Learn the OpenAccess API
Using Python

©Silicon Integration Initiative
Initial Contribution By

James Masters
Intel - 2013
Updates & Additions
by Silicon Integration Initiative - 2020
Section 9:
oaNet, oaTerm, oaPin, oaPinFig

• Any oaShape can have a net attached to it to establish connectivity

• Different connectivity models can be established by using oaTerm, oaPin, and oaPinFig
oaNet

- Any oaShape can have connectivity associated with it using an a type of oaNet:
  - oaNet – wrapper to represent any of the oa*Net types
  - oaScalarNet – represents a non-bit net (e.g. “foo”)
  - oaBusNet – represents a net with many bits (e.g. “foo[0:7]”)
  - oaBusNetBit – represents a net with a single bit (e.g. “foo[1]”)
  - oaBundleNet – represents potentially repeated simple nets
- Most nets are oaScalarNet – single net name
- Can create a net in a block by just using oaNet and the right net type will automatically be detected by the name
  - A created net by itself does not involve any shapes until shapes are added to the net

```python
net = oa.oaNet.create(block, "mynet")
shape.addToNet(net)
...
shape.getNet(net) # to get later on
```
Lab 9.1: Add Shapes to Net

• Goal - Become familiar with creating oaNets inside of an oaBlock and adding shapes to those nets

• Write a script to create oaBlock and add shapes to an oaNet
  – Pretend that we’re creating a “buffer” cell with three nets: “in”, “out”, and “n1” (between the inverters)
    • Will only draw rectangles as an abstract view – not all shapes
  – Create layout oaDesign called “buffer” under library “mylib”
  – Refer to the picture on the next slide to show rectangle locations, cell boundary dimensions, and net names
  – See next page

Compare your script to labs/9.1/buffer.py
Buffer Shapes (Lab 9.1)

- Cell name: buffer/layout
- Rectangle attributes:
  - Layer = m1/drawing
  - Net Name = as shown
  - Width = 0.10
  - Height = 0.50
oaTerm, oaPin, oaPinFig

• **oaTerm**: A *logical* object joining oaPins and exporting nets outside of the block to be connected at a higher level of hierarchy

• **oaPin**: A *logical* object joining oaPinFigs in a way that indicates a desired connectivity model (see next slide for details)

• **oaPinFig**: A *physical* attribute that is attached to an oaShape that is intended to be a connection point outside of the current block level
**OA Connectivity Models**

- Different oaTerm, oaPin, and oaPinFig (PF) combinations indicate the type of connectivity to export to the next hierarchical level.
- **Single Pin**: Single pin to be connected.
- **Strongly Connected**: Can route into either pin and out the other.
- **Weakly Connected**: High resistance connection; can route to one or another pin, but not through.
- **Must Join**: Must connect to both pins at the next hierarchical level (open circuit in the subcell).
Creating `oaTerm`, `oaPin`, `oaPinFig`

- `oaTerm` and `oaPin` have names, but they just need to be unique names and otherwise aren’t immediately exposed to users in a design.

- Example creating single pin:
  ```python
term = oa.oaTerm.create(net, net.getName())
pin = oa.oaPin.create(term)
shape.addToPin(pin)
```

- Example creating strongly connected pins (i.e. feedthrough):
  ```python
term = oa.oaTerm.create(net, net.getName())
pin = oa.oaPin.create(term)
shape1.addToPin(pin)
shape2.addToPin(pin)
```
Pin Layers to Use

- You can create a pin on any layer; however:
  - Pins on primitive components are usually the physical layer (e.g. poly/drawing)
  - Pins at the block design are usually on a “pin” purpose (e.g. m1/pin)
  - Pins on the “pin” purpose have an advantage of having different visibility in an editor and also can export to a separate GDSII layer/ datatype number than the “drawing” purpose
  - Pins on a “pin” purpose are placed on top of existing shapes on the “drawing” purpose
    - **they are not placed alone**
Copying OA Shapes

• In the next lab, you will edit your “buffer” design from the previous lab and promote “in” and “out” shapes to pins
• OA provides a built-in mechanism for copying shapes to avoid having to create a new shape
  – Copy into the same block or into another block
  – Transform the copy (move location, rotate)
• In this case, we will simply copy into the same block at the same location/rotation and change the purpose type to “pin”

```python
ppin = oa.oaPurpose.find(tech, "pin").getNumber()
...
newshape = shape.copy(oa.oaTransform())
newshape.setPurposeNum(ppin)
```
Lab 9.2: Creating Pins

- Goal - Become familiar with creating terminals, pins, and pin figures

- Write a script to:
  - Open existing “buffer” design in “a” mode (edit)
    ```python
    design = oa.oaDesign.open("mylib", "buffer", "layout", "a")
    block = design.getTopBlock()
    ```
  - Loop through all m1/drawing shapes and copy the shapes on “in” and “out” nets to be on m1/pin layer and as single pins (one oaTerm, one oaPin, one oaPinFig)
    ```python
    net = shape.getNet()
    if net:
        if net.getName() in ("in", "out"):
            # copy to m1/pin, create term, pin
    ```
  - Use design.saveAs() to save to a different view to avoid constantly copying into the same design as you run multiple times
    ```python
    design.saveAs("mylib", "buffer", "layout1")
    ```

See next page
compare your script with labs/9.2/pins.py
Buffer Shapes (Lab 9.2)

- **Rectangle attributes:**
  - Layer = m1/drawing
  - Net Name = as shown
  - Width = 0.10
  - Height = 0.50

- Create two pins labeled “in” and “out” as shown
  - Create m1/pin shape on top of the existing pin also with the same net name (each pin has m1/drawing and m1/pin superimposed)
  - Add m1/pin shapes to their own respective oaTerm

- ![Diagram showing pin connections](image)

  - ![In connection](image)
  - ![Out connection](image)
Section 9 Summary

• Learned about the relationship between nets and shapes
• Understood the relationship between oaTerm, oaPin, and oaPinFig
• Investigated the connectivity model for pin connections
  – Must Join, Strongly Connected, and Weakly Connected
• Discussed advantages of using the pin purpose
• Used oaTransform to copy existing shapes
Silicon Integration Initiative

www.si2.org

For details contact Marshall Tiner
Director of Production Standards
mtiner@si2.org