Note on GDS File

Created on 2024-2-7
Author Di Yu

Introduction

GDSII is a database file format (filename extension `.gds`) which is used for storing and transferring design layout of integrated circuits. A GDS file contains a series of planar geometries, each of them is assigned with a layer index. These geometries define the shape of devices, and the corresponding layer indices specify the order in which these devices should be fabricated. The GDS format is widely used in nanofabrication, especially for optical lithography, where one needs to upload a gds file as input design file to a lithography system to obtain desired patterns on wafers. Therefore, it's important to learn about how to create, edit, and view a gds file, if one wants to engage in nanofabrication. This note will cover these topics, with a focus on the applications of gds files in integrated photonics.

Table of Contents

Edit and View `gds` File

There are several Python modules that support creating and editing GDS files, such as `gdspy` and `PICwriter`. The most commonly used module in our group is `gdspy`, which allows for the direct creation of common building blocks for photonic integrated circuits. To avoid version conflicts with other Python modules, it's recommended to install `gdspy` in an independent `Anaconda` environment. A tutorial on how to use `gdspy` can be found here.

GDS files are binary files that can only be interpreted by certain software. One widely used GDS file viewer is `KLayout`. It's important to note that `KLayout` can also be used to edit GDS files. In fact, `KLayout` supports two quite useful operations on GDS files: inter-layer boolean operations and layer property editing, which I will introduce below.

Boolean Operation with `lydrc` Script

`KLayout` supports boolean operations between different layers in a GDS file. The computation cost of boolean operations in `KLayout` is much lower than in `gdspy`, making it a favorable option. There are multiple ways to perform boolean operations in `KLayout`, and the easiest one is to write a `drc` (or `lydrc`) script in the `Macros - Macro Development - DRC` window.

The basic grammar of `lydrc` scripts is reasonably simple. For example, the following script loads all layers from `input.gds` and calculates the regions that are contained in the 2nd layer but not in the 1st or 3rd layer. The resulting pattern will be saved as a new GDS file named `output.gds`.

```
source("input.gds")
target("output.gds")

layer_output = (input(2, 0) - input(1, 0)) - input(3, 0)
layer_output.merged.output(100)
```

Save and Load Layer Properties with `lyp` Script

You may notice that in a GDS file, each layer is assigned a number index. This layer indexing might not be ideal since it doesn't convey the function or material associated with these layers. In the initial user interface that displays the GDS pattern, you can rename these layers by right-clicking on the layer indices, choosing `rename`, and entering a new layer name.

It's important to note that the newly defined layer indexing will be lost unless you save the layer properties properly. To save the layer properties, go to `File - Save Layer Properties`. This will create a `lyp` script that contains the customized layer indexing. The customized layer indexing can be loaded into `KLayout` by simply dragging the `lyp` script into the `KLayout` window.