



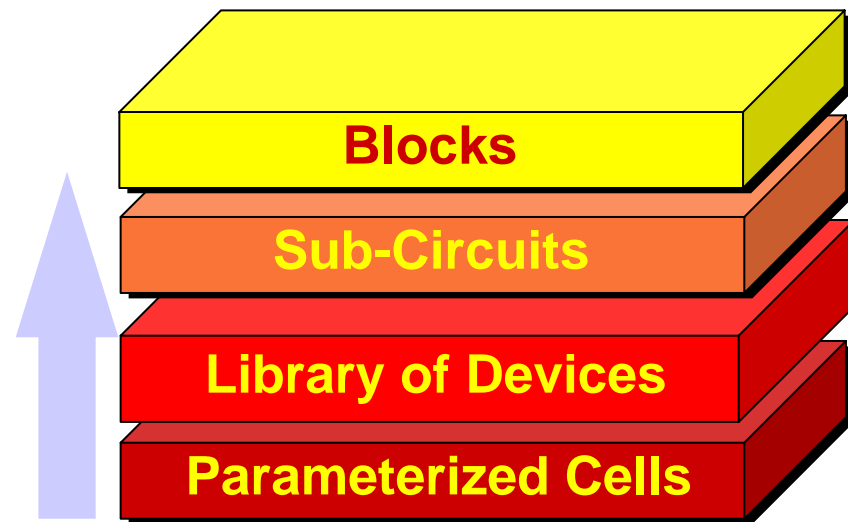
# Open PCells

Ed Petrus

VP Engineering

# About CiraNova

- ▶ CiraNova is developing an OpenAccess based framework for layout generation.
- ▶ Open Access PCells are the building blocks
- ▶ Using Python as the extension language for the framework



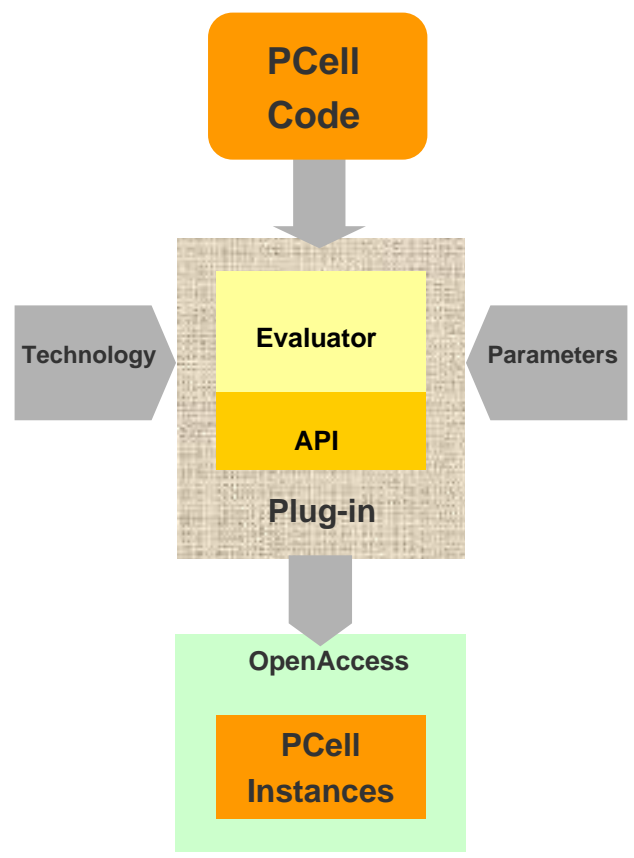
# Universal Parameterized Cells

- ▶ **The quest: PCells that can be instantiated in any EDA tool**
  - Historically difficult to achieve
    - Lack of a standard data model
    - Lack of a standard plug-in mechanism
    - Lack of a standard programming interface for generating desired PCells
    - etc.
  
- ▶ **The quest: PCells that can be instantiated in any EDA tool using OpenAccess as the native in memory data model**
  - *Write once, instantiate everywhere*
  - *One common and standard in-memory data model accompanied by an industrial strength implementation*

**The advent of OpenAccess makes Universal PCells possible**

# Anatomy of PCells

- ▶ **A mechanism for specifying cell parameters**
  - Parameter name, value type, value range, etc.
  - Default parameter values specified at time of authoring
- ▶ **Programming language for scripting geometry construction**
  - Access parameter values
  - Access Technology information
  - Calculate object dimensions and distances between objects
  - Call on interfaces of various subsystems to make instances of PCell
- ▶ **Programming interface for geometry construction**
  - Create, place and manipulate shapes
- ▶ **Programming interface to access technology information**
  - Minimum spacing rules, etc.
- ▶ **A plug-in mechanism and protocol to execute PCell scripts and create parameterized instances of cells**



## Requirements for a Universal PCell solution

- ▶ **Neither authoring nor instantiation of PCells should be locked into a specific vendor solution for physical design**
  - Should be based on industry or technology wide standards
- ▶ **Offers a programming interface for specifying cell parameters that is compatible with OpenAccess and fits well with popular physical design tools**
  - Better still is to have a published specification
- ▶ **Offers layout programming interface with a gradation of abstraction levels**
  - Place(Obj-1 , eastOf , Obj-2)
  - Place(Obj-1 , eastOf, Obj-2, 0.5u)
- ▶ **Offers a programming interface to access technology information**

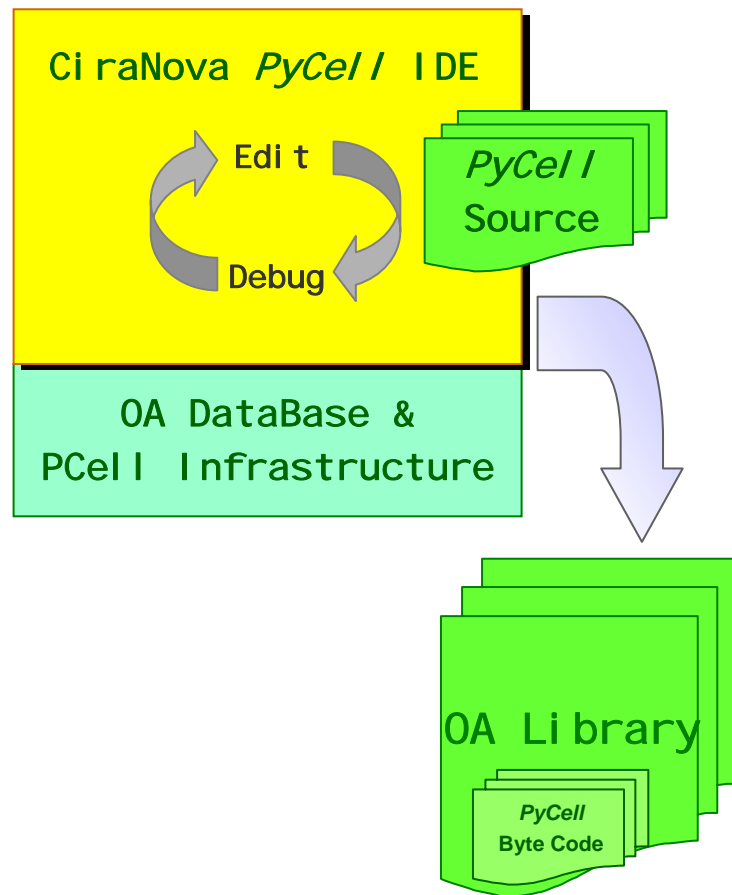
## Requirements, continued

- ▶ **PCell script should be tied to OpenAccess databases (associated with the SuperMaster)**
  - Otherwise it makes existing design management headaches worse
- ▶ **The PCell mechanism and the underlying infrastructure handle interoperability**
  - If a library of PCells works with one OpenAccess based layout editor then it should work with other OpenAccess based layout editors
  - Author of PCell should not be required to address interoperability
- ▶ **PCell executable code should be OS and CPU architecture independent**
  - Otherwise library creators must create PCell executables for every version of OS/CPU/Compiler in play

# CiraNova PCells

- ▶ **OpenAccess PCells are the building blocks of CiraNova's layout generator solution**
  - We see PCells as the mechanism for enabling re-usable and re-targetable IP
- ▶ **Fielding sophisticated authoring tools for PCells and layout generators**
  - Powerful layout generation APIs
  - Integrated Development Environment (IDE)
- ▶ **Easy to use run-time support for instantiating PCells**
  - All you need is a Plug-in delivered as a shared library
- ▶ **Support for authoring in the Python language**
  - The underlying infrastructure is written in C++ and works with OpenAccess
  - Python's OO capability is a natural way to integrate C++ 's class hierarchies into an extension/scripting language – more power to the user
- ▶ ***We call them PyCells™***

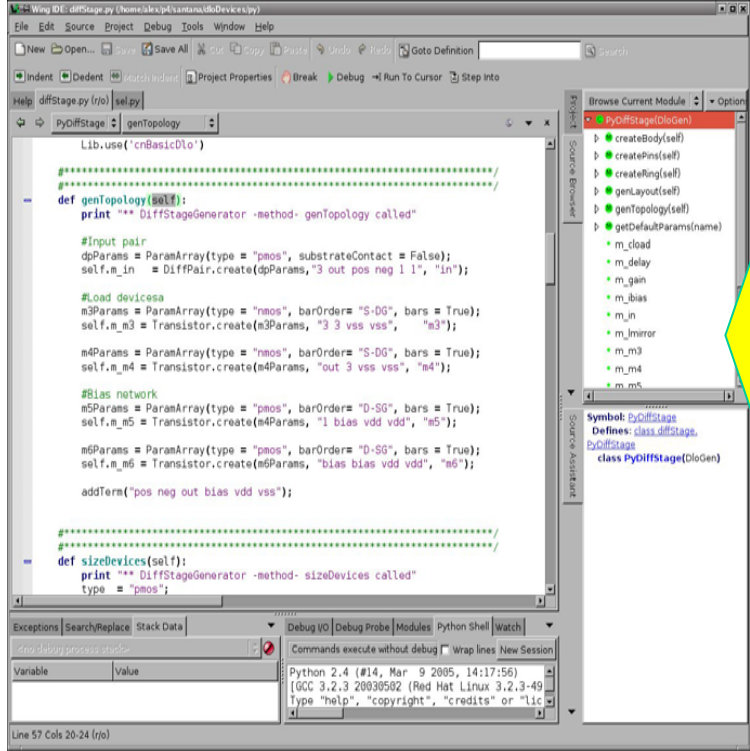
# CiraNova *PyCell* Authoring Diagram



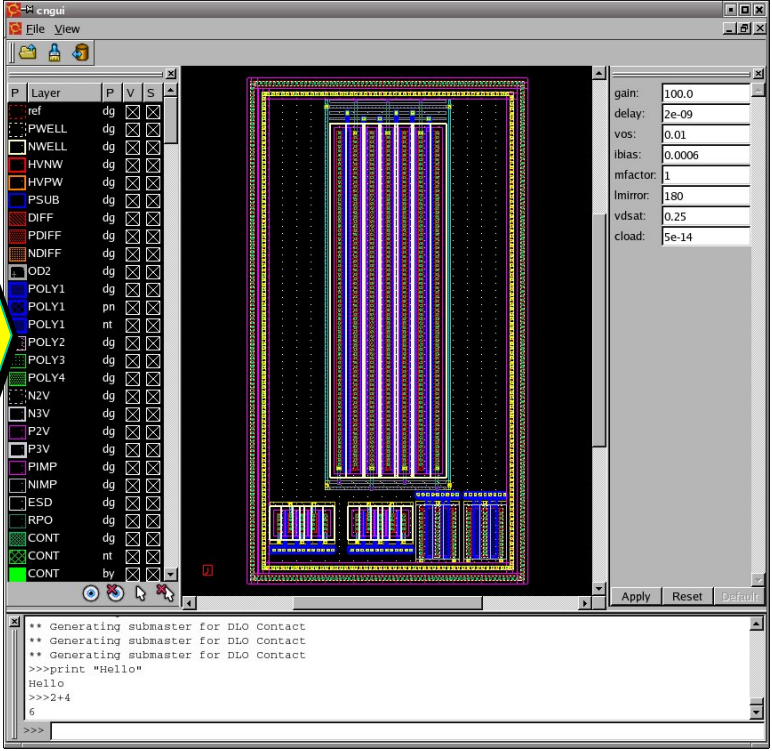
- ▶ *PyCell* source files are used during development
- ▶ *PyCells* are packaged and associated with Libraries for delivery & deployment
- ▶ Libraries contain *PyCells* in Python byte code form
- ▶ Python byte code is saved and is associated with SuperMaster structures in databases
- ▶ Python byte code is OS and CPU architecture independent

# CiraNova PyCell IDE

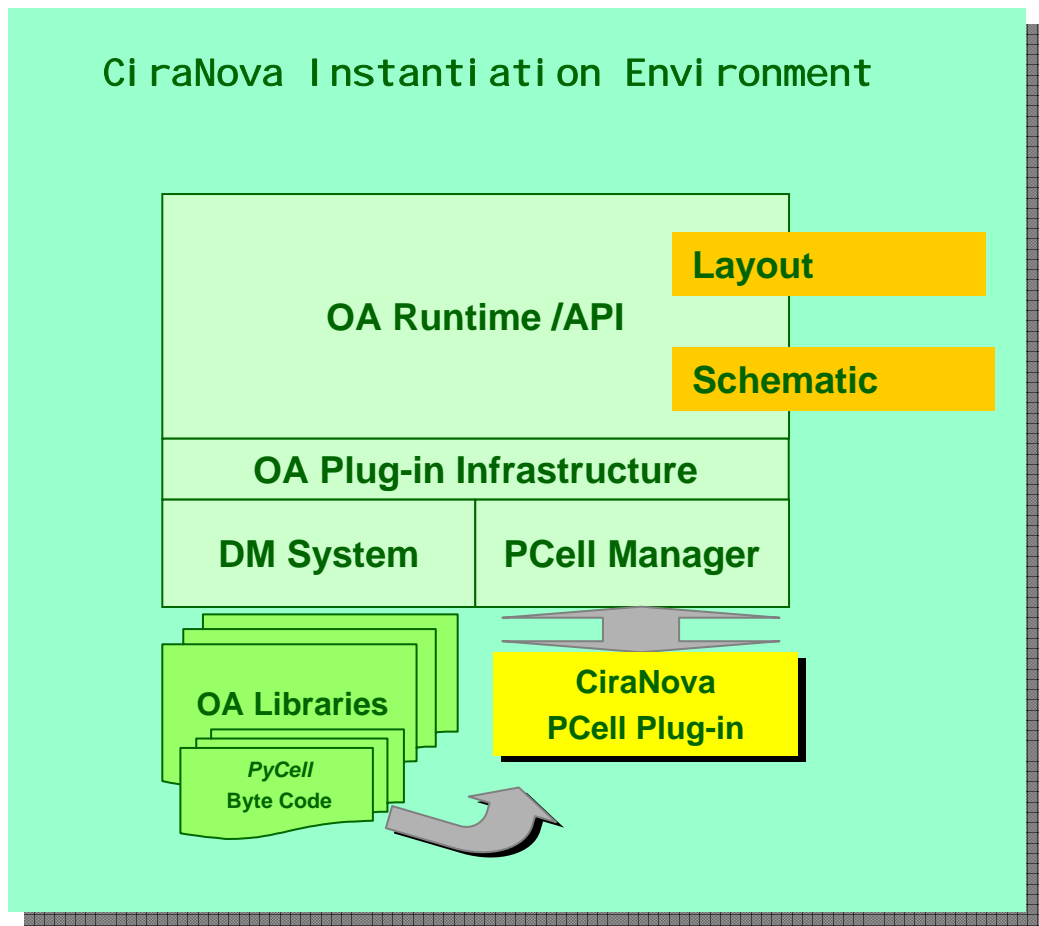
Working with *PyCell* source code:  
 Source code view & edit, break & watch points, trace, etc.



Working with *PyCell* Instances:  
 View layers, pan, zoom, edit params  
 Run commands & scripts in console



# PyCell Instantiation Diagram



# OpenAccess PCell Interoperability

- ▶ **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 Interoperability

- ▶ At CiraNova we have demonstrated that we can write powerful *PyCells* and have them instantiated in tools from Cadence and Silicon Navigator, as well as our own
- ▶ Open Access infrastructure - plug-in, callback mechanisms – make this possible
- ▶ We consider this problem solved – more or less

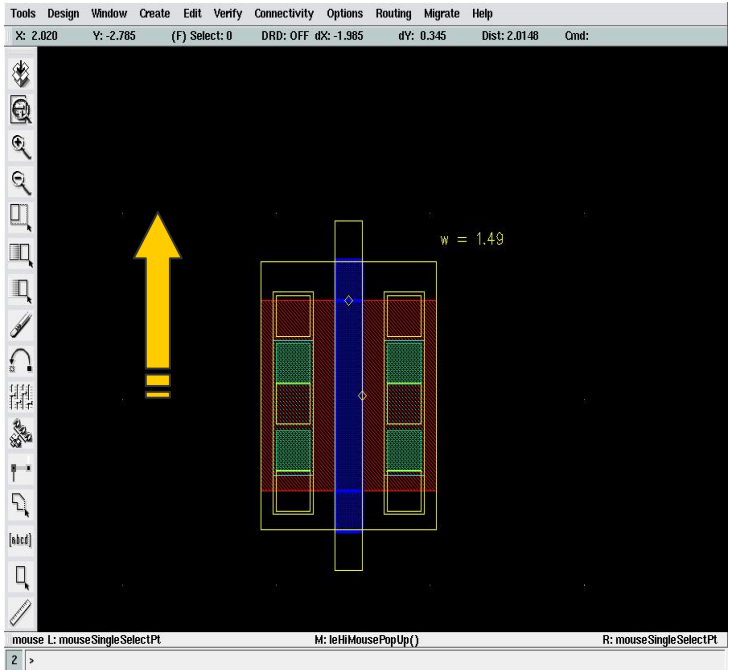
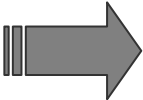
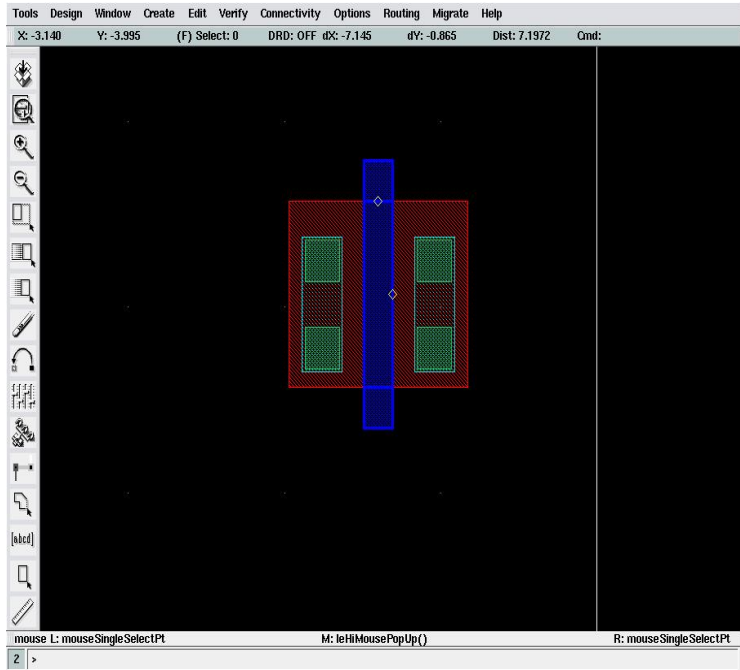
# Conventions of Interoperability

- ▶ **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
- ▶ **Should there be one industry-wide OpenAccess PCell capability ?**
  - C++ , Python, Perl, TCL, SKILL?
  - Raw Open Access APIs or higher level capabilities ?
  - Can one solution satisfy all ?

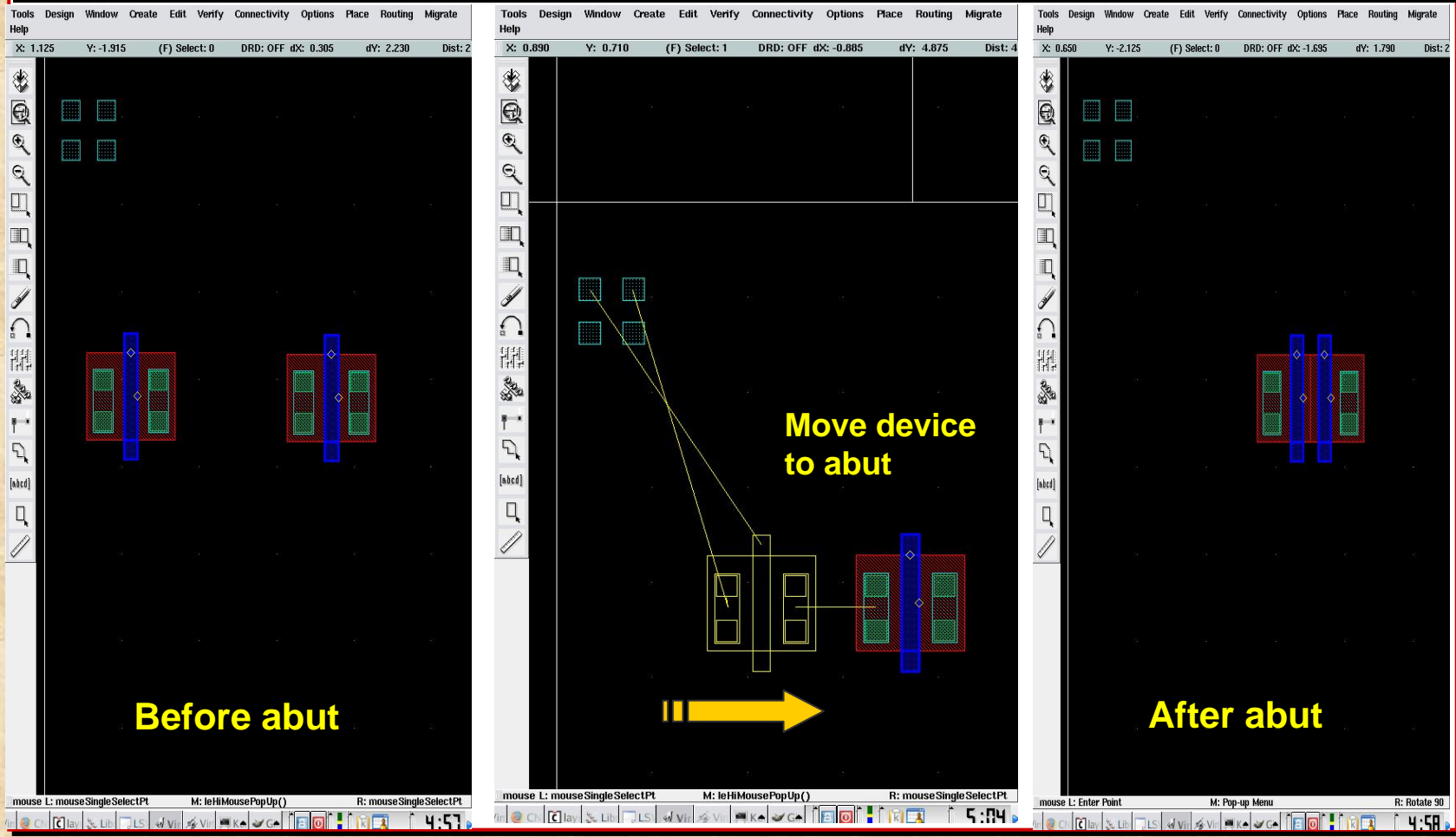
# Protocol: Specify Stretch Handles

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

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



# Protocol: Specify Auto-abutment



**Before abut**

**Move device to abut**

**After abut**

# Summary

- ▶ **Address fundamental issues of PCell openness and interoperability**
  - Too hard to achieve openness without a standard in-memory data model
    - OpenAccess is it
  - Work on standardizing protocols so PCell mechanisms and tools can communicate effectively through the OpenAccess data model
    - Needed are protocols for tools such as layout editors, schematics, P&R, LVS, etc.
  - PCell mechanisms can choose to support one or multiple scripting languages
  - 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!*