GenISys
Advancing the Standard

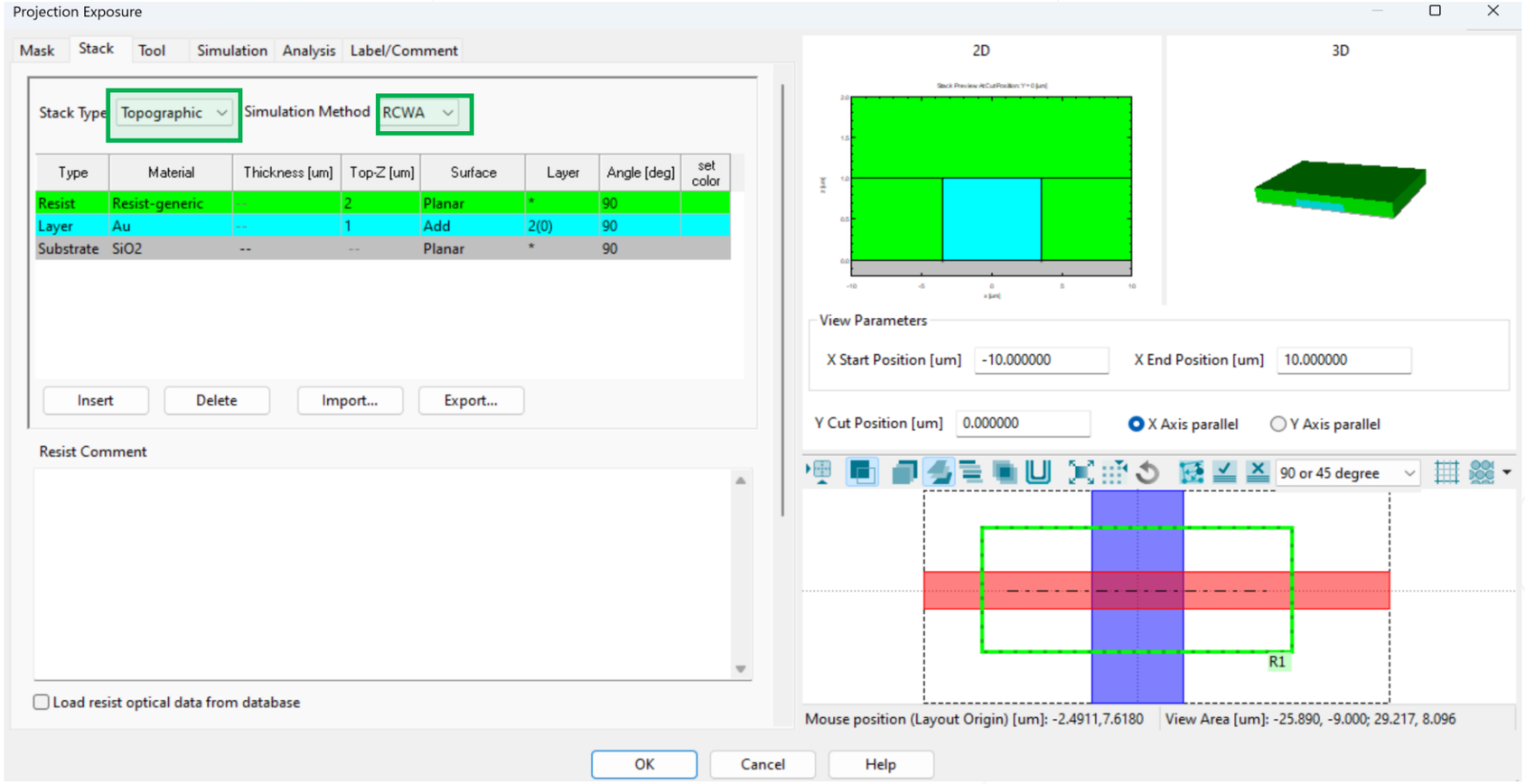
LAB

What‘s new v5.12

Topographic Simulation

RCWA Topographic Simulation

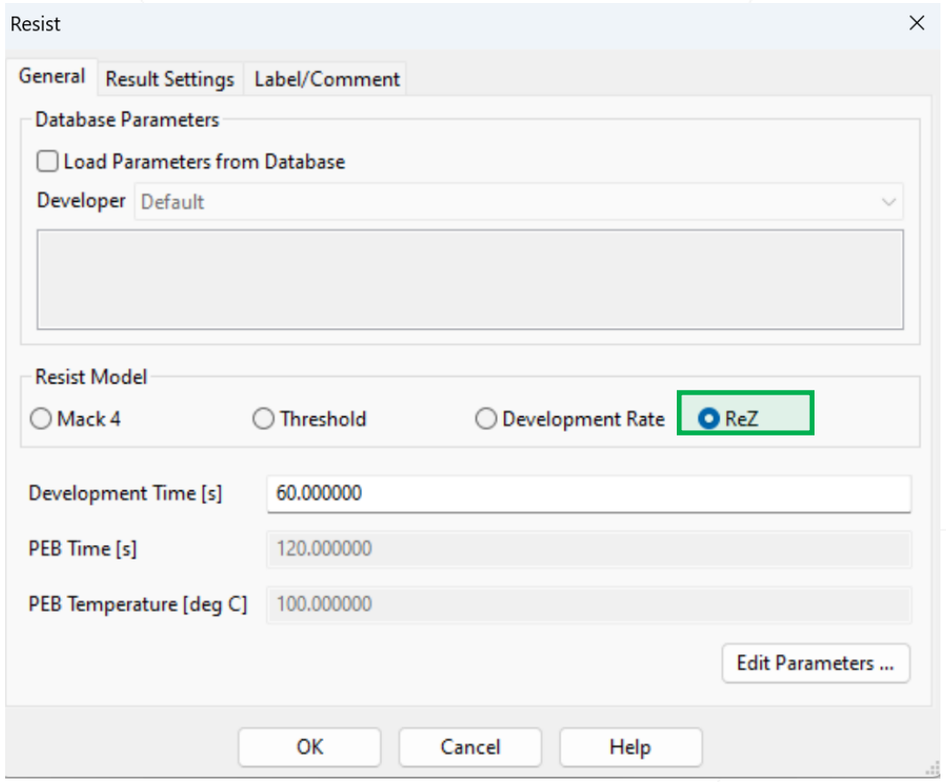
- RCWA (Rigorous Coupled-Wave Analysis) method is introduced for topographic simulation with reduction in simulation noise.
  - The user can choose between the integral method (default) and the RCWA



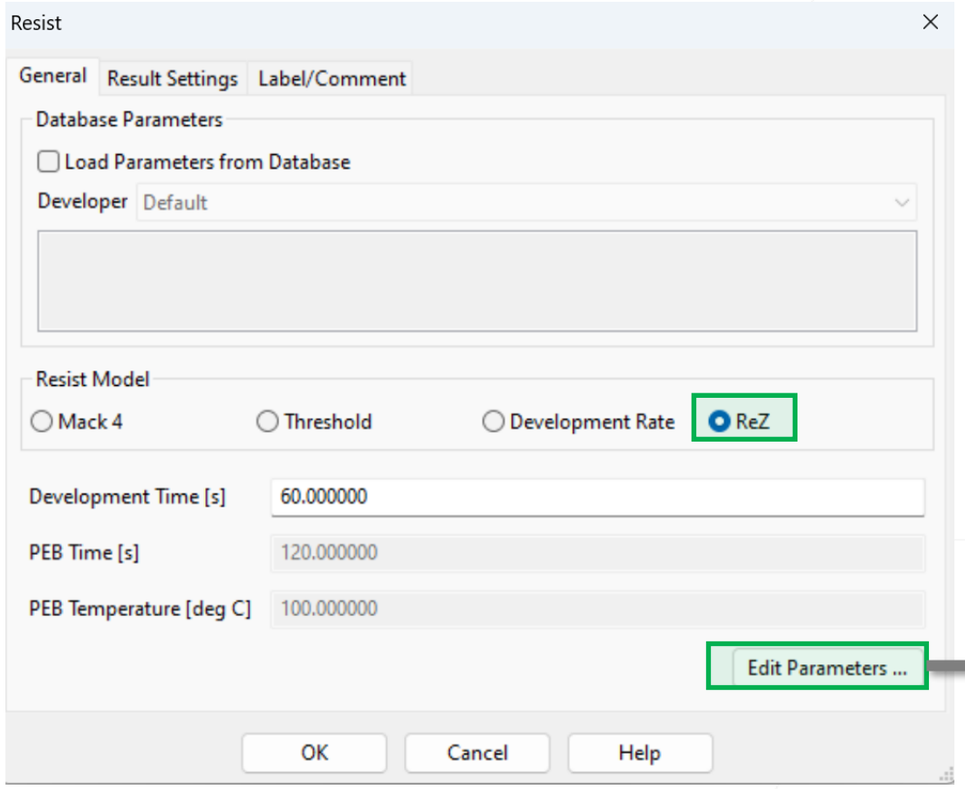
ReZ Resist Model

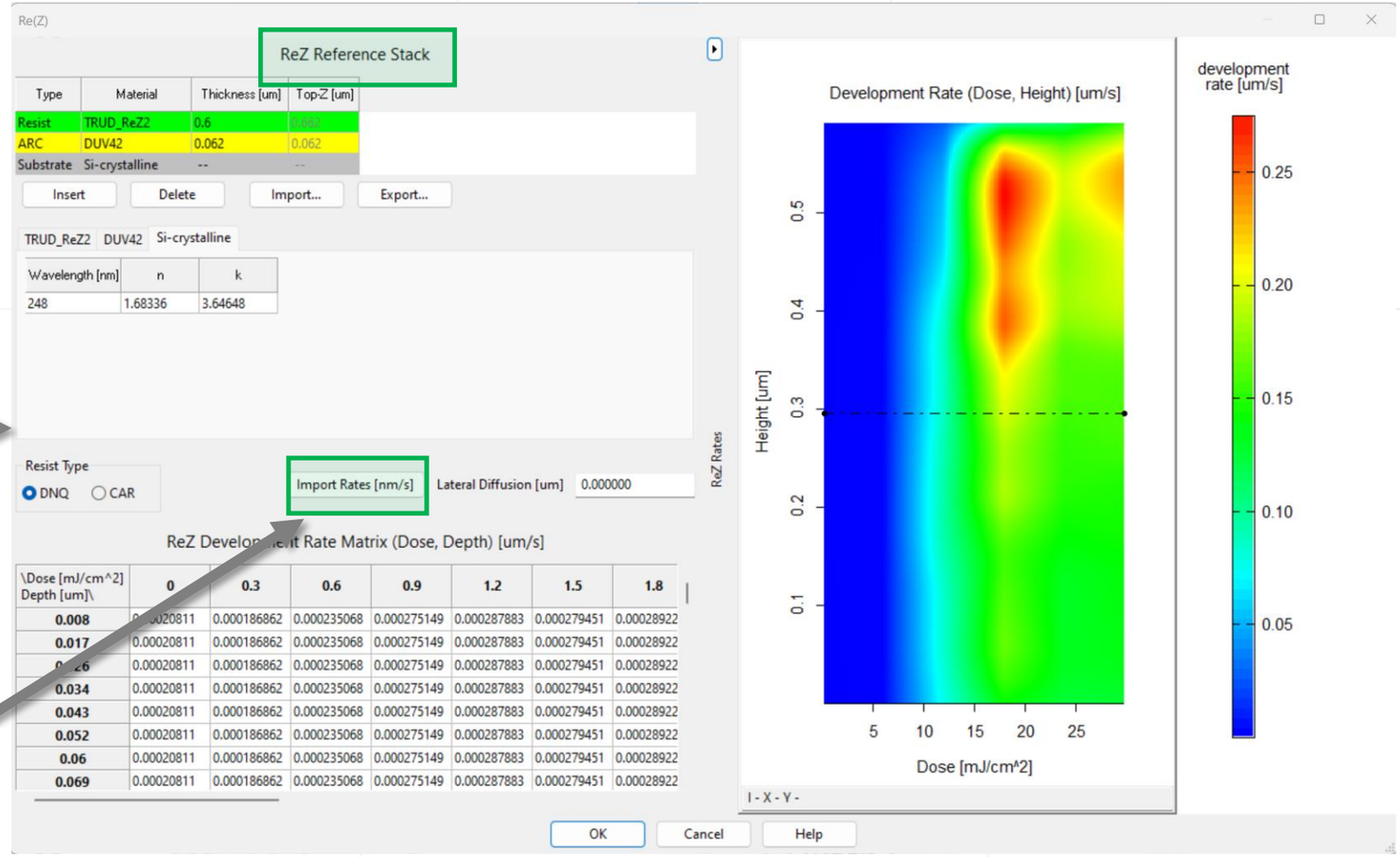
ReZ Model

- In resist model, **ReZ Model** applies directly resist development rates, e.g. measured by LTJ (Litho Tech Japan) RDA(Resist Development Analyzer).
  - Development Rate Matrix is dose and depth into resist dependent.



ReZ Model



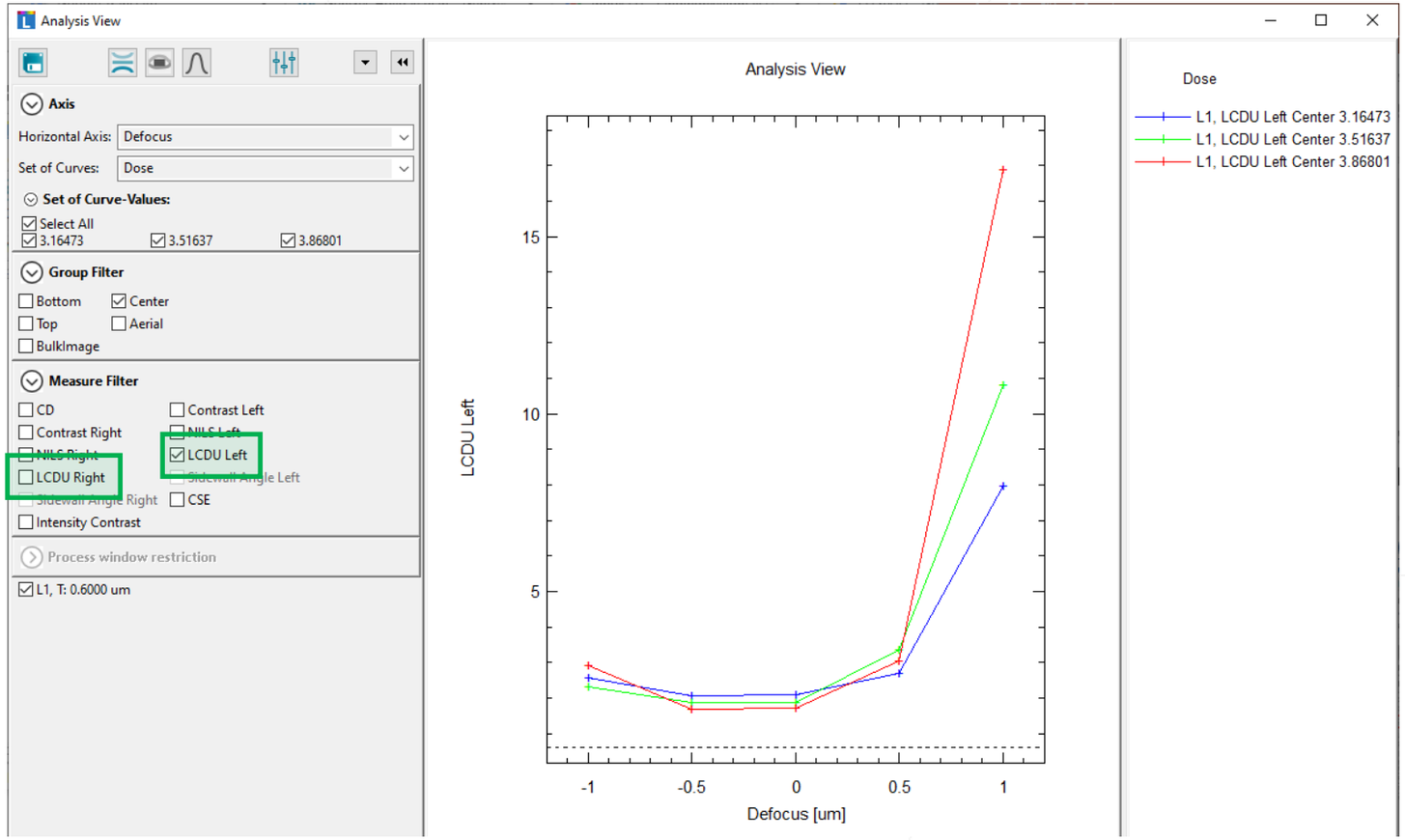


The experimentally measured development rates are imported for the reference stack

Local CD Uniformity

Local CD Uniformity (LCDU)

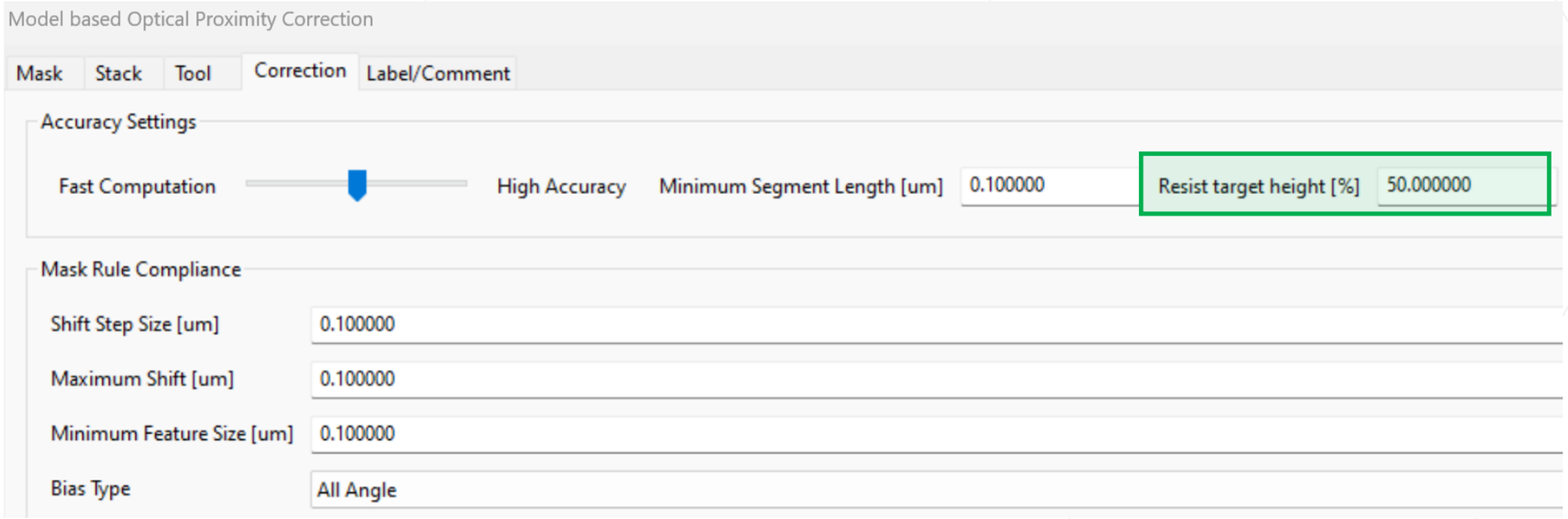
- Local CD Uniformity (LCDU) is available in the analysis view.
- LCDU is based on the NILS value according to the equation in *EUVL: the natural evolution microlithography*.



Model OPC Improvements

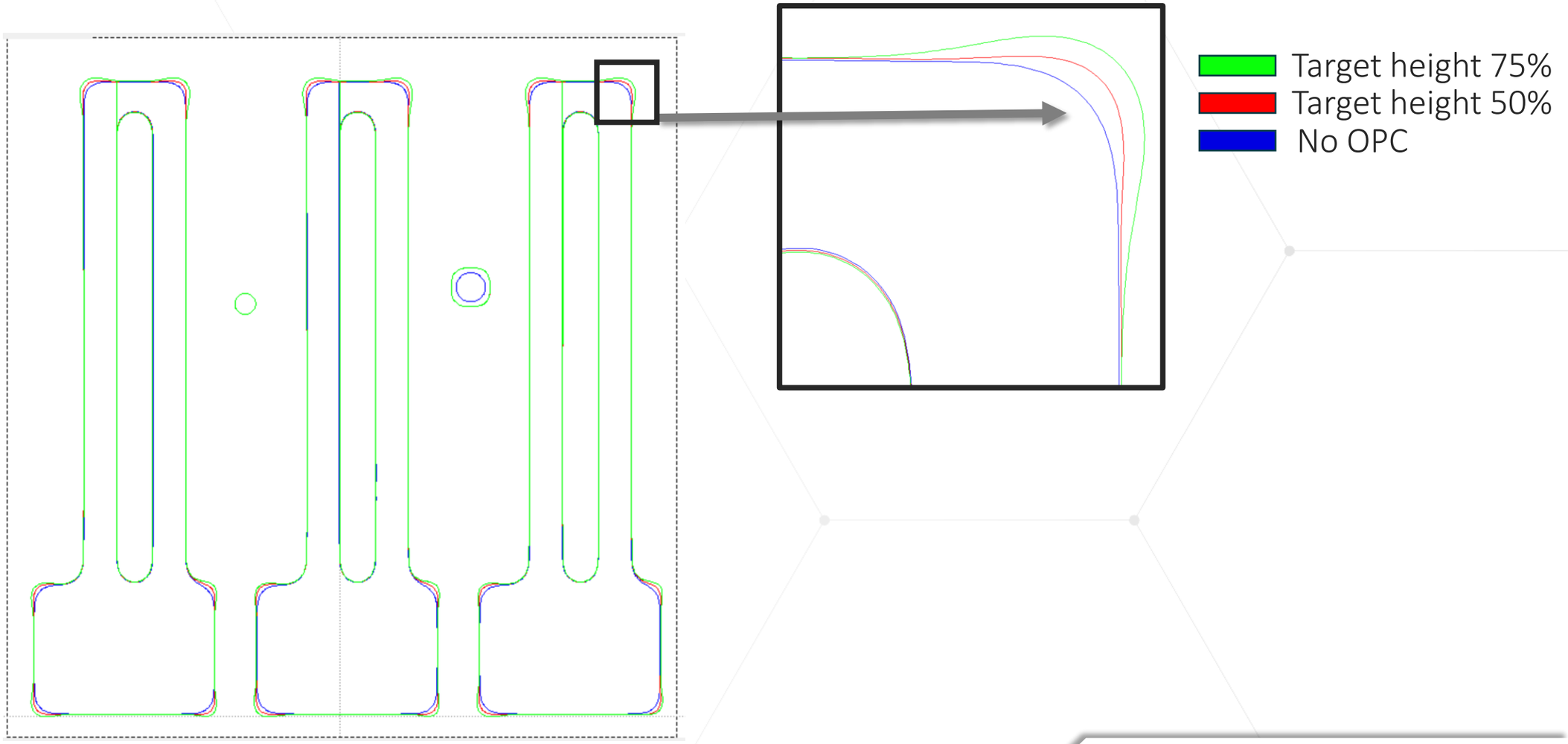
User Defined Height Position

- User can specify the target height in the resist by defining the relative resist height percentage (being 0 the resist bottom and 100% the resist top)



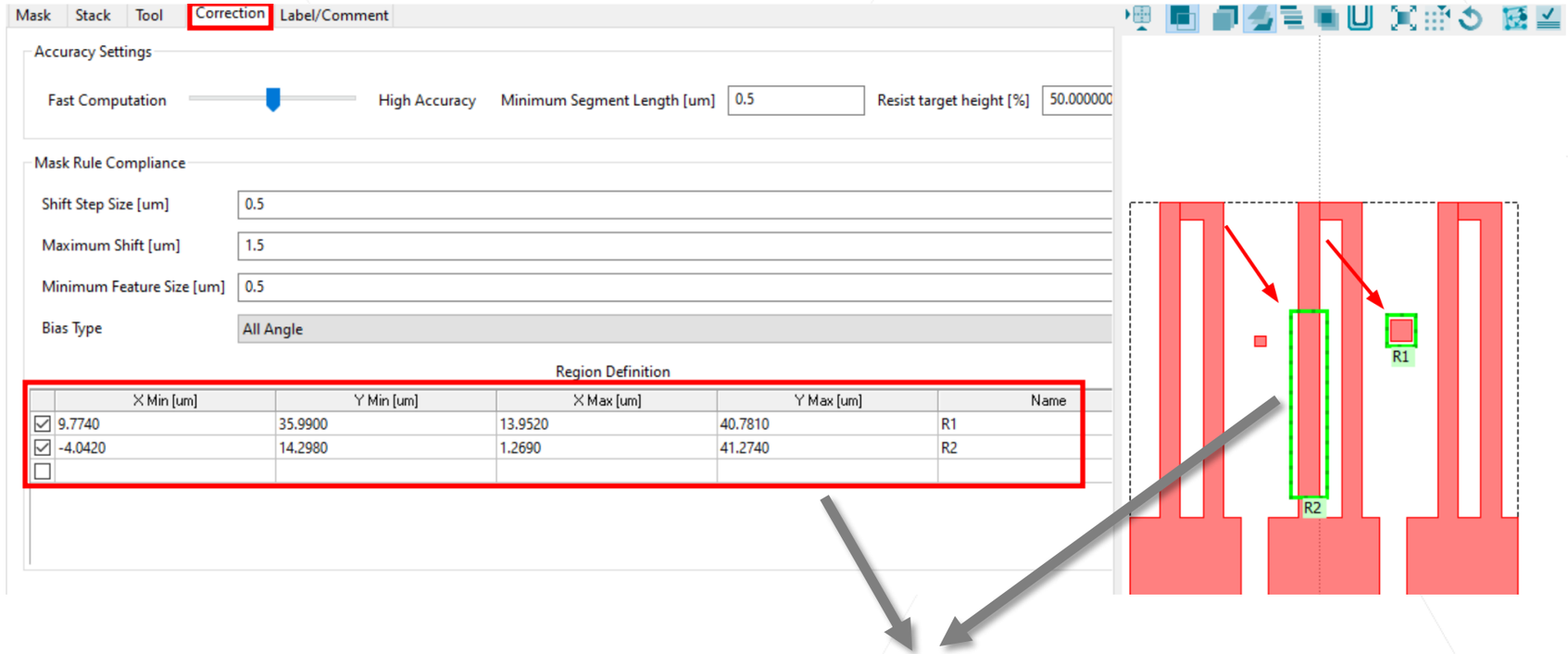
User Defined Height Position

- User gets more comprehensive mask corrections by changing the relative resist height definition.



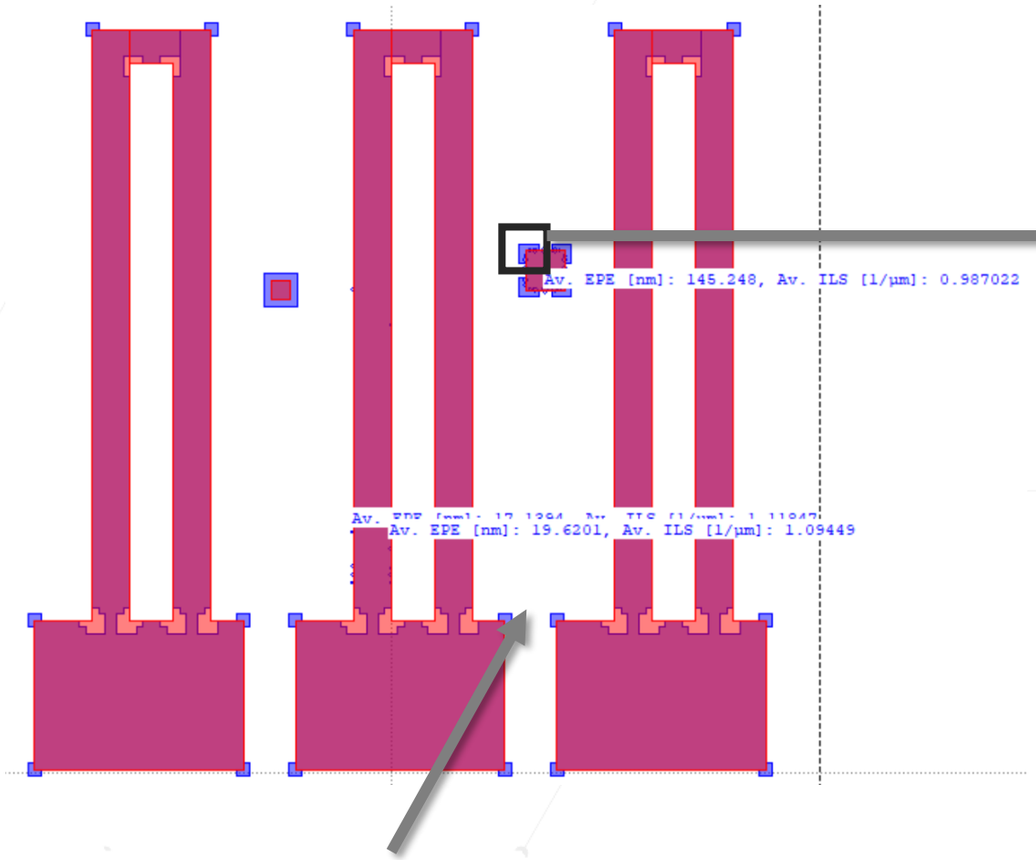
Calculating ILS and EPE during OPC

- OPC now offers the option to calculate image log slope (ILS) and estimated edge placement error (EPE) after OPC correction.
- Adding a metrology region on the correction tab will collect this information and display them in the VIEWER result layout.

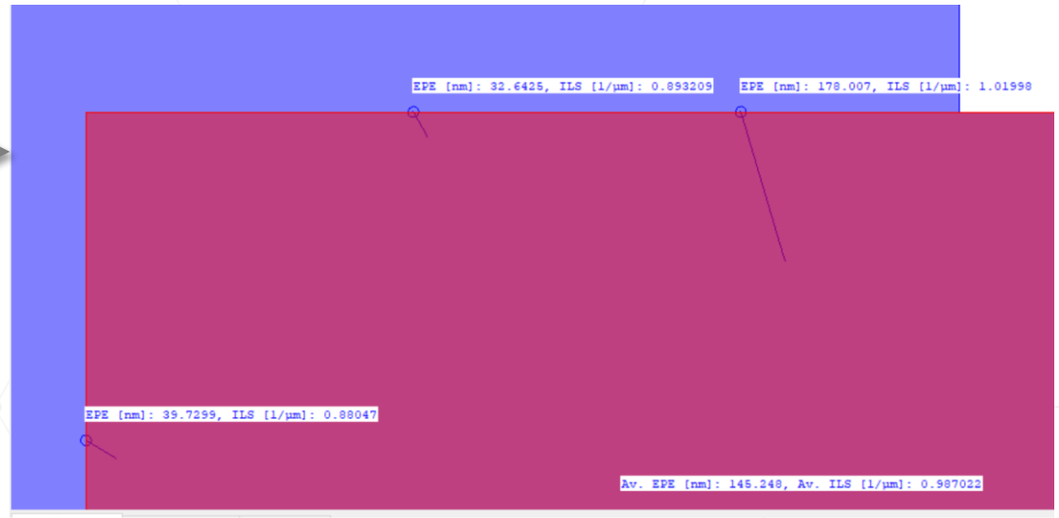


Metrology regions for EPE and ILS

Calculating ILS and EPE during OPC



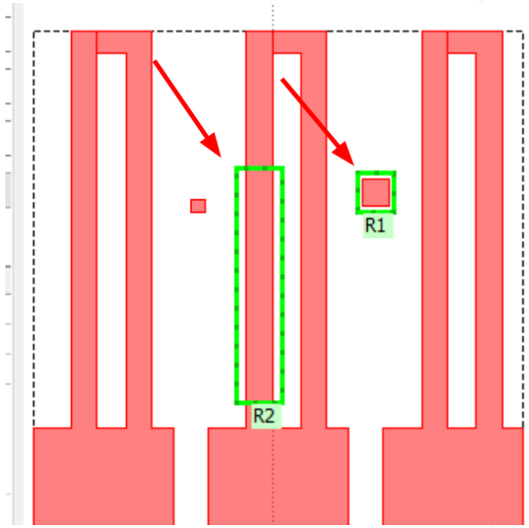
EPE and ILS values are displayed in the VIEWER result at several points of the layout, as well as the average values per region



Zooming in displays the specific region-of-interest EPE and ILS

Calculating ILS and EPE during OPC

- Mean, Variance and a histogram of the distribution are calculated and added to the log output file.



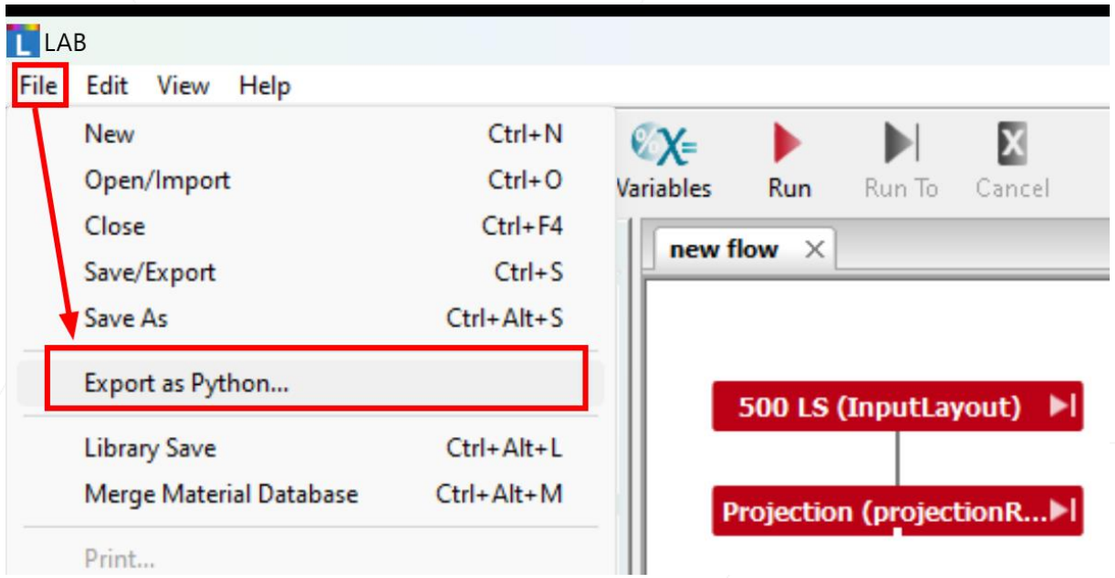

```
Region: R1
EPE: Mean: 145.247859 Variance: 8539.913490
    [32.000000 nm, 33.000000 nm] : 2
    [38.000000 nm, 39.000000 nm] : 2
    [39.000000 nm, 40.000000 nm] : 2
    [46.000000 nm, 47.000000 nm] : 2
    > 102.000000 nm: 12

ILS: Mean: 0.987022 Variance: 0.009130
    [0.000000 µm, 1.000000 µm] : 10
    [1.000000 µm, 2.000000 µm] : 10
Region: R2
  EPE: Mean: 18.379745 Variance: 9.159113
    [13.000000 nm, 14.000000 nm] : 1
    [14.000000 nm, 15.000000 nm] : 1
    [15.000000 nm, 16.000000 nm] : 2
    [19.000000 nm, 20.000000 nm] : 1
    [20.000000 nm, 21.000000 nm] : 3
    [21.000000 nm, 22.000000 nm] : 2
```

Histogram data in the log file

LAB Python API

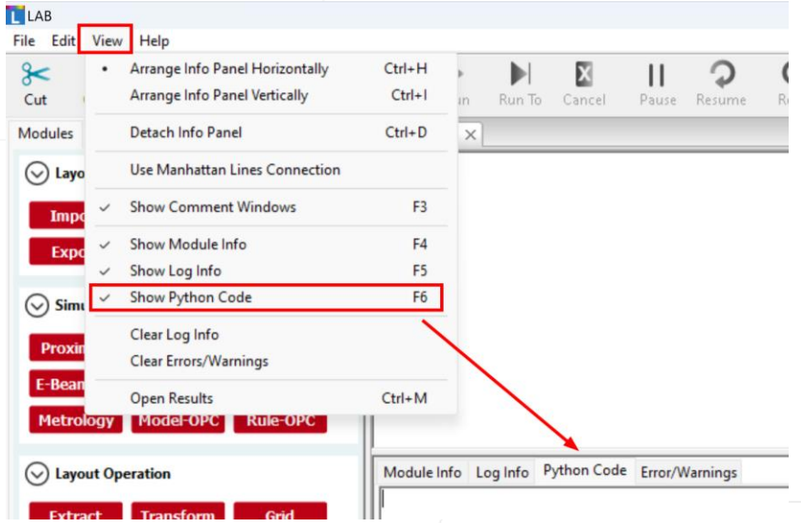
Exporting Python Code

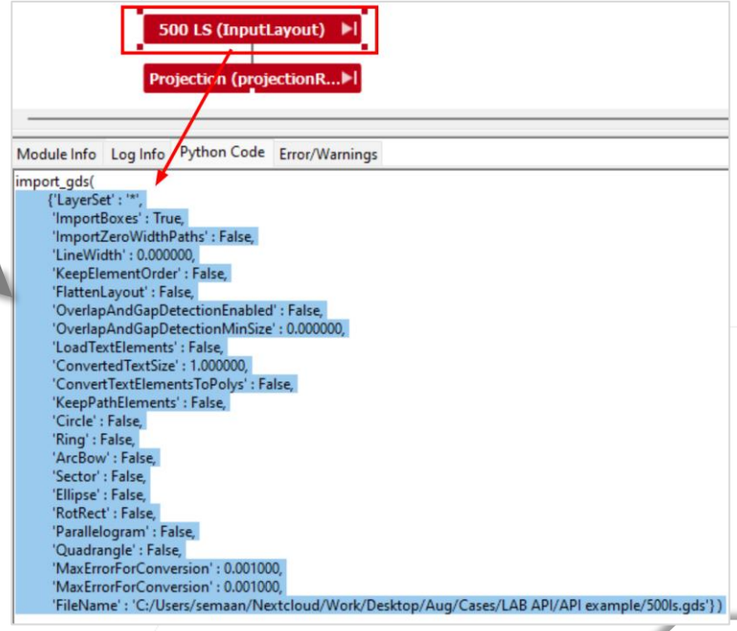
- LAB simulations can run using a Python API
- There are two ways to write the Python code:
  1. Using the ***Export as Python*** option to use a Python script of an entire LAB flow



Writing Python Code

2. Editing an example Python code or parameters of a module:
   - Go to the view tab in the LAB GUI and enable the option "**Show Python Code**“
   - The Python code and the parameters for each module will appear in the Python code section once a module is selected
   - Copy the full code or specific lines from this section as needed to your Python code





Python API configuration

- Framework to use in the preferred Python IDE:
  - # Specifying the directory path where LAB is installed
    - Import sys
    - sys.path.append('installation directory path'). For example:
      - sys.path.append ('C:/Program Files/LAB/v5.12.0.beta_x64')
  - #Imoprt the LABpy library
    - import LABpy
  - # load the python LAB object
    - LAB = LABpy.GLAB()

```
# load necessary system libraries
import sys

# set the path to the LAB python interface libraries
sys.path.append("C:/Program Files/LAB/v5.12.0.beta_x64")

import LABpy
try :

    # load the python LAB object
    LAB = LABpy.GLAB()
```

LAB object method: LAB.import_gds()

- **LAB.import_gds()** is a method for importing a layout

Example: importing the 500ls.gds

```
InputLayout = LAB.import_gds(
     {'LayerSet' : '*',
      'ImportBoxes' : True,
      'ImportZeroWidthPaths' : False,
      'LineWidth' : 0.000000,
      'KeepElementOrder' : False,
      'FlattenLayout' : False,
      'OverlapAndGapDetectionEnabled' : False,
      'OverlapAndGapDetectionMinSize' : 0.000000,
      'LoadTextElements' : False,
      'ConvertedTextSize' : 1.000000,
      'ConvertTextElementsToPolys' : False,
      'KeepPathElements' : False,
      'Circle' : False,
      'Ring' : False,
      'ArcBow' : False,
      'Sector' : False,
      'Ellipse' : False,
      'RotRect' : False,
      'Parallelogram' : False,
      'Quadrangle' : False,
      'MaxErrorForConversion' : 0.001000,
      'MaxErrorForConversion' : 0.001000,
      'FileName' : '500ls.gds'})
```

Layout file directory and name

LAB object method: LAB.projection()

- **LAB.projection()** is a method for performing projection simulations
- It contains all the projection parameters as a Python dictionary (not all shown here)

```
projectionResults = LAB.projection(InputLayout,
    {'Projection' : True,
     'Spectrum' : [[365, 5, 1]],
     'ExposureDose' : 1.730505,
```

The InputLayout is the output of the previous LAB.import_gds() method

LAB object method: LAB.get_measure_value()

- The **LAB.get_measure_value()** method returns lithography measurements, such as CD, NILS, contrast, etc.

Example: returning the line CD at the center of the stack

```
# CD Measurements: can be perforumed for ("Top", 'Center', 'Bottom', "Aerial")
Line1CenterCD = LAB.get_measure_value(projectionResults, "CD", "Center", "Line1")
print('Line1 Center_CD =',Line1CenterCD)
```

The projectionResults is the output of the previous LAB.projection() method

LAB object method: LAB.get_3d_numpy()

- The **LAB.get_3d_numpy()** returns images as NumPy arrays as a dictionary, with their region names and physical quantity as key
- For example, dictionary keys for a projection model can be: ‘Region1_Bulk Image' or ‘Region1 _Aerial Image'

Example: returning the NumPy arrays of the bulk image

```
numpyImages = LAB.get_3d_numpy(projectionResults)
key = 'Region1_Bulk Image'
myImage = numpyImages[key]
```

The projectionResults is the output of the previous LAB.projection() method

The output: myImage.numpy_array

- **myImage.numpy_array** is a 3D numpy array of the Bulk Image
- myImage.numpy_array.shape returns the shape in (Z,Y,X)

Example: plotting a z-slice extracted from myImage.numpy_array

```
numpyImages = LAB.get_3d_numpy(projectionResults)
key = 'Region1_Bulk Image'
myImage = numpyImages[key]
```

```
# Plot z-slice with coordinate positions
plt.imshow(myImage.numpy_array[int(myImage.numpy_array.shape[0]/2)], extent=myImage.position_map.view_area(), cmap='hot', interpolation='nearest')
plt.show()
```

Choosing z-slice to plot

Setting the image dimensions

LAB object method: LAB.model_opc()

- **LAB.model_opc()** is a method for performing the model-based OPC (Optical Proximity Correction) on the mask layout
- It contains all the Model-OPC parameters defined in the LAB GUI, organized as a Python dictionary (not all parameters are shown here)

```
# Model-OPC
modelOpcResult = LAB.model_opc(InputLayout,
        {'Projection' : True,
         'Spectrum' : [[365, 5, 95], [405, 5, 75], [436, 5, 100]],
         'ExposureDose' : 3.12,
```

The output of the LAB.model_opc() method is the OPC-corrected layout, which can be use for new simulations

The InputLayout is the output of the previous LAB.import_gds() method (the original layout)

Thank You!

support@genisys-gmbh.com

Headquarters

GenISys GmbH
Eschenstr. 66
D-82024 Taufkirchen (Munich)
GERMANY

+49-(0)89-3309197-60

+49-(0)89-3309197-61

info@genisys-gmbh.com

USA Office

GenISys Inc.
P.O. Box 410956
San Francisco, CA
94141-0956
USA

+1 (408) 353-3951

usa@genisys-gmbh.com

Japan / Asia Pacific Office

GenISys K.K.
German Industry Park
1-18-2 Hakusan Midori-ku
Yokohama 226-0006
JAPAN

+81 (0)45-530-3306

+81 (0)45-532-6933

apsales@genisys-gmbh.com