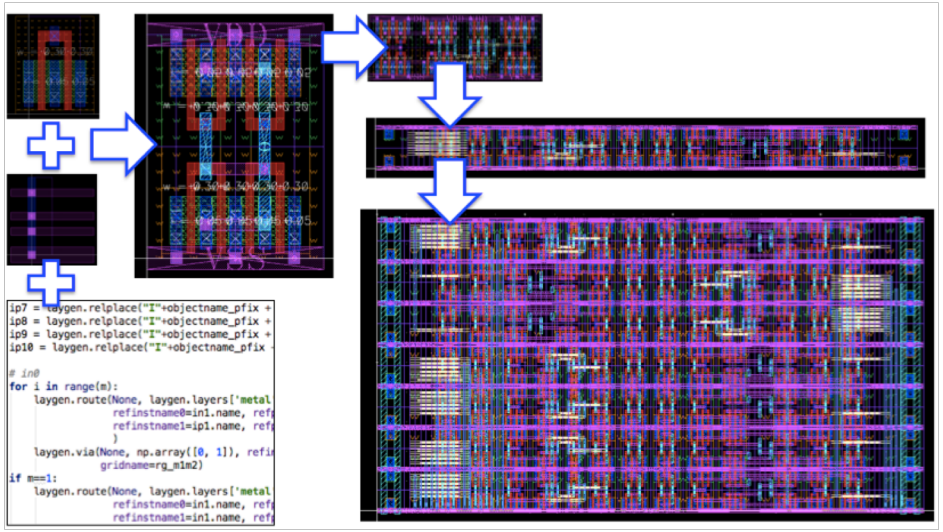
BAG tutorial for schematic generation and simulation

Zhongkai Wang
01/30/2018

About BAG

- BAG is an interface between Python and CAD tools (e.g. Cadence)
- Writing parameterized schematic/layout generators
- Main purpose is for layout

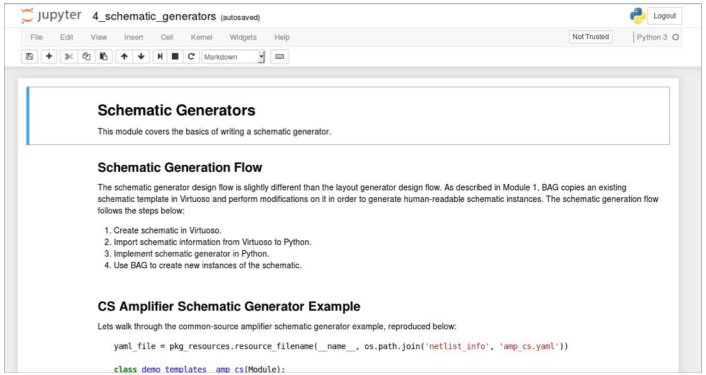


BAG related tools/softwares

- Python
- Git(Hub)
  - Version control software
- Pycharm
  - Python IDE

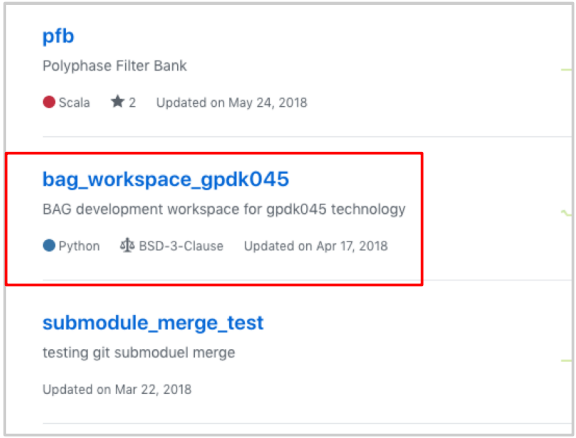
Materials for BAG

- Course page last year
  - http://inst.eecs.berkeley.edu/~ee240b/sp18/homework.html
  - Several tutorials/slides by Eric
- Bootcamp examples
  - Embedded in the workspace
  - Run ./start_tutorial.sh, look at module: schematic_generators



Get workspace repo

- Cadence gpdk45
  - https://github.com/ucb-art
- Workspace
  - git clone https://github.com/ucb-art/bag_workspace_gpdk045.git
- Get submodules
  - Go to the folder just created
  - git submodule update --init --recursive
- tcsh → change to tcsh shell (you could also use bash)
- source .cshrc



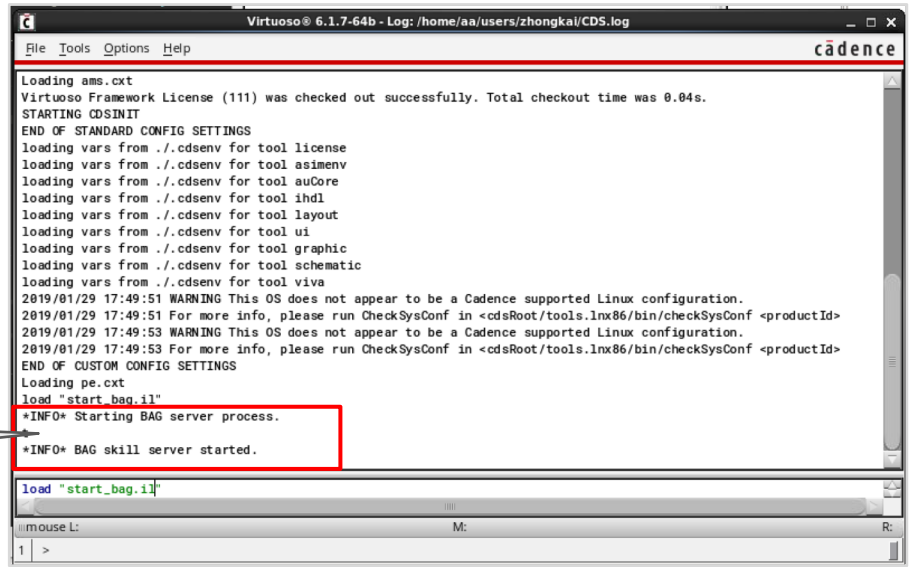
Create your own design folder

- Create design folder -- folder to put all your design in
  - mkdir bag_tutorial
- Create sub-folders
  - mkdir OA scripts

- Purpose of folders
  - OA: Cadence library
  - Scripts: test script
  - Layout: layout scripts
  - BagModules: schematic scripts

Start virtuoso

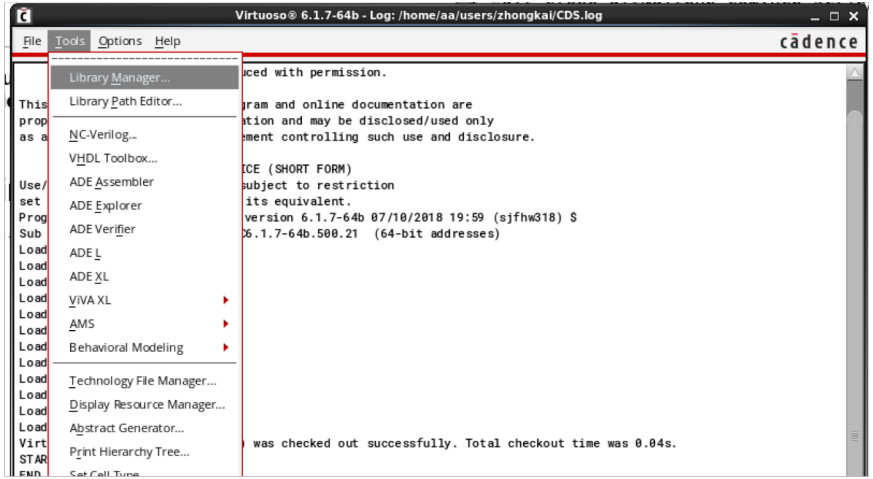
- Go back to your workspace
  - virtuoso&
- Start bag server
  - Run following command in CIW window
  - load “start_bag.il”

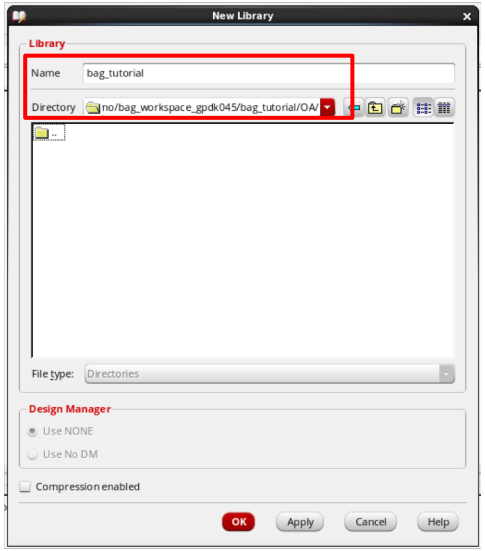


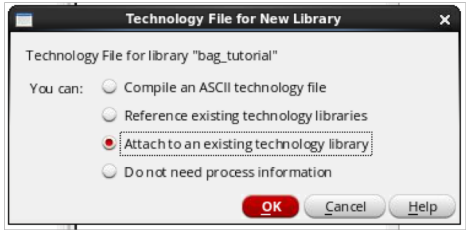
Create template library

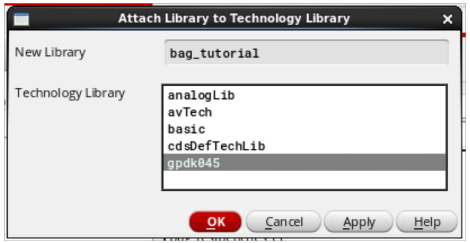
- Start library manager
  - Tools -> library manager

- Create bag_tutorial library at bag_tutorial/OA and attach it to the technology library



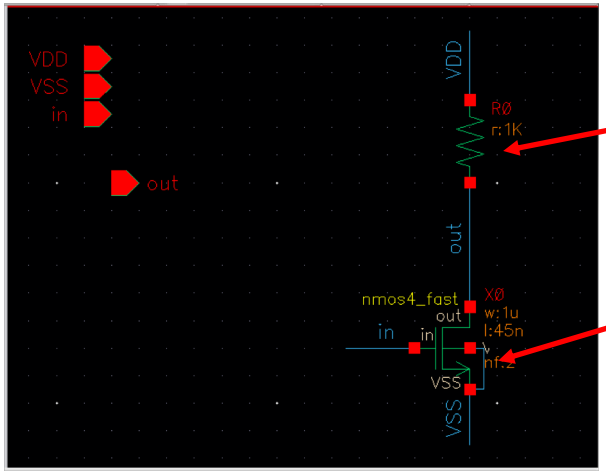






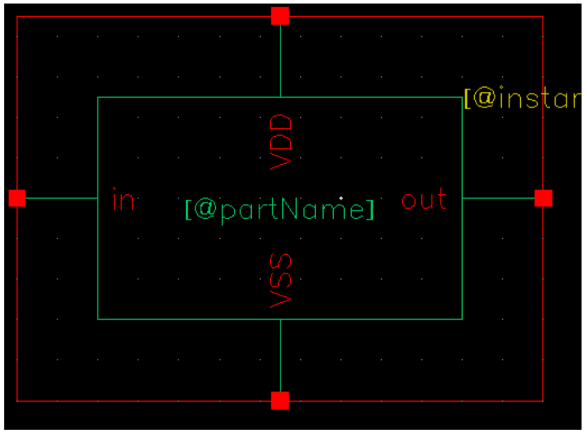
Create CS Amplifier

- File -> New -> Cell View
- Create schematic with name cs_amp
- Create the following schematic and create symbol for it



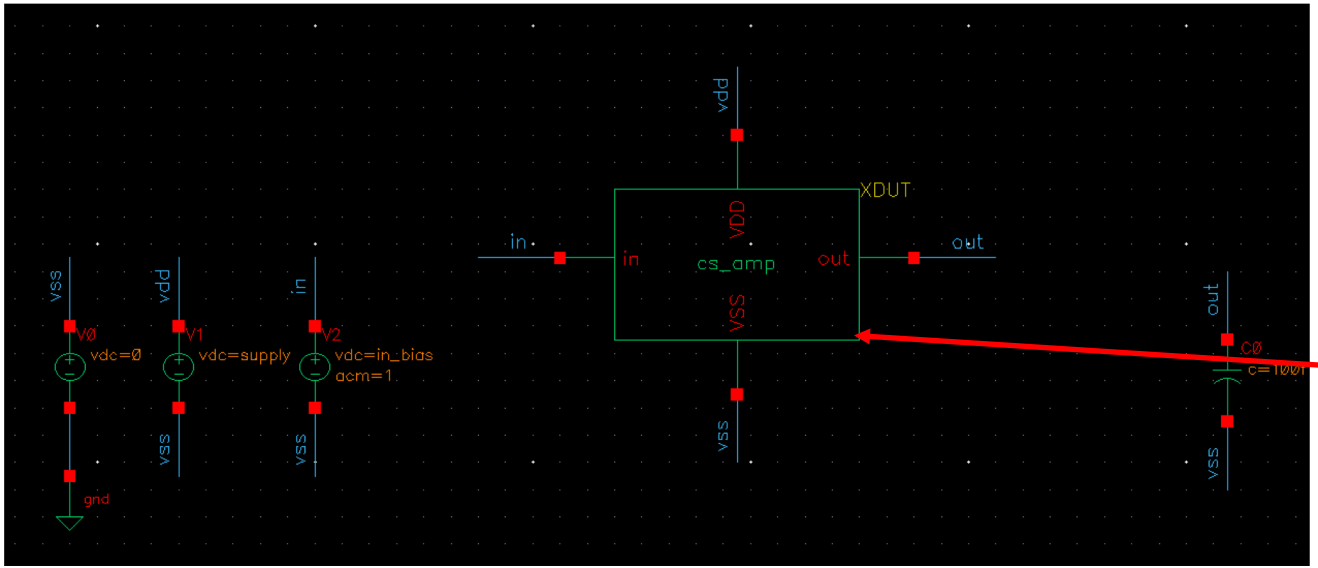
Res:
analogLib library
1K

NMOS:
Bag_prim library
nmos4_fast
nf=nf



Create testbench (cs_amp_tb)

Create testbench for cs_amp , and change instance name to “XDUT”.

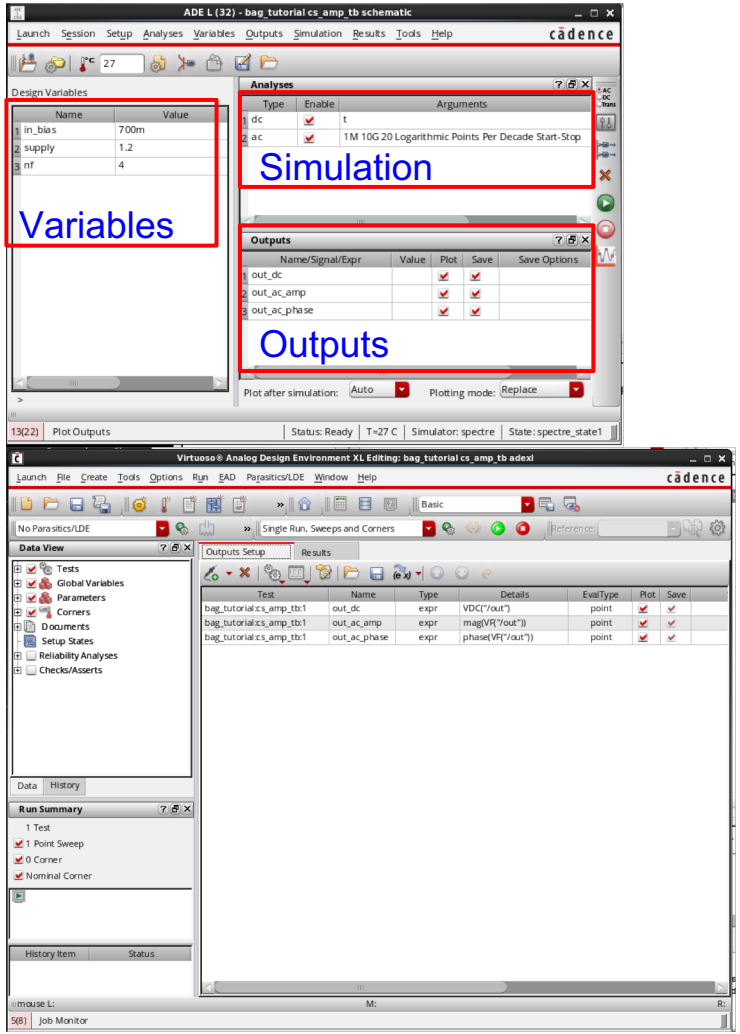


cs_amp we just created

Set simulation

- Create ADEXL
- Set up simulation options
  - Dc and ac simulation
  - Three variables:
    - in_bias, supply, nf
  - Three outputs:
    - Out_dc, out_ac_amp, out_ac_phase
  - Save the state

Two ways to get results:
- *getData* function
- Use *calculator* to save results



Start bag and import library

- Run ./start_bag.sh
- In ipython console
  - import bag
  - prj = bag.BagProject()
  - prj.import_design_library('bag_tutorial')
- Move the generated BagModules into bag_tutorial folder
  - Move all the design script to your design folder
- Change bag_tutorial definition in bag_lib.def

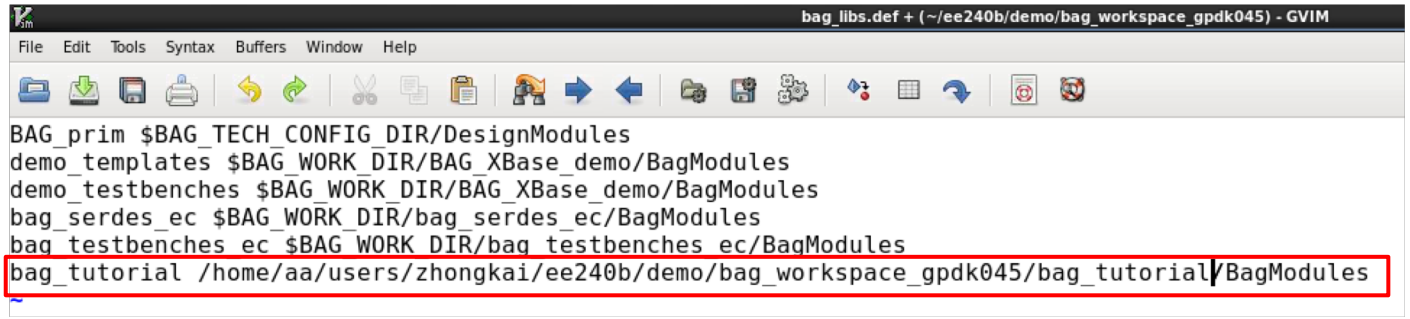
```
[zhongkai@hpse-9 bag_workspace_gpdk045]$ ./start_bag.sh
Python 3.6.3 |Anaconda custom (64-bit)| (default, Oct 13 2017, 12:02:49)
Type 'copyright', 'credits' or 'license' for more information
IPython 6.1.0 -- An enhanced Interactive Python. Type '?' for help.

In [1]: import bag

In [2]: prj = bag.BagProject()
*WARNING*: No TechInfo class defined.  Using a dummy version.

In [3]: prj.import_design_library('bag_tutorial')
```

Create BagModules/bag_tutorial folder



cs_amp schematic design script

- Change bag_tutorial/BagModules/bag_tutorial/cs_amp.py
  - Add your design script to get_params_info and design function

```python
@classmethod
def get_params_info(cls):
    # type: () -> Dict[str, str]
    """Returns a dictionary from parameter names to descriptions.

    Returns
    -------
    param_info : Optional[Dict[str, str]]
        dictionary from parameter names to descriptions.
    """
    return dict(
        mos_l='Channel length',
        mos_w='transistor width',
        mos_nf='number of fingers',
        mos_intent='threshold option',
    )

def design(self, mos_l, mos_w, mos_nf, mos_intent, **kwargs):
    """To be overridden by subclasses to design this module.

    This method should fill in values for all parameters in
    self.parameters.  To design instances of this module, you can
    call their design() method or any other ways you coded.

    To modify schematic structure, call:

    rename_pin()
    delete_instance()
    replace_instance_master()
    reconnect_instance_terminal()
    restore_instance()
    array_instance()
    """
    self.instances['X0'].design(l=mos_l, w=mos_w, nf=mos_nf, intent=mos_intent)
```

Design parameter comments

Give transistor parameters

cs_amp_tb schematic design script

- Change bag_tutorial/BagModules/bag_tutorial/cs_amp_tb.py
  - Add your design script to get_params_info and design function

```python
@classmethod
def get_params_info(cls):
    # type: () -> Dict[str, str]
    """Returns a dictionary from parameter names to descriptions.

    Returns
    -------
    param_info : Optional[Dict[str, str]]
        dictionary from parameter names to descriptions.
    """
    return dict(
        impl_lib='implementation library',
        cell_name='cell name'
    )

def design(self, impl_lib, cell_name):
    """To be overridden by subclasses to design this module.

    This method should fill in values for all parameters in
    self.parameters.  To design instances of this module, you can
    call their design() method or any other ways you coded.

    To modify schematic structure, call:

    rename_pin()
    delete_instance()
    replace_instance_master()
    reconnect_instance_terminal()
    restore_instance()
    array_instance()
    """
    self.replace_instance_master('XDUT', impl_lib, cell_name, static=True)
```

Design parameter comments

Replace with generated schematic

Top level design script (cs_amp_dsn.py)

- Create bag_tutorial/scripts/cs_amp_dsn.py
  (you can also copy it from bCouses)

Generate function (in cs_amp_dsn.py)

- Create a dsn object
- Design with schematic parameters
- Generate schematic to implementation library

```
def generate(prj, temp_lib, impl_lib, cell_name, sch_params):

    print("Generating schematic ...")
    dsn = prj.create_design_module(temp_lib, cell_name)
    dsn.design(**sch_params)
    dsn.implement_design(impl_lib, top_cell_name=cell_name)
```

Simulation function (in cs_amp_dsn.py)

```python
def simulate(prj, temp_lib, impl_lib, tb_name, cell_name, sim_params, show_plot=True):

    # generate tb schematic
    print("Generating testbench ...")
    dsn = prj.create_design_module(temp_lib, tb_name)
    dsn.design(impl_lib=impl_lib, cell_name=cell_name)
    dsn.implement_design(impl_lib, top_cell_name=tb_name)

    # configure tb
    tb = prj.configure_testbench(tb_lib=impl_lib, tb_cell=tb_name)
    tb.set_parameter('supply', sim_params[0])
    tb.set_parameter('in_bias', sim_params[1])
    tb.set_parameter('nf', sim_params[2])
    tb.update_testbench()

    # rum simulation
    tb.run_simulation()
    print(tb.save_dir)
    results = load_sim_results(tb.save_dir)

    out_dc = results['out_dc']
    freq = results['freq']
    out_ac_amp = results['out_ac_amp']
    out_ac_phase = results['out_ac_phase']

    if show_plot:
        print(f"Output dc is {out_dc}")
        plt.figure()
        plt.subplot(2, 1, 1)
        plt.semilogx(freq, out_ac_amp)
        plt.subplot(2, 1, 2)
        plt.semilogx(freq, out_ac_phase)
        plt.show(block=False)

    return out_dc, out_ac_amp, out_ac_phase
```

Similar with cs_amp

Configure tesbench and setup parameters

Run simulation and get results

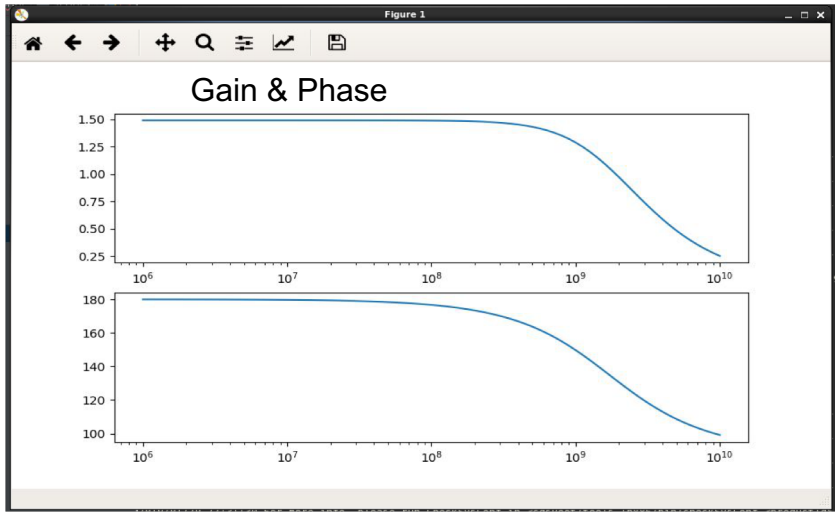
Plot output

Run script

- Change cs_amp_dsn.py

```
generate(bprj, temp_lib, impl_lib, cell_name, sch_params)
simulate(bprj, temp_lib, impl_lib, tb_name, cell_name, sim_params=[1.2, 0.7, 4])
#design1(bprj, temp_lib, impl_lib, tb_name, cell_name, sch_params)
#design2(bprj, temp_lib, impl_lib, tb_name, cell_name, sch_params)
```

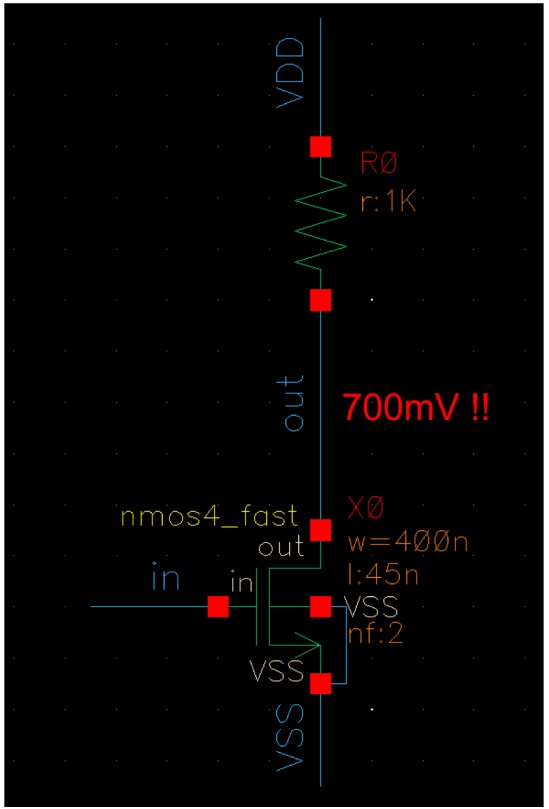
- run –i bag_tutorial/scripts/cs_amp_dsn.py



Design circuit with BAG

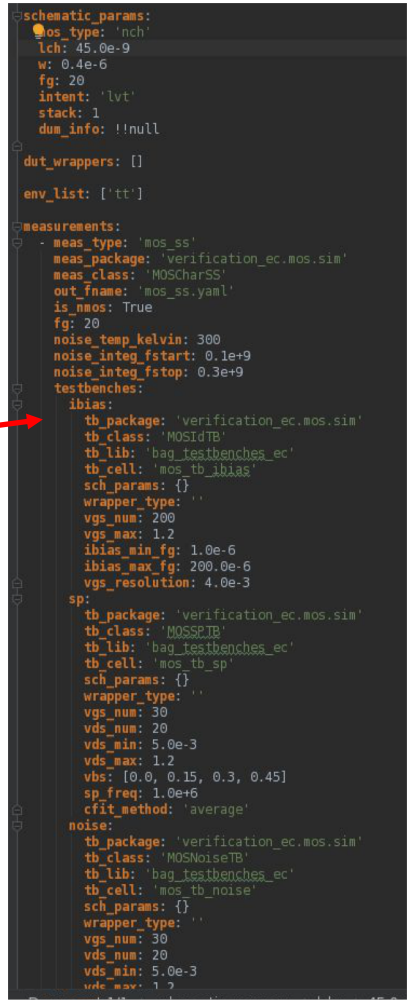
More fun stuff

- Object: set CS amplifier output voltage around 700mV by changing nf (Assume input bias is also 700mV)
  - Hand calculation?
  - Look-up table or database (design 1)
  - Iteration or Sweep. (design 2)



Generate MOS transistor look-up table

- run –i scripts_char/nch_char_90n.py
- Workspace is embedded with a mos sweep script
  - Using MeasurementManager/TestbenchManager to sweep all the bias
  - script_char/nch_char_90n.py
  - parameter/spec file in specs_mos_char/nch_w0d4_45n.yaml (a template file)
  - Save result to database
- Works pretty good with other devices
  - MOMCap, Inductor, CapDAC switches …
  - Not favorable for larger circuit
- More details in Eric’s discussion slides last year!!



```yaml
schematic_params:
  mos_type: 'nch'
  lch: 45.0e-9
  w: 0.4e-6
  fg: 20
  intent: 'lvt'
  stack: 1
  dum_info: !!null

dut_wrappers: []

env_list: ['tt']

measurements:
  - meas_type: 'mos_ss'
    meas_package: 'verification_ec.mos.sim'
    meas_class: 'MOSCharSS'
    out_fname: 'mos_ss.yaml'
    is_nmos: True
    fg: 20
    noise_temp_kelvin: 300
    noise_integ_fstart: 0.1e+9
    noise_integ_fstop: 0.3e+9
    testbenches:
      ibias:
        tb_package: 'verification_ec.mos.sim'
        tb_class: 'MOSIdTB'
        tb_lib: 'bag_testbenches_ec'
        tb_cell: 'mos_tb_ibias'
        sch_params: {}
        wrapper_type: ''
        vgs_num: 200
        vgs_max: 1.2
        ibias_min_fg: 1.0e-6
        ibias_max_fg: 200.0e-6
        vgs_resolution: 4.0e-3
      sp:
        tb_package: 'verification_ec.mos.sim'
        tb_class: 'MOSSPTB'
        tb_lib: 'bag_testbenches_ec'
        tb_cell: 'mos_tb_sp'
        sch_params: {}
        wrapper_type: ''
        vgs_num: 30
        vds_num: 20
        vds_min: 5.0e-3
        vds_max: 1.2
        vbs: [0.0, 0.15, 0.3, 0.45]
        sp_freq: 1.0e+6
        cfit_method: 'average'
      noise:
        tb_package: 'verification_ec.mos.sim'
        tb_class: 'MOSNoiseTB'
        tb_lib: 'bag_testbenches_ec'
        tb_cell: 'mos_tb_noise'
        sch_params: {}
        wrapper_type: ''
        vgs_num: 30
        vds_num: 20
        vds_min: 5.0e-3
```

Query function

- Search for transistor results with given bias
  - scripts_char/mos_query.py

```
interp_method = 'spline'
sim_env = 'tt'
nmos_spec = 'specs_mos_char/nch_w0d5_90n.yaml'

intent = 'lvt'

db = get_db(nmos_spec, intent, interp_method=interp_method, sim_env=sim_env)

pprint.pprint(db.query(vbs=0, vds=0.45, vgs=0.45))
pprint.pprint(db.query(vbs=0, vds=0.5, vgs=0.5))
pprint.pprint(db.query(vbs=0.15, vds=0.55, vgs=0.55))
pprint.pprint(db.query(vbs=0.15, vds=0.55, vstar=0.2))
```

Read database and run interpolation

Query function with given bias

- gm/Id and V* design methodology

Design with look-up table

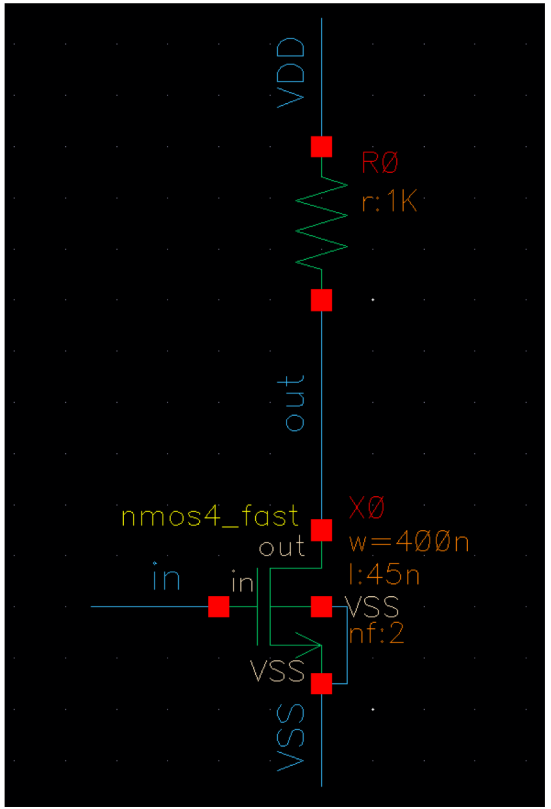
- Change cs_amp_dsn.py

```
#generate(bprj, temp_lib, impl_lib, cell_name, sch_params)
#simulate(bprj, temp_lib, impl_lib, tb_name, cell_name, sim_params=[1.2, 0.7, 4])
design1(bprj, temp_lib, impl_lib, tb_name, cell_name, sch_params)
#design2(bprj, temp_lib, impl_lib, tb_name, cell_name, sch_params)
```

- run –i bag_tutorial/scripts/cs_amp_dsn.py
  - Could get current Idc from output voltage and resistor value
  - From query function, get current Id of 1 finger transistor
  - nf = Idc/Id

```
# get mos bias information
mos_info = db.query(vbs=0, vds=0.7, vgs=0.7)
# print mos bias information
pprint.pprint(mos_info)
print("\n")

# calculate
head_room = 0.5
res = 1e3
nf = round(head_room / res / mos_info['ibias'])
sch_params['nf'] = nf
print(f"MOS finger number is {nf}")
```



Iteration or Sweep

- Change cs_amp_dsn.py

```
#generate(bprj, temp_lib, impl_lib, cell_name, sch_params)
#simulate(bprj, temp_lib, impl_lib, tb_name, cell_name, sim_params=[1.2, 0.7, 4])
#design1(bprj, temp_lib, impl_lib, tb_name, cell_name, sch_params)
design2(bprj, temp_lib, impl_lib, tb_name, cell_name, sch_params)
```

- run –i bag_tutorial/scripts/cs_amp_dsn.py
  - Start from 1 finger and add finger number until we get the voltage we need
  - Usually use it for *larger circuit*

```
out_dc = 1.2
nf = 1
nf_list = []
out_dc_list = []
while out_dc > 0.7:
    out_dc, out_ac_amp, out_ac_phase = \
        simulate(prj, temp_lib, impl_lib, tb_name, cell_name, sim_params=[1.2, 0.7, nf], show_plot=False)

    # add to lists
    nf_list.append(nf)
    out_dc_list.append(out_dc)
    print(f"Output dc is {out_dc} @ nf is {nf}")

    # change nf
    nf = nf+1
```

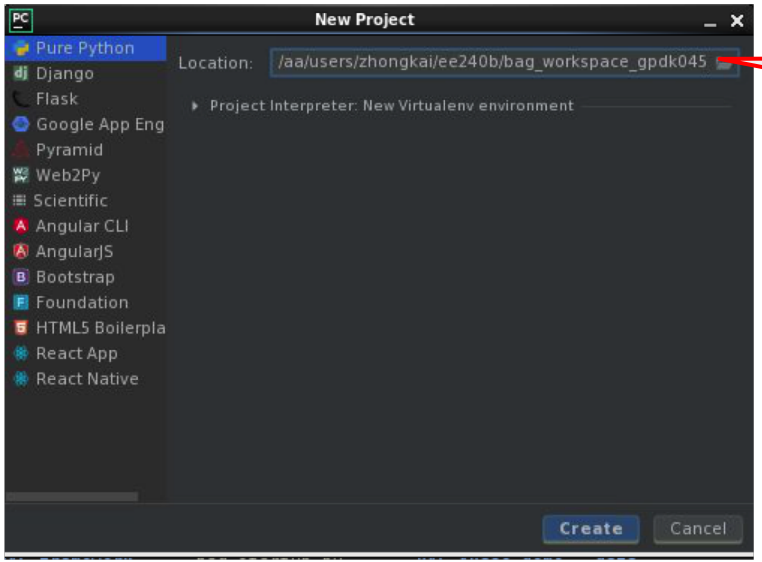
Thanks!
Have fun with BAG!

Zhongkai Wang (zhongkai@berkeley.edu)
Ayan Biswas (ayan_biswas@berkeley.edu)
Eric Chang (pkerichang@berkeley.edu)

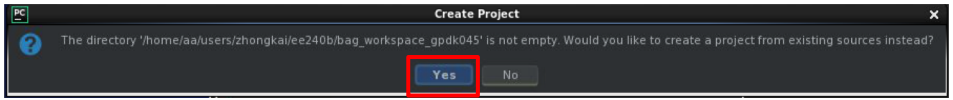
Backup slides

- Pycharm setup

Step1: Create pycharm project

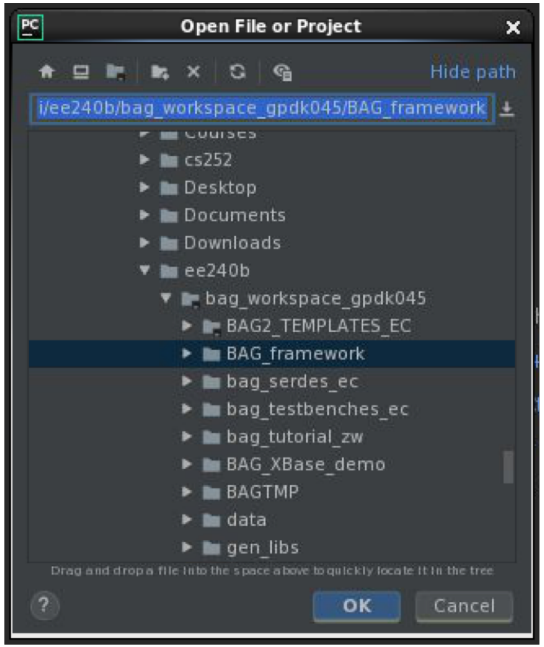


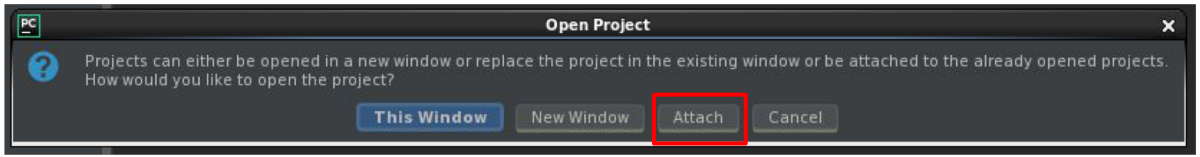
Your BAG workspace from git clone ..



Add python folders to project

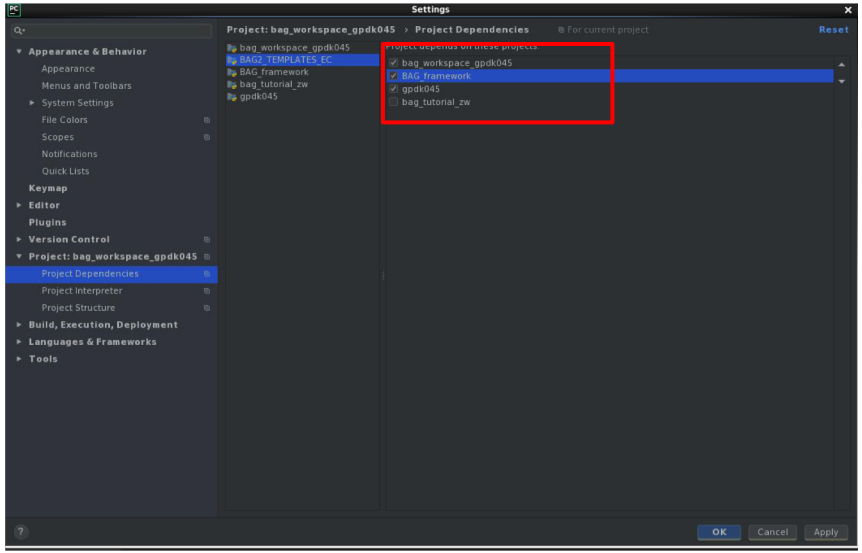
- Open the following folders and add to project
  - BAG2_TEMPLATES_EC -- BAG templates
  - BAG_framework -- BAG framework
  - gpdk045 -- technology folder
  - Other design folders if you are going to use them
    - E.g. bag_serdes_ec
    - Include your own design folder!





Set project dependency

- Set depends on the folder is use scripts from other folders
  - BAG2_TEMPLATES_EC
    - Bag_work_space_gpdk045, BAG_framework, gpdk045
  - BAG framework
    - Bag_work_space_gpdk045, BAG2_TEMPLATES_EC, gpdk045
  - Gpdk045
    - Bag_work_space_gpdk045, BAG_framework, BAG2_TEMPLATES_EC
  - Bag_tutorial_zk (My design folder)
    - Bag_work_space_gpdk045, BAG_framework, BAG2_TEMPLATES_EC, gpdk045



Project Interpreter

- Change to available python verision (Python 3.6) for each folders