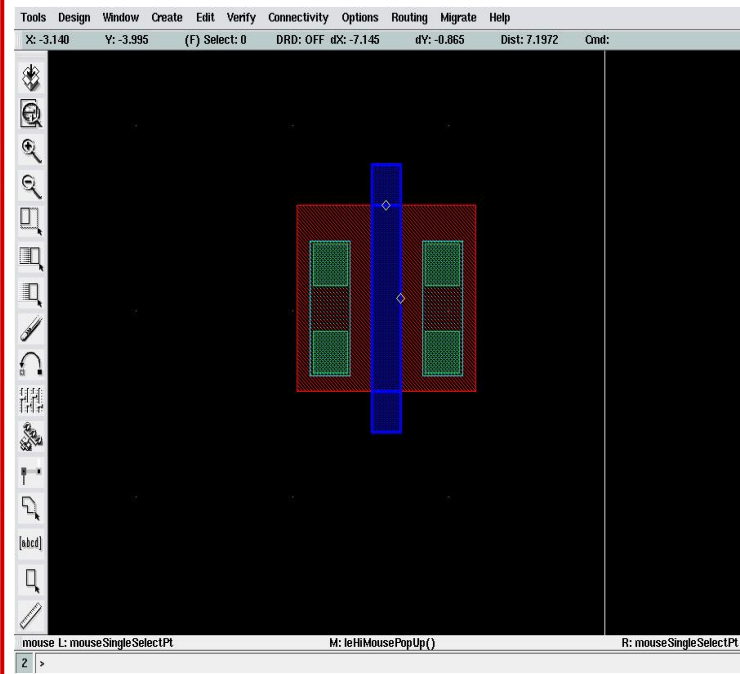
The Conventions

- ▶ *Let us illustrate the problem using a number of examples*
- ▶ **Example1** : which layer purpose pair to associate with pin data structures created by the PCell ?
- ▶ **Example2** : how should the API define auto-abutment capabilities for devices? Should all OpenAccess based layout design tools understand one protocol? Should the PCell capability support multiple protocols?
- ▶ **Example3** : Whose protocol should we use to tell routers the limitations of routing over devices?
- ▶ **Example4**: Layer Purpose Pairs (LPPs) as defined in techfiles – should all physical design tools understand this concept or will it remain a Cadence only thing ?

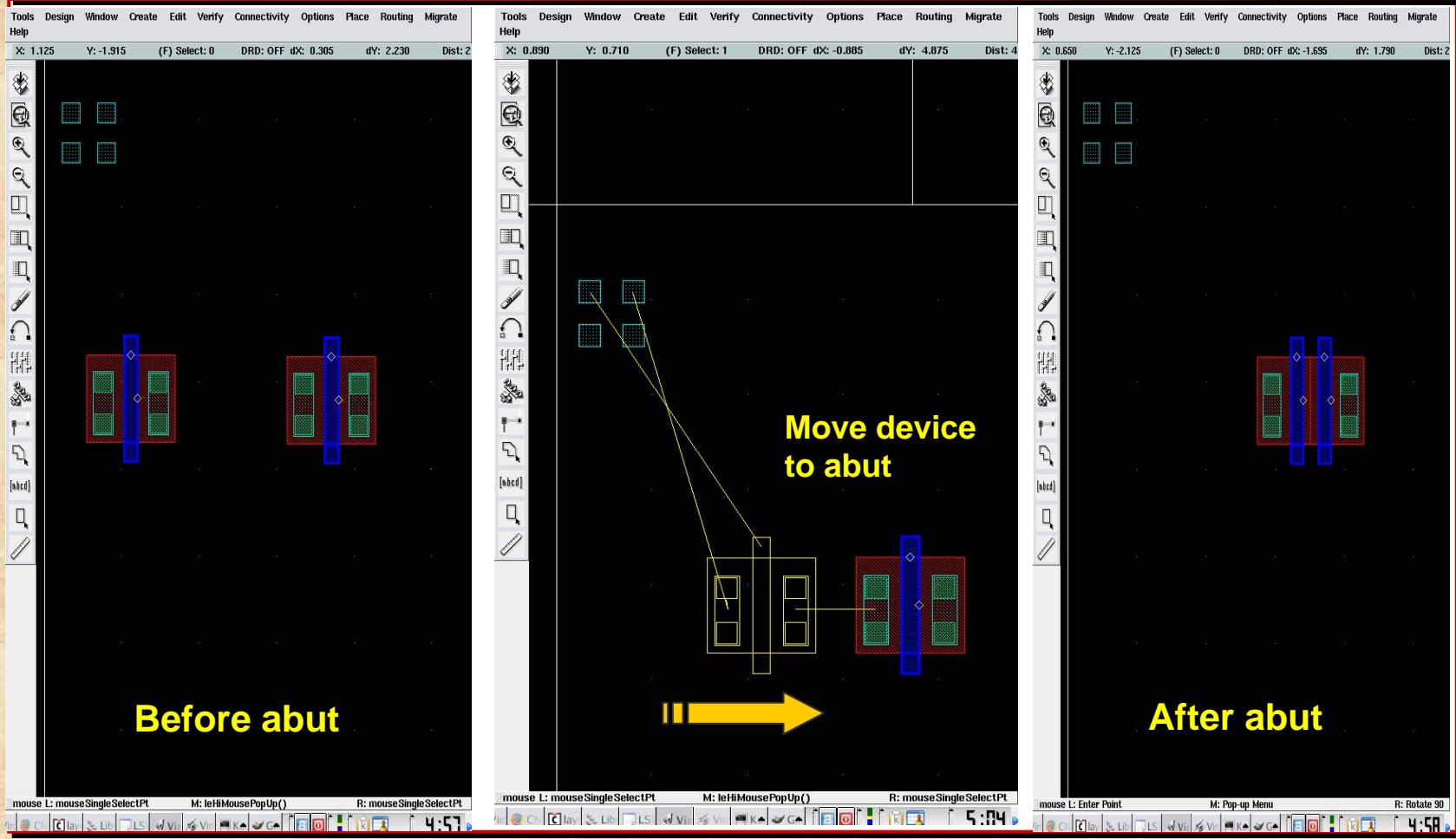
Stretch Handles

PyCell device with stretch handles in Cadence's Virtuoso layout editor

Effect of stretching increases gate width & number of source / drain contacts



Auto-abutment Example



The image displays three sequential screenshots of a PCB design software interface, illustrating the auto-abutment process. Each screenshot shows a dark workspace with a grid and various components. The components are highlighted with blue selection boxes and red dashed outlines.

- Before abut:** Two components are positioned side-by-side. The text "Before abut" is written in yellow at the bottom of the panel.
- Move device to abut:** A component is being moved towards the other. A yellow arrow points to the right, and the text "Move device to abut" is written in yellow. The text "Move device to abut" is written in yellow in the center of the panel.
- After abut:** The two components are now abutted together. The text "After abut" is written in yellow at the bottom of the panel.

The software interface includes a menu bar (Tools, Design, Window, Create, Edit, Verify, Connectivity, Options, Place, Routing, Migrate) and a status bar at the bottom. The status bar shows coordinates (X, Y), selection status ((F) Select: 0 or 1), and other parameters (DRD: OFF, dX, dY, Dist: 2).

Conventional Challenges

- ▶ **Standardizing information relevant to fabrication process technology**
 - Decades of Techfile based complexity from multiple EDA tool vendors
- ▶ **Standardizing the protocols by which PCells “communicate” with physical design tools**
 - Work on specifying necessary protocols, OR
 - Go with defacto standards – adopt the protocols of the dominant tools
- ▶ **OpenAccess views PCells as physical entities only; representation for PCells as electrical entities must be layered on top of OpenAccess**
 - The complete design loop for PCell representation must take into account tools such as schematic entry, simulation, test-bench environment where most of the work of sizing devices takes place (needs expression definition & evaluation)
- ▶ **Should there be one industry-wide OpenAccess PCell capability ?**
 - C++ , Python, Perl, Tcl, SKILL?
 - Raw OpenAccess APIs or higher level capabilities ?
 - Can one solution satisfy all ?

Future Direction

- ▶ **The EDA industry – suppliers and consumers – is on an unstoppable path towards advancing the software architecture of the tools**
 - Vertical integration gives way to horizontal integration
 - Componentizing the architecture will benefit all constituents
 - The OpenAccess PCell mechanism is an excellent example of the future of component based software architecture
 - The same concepts can be extended and applied to tool integration
 - It happened to enterprise software during 1990s; is EDA immune from this?

Summary

- ▶ **OpenAccess needs the rest of us to pitch in and fill the areas that fall immediately outside OpenAccess**
 - PCell interoperability is just one piece in a large puzzle

- ▶ **Address fundamental issues of PCell interoperability**
 - Work on standardizing protocols so PCell mechanisms and tools can communicate effectively
 - Establish as standards information about and related to process technology
 - Make sure interoperability is guaranteed by the PCell mechanism
 - That is, whatever the solution, please make sure the burden of interoperability is removed from the author of PCell libraries

... may the best PyCell win!