Learn the OpenAccess API

Using Python

Initial Contribution By

James Masters
Intel - 2013
Updates & Additions
by Silicon Integration Initiative - 2020

Section 10: oaInst

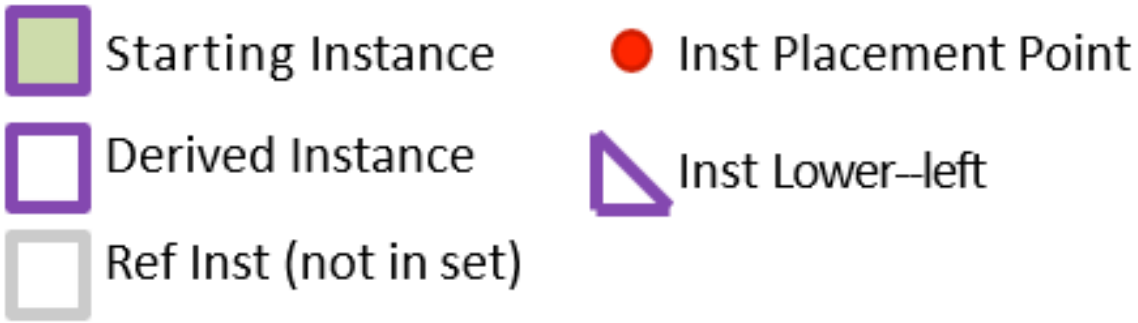
- Subcells are placed into a top block using an oaInst:
  - **oaScalarInst:** a single instance
  - **oaVectorInst:** several copies of an instance master with index numbers to differentiate
  - **oaArrayInst:** represents an array of instances in one master design where every cell has the same orientation (a “mosaic”)

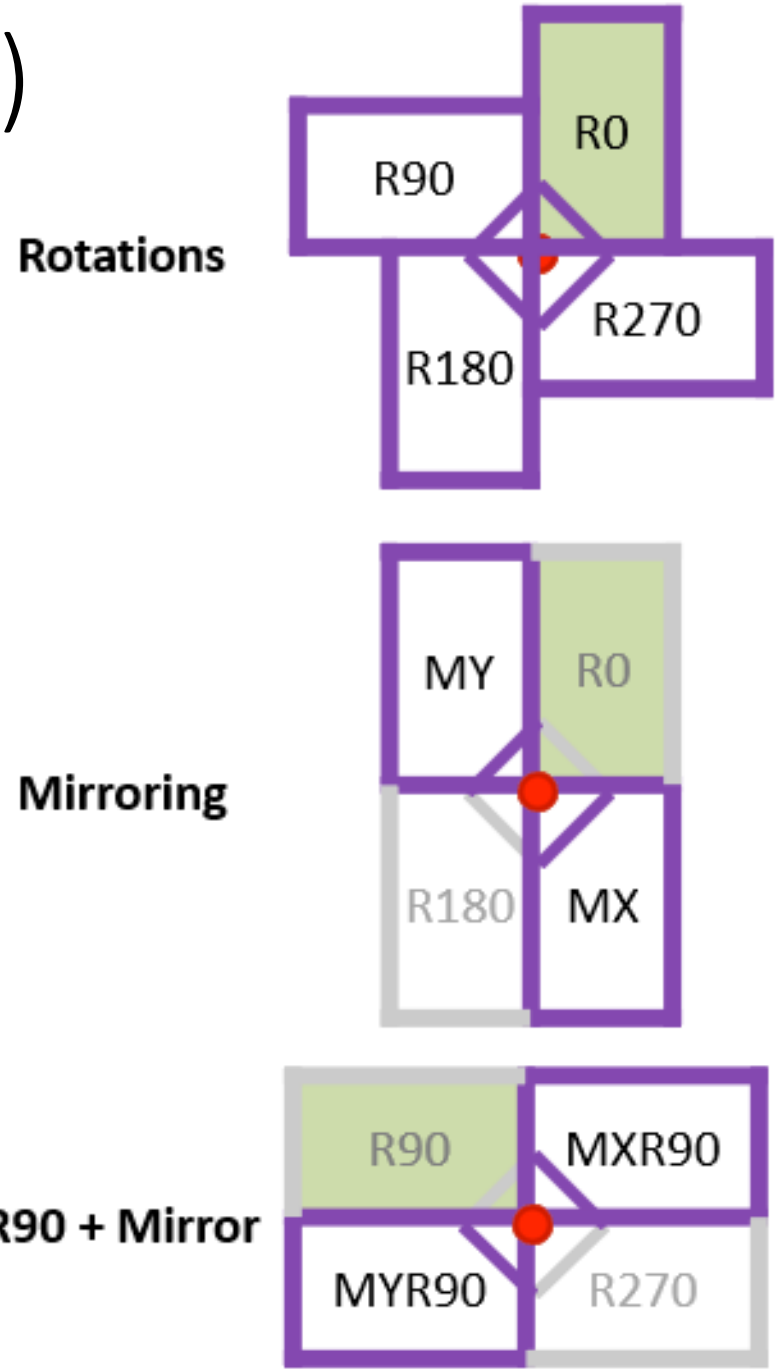
Terminology and Concepts

- **Instance:** Placement of a subcell into a top-level design
- **Instance "master":** the oaDesign of the subcell being placed
- **Instance name:** a unique name given to an instance when placed within a design to differentiate against other instances
- **Instance transformation:** the instance placement location and orientation (rotation and/or mirroring)

Orientations (oaOrient)

- A transformation may have any of the shown orientations (see oaOrient for details)
- Default orientation without any transposition is “R0” (`oa.oacR0`)





oaScalarInst

- The most common instance placement is an oaScalarInst which represents a single placement
- There are a few different methods to create an instance; two common ones are:
  - **Option 1:** Place instance after the instance "**master**" (oaDesign) is opened. This ensures that the instance "master" is found (validated).

```
subcell = oa.oaDesign.open("mylib", "subcell", "layout", "r")  inst =
   oa.oaScalarInst.create(block, subcell, "i0",
[cd2dbu(0), cd2dbu(0), "R0"])
```

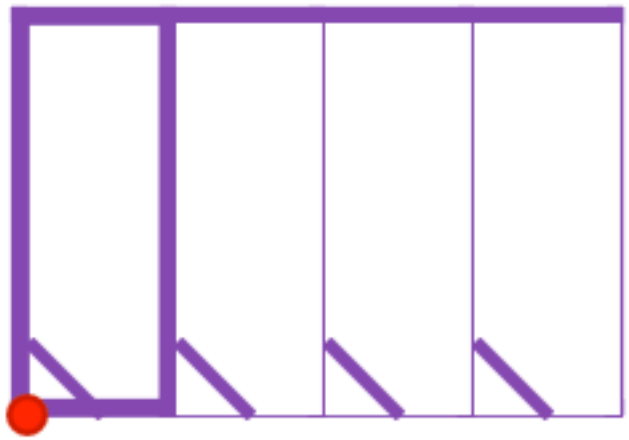
  - **Option 2:** Place the instance with just the library, cell, and view name. The design may or may not exist (not validated).

```
inst = oa.oaScalarInst.create(block, "mylib", "subcell", "layout",  "i0",
   [cd2dbu(0), cd2dbu(0), "R0"])
```

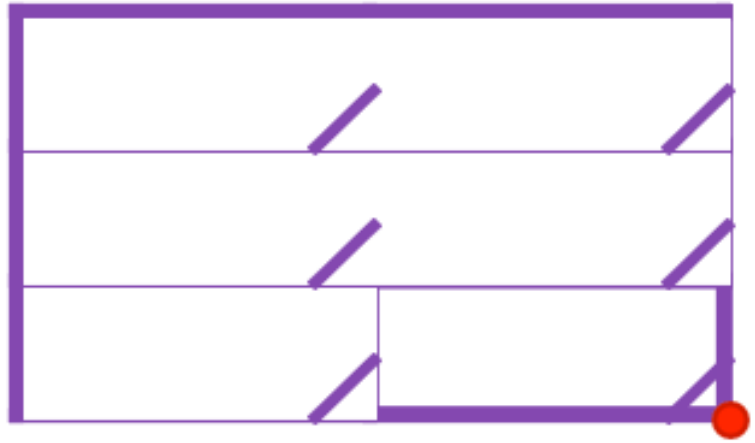
- There are legitimate reasons to use either Option 1 or Option 2; however, if possible, it is probably a good idea to use Option 1 to validate the existence of the subcell

oaArrayInst

- The oaArrayInst has the same basic creation options as oaScalarInst except has these properties:
  - Distance X: Spacing in the X direction
  - Distance Y: Spacing in the Y direction
  - Num Rows: Number of rows in the array
  - Num Columns: Number of columns in the array
  - The X/Y distance and rows/columns follow an orientation change (e.g. for a R90 rotation, the X distance is in the Y direction and the "columns" will look like rows)



R0, rows=1, columns=4



R90, rows=2, columns=3

Lab 10.1: Creating Instances

- Goal - Become familiar with creating oaScalarInst and oaArrayInst objects and with transformation orientation (rotation and/or mirroring) concepts
- Write a script to:
  - Create a new “insts” design in “w” mode (overwrite)
  - Open existing “buffer” design (from earlier lab) and place a few instances in the design
  - Open existing “res” design (from earlier lab) and place 1 row and 8 columns of resistors spaced at 0.50um all rotated at R90

Compare your script with labs.10.1/insts.py

Finding Instances

- Instances may be found by name using the oaInst::find() method:

```
inst = oa.oaInst.find(block, "instname")
```

- All instances may be iterated in a block using the oaBlock::getInsts() method:

```
for inst in block.getInsts():
    # do something ...
```

Lab 10.2: Reporting on Instances

- Goal - Become familiar with iterating over oaInst objects and reporting information about those instances

- Write a script to:
  - Open existing “insts” design **in “r” mode (read)**
  - Get the oaBlock
  - Iterate over all instances using block.getInsts() and report the following informa,on for each instance:
    1. Instance name
    2. Library, cell, and view name
    3. Instance origin and rotation
    4. Bonus: Instance array size (if an oaInstArray)

Compare your script to labs/10.2/instlist.py

Remastering Instances

- The master instance contains a reference to the design being instantiated
- This reference can be changed to reference another design

```
for inst in block.getInsts():
    if(inst.getName() == "i0"):
       print("Inst found")
       inst.setMaster(<libname>, <design_ref_name>, "layout")

<your design>.save()
```

Lab 10.3: Remastering an Instance

- Goal - Learn how to switch the referenced design instantiated from the instance level

- Write a script to:
  - Use the script from lab 10.2 to list all the design instances
  - Open existing “insts” design **in “a” mode (append)**
  - Get the oaBlock
  - Add a step remaster instance “i0” from a buffer to a resistor
  - Save the design
  - Iterate over all instances using block.getInsts() and report the following informa,on for each instance:
    1. Instance name
    2. Library, cell, and view name
    3. Instance origin and rotation

Compare your script to labs/10.3/remaster_inst.py

Parameterized Cells (Pcells)

- Instances can have parameters attached to them (oaParamArray)
- Parameters are used control shapes inside of the cell by executing parameterized cell (Pcell) code
- Pcells are outside of the scope of this training
- Note: Some Pcell evaluators are considered proprietary by their authors as the Pcell may be sold as a product.

Section 10 Summary

- Learned how to create instances and change their orientation using oaTransform
- Investigated two methods for placing an instance
- Looked at the oaArrayInst and it's orientation
- Learned how to find instances by name and how to iterate over instances
- We found out how to remaster an instance
- And we introduced Parameterized Cells

Silicon Integration Initiative

www.si2.org

For details contact Marshall Tiner

Director of Production Standards

mtiner@si2.org