First code for all masks

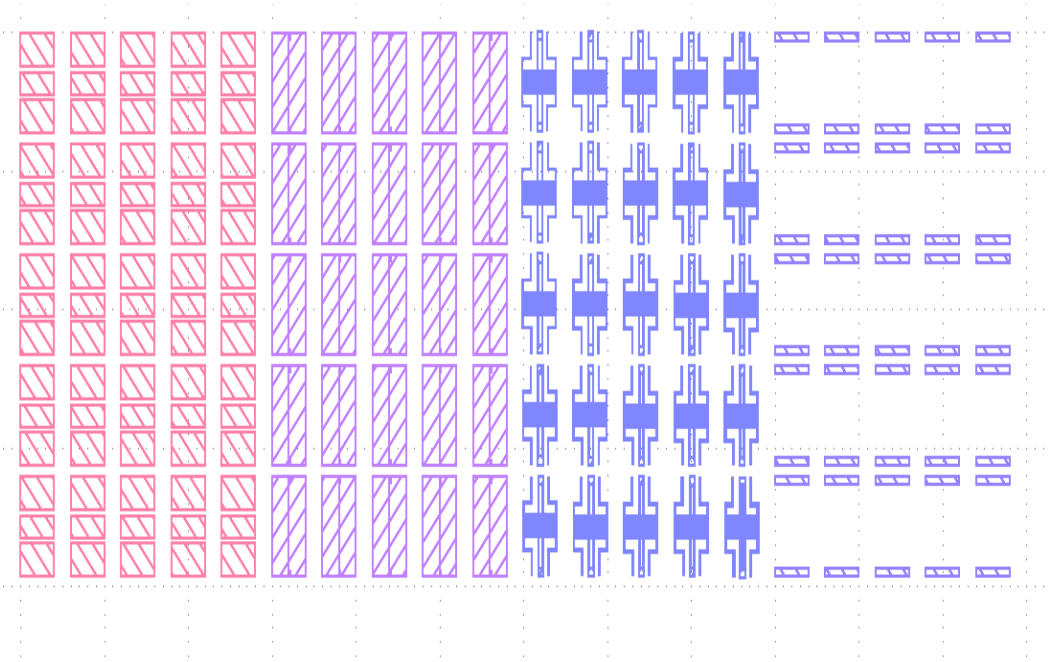
```
import phidl.geometry as pg
from phidl import Device, Layer, Path, CrossSection
from phidl import quickplot as qp
import numpy as np
layer_bottommetal = Layer(1, name='bottommetal')
layer_dielectric = Layer(2, name='dielectric')
layer_dielectric_2 = Layer(3, name='dielectric_2')
layer_topmetal = Layer(4, name='topmetal')
switch = Device()
switch_1 = Device()
columns_1= 5
rows_1=5
for y in range(columns_1):
   for z in range(rows_1):
      i= y+z
      # top layer
      Rectangle5 = switch.add_ref(pg.rectangle(size=(4.5, 0.1*i), layer=layer_topmetal))
      Rectangle6 = switch.add_ref(pg.rectangle(size=(0.1*i,5), layer=layer_topmetal))
      Rectangle7 = switch.add_ref(pg.rectangle(size=(9+(0.1*i), 0.1*i), layer=layer_topmetal))
      Rectangle5.move([9.5+(z*40)-((5-i)*0.1),300+(y*30)-((5-i)*0.1)])
      Rectangle6.move([9.5+(z*40)-((5-i)*0.1),300+(y*30)-((5-i)*0.1)])
      Rectangle7.move([0.5+(z*40)-((5-i)*0.1),304.5+(y*30)])
      Rectangle8 = switch.add_ref(pg.rectangle(size=(4.5, 0.1*i), layer=layer_topmetal))
      Rectangle9 = switch.add_ref(pg.rectangle(size=(0.1*i,5), layer=layer_topmetal))
      Rectangle10 = switch.add_ref(pg.rectangle(size=(9+(0.1*i), 0.1*i), layer=layer_topmetal))
      Rectangle8.move([9.5+(z*40)-((5-i)*0.1),319.5+(y*30)])
      Rectangle9.move([9.5+(z*40)-((5-i)*0.1),315+(y*30)-((5-i)*0.1)])
```

```
Rectangle10.move([0.5+(z*40)-((5-i)*0.1),315+(y*30)-((5-i)*0.1)])
Rectangle12 = switch.add_ref(pg.rectangle(size=(4.5+(0.1*i), 0.1*i), layer=layer_topmetal))
Rectangle13 = switch.add_ref(pg.rectangle(size=(0.1*i,5+(0.1*i)), layer=layer_topmetal))
Rectangle14 = switch.add_ref(pg.rectangle(size=(9, 0.1*i), layer=layer_topmetal))
Rectangle12.move([22+(z*40)-((5-i)*0.1),300+(y*30)-((5-i)*0.1)])
Rectangle13.move([26.5+(z*40)-((5-i)*0.1),300+(y*30)-((5-i)*0.1)])
Rectangle14.move([26.5+(z*40)-((5-i)*0.1),305+(y*30)-((5-i)*0.1)])
Rectangle15 = switch.add_ref(pg.rectangle(size=(4.5+(0.1*i), 0.1*i), layer=layer_topmetal))
Rectangle16 = switch.add_ref(pg.rectangle(size=(0.1*i,5), layer=layer_topmetal))
Rectangle17 = switch.add_ref(pg.rectangle(size=(9, 0.1*i), layer=layer_topmetal))
Rectangle15.move([22+(z*40)-((5-i)*0.1),319.5+(y*30)])
Rectangle16.move([26.5+(z*40)-((5-i)*0.1),315+(y*30)-((5-i)*0.1)])
Rectangle17.move([26.5+(z*40)-((5-i)*0.1),315+(y*30)-((5-i)*0.1)])
Rectangle18 = switch.add_ref(pg.rectangle(size=(11.5, 0.1*i), layer=layer_topmetal))
Rectangle18.move([22+(z*40)-((5-i)*0.1),311.5+(y*30)])
Rectangle19 = switch.add_ref(pg.rectangle(size=(11.5, 0.1*i), layer=layer_topmetal))
Rectangle19.move([22+(z*40)-((5-i)*0.1),309+(y*30)-((5-i)*0.1)])
Rectangle20 = switch.add_ref(pg.rectangle(size=(11.5, 0.1*i), layer=layer_topmetal))
Rectangle20.move([2.5+(z*40)-((5-i)*0.1),311.5+(y*30)])
Rectangle21 = switch.add_ref(pg.rectangle(size=(11.5, 0.1*i), layer=layer_topmetal))
Rectangle21.move([2.5+(z*40)-((5-i)*0.1),309+(y*30)-((5-i)*0.1)])
Rectangle22 = switch.add_ref(pg.rectangle(size=(3, 3), layer=layer_topmetal))
Rectangle22.move([33+(z*40)-((5-i)*0.1),309+(y*30)-((5-i)*0.1)])
Rectangle23 = switch.add_ref(pg.rectangle(size=(3, 3), layer=layer_topmetal))
Rectangle23.move([0+(z*40)-((5-i)*0.1),309+(y*30)-((5-i)*0.1)])
D = Device()
D1 =Device()
D2 = Device()
Rectangle4_size = (8,20)
```

```
    small_square_size = 0.5  # Size of the smaller squares
    rows = 16  # Number of rows of squares
    columns = 8  # Number of columns of squares
    line_distance = 1
    # Calculate the total width and height occupied by the smaller squares
    total_width = columns * small_square_size
    total_height = rows * small_square_size
    # Calculate the spacing between the smaller squares
    spacing_x = (Rectangle4_size[0] - total_width) / (columns + 1.2)
    spacing_y = (Rectangle4_size[1] - total_height) / (rows + 0.7)
    # Draw the larger square
    Rectangle4 = D1 << pg.rectangle(size=Rectangle4_size, layer=layer_topmetal)
    # Calculate the position for each smaller square and draw them
    for x in range(rows):
      for j in range(columns):
        x_offset = (j + 1.5) * spacing_x + j * small_square_size
        y_offset = (x + 1.2) * spacing_y + x * small_square_size
        inner_square = D << pg.rectangle(size=(small_square_size, small_square_size),
layer=layer_topmetal)
        inner_square.move((x_offset, y_offset))
    # Draw the larger square
    Rectangle4 =D1<< pg.rectangle(size=Rectangle4_size, layer=layer_topmetal)
    X6 =pg.xor_diff(B=D,A= Rectangle4, precision = 1e-6)
    D2.add_ref(X6)
    # Calculate the dimensions for the lines
    inner_square_size = (Rectangle4_size[0] - 2 * line_distance, Rectangle4_size[1] - 2 *
line_distance)
    switch.add_ref(D2)
    X6.move([14+(z*40)-((5-i)*0.1),300+(y*30)-((5-i)*0.1)])
```

```
#bottom layer
Rectangle1 = switch.add_ref(pg.rectangle(size=(8, 20), layer=layer_bottommetal))
Rectangle2 = switch.add_ref(pg.rectangle(size=(12, 20), layer=layer_bottommetal))
Rectangle3 = switch.add_ref(pg.rectangle(size=(12, 20), layer=layer_bottommetal))
Rectangle1.move([14+(z*40),0+(y*30)])
Rectangle2.move([24+(z*40),0+(y*30)])
Rectangle3.move([0+(z*40),0+(y*30)])
#dielectric layer
Rectangle24 = pg.rectangle(size=(36, 20),layer=layer_dielectric)
Rectangle24. move([0+(z*40),150+(y*30)])
Rectangle25 = pg.rectangle(size=(1, 1),layer=layer_dielectric)
Rectangle25.move([34+(z*40),160+(y*30)])
Rectangle26 = pg.rectangle(size=(1, 1),layer=layer_dielectric)
Rectangle26.move([1+(z*40),160+(y*30)])
X = pg.xor_diff(A = Rectangle24, B = Rectangle25, precision = 1e-6)
X1 = pg.xor_diff(A = Rectangle24, B = Rectangle26, precision = 1e-6)
X2 =pg.xor_diff(A=X,B=X1, precision = 1e-6)
X3 = pg.xor_diff(A=Rectangle24,B=X2, precision = 1e-6)
switch.add_ref(X3)
#final etching
Rectangle27 = pg.rectangle(size=(3, 20),layer=layer_dielectric_2)
Rectangle27.move([0+(z*40),450+(y*30)])
Rectangle28 = pg.rectangle(size=(3, 20),layer=layer_dielectric_2)
Rectangle28.move([33+(z*40),450+(y*30)])
Rectangle29 = pg.rectangle(size=(36, 20),layer=layer_dielectric_2)
Rectangle29.move([0+(z*40),450+(y*30)])
X4 = pg.xor_diff(A = Rectangle29, B = Rectangle27, precision = 1e-6)
```

```
    X5 = pg.xor_diff(A = Rectangle29, B = Rectangle28, precision = 1e-6)
    X6 =pg.xor_diff(A=X4,B=X5, precision = 1e-6)
    switch.add_ref(X6)
switch.write_gds('7.gds')
qp(switc)
```



Top view

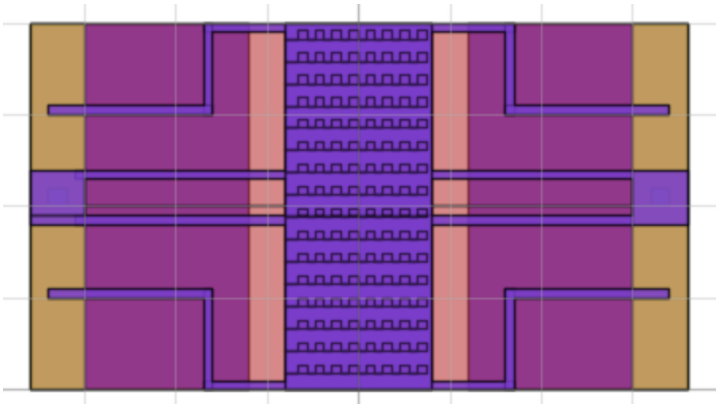
```
import phidl.geometry as pg
from phidl import Device, Layer, Path, CrossSection
from phidl import quickplot as qp
import numpy as np
layer_bottommetal = Layer(1, name='bottommetal')
layer_dielectric = Layer(2, name='dielectric')
layer_dielectric_2 = Layer(3, name='dielectric_2')
layer_topmetal = Layer(4, name='topmetal')
```

```
switch = Device()
switch_1 = Device()
Rectangle1 = switch.add_ref(pg.rectangle(size=(8, 20), layer=layer_bottommetal))
Rectangle2 = switch.add_ref(pg.rectangle(size=(12, 20), layer=layer_bottommetal))
Rectangle3 = switch.add_ref(pg.rectangle(size=(12, 20), layer=layer_bottommetal))
Rectangle1.move([-4, 0])
Rectangle2.move([6,0])
Rectangle3.move([-18,0])
Rectangle4 = switch.add_ref(pg.rectangle(size=(8, 20), layer=layer_topmetal))
Rectangle4.move([-4,0])
Rectangle5 = switch.add_ref(pg.rectangle(size=(4.5, 0.5), layer=layer_topmetal))
Rectangle6 = switch.add_ref(pg.rectangle(size=(0.5,5.5), layer=layer_topmetal))
Rectangle7 = switch.add_ref(pg.rectangle(size=(9, 0.5), layer=layer_topmetal))
Rectangle5.move([-8.5,0])
Rectangle6.move([-8.5,0])
Rectangle7.move([-17,5])
Rectangle8 = switch.add_ref(pg.rectangle(size=(4.5, 0.5), layer=layer_topmetal))
Rectangle9 = switch.add_ref(pg.rectangle(size=(0.5,5), layer=layer_topmetal))
Rectangle10 = switch.add_ref(pg.rectangle(size=(9, 0.5), layer=layer_topmetal))
Rectangle8.move([-8.5,19.5])
Rectangle9.move([-8.5,15])
Rectangle10.move([-17,15])
Rectangle12 = switch.add_ref(pg.rectangle(size=(4.5, 0.5), layer=layer_topmetal))
Rectangle13 = switch.add_ref(pg.rectangle(size=(0.5,5.5), layer=layer_topmetal))
Rectangle14 = switch.add_ref(pg.rectangle(size=(9, 0.5), layer=layer_topmetal))
Rectangle12.move([4,0])
Rectangle13.move([8,0])
Rectangle14.move([8,5])
Rectangle15 = switch.add_ref(pg.rectangle(size=(4.5, 0.5), layer=layer_topmetal))
```

```
Rectangle16 = switch.add_ref(pg.rectangle(size=(0.5,5), layer=layer_topmetal))
Rectangle17 = switch.add_ref(pg.rectangle(size=(9, 0.5), layer=layer_topmetal))
Rectangle15.move([4,19.5])
Rectangle16.move([8,15])
Rectangle17.move([8,15])
Rectangle18 = switch.add_ref(pg.rectangle(size=(11, 0.5), layer=layer_topmetal))
Rectangle18.move([4,11.5])
Rectangle19 = switch.add_ref(pg.rectangle(size=(11, 0.5), layer=layer_topmetal))
Rectangle19.move([4,9])
Rectangle20 = switch.add_ref(pg.rectangle(size=(11.5, 0.5), layer=layer_topmetal))
Rectangle20.move([-15.5,11.5])
Rectangle21 = switch.add_ref(pg.rectangle(size=(11.5, 0.5), layer=layer_topmetal))
Rectangle21.move([-15.5,9])
Rectangle22 = switch.add_ref(pg.rectangle(size=(3, 3), layer=layer_topmetal))
Rectangle22.move([15,9])
Rectangle23 = switch.add_ref(pg.rectangle(size=(3, 3), layer=layer_topmetal))
Rectangle23.move([-18,9])

switch_joined_by_layer = pg.union(switch, by_layer = True)
switch.add_ref(switch_joined_by_layer)
D = Device()
D1 =Device()
D2 = Device()
Rectangle4_size = (8,20)
small_square_size = 0.5  # Size of the smaller squares
rows = 16  # Number of rows of squares
columns = 8  # Number of columns of squares
line_distance = 1
# Calculate the total width and height occupied by the smaller squares
```

```
total_width = columns * small_square_size
total_height = rows * small_square_size
# Calculate the spacing between the smaller squares
spacing_x = (Rectangle4_size[0] - total_width) / (columns + 1.2)
spacing_y = (Rectangle4_size[1] - total_height) / (rows + 0.7)
# Draw the larger square
Rectangle4 = D1 << pg.rectangle(size=Rectangle4_size, layer=layer_topmetal)
# Calculate the position for each smaller square and draw them
for x in range(rows):
  for j in range(columns):
    x_offset = (j + 1.5) * spacing_x + j * small_square_size
    y_offset = (x + 1.2) * spacing_y + x * small_square_size
    inner_square = D << pg.rectangle(size=(small_square_size, small_square_size),
layer=layer_topmetal)
    inner_square.move((x_offset, y_offset))
# Draw the larger square
Rectangle4 =D1<< pg.rectangle(size=Rectangle4_size, layer=layer_topmetal)
X9 =pg.xor_diff(B=D,A= Rectangle4, precision = 1e-6)
D2.add_ref(X9)
# Calculate the dimensions for the lines
inner_square_size = (Rectangle4_size[0] - 2 * line_distance, Rectangle4_size[1] - 2 * line_distance)
X9.movex(-4)
switch.add_ref(D2)


Rectangle24 = pg.rectangle(size=(36, 20),layer=layer_dielectric)
Rectangle24. move([-18,0])
Rectangle25 = pg.rectangle(size=(1, 1),layer=layer_dielectric)
Rectangle25.move([16,10])
Rectangle26 = pg.rectangle(size=(1, 1),layer=layer_dielectric)
```

```
Rectangle26.move([-17,10])
X = pg.xor_diff(A = Rectangle24, B = Rectangle25, precision = 1e-6)
X1 = pg.xor_diff(A = Rectangle24, B = Rectangle26, precision = 1e-6)
X2 =pg.xor_diff(A=X,B=X1, precision = 1e-6)
X3 = pg.xor_diff(A=Rectangle24,B=X2, precision = 1e-6)
switch.add_ref(X3)
Rectangle27 = pg.rectangle(size=(3, 20),layer=layer_dielectric_2)
Rectangle27.move([-18,0])
Rectangle28 = pg.rectangle(size=(3, 20),layer=layer_dielectric_2)
Rectangle28.move([15,0])
Rectangle29 = pg.rectangle(size=(36, 20),layer=layer_dielectric_2)
Rectangle29.move([-18,0])
X4 = pg.xor_diff(A = Rectangle29, B = Rectangle27, precision = 1e-6)
X5 = pg.xor_diff(A = Rectangle29, B = Rectangle28, precision = 1e-6)
X6 =pg.xor_diff(A=X4,B=X5, precision = 1e-6)
switch.add_ref(X6)
switch.write_gds('TOPviewmask.gds')
qp(switch)
```



variations

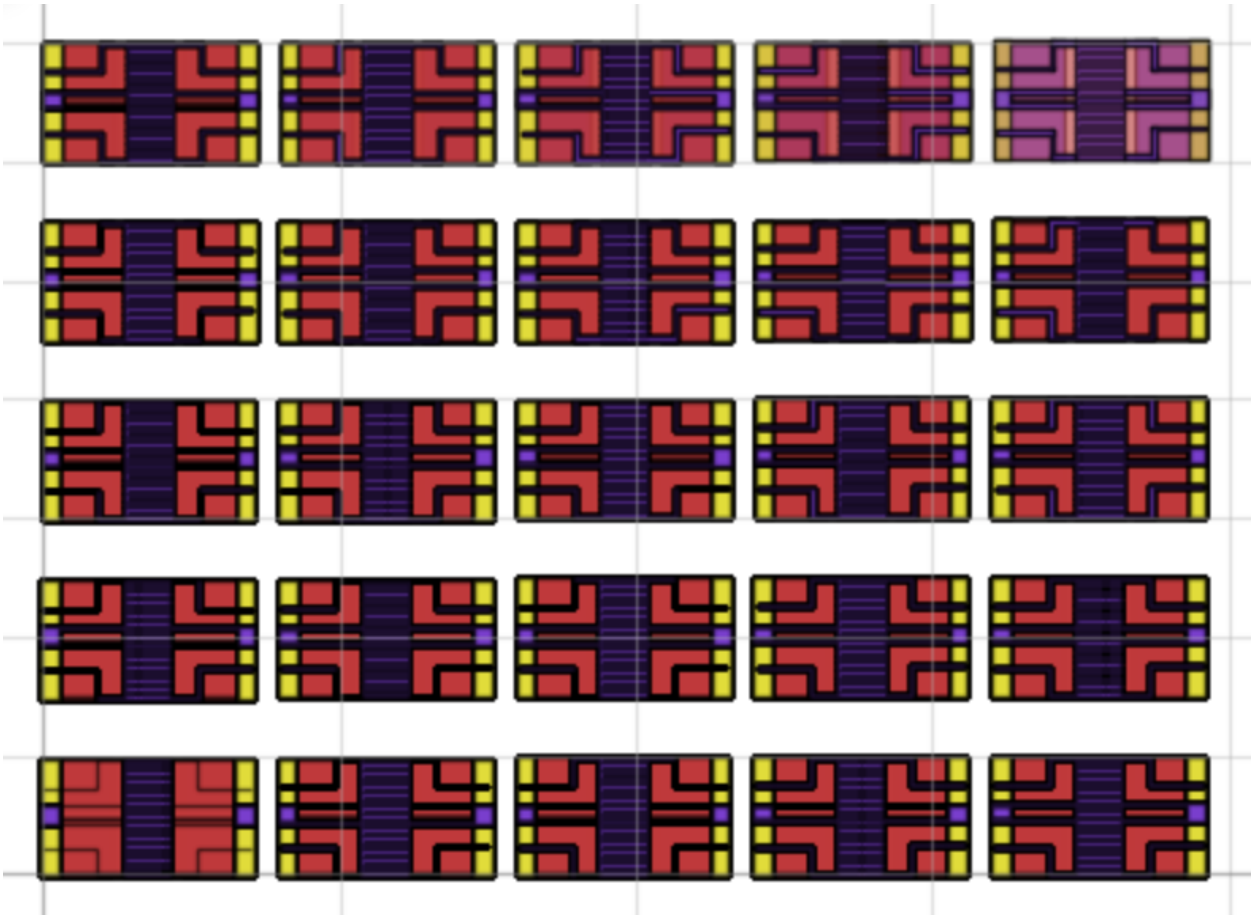
```
import phidl.geometry as pg
from phidl import Device, Layer, Path, CrossSection
from phidl import quickplot as qp
import numpy as np
layer_bottommetal = Layer(1, name='bottommetal')
layer_dielectric = Layer(2, name='dielectric')
layer_dielectric_2 = Layer(3, name='dielectric_2')
layer_topmetal = Layer(4, name='topmetal')
switch = Device()
switch_1 = Device()
columns_1= 5
rows_1=5
for y in range(columns_1):
  for z in range(rows_1):
    i= y+z
    # top layer
    Rectangle5 = switch.add_ref(pg.rectangle(size=(4.5, 0.1*i), layer=layer_topmetal))
    Rectangle6 = switch.add_ref(pg.rectangle(size=(0.1*i,5), layer=layer_topmetal))
    Rectangle7 = switch.add_ref(pg.rectangle(size=(9+(0.1*i), 0.1*i), layer=layer_topmetal))
    Rectangle5.move([9.5+(z*40)-((5-i)*0.1),0+(y*30)-((5-i)*0.1)])
    Rectangle6.move([9.5+(z*40)-((5-i)*0.1),0+(y*30)-((5-i)*0.1)])
    Rectangle7.move([0.5+(z*40)-((5-i)*0.1),4.5+(y*30)])
    Rectangle8 = switch.add_ref(pg.rectangle(size=(4.5, 0.1*i), layer=layer_topmetal))
    Rectangle9 = switch.add_ref(pg.rectangle(size=(0.1*i,5), layer=layer_topmetal))
    Rectangle10 = switch.add_ref(pg.rectangle(size=(9+(0.1*i), 0.1*i), layer=layer_topmetal))
    Rectangle8.move([9.5+(z*40)-((5-i)*0.1),19.5+(y*30)])
    Rectangle9.move([9.5+(z*40)-((5-i)*0.1),15+(y*30)-((5-i)*0.1)])
    Rectangle10.move([0.5+(z*40)-((5-i)*0.1),15+(y*30)-((5-i)*0.1)])
    Rectangle12 = switch.add_ref(pg.rectangle(size=(4.5+(0.1*i), 0.1*i), layer=layer_topmetal))
```

```
Rectangle13 = switch.add_ref(pg.rectangle(size=(0.1*i,5+(0.1*i)), layer=layer_topmetal))
Rectangle14 = switch.add_ref(pg.rectangle(size=(9, 0.1*i), layer=layer_topmetal))
Rectangle12.move([22+(z*40)-((5-i)*0.1),0+(y*30)-((5-i)*0.1)])
Rectangle13.move([26.5+(z*40)-((5-i)*0.1),0+(y*30)-((5-i)*0.1)])
Rectangle14.move([26.5+(z*40)-((5-i)*0.1),5+(y*30)-((5-i)*0.1)])
Rectangle15 = switch.add_ref(pg.rectangle(size=(4.5+(0.1*i), 0.1*i), layer=layer_topmetal))
Rectangle16 = switch.add_ref(pg.rectangle(size=(0.1*i,5), layer=layer_topmetal))
Rectangle17 = switch.add_ref(pg.rectangle(size=(9, 0.1*i), layer=layer_topmetal))
Rectangle15.move([22+(z*40)-((5-i)*0.1),19.5+(y*30)])
Rectangle16.move([26.5+(z*40)-((5-i)*0.1),15+(y*30)-((5-i)*0.1)])
Rectangle17.move([26.5+(z*40)-((5-i)*0.1),15+(y*30)-((5-i)*0.1)])
Rectangle18 = switch.add_ref(pg.rectangle(size=(11.5, 0.1*i), layer=layer_topmetal))
Rectangle18.move([22+(z*40)-((5-i)*0.1),11.5+(y*30)])
Rectangle19 = switch.add_ref(pg.rectangle(size=(11.5, 0.1*i), layer=layer_topmetal))
Rectangle19.move([22+(z*40)-((5-i)*0.1),9+(y*30)-((5-i)*0.1)])
Rectangle20 = switch.add_ref(pg.rectangle(size=(11.5, 0.1*i), layer=layer_topmetal))
Rectangle20.move([2.5+(z*40)-((5-i)*0.1),11.5+(y*30)])
Rectangle21 = switch.add_ref(pg.rectangle(size=(11.5, 0.1*i), layer=layer_topmetal))
Rectangle21.move([2.5+(z*40)-((5-i)*0.1),9+(y*30)-((5-i)*0.1)])
Rectangle22 = switch.add_ref(pg.rectangle(size=(3, 3), layer=layer_topmetal))
Rectangle22.move([33+(z*40)-((5-i)*0.1),9+(y*30)-((5-i)*0.1)])
Rectangle23 = switch.add_ref(pg.rectangle(size=(3, 3), layer=layer_topmetal))
Rectangle23.move([0+(z*40)-((5-i)*0.1),9+(y*30)-((5-i)*0.1)])
switch_joined_by_layer = pg.union(switch, by_layer = True)
switch.add_ref(switch_joined_by_layer)
D = Device()
D1 =Device()
D2 = Device()
Rectangle4_size = (8,20)
```

```
    small_square_size = 0.5  # Size of the smaller squares
    rows = 16  # Number of rows of squares
    columns = 8  # Number of columns of squares
    line_distance = 1
    # Calculate the total width and height occupied by the smaller squares
    total_width = columns * small_square_size
    total_height = rows * small_square_size
    # Calculate the spacing between the smaller squares
    spacing_x = (Rectangle4_size[0] - total_width) / (columns + 1.2)
    spacing_y = (Rectangle4_size[1] - total_height) / (rows + 0.7)
    # Draw the larger square
    Rectangle4 = D1 << pg.rectangle(size=Rectangle4_size, layer=layer_topmetal)
    # Calculate the position for each smaller square and draw them
    for x in range(rows):
      for j in range(columns):
        x_offset = (j + 1.5) * spacing_x + j * small_square_size
        y_offset = (x + 1.2) * spacing_y + x * small_square_size
        inner_square = D << pg.rectangle(size=(small_square_size, small_square_size),
layer=layer_topmetal)
        inner_square.move((x_offset, y_offset))
    # Draw the larger square
    Rectangle4 =D1<< pg.rectangle(size=Rectangle4_size, layer=layer_topmetal)
    X6 =pg.xor_diff(B=D,A= Rectangle4, precision = 1e-6)
    D2.add_ref(X6)
    # Calculate the dimensions for the lines
    inner_square_size = (Rectangle4_size[0] - 2 * line_distance, Rectangle4_size[1] - 2 *
line_distance)
    switch.add_ref(D2)
    X6.move([14+(z*40)-((5-i)*0.1),0+(y*30)-((5-i)*0.1)])
```

```
#bottom layer
Rectangle1 = switch.add_ref(pg.rectangle(size=(8, 20), layer=layer_bottommetal))
Rectangle2 = switch.add_ref(pg.rectangle(size=(12, 20), layer=layer_bottommetal))
Rectangle3 = switch.add_ref(pg.rectangle(size=(12, 20), layer=layer_bottommetal))
Rectangle1.move([14+(z*40)-((5-i)*0.1),0+(y*30)-((5-i)*0.1)])
Rectangle2.move([24+(z*40)-((5-i)*0.1),0+(y*30)-((5-i)*0.1)])
Rectangle3.move([0+(z*40)-((5-i)*0.1),0+(y*30)-((5-i)*0.1)])
#dielectric layer
Rectangle24 = pg.rectangle(size=(36, 20),layer=layer_dielectric)
Rectangle24. move([0+(z*40)-((5-i)*0.1),0+(y*30)-((5-i)*0.1)])
Rectangle25 = pg.rectangle(size=(1, 1),layer=layer_dielectric)
Rectangle25.move([34+(z*40)-((5-i)*0.1),10+(y*30)-((5-i)*0.1)])
Rectangle26 = pg.rectangle(size=(1, 1),layer=layer_dielectric)
Rectangle26.move([1+(z*40)-((5-i)*0.1),10+(y*30)-((5-i)*0.1)])
X = pg.xor_diff(A = Rectangle24, B = Rectangle25, precision = 1e-6)
X1 = pg.xor_diff(A = Rectangle24, B = Rectangle26, precision = 1e-6)
X2 =pg.xor_diff(A=X,B=X1, precision = 1e-6)
X3 = pg.xor_diff(A=Rectangle24,B=X2, precision = 1e-6)
switch.add_ref(X3)
#final etching
Rectangle27 = pg.rectangle(size=(3, 20),layer=layer_dielectric_2)
Rectangle27.move([0+(z*40)-((5-i)*0.1),0+(y*30)-((5-i)*0.1)])
Rectangle28 = pg.rectangle(size=(3, 20),layer=layer_dielectric_2)
Rectangle28.move([33+(z*40)-((5-i)*0.1),0+(y*30)-((5-i)*0.1)])
Rectangle29 = pg.rectangle(size=(36, 20),layer=layer_dielectric_2)
Rectangle29.move([0+(z*40),0+(y*30)])
X4 = pg.xor_diff(A = Rectangle29, B = Rectangle27, precision = 1e-6)
```

```
    X5 = pg.xor_diff(A = Rectangle29, B = Rectangle28, precision = 1e-6)
    X6 =pg.xor_diff(A=X4,B=X5, precision = 1e-6)
    switch.add_ref(X6)
switch.write_gds('masks_1.gds')
qp(switch)
```



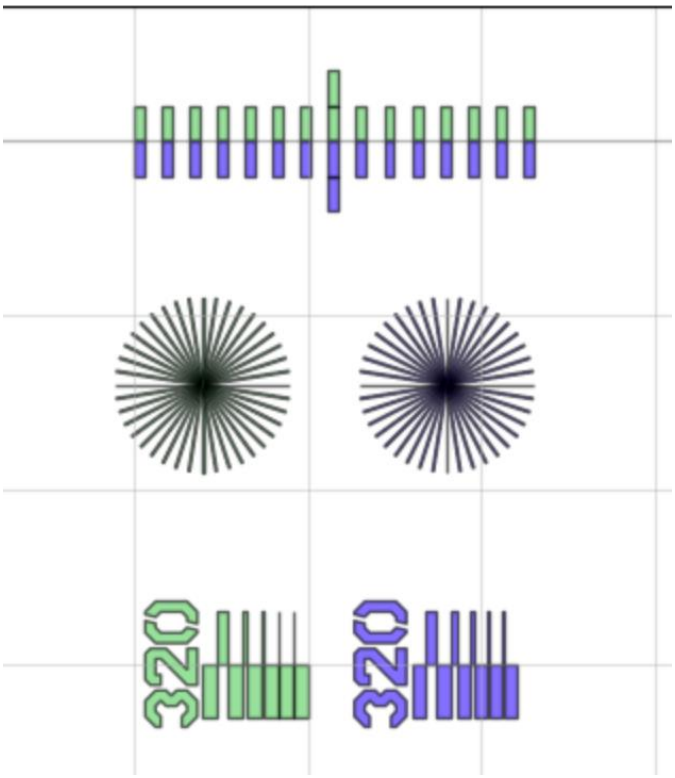
Test _st

```
from phidl import Device, geometry as pg
from phidl import quickplot as qp

DT= Device()
```

```
# Create and add the new pattern
callipers = pg.litho_calipers(
    notch_size=[300, 1000],
    notch_spacing=500,
    num_notches=7,
    offset_per_notch=0.1,
    row_spacing=0,
    layer1=0,
    layer2=1)
callipers_ref = DT.add_ref(callipers)

callipers_ref.move((25000, 0))

#Lithographic star

lithostar = pg.litho_star(
    num_lines = 20,
    line_width = 40,
    diameter = 5000,
    layer = 0
    )
lithostar_ref = DT.add_ref(lithostar)

lithostar_ref.move((27000, -7000))

#Lithographic star

lithostar = pg.litho_star(
    num_lines = 20,
```

```
    line_width = 40,
    diameter = 5000,
    layer = 1
    )
lithostar_ref = DT.add_ref(lithostar)

lithostar_ref.move((34000, -7000))



#Step-resolution

Step_resolution = pg.litho_steps(
    line_widths = [10*2,20*2,40*2,80*2,160*2],
    line_spacing = 400,
    height = 3000,
    layer = 0
    )
Step_resolution_ref = DT.add_ref(Step_resolution)

Step_resolution_ref.move((27000, -15000))

#Step-resolution

Step_resolution = pg.litho_steps(
    line_widths = [10*2,20*2,40*2,80*2,160*2],
    line_spacing = 400,
    height = 3000,
    layer = 1
    )
```

```
Step_resolution_ref = DT.add_ref(Step_resolution)

Step_resolution_ref.move((33000, -15000))

# Quickplot the geometry
qp(DT)

D.write_gds('Finalproject.gds')
```



Other st for testing

```
import phidl.geometry as pg
from phidl import Device, quickplot as qp

def shape_with_ground_pad():
```

```
    D = Device()
    s = D << pg.snspd_expanded(layer = 0).rotate(-90)
    contact_pad = D << pg.compass(size = (10,10), layer = 1)
    ground_pad = D << pg.compass(size = (10,10), layer = 50)
    contact_pad.connect('S',s.ports[1])
    ground_pad.connect('N',s.ports[2])
    return D

def three_shapes_with_ground_pad():
    Structures = Device()
    s1 = Structures << shape_with_ground_pad()
    s2 = Structures << shape_with_ground_pad()
    s3 = Structures << shape_with_ground_pad()
    group = s1 + s2 + s3
    group.distribute(direction = 'x', spacing = 10)
    return Structures

qp(three_shapes_with_ground_pad())
D.write_gds('three_shapes_with_ground_pad.gds')
```