Owen-Ethan_905452983_palatics_Lab4

Ethan Owen
SID: 905452983
EE201A, Winter 2026, MSOL

Part 1: Min Area

Initial Steps

I started by adding in all the files that we were given in the project as follows, and ensured they were referenced as needed in the script

- Gate-level netlist (`.v` file, generated by synthesis tool, e.g., Cadence Genus)
- Timing constraints (`.sdc` file, generated by synthesis tool)
- LEF for library/technology (`.lef` file, Nangate/FreePDK)
- Liberty timing specs for library (`.lib` file, Nangate)
- GDS for Nangate library cells (`.gds` file, to write final design GDS for tapeout

UTIL Value Adjustments

I adjusted the UTIL value in the script and observed the resulting area that was generated from it.

Target Util	Core Area (um^2)	Chip Area (um^2)	Setup Slack (ns)	Hold Slack (ns)	DRC Violations	Result
0.90	383.838	1,005.480	0.527	0.017	0	PASS
0.99	349.258	948.024	0.518	0.017	0	PASS
0.991	349.258	948.024	0.518	0.017	0	PASS
0.992	(smaller)	--	--	--	781+	FAIL
0.995	(smaller)	--	--	--	781+	FAIL

I watched for improved cell utilizations as well, since a high cell utilization means that the design is crammed into a smaller footprint, and thus a smaller core area + chip area.

I landed on `0.991` being the best that I could go without introducing massive DRC violations.

Results

```
INITIAL_UTILL: 0.991
FINAL_UTIL: 0.98172
FINAL_SETUP_SLACK: 0.518
```

Part 2: No Timing Driven

Timing (hold, setup) and Power Reports

These were the end places where I saved results for power, setup time, and hold time, while running the

TCL script. These save points are consistent between the part 2 case with `timingDriven true` and `timingDriven false`.

- `before_placement`
- `after_placement`
- `after_opt_prects`
- `after_extractrc_postcts`
- `after_opt_postcts_hold`
- `after_extractrc_preroute`
- `after_global_route`
- `after_detail_route`
- `after_opt_postroute_setup`
- `after_opt_postroute_hold`
- `after_extractrc_postroute`
- `final_postroute`

Table of Information

This table was generated after running `lab4_part2` followed by `lab4_part2` with `timingDriven` set to false. The results are compared for hold & setup timing and power.

```
Step                          |   Setup TD Setup NoTD |    Hold TD  Hold NoTD |      Power TD   Power NoTD
-----------------------------------------------------------------------------------------------------------
Before Placement              |      0.754      0.754 |      0.013      0.013 |  0.24192193   0.24192193
After Placement               |      0.601      0.601 |      0.017      0.017 |  0.30237865   0.30237865
After Pre-CTS Opt             |      0.571      0.571 |      0.017      0.017 |  0.16421903   0.16421903
After Post-CTS RC Ext         |      0.582      0.582 |      0.017      0.017 |  0.16616036   0.16616036
After Post-CTS Hold Opt       |      0.582      0.582 |      0.017      0.017 |  0.16616036   0.16616036
After Pre-Route RC Ext        |      0.457      0.457 |      0.020      0.020 |  0.19107963   0.19107963
After Global Route            |      0.470      0.470 |      0.019      0.019 |  0.18469520   0.18469520
After Detail Route            |      0.470      0.470 |      0.019      0.019 |  0.18469520   0.18469520
After Post-Route Setup Opt    |      0.518      0.518 |      0.017      0.017 |  0.16604142   0.16604142
After Post-Route Hold Opt     |      0.518      0.518 |      0.017      0.017 |  0.16604142   0.16604142
Final Post-Route RC Ext       |      0.518      0.518 |      0.017      0.017 |  0.16604142   0.16604142
-----------------------------------------------------------------------------------------------------------
```

I noticed that the output of the final setup time was the same as that of part 1, which was a good sign that the results are consistent between the two steps.

What does `timingDriven` actually do?

Timing driven placement essentially makes timing a component of the placement objective. Normally, without this, we consider wirelength and congestion. But this option factors in timing as well. I think this mainly serves to improve critical timing paths in the place and after-placement optimization steps.

Results

I noticed *no* differences between `timingDriven false` and `timingDriven true`. There are a couple potential reasons for this that I see:

1. The design is pretty small
2. Both runs share the same core floorplan, utilization, libraries, constraints, so maybe `timingDriven` has a lesser effect due to that
3. Later optimization steps dominate the early ones with `timingDriven`
4. In my conclusions, I only compare worse case timing cases and power, there might be minor differences in the other results, but I doubt this.

Part 3: Power Structure

Adding Stripes in CLI

I used the CLI only and no GUI at all. To add the strips I used the following parameters:

```
set STRIPE_LAYER metal2
set STRIPE_DIRECTION vertical
set STRIPE_WIDTH_UM 0.21
set STRIPE_SPACING_UM 4.13
set STRIPE_OFFSET_LEFT_UM 2.0
set STRIPE_SET_DISTANCE_UM 8.26
```

These we extracted from the problem statement. The only outstanding figure here is the set distance, which ends up being double the stripe spacing.

Optimizations and UTIL

When I originally ran the script with no changes in UTIL values, I got huge DRC errors. In response I iteratively lowered the UTIL values from `0.991` to `0.918` where I finally passed DRC. Then I compared final values from here

Table of Comparison to Part 1

To compare against, I used my script from part 1 as the benchmark, and added in the metal strip and compared the results using a shell script.

Metric	Part 1	Part 3 Stripes	Delta (P3-P1)
Target Utilization	0.991	0.918	-0.073000
Final Utilization (%)	98.172	90.967	-7.205000
Worst Setup Slack (ns)	0.518	0.429	-0.089000
Worst Hold Slack (ns)	0.017	0.015	-0.002000
Total Power (mW)	---	0.19797320	---
Power Delta (%)	---	---	---
DRC Violations	0	0	0.000000
Core Area (um^2)	349.258	376.922	27.664000
Chip Area (um^2)	948.024	993.989	45.965000
Legality (Timing+DRC)	PASS	PASS	

Results

These results we accomplished with no DRC violations.

- Highest tested legal target utilization: 0.918
- Corresponding best final utilization (actual): 0.9067
- Worst setup slack: 0.429 ns
- Worst hold slack: 0.015 ns

By adding the power strips we worsened utilization, setup slack, and hold slack in comparison to the original TCL script that did not include the strips.