

類比電路佈局合成自動化

Automatic Layout Synthesis for Analog Circuits

單元一

類比電路佈局合成自動化介紹與設計流程建置

Lecturer: Shao-Yun Fang

The Electronic Design Automation Laboratory

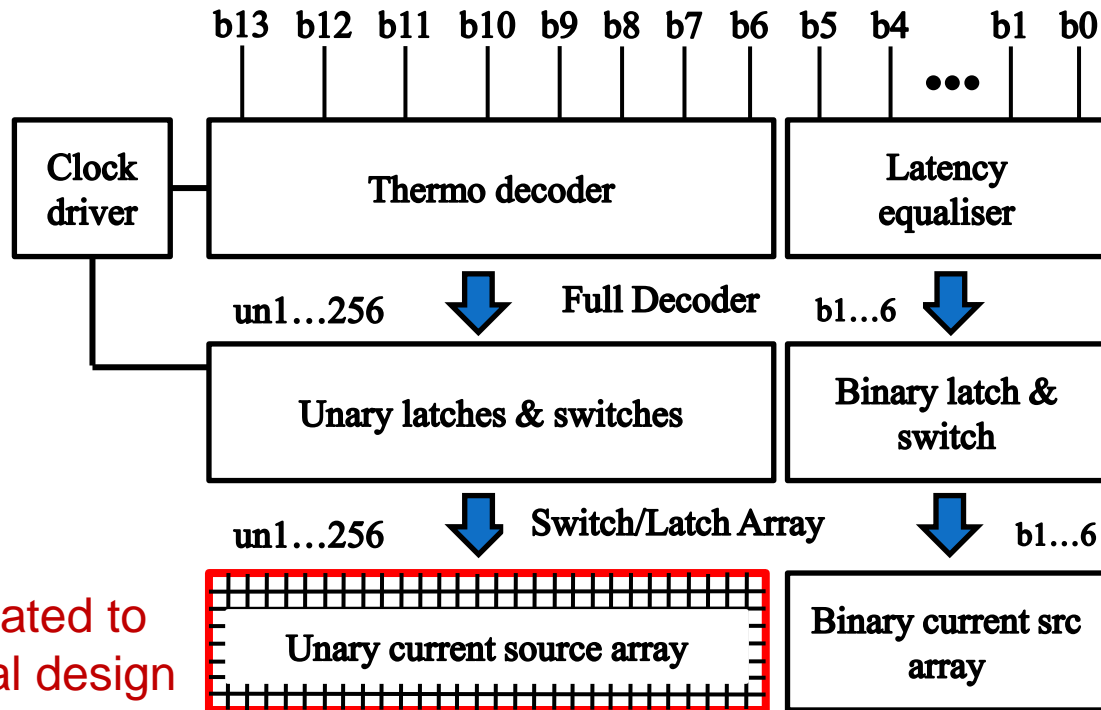
Department of Electrical Engineering

National Taiwan University of Science and Technology

Taipei 106, Taiwan

A Large-Scale Analog Circuit

- A 14-bit current-steering digital-to-analog converter (DAC)
 - The 8 MSBs are decoded from a thermometer decoder that steers a unary-weighted current source array
 - The 6 LSBs steer the binary weighted current source array

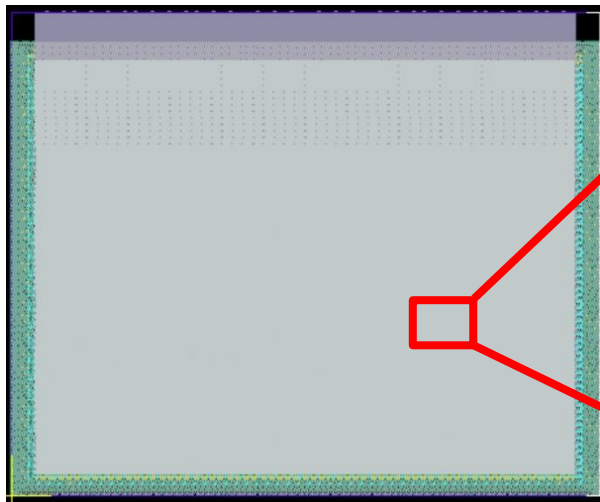


Too complicated to
adopt manual design

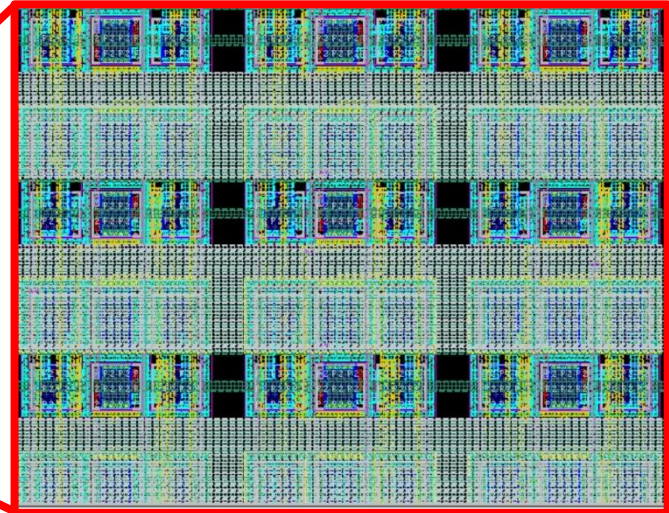
Current Source Array

Why EDA is Required

- **Electronic design automation (EDA)**
 - A category of software tools for designing electronic systems such as digital/**analog** integrated circuits (ICs)
- **The 8-bit unary current source array**
 - 256 current sources, each consists of 16 units
 - 4096 current source units arranged in a 64x64 array



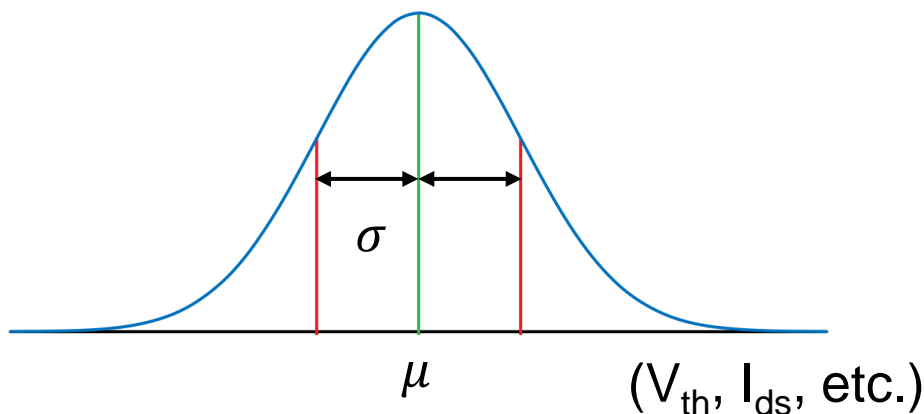
Manually laid out: > 1 month



EDA: < **1 second**

Causes of Mismatches

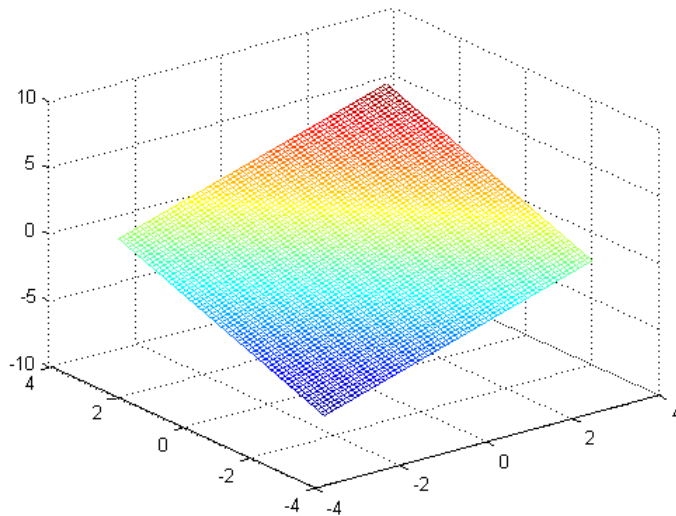
- The performance of an analog circuit is usually dominated by mismatches among devices that should be identical
- Random variations
 - Caused by random fluctuations in pattern dimension, doping, oxide thickness, etc.
 - Usually modeled as a Gaussian distribution
 - Mitigation approaches
 - Increase device area
 - Decompose a device into units and distribute in a layout



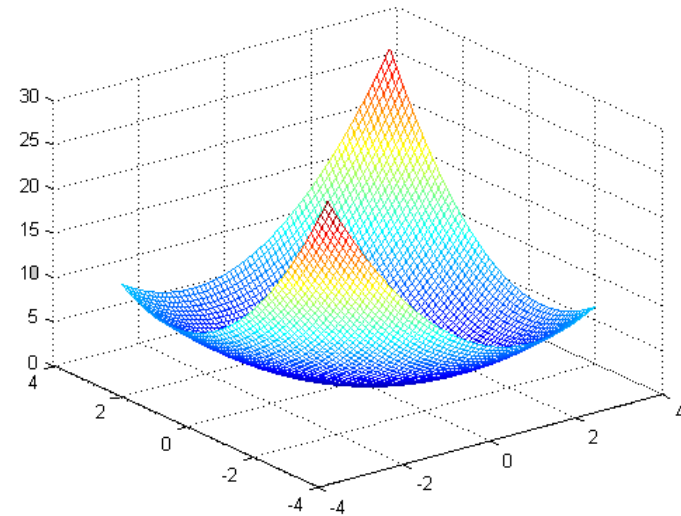
Causes of Mismatches (cont'd)

- **Systematic variations**

- Caused by process variation, thermal distribution, uneven mechanical stress from other circuit layers, etc.
- Show up as spatial gradients in device parameters



First-order gradient



Second-order gradient

Common-Centroid Constraint

- **Systematic mismatch can be compensated by careful layout design**
- **The common-centroid constraint**
 - The centroids of the multiple units of all devices are at the same position patterns
 - Used to eliminate linear gradient errors
 - Nonlinear gradient error mitigation will be introduced later

a_1	b_1	c_1	d_1
a_2	b_2	c_2	d_2
d_3	c_3	b_3	a_3
d_4	c_4	b_4	a_4

- Four devices
- Four units for each device
- Common-centroid

Design Flow Construction

GDSII Introduction

- GDSII is a **binary file** format representing planar geometric shapes, text labels
- GDSII includes
 - Standard cells
 - Metals
 - Vias
 - Polys
 - Diffusions
 - etc

```
0000000: 0006 0002 0005 001c 0102 07e3 0007 000a
0000010: 0015 0006 0025 07e3 0007 000b 0015 000f
0000020: 0029 000e 0206 5961 6f5f 6669 6e61 6c00
0000030: 0014 0305 3e41 8937 4bc6 a7f0 3944 b82f
0000040: a09b 5a50 001c 0502 07e1 0002 0009 0011
0000050: 002c 0015 07e3 0007 000b 0015 000f 0029
0000060: 000a 0606 4353 5f47 454e 0004 0900 0006
0000070: 0d02 0034 0006 0e02 0000 0008 0f03 0000
0000080: 03e8 0014 1003 0013 6f64 0005 c300 0013
0000090: 6aa0 0005 c300 0004 1100 0004 0900 0006
00000a0: 0d02 0036 0006 0e02 0000 0008 0f03 0000
00000b0: 01b8 0014 1003 000d 28f2 0001 3254 000d
00000c0: 28f2 ffff c5b8 0004 1100 0004 0900 0006
00000d0: 0d02 0032 0006 0e02 0000 0008 0f03 0000
00000e0: 01b8 0014 1003 0015 4618 0005 8aac 0015
00000f0: 4618 0001 c520 0004 1100 0004 0900 0006
0000100: 0d02 0032 0006 0e02 0000 0008 0f03 0000
0000110: 01b8 0014 1003 0008 866c 0007 7fec 0008
0000120: 866c 0004 dcec 0004 1100 0004 0900 0006
0000130: 0d02 0034 0006 0e02 0000 0008 0f03 0000
```


DEF Introduction

- Design Exchange Format (DEF)
 - An open specification for representing physical layout of an IC
- DEF includes
 - Die area
 - Components
 - Special nets

```
SPECIALNETS 6144 ;  
- W4822  
+ ROUTED ME4 1000 ( 1273700 377600 ) ( 1272480 * ) ;  
- W2328  
+ ROUTED ME5 440 ( 862450 78420 ) ( * -14920 )  
NEW ME3 440 ( 1394200 363180 ) ( * 116000 ) ;  
- W1517  
+ ROUTED ME3 440 ( 558700 491500 ) ( * 318700 )  
NEW ME4 1000 ( 512730 191100 ) ( 505260 * ) ;  
- W3811  
+ ROUTED ME5 440 ( 1418950 491500 ) ( * 361970 )  
NEW ME3 440 ( 462700 318420 ) ( * 160760 ) ;  
- W761  
+ ROUTED ME3 440 ( 275200 491500 ) ( * 397780 )  
NEW ME5 440 ( 103450 431820 ) ( * 168220 ) ;
```

```
VERSION 5.6 ;  
DIVIDERCHAR "/" ;  
BUSBITCHARS "[" ;  
DESIGN CS_GEN ;  
  
UNITS DISTANCE MICRONS 1000 ;  
  
PROPERTYDEFINITIONS  
COMPONENTPIN text STRING ;  
END PROPERTYDEFINITIONS  
  
DIEAREA ( 0 0 ) ( 1535670 576790 ) ;  
  
COMPONENTS 10496 ;  
- via5922 Via34  
+ PLACED ( 345480 -370 ) N ;  
- via5923 Via34  
+ PLACED ( 405480 -370 ) N ;  
- via5926 Via34  
+ PLACED ( 1101480 -370 ) N ;  
- via5927 Via34  
+ PLACED ( 1185480 -370 ) N ;
```

Open Virtuoso

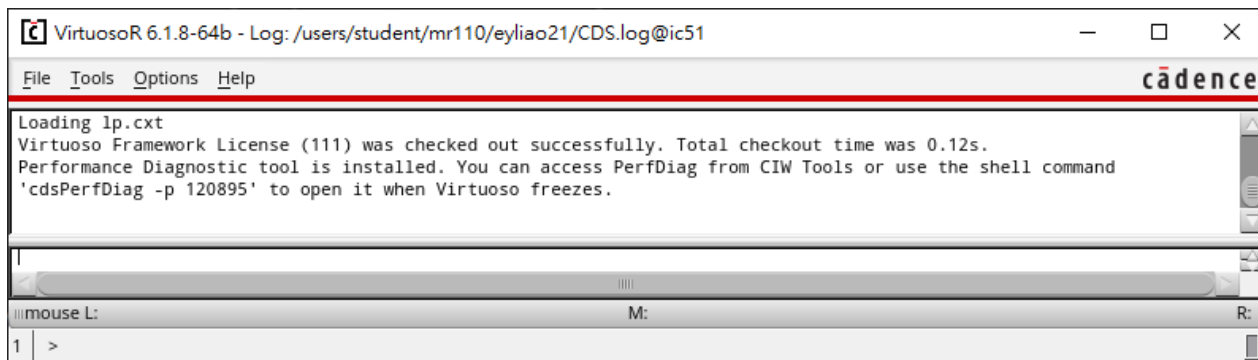
- Open Virtuoso

```
$ cd HW5_visual/
```

```
$ setenv OA_HOME /usr/cad/cadence/IC/cur/oa_v22.60.074
```

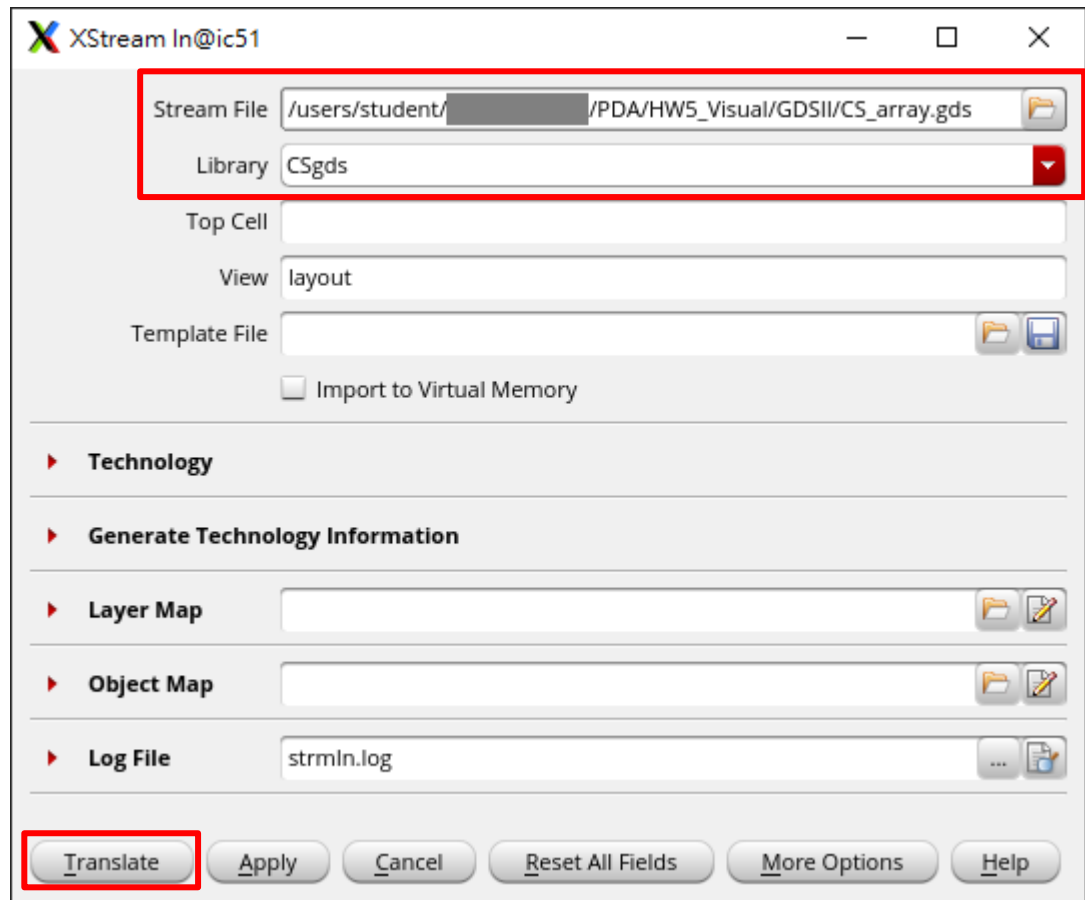
```
$ virtuoso &
```

```
[ @ic51 ~/PDA]$ cd HW5_visual/  
[ @ic51 HW5_visual]$ setenv OA_HOME /usr/cad/cadence/IC/cur/oa_v22.60.074  
[ @ic51 HW5_visual]$ virtuoso &  
[1] 236417  
[ @ic51 HW5_visual]$ C: unknown locale  
[ @ic51 HW5_visual]$
```



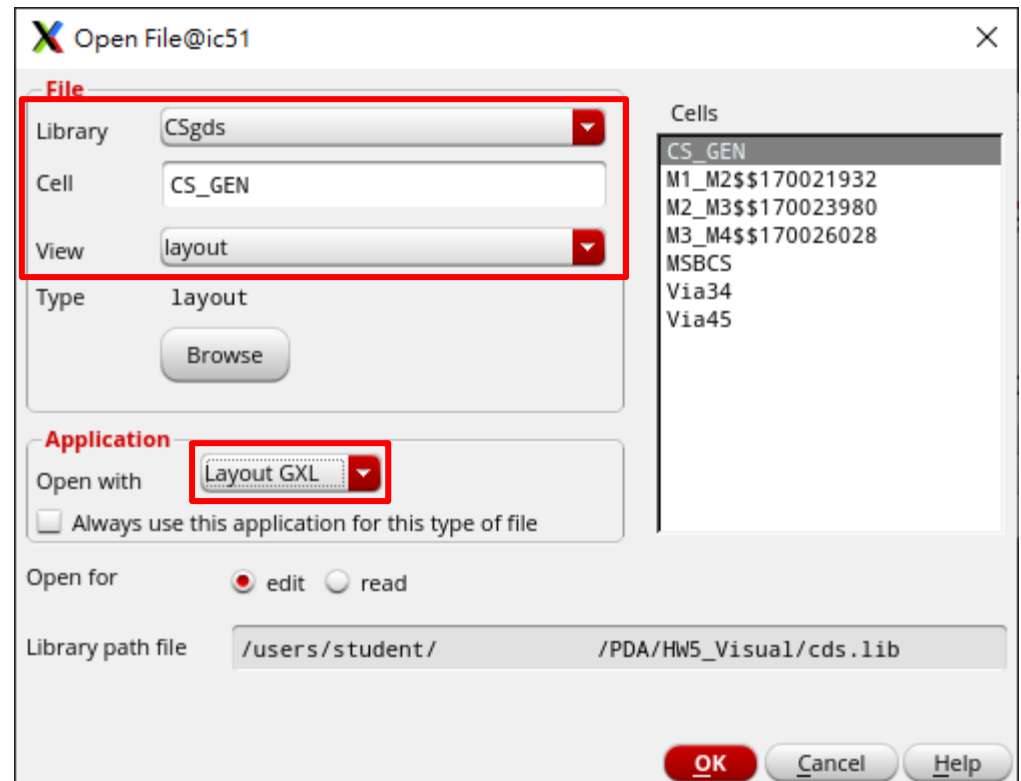
Import GDSII

- Import “CS_array.gds” and specify a new library “CS_gen”
 - Left click: File → Import → Stream...
 - Stream File: GDSII/CS_array.gds
 - Library: CSgds
 - Left click: Translate
- Technology file can be loaded if one is available

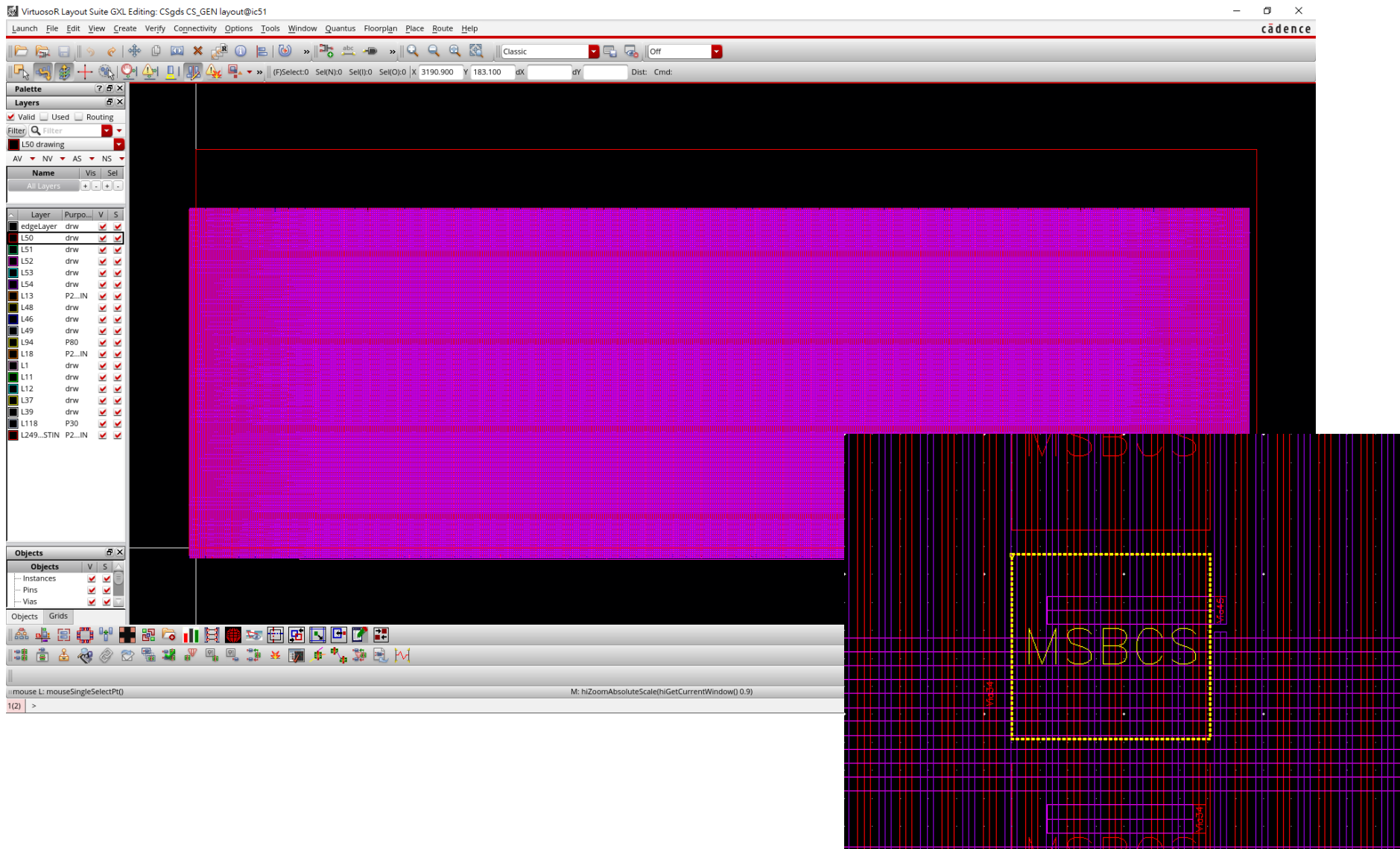


Import GDSII (cont'd)

- **Open the layout**
 - Left click: File → Open...
 - Library: CSgds
 - Cell: CS_GEN
 - View: layout
 - Application: Layout GXL
 - Left click: OK

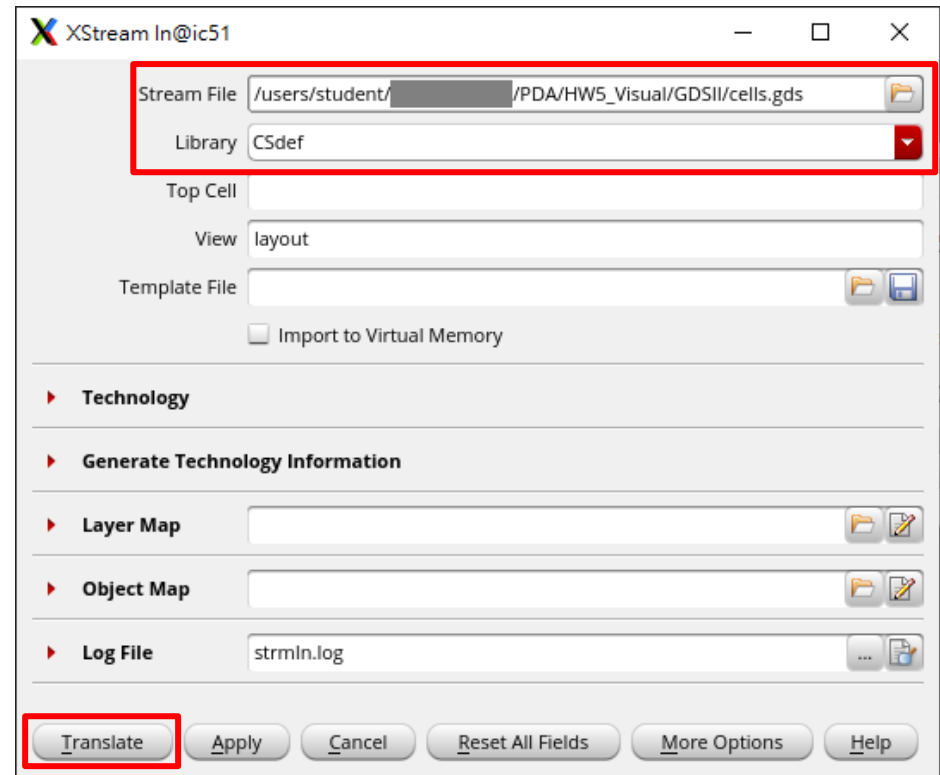


Import GDSII (cont'd)



Import DEF

- The layout automation flow requires the layouts (*.gds) of device components and then generates a DEF file specifying the placement and routing of the components
- Step 1: import “cells.gds” and specify a new library “CSdef”
 - Left click: File → Import → Stream...
 - Stream File: GDSII/cells.gds
 - Library: CSdef
 - Left click: Translate



Import DEF (cont'd)

- **Step 2: import “CS.lef” to “CSdef”**
 - Left click: File → Import → LEF...
 - LEF File Name: LEF/CS.lef
 - Target Library Name: CSdef
 - Left click: OK
- **A LEF file (specifying the design rules of metal layers) is required before importing a DEF file**

LEF In@ic51

LEF File Name: dent / ... / PDA/HW5_Visual/LEF/CS.lef

Target Library Name: CSdef

☐ Overwrite ☐ Share Library

Target Library Path:

Target Tech Library Name:

Target Tech Library Path:

Ref. Technology Libraries:

Macro Target View Name: abstract

Log File Name:

Layer Map File Name:

☐ Use Template File ☒ Use GUI Fields

Template File Name:

Save Template File Name: ... Save

Comment Char:

Pin Purpose:

☐ Use Text Layer same as Pin Layer

Text Layer Name:

Text Purpose:

Text Height:

☐ Compress Compress Level:

☐ Map Conflicts

☐ Import PnR Library Data Only

☐ Import LEF LAYER LEF58 Properties Constraints to 'foundry_innovus' Group

☐ Do not Modify the Original 'display.drf' File

☐ Create Fixed Via Definitions

Verilog File Name: ...

OK Cancel Defaults Apply Help

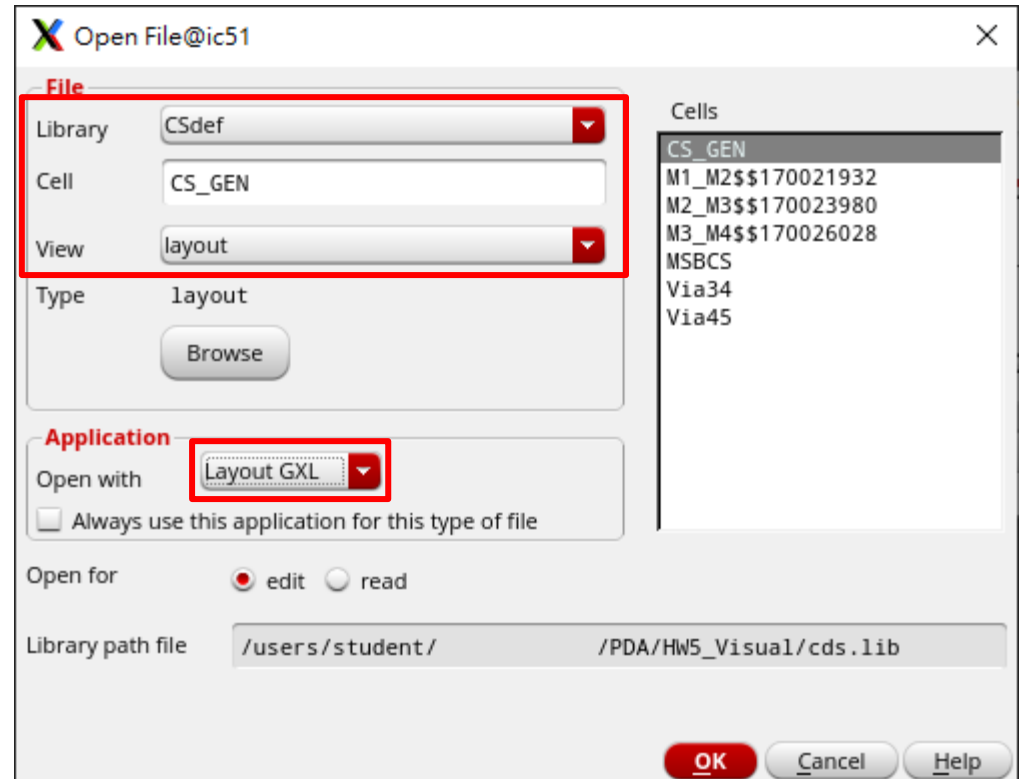
Import DEF (cont'd)

- **Step 3: import DEF to “CSdef”**
 - Left click: File → Import → DEF...
 - DEFIn File Name: DEF/CS.def
 - Target Library Name: CSdef
 - Left click: OK

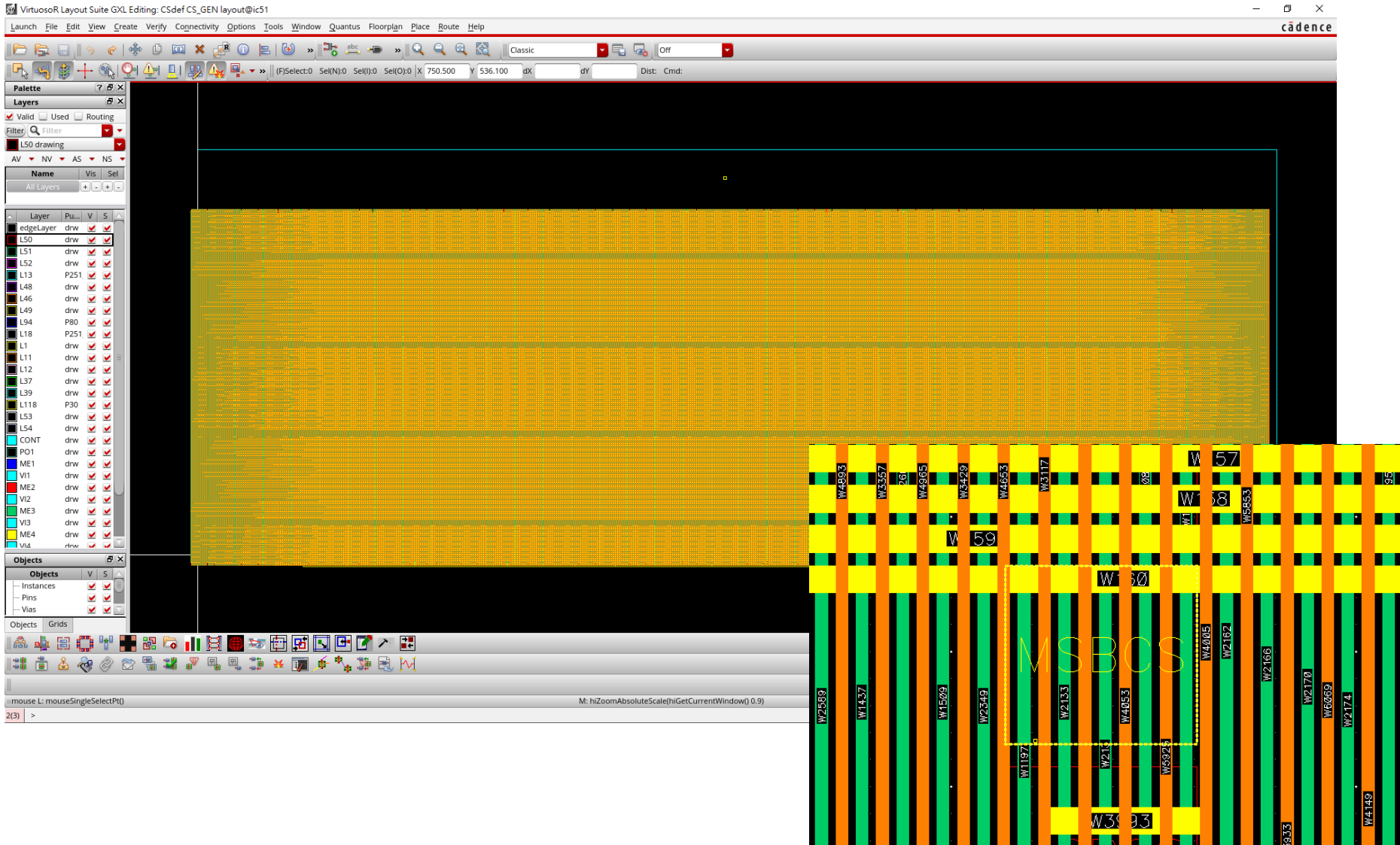
The screenshot shows the 'DEF In@ic51' dialog box. The 'DEFIn File Name' field is set to 'tudent/[redacted]/PDA/HW5_Visual1/DEF/CS.def' and the 'Target Library Name' is set to 'CSdef'. Both fields are highlighted with a red rectangle. The 'OK' button at the bottom is also highlighted with a red rectangle. Other fields include 'Target Library Path', 'Ref. Technology Libraries', 'Create a module hierarchy from hierarchical names', 'Share Library', 'New Library', 'Technology From Library', 'Target Cell Name', 'Target View Name', 'Component View List', 'Master Library List', 'Overwrite Design', 'Log File Name', 'Use Template File', 'Use GUI Fields', 'Template File Name', 'Save Template File Name', 'Comment Char', 'Pin Purpose', 'Do not create any routing data', 'Layer Map File Name', 'User Skill File', 'Compress', 'Compress Level', and 'Ignore DRCFILL Shape Tag & Translate on Drawing Purp'.

Import DEF (cont'd)

- **Open the layout**
 - Left click: File → Open...
 - Library: CSdef
 - Cell: CS_GEN
 - View: layout
 - Application: Layout GXL
 - Left click: OK



Import DEF (cont'd)

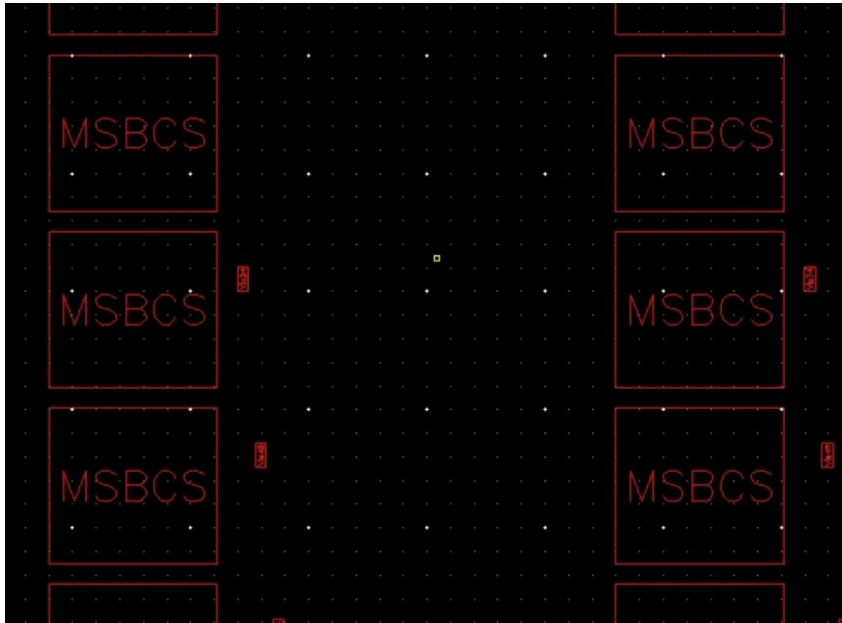


Shortcuts

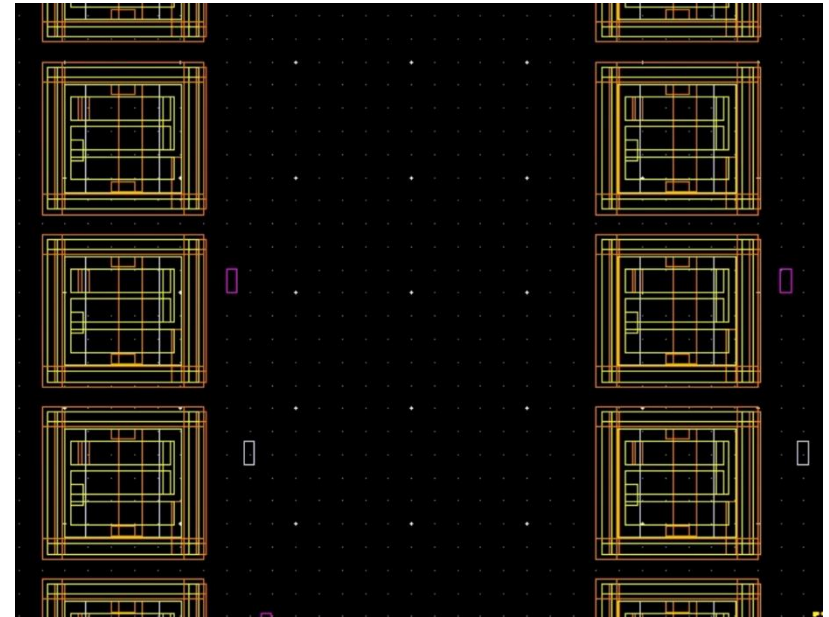
- **shift+z**
 - Zoom out
- **ctrl+z**
 - Zoom in
- **z**
 - Zoom to area
- **f**
 - Zoom to fit all
- **shft+f**
 - Show cell layouts
- **ctrl+f**
 - Hide cell layouts

Reminding

- Cell layouts can only be seen after importing the gds file of device components



Hide cell layouts (ctrl+f)



Show cell layouts (shift+f)

Python Introduction

Python

- **High level interpreted programming language**
- **We will use python to accomplish CS placement and routing automation**
- **Pre-training**
 - Run python
 - Define variable
 - Different type of variables
 - Operator computation
 - For loop and list of variables definition
 - Instance instantiation
 - Run python script

Run Python

- Just type “python” in your terminal

\$ python3

```
[ @ic51 python]$ python3
Python 3.6.8 (default, Nov 16 2020, 16:55:22)
[GCC 4.8.5 20150623 (Red Hat 4.8.5-44)] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> █
```

- Exit python

>>> quit()

```
[ @ic51 python]$ python3
Python 3.6.8 (default, Nov 16 2020, 16:55:22)
[GCC 4.8.5 20150623 (Red Hat 4.8.5-44)] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> quit()
[ @ic51 python]$ █
```

Define Variable

- Give a name and initial value

```
Python 3.6.8 (default, Nov 16 2020, 16:55:22)
[GCC 4.8.5 20150623 (Red Hat 4.8.5-44)] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> x = 10
>>> y = 20
>>> 
```

- Print variables

```
>>> x = 10
>>> y = 20
>>> print(x)
10
>>> print(y)
20
>>> 
```


Different Type of Variables

- The variable types can be int, float, string, or even an instance defined by users

```
>>> x = 5
>>> y = 5.7
>>> z = 'current source array'
>>> type(x)
<class 'int'>
>>> type(y)
<class 'float'>
>>> type(z)
<class 'str'>
>>> █
```

```
>>> from myObject import Component
>>> c = Component('MCBCS', 'C1', 0, 0)
>>> type(c)
<class 'myObject.Component'>
>>> █
```

Operator Computation

- Let's try all operators

```
>>> 10 + 7
17
>>> 10 - 7
3
>>> 10 * 7
70
>>> 10 / 7
1.4285714285714286
>>> 10 // 7
1
>>> 10 ** 7
10000000
>>> pow(10, 7)
10000000
>>> █
```

```
>>> import math
>>> math.sqrt(100)
10.0
>>> math.sin(math.radians(90))
1.0
>>> █
```

For Loop

- Single and double for loop example

```
>>> for i in range(10):  
...     x = i * 10  
...     print(x)  
...  
0  
10  
20  
30  
40  
50  
60  
70  
80  
90  
>>>
```

```
>>> for i in range(3):  
...     for j in range(3):  
...         print('{:d} {:d}'.format(i, j))  
...  
(0 0)  
(0 1)  
(0 2)  
(1 0)  
(1 1)  
(1 2)  
(2 0)  
(2 1)  
(2 2)  
>>>
```

List of Variables

- Initialize a one/two-dimensional list

```
>>> listA = [1, 2, 3]
>>> type(listA)
<class 'list'>
>>> listA[0]
1
>>> listA[1]
2
>>> listA[2]
3
>>> █
```

```
>>> listA = [[1, 2, 3], [4, 5, 6]]
>>> listA[0]
[1, 2, 3]
>>> listA[1]
[4, 5, 6]
>>> listA[0][0]
1
>>> listA[1][2]
6
>>> █
```

List of Variables (cont'd)

- Initialize a one/two-dimensional empty list with designated type

```
>>> listA = [float for i in range(4)]
>>> listA
[<class 'float'>, <class 'float'>, <class 'float'>, <class 'float'>]
>>> len(listA)
4
>>> █
```

```
>>> listA = [[int for j in range(3)] for i in range(2)]
>>> listA
[<class 'int'>, <class 'int'>, <class 'int'>], [<class 'int'>, <class 'int'>, <class 'int'>]]
>>> len(listA)
2
>>> len(listA[0])
3
>>> █
```

List Append

- Incrementally append elements to a list

```
>>> listA = [1, 2, 3]
>>> listA
[1, 2, 3]
>>> listA.append(4)
>>> listA
[1, 2, 3, 4]
>>> █
```

Instance Instantiation

- Three pre-defined instances will be applied

- Die

- design_name
 - _x1
 - _y1
 - _x2
 - _y2

```
>>> from myObject import Die
>>> die = Die('cs_array', 0, 0, 10000, 10000)
>>> die.design_name
'cs_array'
>>> die._x1
0
>>> die._x2
10000
>>> die._y1
0
>>> die._y2
10000
>>> █
```

Instance Instantiation (cont'd)

- Component
 - lib_name
 - inst_name
 - _x
 - _y

```
>>> from myObject import Component
>>> cs_unit = Component('MSBCS', 'cs1', 0, 0)
>>> cs_unit.lib_name
'MSBCS'
>>> cs_unit.inst_name
'cs1'
>>> cs_unit._x
0
>>> cs_unit._y
0
>>> █
```


Instance Instantiation (cont'd)

- SpecialNet
 - inst_name
 - layer
 - _x1
 - _y1
 - _x2
 - _y2

```
>>> from myObject import SpecialNet
>>> net = SpecialNet('n1', 'ME3', 0, 0, 440, 700)
>>> net.inst_name
'n1'
>>> net.layer
'ME3'
>>> net._x1
0
>>> net._y1
0
>>> net._x2
440
>>> net._y2
700
>>> █
```

Run Python Script

- Create a “test.py” file

```
$ vim test.py
```

- Write some python command in “test.py” and save

```
1 import sys
2
3 argc = len(sys.argv)
4 print('read {:d} args'.format(argc))
5
6 for i, arg in enumerate(sys.argv):
7     print('arg{:}: {}'.format(i, arg, type(arg)))
8
```

- Run “test.py” script in terminal

```
$ python3 test.py abc 123
```

```
[ @ic51 python]$ vim test.py
[ @ic51 python]$ python3 test.py abc 123
read 3 args
arg0: test.py, type: <class 'str'>
arg1: abc, type: <class 'str'>
arg2: 123, type: <class 'str'>
[ @ic51 python]$
```