Learn the OpenAccess API

Using Python

Initial Contribution By

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

Section 11 oaPartitions

- What are oaPartitions and why we use them
- Learned how to add oaPartitions to a design
- Learned how to read oaPartitions from a design

What is an oaPartition

- An oaPartition is a means of grouping objects within the database for later use.
- oaPartitions are Persistent
  - They don’t go away when you close the design
- Your application can create oaPartitions by any criteria
  - By layer, location, Name, etc.
  - Your application loads the data into an oaPartitionArray

About OpenAccess

- Within OpenAccess object data is stored in tables
- Different types of objects are stored in different tables
  - Shapes are in one table while nets are in another
- Each table consists of a set of member tables
  - Each member table holds one of the attributes of every object
- The table for a given object type can require considerable local memory
- oaPartitions gives you the ability to load only the objects you are interested in from the table
  - Called “partial loading”

About OpenAccess

- OpenAccess is not simply a place to store numbers
- OpenAccess supports simultaneous users
  - Requires file locking to prevent data collisions between users
- File locking makes multi-process applications difficult as the lock management gets in the way
- With oaPartitions the application can create partitions that insure no processes will be accessing the same data
  - So the file locking can be temporarily disabled to allow parallel processing!
  - The partial loading makes each of the parallel processes use less local memory (“light”)

Making oaPartitions

- Three of the object tables are enabled for oaPartitions
  - The Via table, the Shape Table, and the Instance Table
  - These are the most common objects you might iterate over
- First create a partition Array

```
InstArray = oa.oaPartitionArray_oaInst()
ShapeArray = oa.oaPartitionArray_oaShape()
ViaArray = oa.oaPartitionArray_oaVia()
```

- Now load the array as follows:

```
InstArray.add(inst_object)
ShapeArray.add(shape_object)
ViaArray.add(via_object)
```

- Save the design

Note: The C++ implementation is slightly different, see documentation

Removing oaPartitions

- To remove oaPartitions from a design

```
if(oa.oaPartition_oaInst.getPartitions(design)):
    oa.oaPartition_oaInst.destroyAll(design)
```

- An object can only be in one partition at a time
- You can make as many as 256 oaPartitions in a design

Lab 11.1 – Making oaPartitions

- Write a python script to:
  - Open existing "insts" design in "a" mode (append)
  - Destroy an existing partitions
  - Create two partitions "Res" and "Buf"
  - Iterate over all instances using block.getInsts()
  - if the instance name is "res" add it to the Res partition, if the name is "buffer" add it to Buf
  - Save the design with the oaPartitions as “part_des”

Compare your results with labs/11.1/makePartitions.py

Reading From oaPartitions

- First set the load model
  - Default is loadAll

```
loadmodelShape = oa.oasShapeOnDemandLoadControl()
loadmodelVia = oa.oasViaOnDemandLoadControl()
loadmodelInst = oa.oasInstOnDemandLoadControl()
```

- Get all the partitions of a type

```
shapeParts = oa.oaPartition_oaShape.getPartitions(design)
viaParts = oa.oaPartition_oaVia.getPartitions(design)
instParts = oa.oaPartition_oaInst.getPartitions(design)
```

- Load the partitions

```
for part in instParts:
    part.load()
```

Lab 11.2 – Reading oaPartitions

- Write a python script to:
  - Open existing "part_des" design in "r" mode (read)
  - Get the oaPartitions and print their names and contents

Compare your results with labs/11.2/readPartitions.py

Section 11 Summary

- Understand what an oaPartition is and how they work
- Understand uses for oaPartitions
- Learned how to make oaPartitions
- And we learned how to load the oaPartitions in a design

Silicon Integration Initiative

www.si2.org

For details contact Marshall Tiner

Director of Production Standards

mtiner@si2.org