IceLab: A Python-based framework for semiconductor device measurement and analysis

by Arnold van der Wal

Abstract

Icelab provides designers of semiconductor components or devices with Python tools that help them define and carry out highly-automated device measurements and data analysis.
It supports local and remote operation of semiconductor wafer probe-stations and measurement equipment.
Data is stored in a **MongoDB** database for subsequent analysis, including interactive visualization with **Bokeh** or **Matplotlib**.

Features

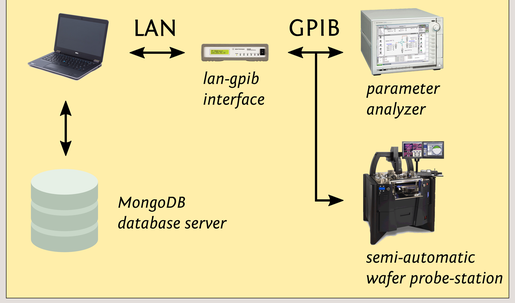
- Replaces labor-intensive manual measurements, thereby saving valuable time.
- Is flexible in type of measurements and instruments and ability to define measurement sequences.
- Uses structured storage of measured data in database for later (statistical) analysis.
- Powerful query and plotting methods enable visualization of almost any combination of parameters.

Rationale

- Commercial alternatives do not always offer the required measurement flexibility.
- Licenses can be too expensive to justify.

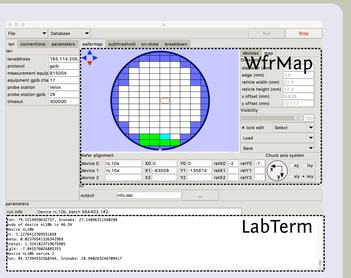
Infrastructure

- Multiple network options: LAN-GPIB, USB-GPIB, or fixed GPIB network card.
- Network communication with **pyVisa** in combination with **VISA** library (from **Keysight** or **National Instruments**), or **SICL** library (**Agilent/Keysight**).

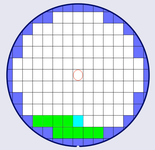


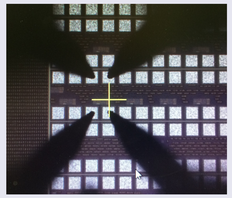
Measurement

- User builds own GUI-based measurement application.
- **Icelab** provides specialized super-widgets and simplified placement scheme.
- **Icelab** also provides drivers for communication with measurement instruments and probe-stations.




```python
from icelab.core import labui as ui
from icelab.langpib import connect
from icelab.drivers import B1500A, Nucleus
from icelab.db.mongolib import ConnectDB
from icelab.measurements import bvds

def setup(g, wnd):
    # place widgets on grid
    nb = ui.LabNotebook(wnd, row=0, column=1, ...)
    g.wfr = ui.WfrMap(nb, toplabel='wafermap', ...)
    g.pl = ui.LabPlot2D(nb, toplabel='breakdown')
    ...
    g.term = ui.LabTerm(wnd, row=3, column=0, columnspan=2, ...)

def enable(g):
    # open instrument connections
    g.par = connect(driver=B1500, gpib=17, url='apc1221').open(timeout=10000)
    g.probe = connect(driver=Nucleus, gpib=28, url='apc1221').open(timeout=10000)
    ...
    # open database connection
    g.db = ConnectDB(**ui.menu.database)


def execute(g):
    gate = g.par.module(slot=1)
    drain = g.par.module(slot=2)
    source = g.par.module(slot=3)

    for device in g.wfr.get_devices(db=g.db):
        # move probe to position and contact
        xpos, ypos = device.position()
        g.probe.move(xpos, ypos)
        g.probe.contact()

        # do breakdown measurement
        data = bvds(g.par, source=source, drain=drain, gate=gate)
        g.term.write('measured breakdown {}\n'.format(data.bvds))

        # store result in MongoDB database
        g.db.insert_db(device=device, data=data, ...)

        # plot leakage current
        g.pl.plot('idvd', data.vd, abs(data.idrain), color='red',
        g.pl.draw()

def close(g):
    # close resources
    g.par.close()
    g.probe.separate()
    g.probe.close()
    g.db.close()

main = ui.main(setup=setup, enable=enable, execute=execute,
               close=close)
# start event loop
main.start()
```



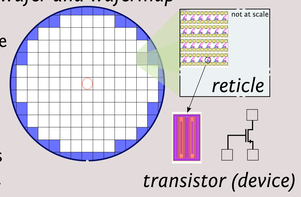


```
DV 1,0,0.0,2.0E-7
DV 3,0,0.0,2.0E-7
WV 2,0,0.0,100,201,1E-7
MM 2,1,2,3
XE
```

```
{ 'run_id':<sequence number>,
  'device_id':<ObjectId>,
  'batch_id':<ObjectId>,
  'abc':'nL22b','bvds':45,
  'idrain':{'type':'ndarray',
            'data': <numpy.array>}
  ...}
```

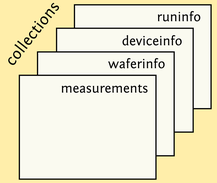
MongoDB

Silicon wafers are processed in batches. Patterning of structures is done through a photo-lithographic process, using multiple masks. A set of masks needed for a fully processed wafer is called a *maskset*. A maskset represents the design of a *reticle* that consists of many components and/or electronic circuits. A reticle design is repeated multiple times on a wafer surface.

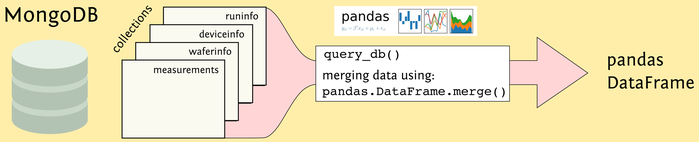


Database

- **MongoDB** used as default database with easy key/value datastore
- Flexible schema for handling variety of measurement data
- Collection *runinfo* keeps information of each measurement session.
- Device names, types, reticle positions, and geometry stored in *deviceinfo* collection.
- Wafer number, maskset name and processing conditions stored in *waferinfo* collection.
- All measurement data together with position on wafer, wafer reference, and device reference are stored in one *measurements* collection.



Querying the database

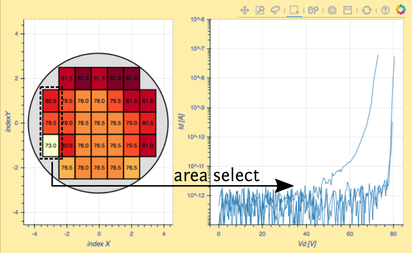


Visualization

- **Bokeh** for interactive plots in **Jupyter** notebook.
- **Matplotlib** for high-quality plots. Ideal for use in reports and for batch plotting.
- **Icelab** provides special plotting functions for plotting in **Bokeh** and **Matplotlib**.

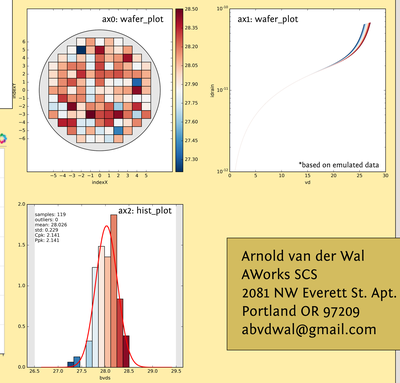
Bokeh

```python
from icelab.plotting.bokehlib import WaferComboPlot
from icelab.db.mongolib import query_db

# query database and import
qry = query_db(database='nmos', host='localhost', masksets='Jalapeno',
               run_id=(1,), devices=('nL22a', 'nL22b'),
               measurements={'bvds', 'vt', 'stress'})
# select data for plotting
data = qry.get('vd', 'idrain', 'bvds',
               conditions={'measurement':'bvds', 'abc':'nL22b'})
# plot data
p = WaferComboPlot(data, 'vd', 'idrain', field='bvds', **wafer_prop,
                   print_values=True, y_axis_type='log',
                   x_label_right='Vd [V]', y_label_right='Id [A]')
show(p)
```



Matplotlib

```python
from icelab.plotting.mpllib import sfigure
from icelab.db.mongolib import query_db

# query database and import
qry = query_db(database='nmos', host='localhost',
               run_id=1, devices='nA08', measurement='bvds')
# select data for plotting
data = qry.get('vd', 'idrain', 'bvds', 'abc')

# plot data
f, ((ax0, ax1), (ax2, _)) = sfigure(ncols=2, nrows=2,
                                    remove_last=True)
ax0.wafer_plot(data, 'bvds', diameter=300, pitchX=23,
               pitchY=20, color_bar=True)
ax1.line_plot(data, 'vd', 'idrain', 'bvds')
ax1.autoscale(enable=True, axis='both', tight=None)
ax1.set_yscale('log'); ax1.set_ylim([1e-12, 1e-10])
ax2.hist_plot(data, 'bvds', cb_colors=True, bins=10,
              lsl=26,5, usl=29,5, skip_outliers=False)
```



Arnold van der Wal
AWorks SCS
2081 NW Everett St. Apt. 405
Portland OR 97209
abvdwal@gmail.com