Tuesday Mar 18, 2025	EE 705: VLSI DESIGN LAB 2024-2025 ASSIGNMENT-4	Due On Mar 25, 2025

Note:

1. This assignment will be done in groups with one submission per group.
2. Submission is **only** through Moodle in the form of a PDF file upload.
3. All plots/screenshots must be legible/readable in the submission (axes values and units).

TA's: Ankur, Arpit and Lakshmi

Introduction:

In this assignment you will generate memory using OpenRAM compiler and perform RTL to GDS of the created memory in OpenLane based open-source IC design flow.

Instructions to login to the VLSI lab machine:

- Each group is assigned a group id for login. Refer to your login id from here.
- If you are on any unix based system, open a terminal for subsequent steps. If you are on Windows, download MobaXterm and open a terminal via MobaXterm. Ensure you are connected to IITB network.
- To login, type on the terminal: *ssh -X EE705_X@10.107.90.73* (X corresponds to the number from your login id assigned to your group). Enter the password and you are good. *Default password is EE705*.
- After the first login, type on the terminal: *passwd* and change your password immediately.

PART A: OpenRAM

1. Firstly, we will install the OpenRAM compiler. Kindly refer to the ***OpenRAM installation document***
2. Read about OpenRAM: Read OpenRAM.
3. Once installed, let's generate memory. For RISC-V we will be generating memories for ICACHE and DCACHE having the following specification.

   DCACHE SPECIFICATION: Data size is 32 bits
   Address size is 11 bits
   ICACHE SPECIFICATION: Data size is 20 bits
   Address size is 8 bits

4. Let's create a configuration file to generate the memory. An example is shown below:
   cd OpenRAM
   Make sure that conda is activated then type → gedit myconfig_sky130.py (you can use nano or any other editor as well)

```
(base) lakshmi_ubuntu@LAPTOP-NATIL47D:/mnt/c/Users/Iyer/OpenRAM$ gedit myconfig_sky130.py
```

   The contents of this file is mentioned in this myconfig_sky130.py
5. To run the compiler, go to OpenRAM folder. Activate the miniconda and then type the below command shown

```
(base) lakshmi_ubuntu@LAPTOP-NATIL47D:/mnt/c/Users/Iyer/OpenRAM$ python3 sram_compiler.py myconfig_sky130.py -k
```

6. Once finished, check the output folder to see the files generated.

Repeat the procedure for Dcache specification and generate the memory.

Your report should have the following:

1. OpenRAM execution on terminal screenshot along with output folder contents screenshot for both Icache and Dcache memories.

PART B : OpenLane

1. Once both memories are generated you will perform RTL to GDS flow. Login to your account.

```
13/03/2025  12:43.11  /home/mobaxterm  ssh -X EE705_34@10.107.90.73
Warning: Permanently added '10.107.90.73' (ED25519) to the list of known hosts.
EE705_34@10.107.90.73's password:
```

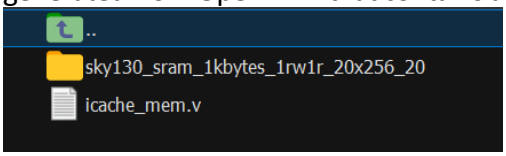
2. Type *ls* to list the files. You should see a folder named OpenLane. Go into the directory by typing the command *cd OpenLane*

3. Type *ls* to list the files. You will find a folder named designs. Create your folder inside it (say *icache_flow*: to create rtl-gds flow for the icache memory generated using OpenRAM)

```
[EE705_34@vlsi73 OpenLane]$ cd designs
[EE705_34@vlsi73 designs]$ mkdir icache_flow
```

4. You need to follow the procedure similar to EE671. The memory generated by the compiler will be used as a macro and needs to be instantiated in a top-level design.
   Refer to the tutorial: ***Hierarchical chip design (with macros)***

5. Inside the *src* folder you will have to create a top module (**say icache_mem.v**) and also you will upload the folder generated from OpenRAM that contains all the views generated for the memory.



6. Inside the OpenRAM generated sky130_sram_1kbytes_1rw1r_20x256_20.v file, add the line as shown.

```
// OpenRAM SRAM model
// Words: 256
// Word size: 20
/// sta-blackbox        <-- Must be added
module sky130_sram_1kbytes_1rw1r_20x256_20(
`ifdef USE_POWER_PINS
    vccd1,
    vssd1,
`endif
// Port 0: RW
    clk0,csb0,web0,addr0,din0,dout0,
// Port 1: R
    clk1,csb1,addr1,dout1
  );
```

7. Few points from the tutorial:
   (a) You will be instantiating your memory (say sky130_sram_1kbytes_1rw1r_20x256_20.v) ***only once*** inside the top module (**say icache_mem.v**) (Kindly refer to the part **Create the Verilog Files** mentioned here). Your top module must

look like the one given here.
Also note: You need to adjust the address and data size according to your design

```
sky130_sram_1kbyte_1rw1r_32x256_8 sram0(
    .clk0(clk),
    .csb0(!cs),
    .web0(!we),
    .wmask0(write_allow[3:0]),   <-- Not needed for your design
    .addr0(addr),
    .din0(datain[31:0]),
    .dout0(dataout_int[31:0]),

    .clk1(1'b0),
    .csb1(1'b1),
    .addr1(1'b0),
    .dout1(dout1[31:0])
);
```

Our design is also has a 1rw1r port, but we require 1rw port only, so we will also be disabling the 1r port section

(b) To create *config.json* file refer here.
Note: Remove the **"EXTRA_LIBS" line**.
Also, **You can change the "DIE_AREA":** "0 0 500 500" **(You can change this as per your macro size (which will be mentioned in the** sky130_sram_1kbytes_1rw1r_20x256_20.lef **)**

(c) To create *macro_placement.cfg* file refer here.

8. Once all files are ready, you can run the rtl-gds flow in Openlane. We use a docker based environment to run OpenLane. To enter the docker container, type *make mount* on the terminal

```
[EE705_34@vlsi73 OpenLane]$ make mount
```

9. You are now inside the docker container. From this container, we will run all the tcl scripts. To call the OpenLane flow, we need to run the flow.tcl script with some arguments. On the container terminal, type *./flow.tcl -design icache_flow -tag full_guide -interactive -overwrite* . (See Point 3 to check the design name)

```
OpenLane Container (1.1.1):/openlane% ./flow.tcl -design icache_flow -tag full_guide -interactive -overwrite
```

10. Now, run the following commands in the openlane container one by one.
    a. **run_synthesis**
    b. **run_floorplan**
    c. **run_placement**
    d. **run_cts**
    e. **run_routing**
    f. **run_parasitics_sta**
    g. **run_magic**
    h. **run_magic_spice_export**
    i. **run_magic_drc**
    j. **run_lvs**
    k. **run_antenna_check**

    Exit the container after running these steps and go to the "runs" folder to see the output files

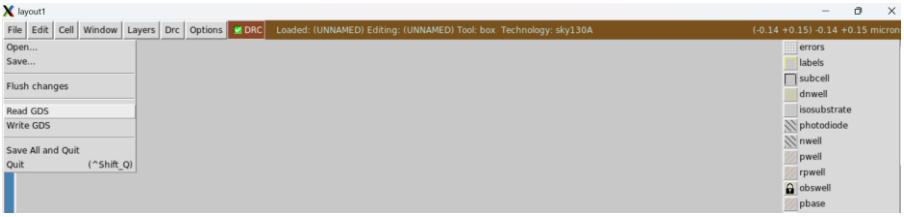
11. To view the layout, open magic tool

    First type on terminal : **source ~/.bashrc**

Then invoke magic

```
[EE705_34@vlsi73 OpenLane]$ magic
```

12. Read the gds file: (present inside /runs/full_guide/results/signoff)



13. You will notice that there are drc errors (which are expected). Once the file is loaded in magic, at the top panel go to *Options→Cell manager*. You should be bale to view your design name. Click on *Expand .*

Your report should have the following (For both the Icache and Dcache) :

1. Go to reports/signoff and paste the screenshot of .lvs.rpt file.
2. Top level module in which you have instantiated the macros
3. Config.json and macro_placement.cfg file screenshot
4. Layout of the design on Magic screenshot.
5. Go to the logs/routing folder, open the design_sta.log and fill up the table below:

Clock Frequency (MHz)	
Worst case setup slack (ns)	
Worst case hold slack (ns)	
Design area (μm^2)	
Total Power Consumption (µW)	