`Faculty:	**FACULTY OF ELECTRICAL ENGINEERING**		
Subject : **ELECTRICAL ENGINEERING LABORATORY** Subject Code : **SKEL 3742**		Review : Release Date : 16 March 2026 Last Amendment Procedure Number	

FAKULTI KEJURUTERAAN ELEKTRIK
UNIVERSITI TEKNOLOGI MALAYSIA
KAMPUS SKUDAI
JOHOR

SKEL 3742
VLSI SYSTEM DESIGN LAB

Student Pack : VLSI Physical Design with OpenLane EDA Tutorial

Prepared by : 1. Prof. Madya Ir. Dr. Ab Al-Hadi bin Ab Rahman Signature & Stamp : ASSOC. PROF. IR. DR. AB AL-HADI AB RAHMAN Faculty of Electrical Engineering Universiti Teknologi Malaysia 81310 Johor Bahru Johor Date: 16 March 2026	Approved by: Name: PROF. IR. DR. RUBITA SUDIRMAN HEAD OF DEPARTMENT DEPT. OF ELECTRONIC AND COMPUTER ENGINEERING FACULTY OF ELECTRICAL ENGINEERING UNIVERSITI TEKNOLOGI MALAYSIA 81310 JOHOR BAHRU, JOHOR email: rubita@utm.my Signature & Stamp : Date: 19/3/2026

VLSI Physical Design Lab Tutorial

Faculty of Electrical Engineering

Universiti Teknologi Malaysia

16 March 2026

Platform: VMWare Workstation

Linux: Almalinux 9.5

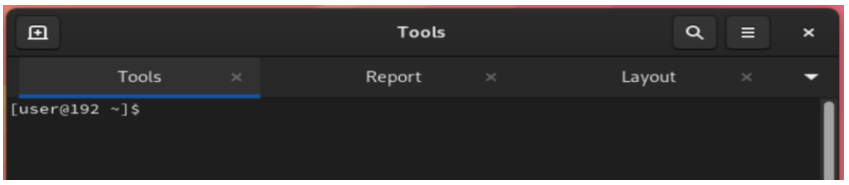
Design Example: 32-bit multiplier

Task: Complete this tutorial by filling in the blank spaces in red fonts

---

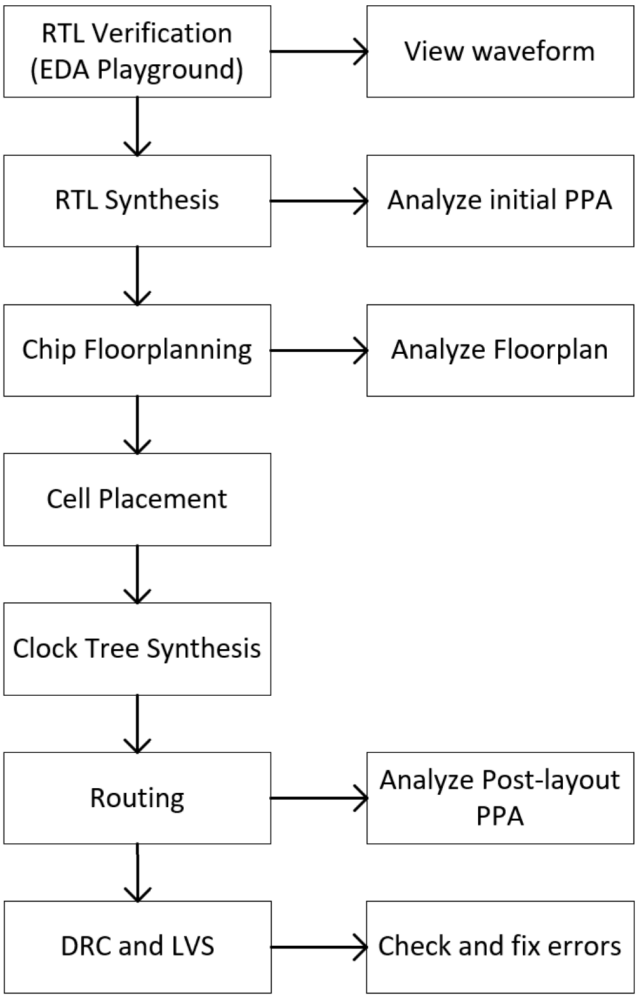
Preliminary

a. Familiarize with basic Linux commands and operations, such as, cd, pwd, ls, mkdir, etc.
b. Open Linux Terminal and setup with three Terminal tabs for Tools, Report, and Layout.
   Tools: For running the tools command (e.g., run_synthesis etc.)
   Report: For opening the logs and results files to view design parameters
   Layout: For viewing chip layout using the KLayout tool



c. Main tools and project directory of OpenLane is at **~/Projects/OpenLane**. Designs are stored at **~/Projects/OpenLane/designs**.

d. Overall Project flow is given below:



1. **Create the Design Files**
   a. Navigate to your OpenLane directory (in **Report** tab) and set up the structure:
   `~/Projects/OpenLane/designs/multiplier/src`

   b. Create or copy the Verilog design file **multiplier.v** in
   `~/Projects/OpenLane/designs/multiplier/src/multiplier.v`

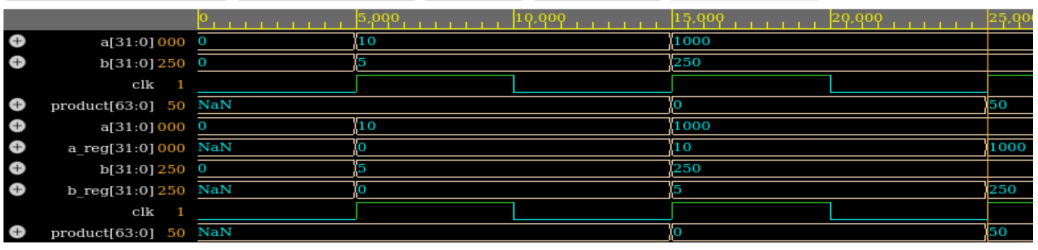
   c. Create or copy the Verilog testbench file tb_multiplier.v in
   `~/Projects/OpenLane/designs/multiplier/tb/tb_multiplier.v`

   d. The design is basically a synchronous 32-bit multiplier with registers. We use input and output registers. This is crucial for timing closure; without

them, the "combinational path" from input pin to output pin would be too long for high clock speeds.

2. **Simulating the design**
    a. For functional simulation, we will be using the open-source EDA Playground.
    b. **Open EDA Playground using Firefox:** Go to edaplayground.com.
    c. **Input the Code:**
        i. **Right Pane (multiplier.v):** Paste your module multiplier code here.
        ii. **Left Pane (tb_multiplier.v):** Paste the testbench code here.

    d. **Configure the Sidebar (Tools & Simulators):**
        i. **Testbench + Design:** Select **SystemVerilog/Verilog**.
        ii. **Tools & Simulators:** Choose **Icarus Verilog X.0**
        iii. **Check the box: Open EPWave after run** (this is crucial for seeing the waveform).
        iv. **Click Save.**
    e. Click the **Run** button in the top header.
    f. **View Results:**
        i. **Console View:** Look at the bottom "Logs" pane. You will see the $monitor output showing the calculations.
        ii. **Waveform View:** A new tab (EPWave) will open automatically. Click "Get Signals" or look at the tree on the left to add a, b, and product to the window.



3. **Configure the Design (config.json)**
    a. Create the configuration file below in **~/Projects/OpenLane/designs/multiplier/config.json**

```
{
    "DESIGN_NAME": "multiplier",
    "VERILOG_FILES": "dir::src/multiplier.v",
    "CLOCK_PORT": "clk",
    "CLOCK_PERIOD": 10.0,
    "FP_SIZING": "relative",
    "FP_CORE_UTIL": 40,
    "PL_TARGET_DENSITY": 0.45,
    "SYNTH_STRATEGY": "AREA 0"
}
```

    b. This file controls your constraints. We will start with a conservative 100MHz clock (period = 10ns). **SYNTH_STRATEGY: AREA 0** tells the tools to focus on making the circuit as small as possible.

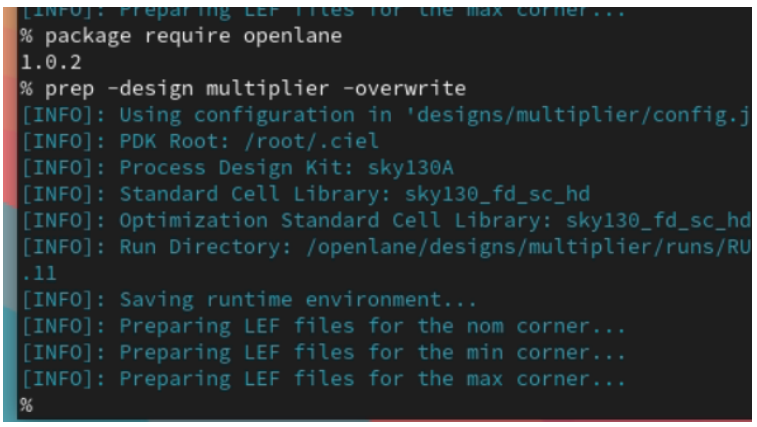
4. **Run Synthesis & Initial PPA Analysis**
    a. In the **Tools** tab, Go to the **~/Projects/OpenLane**, and run the command to load the tools:

```
make mount PDK_ROOT=/opt/sky130_pdk

./flow.tcl -design multiplier -interactive

package require openlane

prep -design multiplier -overwrite
```



    Running the flow.tcl script “loads” the tool where tool commands can be run at “%”.

b. Next, we will synthesize the multiplier design. Run the command:

`run_synthesis -overwrite`

```
% run_synthesis -overwrite
[STEP 1]
[INFO]: Running Synthesis (log: designs/multiplier/runs/RUN_2026.03.09_14.25.
11/logs/synthesis/1-synthesis.log)...
[STEP 2]
[INFO]: Running Single-Corner Static Timing Analysis (log: designs/multiplier
/runs/RUN_2026.03.09_14.25.11/logs/synthesis/2-sta.log)...
%
```

This runs the synthesis process to obtain the initial power, performance, and area parameters. As we can see, the synthesis (**1-synthesis.log**) and timing analysis (**2-sta.log**) run log is written. Any errors or status can be found from these log files.
Take note of the run directory, for example above, **RUN_2026.03.09_14.25.11**. This is where all the log and results are stored.

5. Analyzing synthesis PPA results

a. Once synthesis finishes, it will print a summary of the gates used. You can find the detailed reports in:

~/Projects/OpenLane/designs/multiplier/runs/RUN_XXX/reports/synthesis/

b. Open the file **1-synthesis_pre.stat** from the above directory (using any text editor such as vim or gedit), e.g.:

```
[user@localhost OpenLane]$ gvim designs/multiplier/runs/RUN_2026.03.09_14.25.
11/reports/synthesis/1-synthesis_pre.stat
```

Take note of the cell count:

Number of cells:

Sequential logic cells:

Combinational logic cells:

c. Hint: you can use the Grep command to easily find the “cells” information:

d. Open the file **1-synthesis.AREA_0.stat.rpt** from the above directory and find the post synthesis resource information.

```
[user@localhost OpenLane]$ gvim designs/multiplier/runs/RUN_2026.03.09_14.25.
11/reports/synthesis/1-synthesis.AREA_0.stat.rpt
```

post-synthesis cell count and area**:**

Number of cells:

Chip area:

e. Timing and area information can be found here:

~/Projects/OpenLane/designs/multiplier/runs/RUN_XXX/logs/synthesis/x-sta.log

Find the slack and power information:

Internal power:

Switching power:

Leakage power:

Total power:

f. Also in the same log file, find the timing information:

Worst negative slack (Setup):

Worst slack (hold):

Positive slack means timing is achieved. Negative slack needs to be fixed.

g. Exercise: Modify the clock period (**config.json**) to 5 ns and record new the PPA results.

(Hint: you need to unmount the tool, modify the config.json file, and re-run synthesis).

Pre-synthesis cell count:

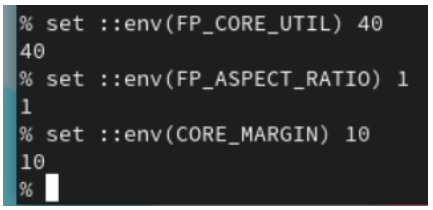
Post-synthesis cell count:

Total power:

WNS (Setup):

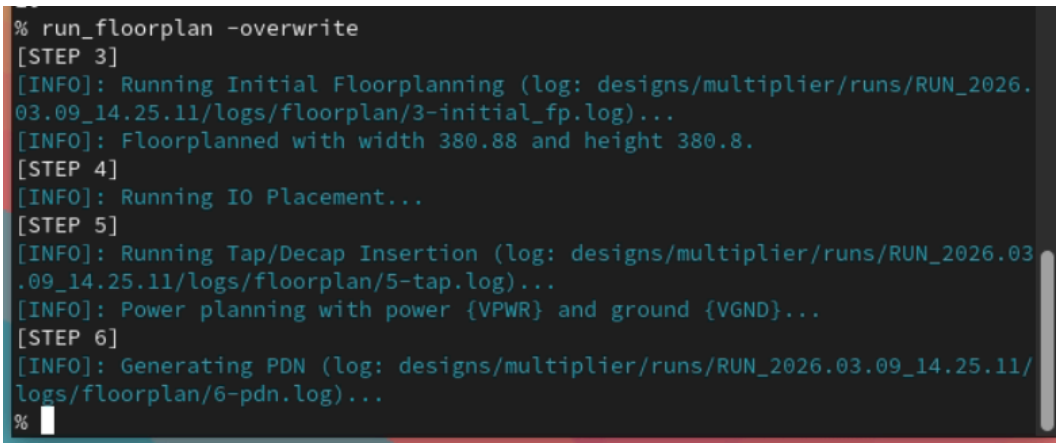
WNS (Hold):

6. **Running chip floorplanning**
    a. Floorplanning prepares the chip layout in terms of core utilization, aspect ratio, and core margin, etc.
    b. First, we need add/modify some constraints. We can modify the constraint directly in the tool.



In the above, we are setting a 40% core utilization, chip aspect ratio of 1 (perfect square), and core margin (space between I/O pins and wiring) of 10um.

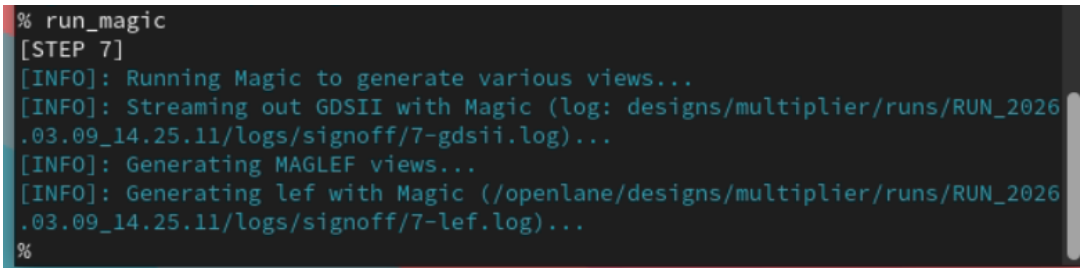
    c. Run the command **`run_floorplan`** to create the floorplan based on our constraints.



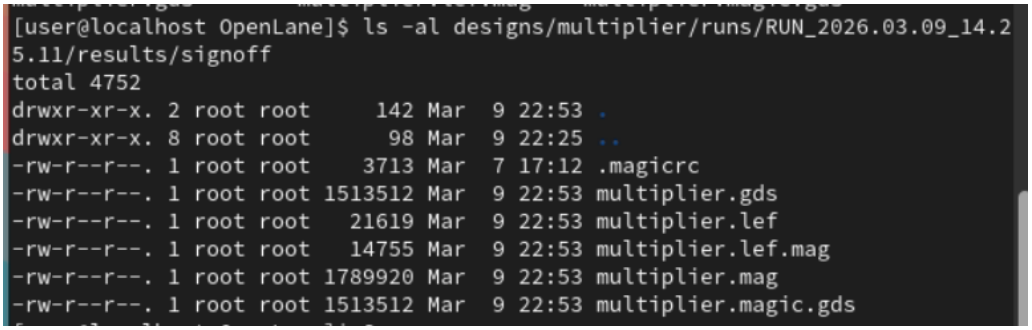
Again, several log files are generated to check for errors and status. Reports and results can also be found within the directory above.
The tool is doing three major things right now:
Defining the Boundary: It calculates the total area needed based on your gate count from synthesis divided by 0.40.
Placing the I/O Pins: It spreads your 32-bit a, 32-bit b, and 64-bit product pins around the edges.

Creating the PDN (Power Distribution Network): It lays down the "Power Stripes" (VDD and VSS) so every gate in the multiplier has electricity.

d. Lets view the generated floorplan. For this, we need to generate the GDS of the floorplan. Type in **`run_magic`**.



After running run_magic, we can see that the GDS is generated in the "**results/signoff**" directory, called **multiplier.gds**.



e. To open and view the GDS file, run the commands below in the **`~/Projects/OpenLane`** directory:

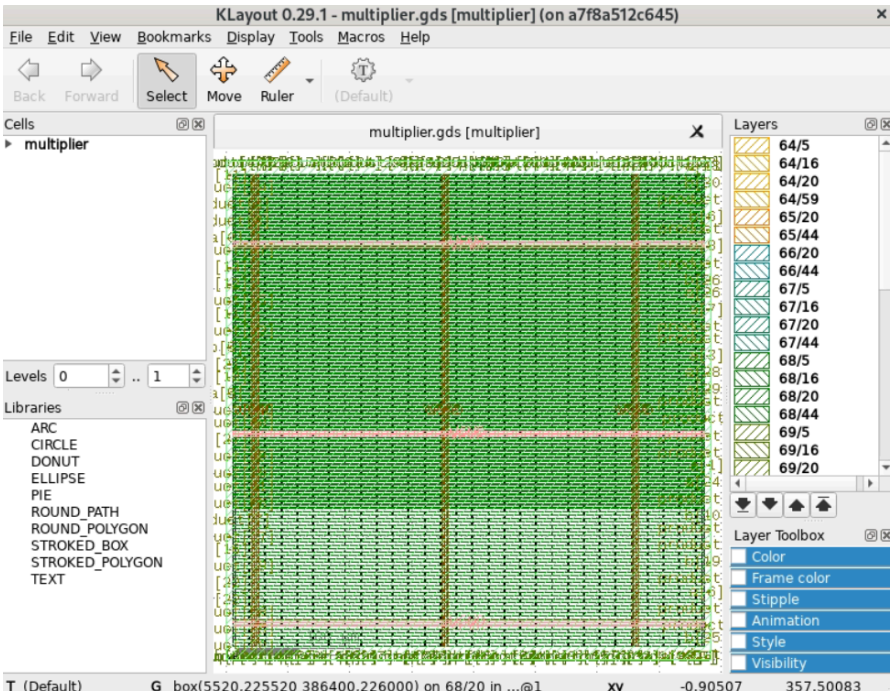
```
xhost +local:docker

sudo docker run -it -v $(pwd):/openlane -v
/tmp/.X11-unix:/tmp/.X11-unix -e DISPLAY=$DISPLAY
efabless/openlane:latest

klayout -rd design_name=multiplier
```

```
[user@localhost OpenLane]$ xhost +local:docker
non-network local connections being added to access control list
[user@localhost OpenLane]$ sudo docker run -it -v $(pwd):/openlane -v /tmp/.X
11-unix:/tmp/.X11-unix -e DISPLAY=$DISPLAY efabless/openlane:latest
[sudo] password for user:
OpenLane Container (1.1.1):/openlane% klayout -rd design_name=multiplier
Fontconfig error: Cannot load default config file: No such file: (null)
```

f. In the KLayout editor, Click **File** □ **Open** and open the **multiplier.gds**.



Note that this is a chip layout without any cell placement and routing. Verify that you have VDD and GND connections in the chip layout.

g. To see the actual dimensions of your chip. Look at the log output or check this file:
designs/multiplier/runs/RUN_XXX/results/floorplan/multiplier.def

```
[user@localhost OpenLane]$ gvim designs/multiplier/runs/RUN_2026.03.09_14.25.
11/results/floorplan/multiplier.def
[user@localhost OpenLane]$
```

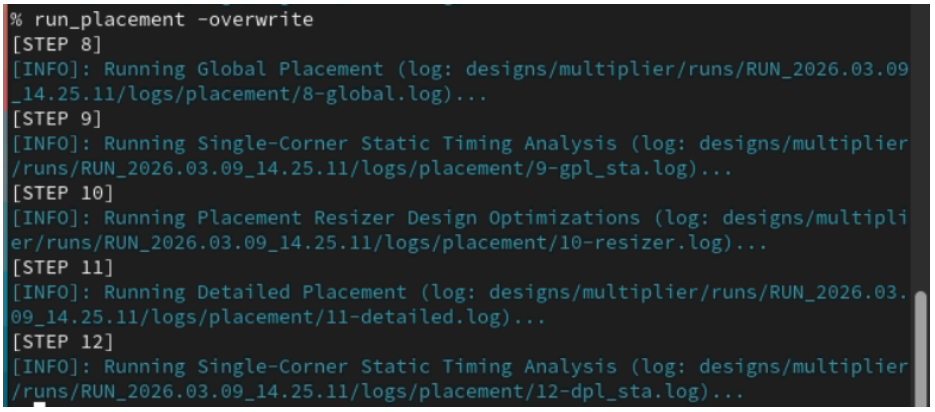
Look for the line:

DIEAREA (0 0) (X Y) ;

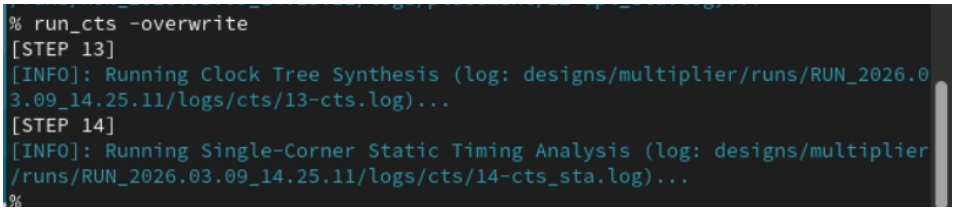
The X and Y coordinates are in nanometer.
Take note of the floorplan width and height:
Width:
Height:

7. **Running the Placement, CTS and routing**
   a. Once floorplanning is complete, the next step is to perform cell placement, clock and signal routing.

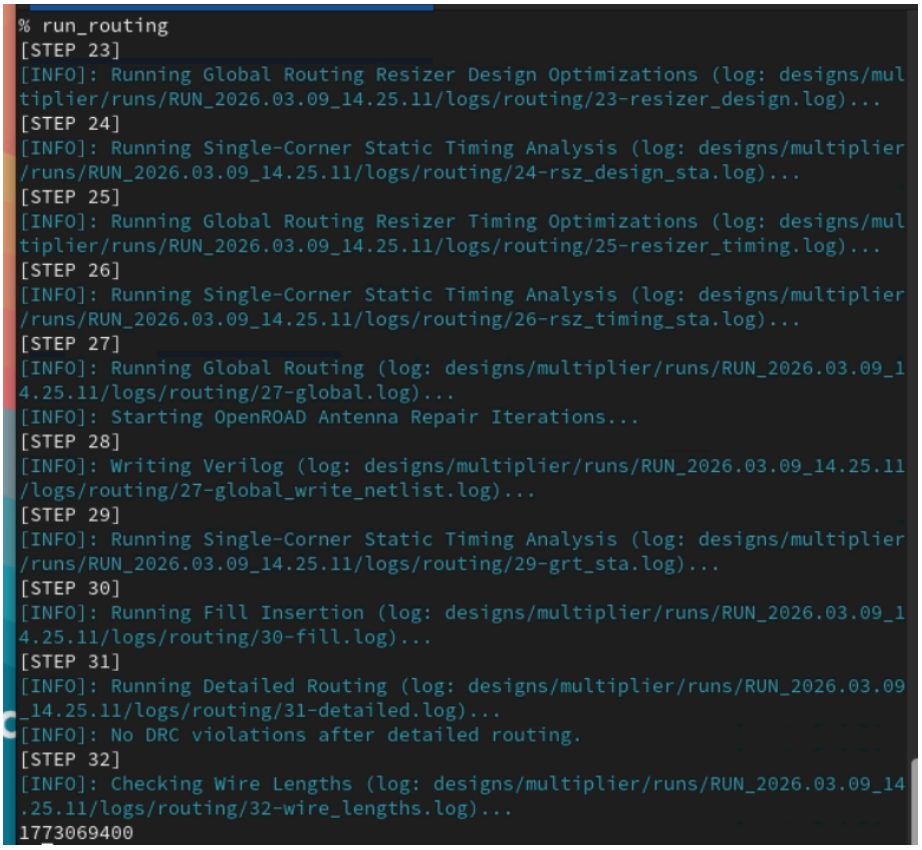
   b. To run placement of cells:
   `run_placement -overwrite`



   c. To run clock tree synthesis:
   `run_cts -overwrite`



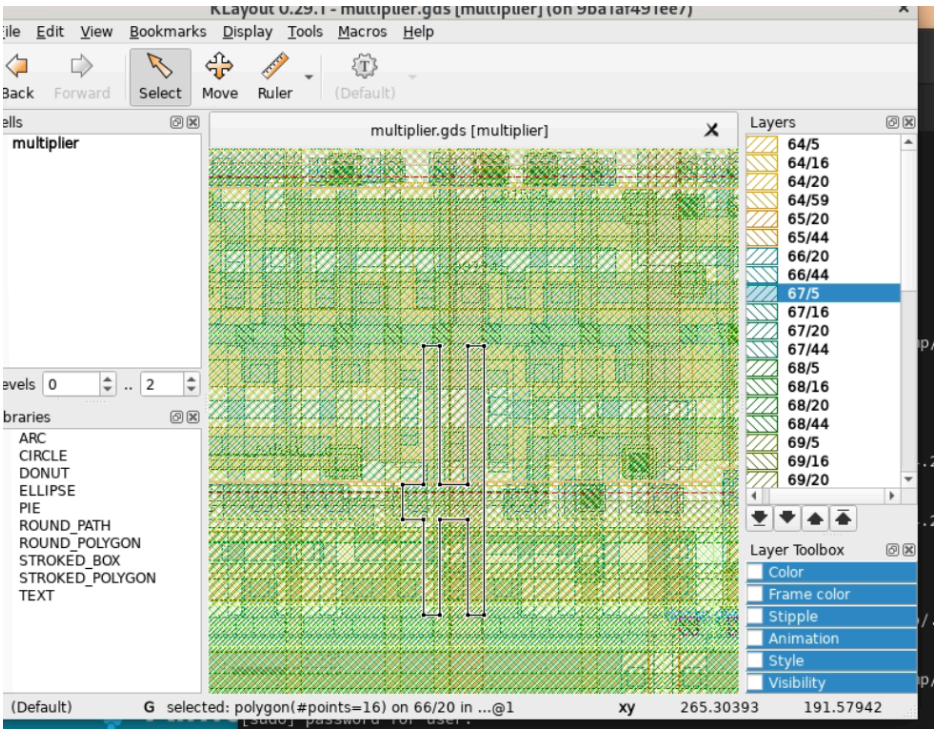
   d. To run routing:
   `run_routing -overwrite`



If there are errors, re-run CTS (**run_cts**) and routing (**run_routing**) command again to perform a second round of routing.

e. Finally, generate the fully placed and route GDS:
`run_magic -overwrite`

f. Check the layout multiplier.gds (“results/signoff” directory) and verify that the cells are placed and signals are routed.



g. Analyze and compare power and timing results before and after physical design. Look for the file **xx-grt_sta.log (logs/routing)**. This log contains the final timing and power information.

Total power:

WNS (Setup):

WNS (Hold):

8. Running DRC and LVS

a. Next is to perform physical verification of the design. For this we need to run a DRC (design rule check) and the LVS (layout vs schematic check).

b. To run DRC check:

`run_magic_drc`

```
% run_magic_drc
[STEP 34]
[INFO]: Running Magic DRC (log: designs/multiplier/runs/RUN_2026.03.09_14.25.11/logs/signoff/34-drc.log)...
[INFO]: Converting Magic DRC database to various tool-readable formats...
[INFO]: No DRC violations after GDS streaming out.
%
```

If there are DRC errors, you may change the configuration setttings (e.g., adjust the utilization and density to be smaller). For example:

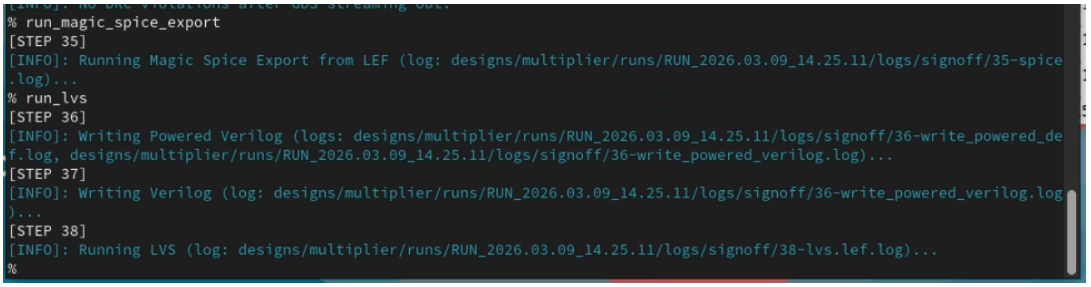
```
set ::env(FP_CORE_UTIL) 35
set ::env(PL_TARGET_DENSITY) 0.4
set ::env(FP_TAPCELL_DIST) 13
```

After applying these settings, you need to re-run the whole flow from synthesis to routing, and re-run **magic_drc**.

c. Once DRC is passed, we can now run LVS.

run_magic_spice_export

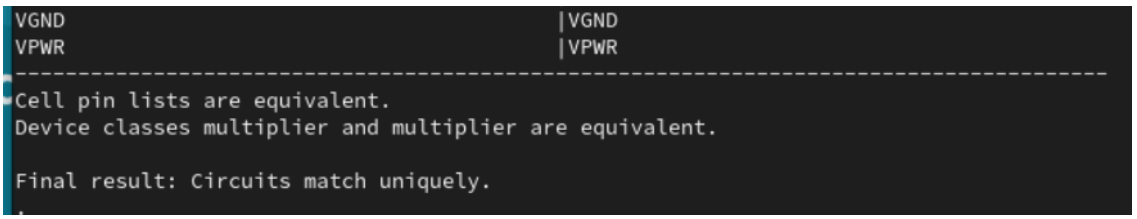
run_lvs



First command is to extract parasitics and netlist from the layout, and second command is to run the LVS check.

Check the report here if LVS is passed:

designs/multiplier/runs/RUN_XXX/logs/signoff/XX.lef.lvs.log



Look for the last line of the file, “**Circuits match uniquely**”