GDSII Feature Extraction with OpenLane and KLayout

This markdown document outlines the full workflow to: - Run synthesis and P&R with OpenLane - Generate the GDSII file - View it in KLayout - Extract per-layer layout features and density metrics - Export data to CSV for further analysis (e.g., ML)

📁1. OpenLane Setup and GDS Generation

Step 1: Mount the OpenLane Docker environment

```
cd ~/OpenLane
make mount
```

Step 2: Inside the container, run the flow interactively

```
./flow.tcl -interactive
```

Step 3: Run the design

```
prep -design spm
run_flow
```

Or run individual steps if needed:

```
run_synthesis
run_floorplan
run_placement
run_cts
run_routing
run_magic
run_klayout
```

📫2. View GDS in KLayout (Host Machine)

Step 1: Exit the OpenLane container

Step 2: Locate the GDS file

```
ls ~/OpenLane/designs/spm/runs/*/results/signoff/*.gds
```

Step 3: Open it using KLayout

```
klayout ~/OpenLane/designs/spm/runs/RUN_2025.07.10_10.50.49/results/signoff/
spm.gds
```

---

3. Python Script to Extract Features and Density

Save as `export_gds_features.py` in the same folder as the GDS file:

```
import pya
import csv
import math

layout = pya.Layout()
layout.read("spm.gds")
top_cell = layout.top_cell()
dbu = layout.dbu

GRID_SIZE_UM = 10

layers = [
    ("met1", layout.layer(67, 20)),
    ("met2", layout.layer(68, 20)),
    ("met3", layout.layer(69, 20)),
    ("via1", layout.layer(67, 44)),
    ("via2", layout.layer(68, 44)),
]

bbox = top_cell.bbox()
x_min = bbox.left * dbu
y_min = bbox.bottom * dbu
x_max = bbox.right * dbu
y_max = bbox.top * dbu
```

```python
cols = math.ceil((x_max - x_min) / GRID_SIZE_UM)
rows = math.ceil((y_max - y_min) / GRID_SIZE_UM)

density_grid = {}
for r in range(rows):
    for c in range(cols):
        density_grid[(r, c)] = 0.0

with open("layout_features.csv", "w", newline="") as csvfile:
    fieldnames = ["layer", "x1", "y1", "x2", "y2", "width", "height", "area",
"feature_type", "grid_row", "grid_col"]
    writer = csv.DictWriter(csvfile, fieldnames=fieldnames)
    writer.writeheader()

    for lname, layer in layers:
        shapes = top_cell.shapes(layer)
        for s in shapes.each():
            if s.is_box():
                box = s.bbox()
                x1 = box.left * dbu
                y1 = box.bottom * dbu
                x2 = box.right * dbu
                y2 = box.top * dbu
                width = x2 - x1
                height = y2 - y1
                area = width * height

                col = int((x1 - x_min) / GRID_SIZE_UM)
                row = int((y1 - y_min) / GRID_SIZE_UM)

                density_grid[(row, col)] += area

                writer.writerow({
                    "layer": lname,
                    "x1": x1, "y1": y1, "x2": x2, "y2": y2,
                    "width": width, "height": height,
                    "area": area,
                    "feature_type": "via" if "via" in lname else "wire",
                    "grid_row": row, "grid_col": col
                })

with open("layout_density_grid.csv", "w", newline="") as csvfile:
    writer = csv.DictWriter(csvfile, fieldnames=["grid_row", "grid_col",
"density_um2"])
    writer.writeheader()
    for (r, c), area in density_grid.items():
        writer.writerow({"grid_row": r, "grid_col": c, "density_um2": area})
```

```
print("✅ Extraction complete: layout_features.csv and layout_density_grid.csv
created.")
```

📠4. Run the Script in OpenLane Container

```
klayout -b -r export_gds_features.py
```

💾5. Export CSVs to Host Machine

```
docker ps

docker cp <container_id>:/openlane/designs/spm/runs/RUN_2025.07.10_10.50.49/
results/signoff/layout_features.csv ~/Downloads/
docker cp <container_id>:/openlane/designs/spm/runs/RUN_2025.07.10_10.50.49/
results/signoff/layout_density_grid.csv ~/Downloads/
```

🔗Final Output

- `layout_features.csv`: Each metal/via shape with coordinates and area
- `layout_density_grid.csv`: Grid-wise routing density in microns^2

Use these for layout analysis, ML models, congestion checks, or floorplan planning.

Let me know if you want to convert this into a GitHub-friendly README or extend the feature extractor! 🚯