BPG Documentation

Release 0.8.5

Pavan Bhargava

Dec 11, 2019

CONTENTS

CHAPTER

ONE

GETTING STARTED

This chapter contains a step-by-step guide to installing BPG and producing your first gds and lsf file. You will also learn the basic structure of a BPG generator, and run a few examples that we have provided. This will utilize the built in example PDK.

1.1 Installing BPG

1.1.1 Prerequisites

We highly recommend you use an Anaconda environment with a Python version greater than 3.6. BPG generally will not function with Python versions less than 3.6, and requires packages with C/C++ dependencies that are most easily installed using Anaconda.

Once Anaconda is set up, please run the following commands to install packages with C/C++ dependencies:

```
conda install numpy
conda install rtree
conda install shapely
```

BPG generally generates output layouts in the GDSII format. To view these layouts, we recommend you install and use the free open-source software package, Klayout.

1.1.2 Installation

WARNING: Installation instructions are currently in flux.

We highly recommend you use an Anaconda environment with a Python version greater than 3.6. BPG generally will not function with Python versions less than 3.6, and requires packages with C/C++ dependencies that are most easily installed using Anaconda.

Once Anaconda is set up, please run the following commands to install packages with C/C++ dependencies:

```
conda install numpy
conda install rtree
conda install shapely
```

Then clone and install BAG with in any folder with:

```
git clone git@github.com:ucb-art/BAG_Framework.git
cd BAG_Framework
pip install .
```

Finally clone and install BPG in any folder with:

```
git clone git@github.com:BerkeleyPhotonicsGenerator/BPG.git
cd BPG
pip install .
```

BPG generally generates output layouts in the GDSII format. To view these layouts, we recommend you install and use the free open-source software package, Klayout.

1.1.3 Quick Workspace Setup

BPG requires specific environmental variables and a PDK for your technology in order to run. We provide an example workspace and PDK for you to quickly get started. To set up your workspace run the following:

1. `bpg setup_workspace`. This copies over the example technology and environment variable setup file
2. `source sourceme.sh`. This will setup all of the necessary environment variables
3. Now you can execute any BPG based python script by running `python <INSERT PATH TO PYTHON FILE HERE>`

To get some example generators and learn more about how BPG works, please check out the tutorial section in getting started.

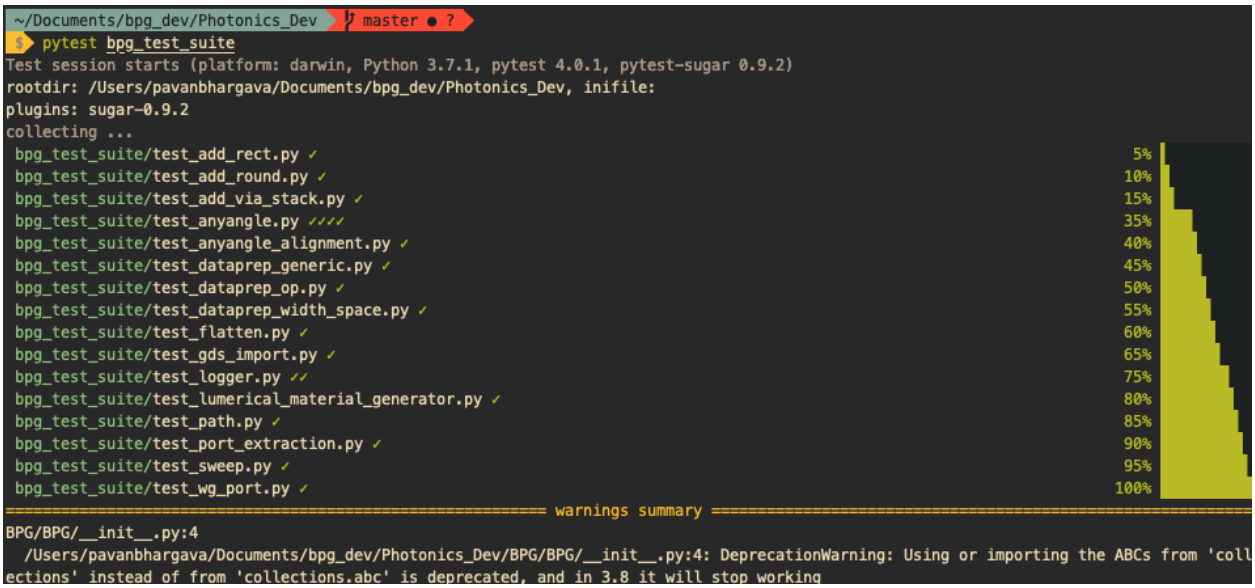
1.1.4 Testing BPG

WARNING: This under certain conditions, this may copy the wrong tests.

You may wish to run BPG's test suite to make sure that it functions properly on your system. To do so, first make sure that you have the example tech setup by running the instructions in the `Quick Workspace Setup` section, and that you. have `pytest` installed. Then run

```
bpg setup_test
pytest bpg_test_suite
```

If you do not have pytest installed, do so by running `pip install pytest`, then re-run `pytest bpg_test_suite`. After running the test suite, you should see something similar to this, with a few warnings and messages below:


~/Documents/bpg_dev/Photonics_Dev
master ● ?
$ pytest bpg_test_suite
Test session starts (platform: darwin, Python 3.7.1, pytest 4.0.1, pytest-sugar 0.9.2)
rootdir: /Users/pavanbhargava/Documents/bpg_dev/Photonics_Dev, inifile:
plugins: sugar-0.9.2
collecting ...
bpg_test_suite/test_add_rect.py ✓ 5%
bpg_test_suite/test_add_round.py ✓ 10%
bpg_test_suite/test_add_via_stack.py ✓ 15%
bpg_test_suite/test_anyangle.py ✓✓✓✓ 35%
bpg_test_suite/test_anyangle_alignment.py ✓ 40%
bpg_test_suite/test_dataprep_generic.py ✓ 45%
bpg_test_suite/test_dataprep_op.py ✓ 50%
bpg_test_suite/test_dataprep_width_space.py ✓ 55%
bpg_test_suite/test_flatten.py ✓ 60%
bpg_test_suite/test_gds_import.py ✓ 65%
bpg_test_suite/test_logger.py ✓✓ 75%
bpg_test_suite/test_lumerical_material_generator.py ✓ 80%
bpg_test_suite/test_path.py ✓ 85%
bpg_test_suite/test_port_extraction.py ✓ 90%
bpg_test_suite/test_sweep.py ✓ 95%
bpg_test_suite/test_wg_port.py ✓ 100%
warnings summary
BPG/BPG/__init__.py:4
/Users/pavanbhargava/Documents/bpg_dev/Photonics_Dev/BPG/BPG/__init__.py:4: DeprecationWarning: Using or importing the ABCs from 'coll
ections' instead of from 'collections.abc' is deprecated, and in 3.8 it will stop working

CHAPTER

TWO

CONFIGURATION FILES

This chapter contains a information on special options to configure the operation of BPG and BAG.

2.1 BPG Configuration

Placeholder text

2.1.1 Anchor1

2.1.2 Anchor2

CHAPTER

THREE

DATAPREP

This chapter contains an in-depth explanation of how dataprep works, and how to customize the dataprep routine to support photonic layout compilation for your specific PDK

3.1 Dataprep

Placeholder text

CHAPTER

FOUR

BPG

4.1 BPG package

4.1.1 Subpackages

BPG.compiler package

Submodules

BPG.compiler.dataprep_gdspy module

BPG.compiler.dataprep_shapely module

BPG.compiler.dataprep_skill module

BPG.compiler.manh_shapely module

BPG.compiler.point_operations module

BPG.compiler.poly_simplify module

Module contents

BPG.gds package

Submodules

BPG.gds.core module

BPG.gds.io module

Module contents

BPG.lumerical package

Submodules

BPG.lumerical.code_generator module

BPG.lumerical.core module

BPG.lumerical.design_manager module

BPG.lumerical.objects module

BPG.lumerical.simulation module

BPG.lumerical.testbench module

Module contents

BPG.oa package

Submodules

BPG.oa.core module

Module contents

BPG.skill package

Submodules

BPG.skill.photonic_skill module

Module contents

BPG.workspace_setup package

Submodules

BPG.workspace_setup.setup module

BPG.workspace_setup.setup_submodules module

Module contents

4.1.2 Submodules

4.1.3 BPG.abstract_plugin module

4.1.4 BPG.bpg_custom_types module

4.1.5 BPG.content_list module

4.1.6 BPG.db module

4.1.7 BPG.flow module

4.1.8 BPG.geometry module

4.1.9 BPG.layout_manager module

4.1.10 BPG.logger module

class BPG.logger.**DontRepeatFilter**
Bases: object

add_key(*key*)

clear_history()

filter(*record*)

BPG.logger.**setup_logger**(*log_path: str*, *log_filename: str = 'bpg.log'*) → None
Configures the root logger so that all other loggers in BPG inherit from its properties.

Parameters

- **log_path** (*str*) – The path to save the log files.
- **log_filename** (*str*) – The name of the primary output log file.

4.1.11 BPG.objects module

4.1.12 BPG.photonic_core module

4.1.13 BPG.port module

4.1.14 BPG.template module

4.1.15 Module contents

PYTHON MODULE INDEX

b

INDEX