CS6135 VLSI Physical Design Automation

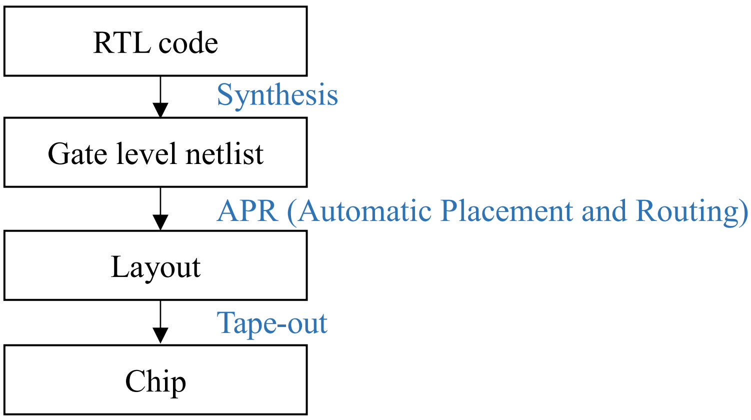
Homework 1: P&R Tool

Due: 23:59, October 17, 2021

1. Goal

In this homework, you are asked to use Synopsys IC Compiler to complete the P&R (Place and Route) flow for a given synthesized standard-cell design. The goal is to acquaint you with IC Compiler and the P&R flow. Besides, you are encouraged to try to optimize timing, total area, and total wirelength without violating any timing or DRC constraints.

In a clear way, we want you to utilize a P&R tool (Synopsys IC Compiler) to do APR to generate a layout.



2. Working Items

- Follow the procedures described in Section 6 step by step to get familiar with IC Compiler.
- Try your best to use IC Compiler for optimizing the timing (clock period in the SDC file) and the area (try to adjust the core utilization in Design Planning stage) of the circuit. (Hint: You need to adjust the value of the clock period (2x and x) in the .sdc file to optimize the timing)
    - `create_clock [get_ports CLK] -period 2x -waveform {0 x}`
- Try to switch the timing-driven option and congestion-driven option on and off, respectively, when placing standard cells to see how they affect the result. Please also try to explain the difference(s) between the timing-driven placement and congestion-driven placement in your submitted report.
- Please explain why we need to insert filler cells into the design.

- Show the information of clock period (.sdc), slack (timing.report), core area (area.report), and total wirelength (route.report) in the report.
- Make sure that the slack must be non-negative and the number of DRC violations must be zero for your physical implementation to be successful.

3. Report

Your report should contain the following contents, and you can add as more as you wish.

(1) Your name and student ID

(2) A comparison table like the following one, and an explanation of the result (The table should be built under a fixed core utilization and clock period and you should specify them in the report.)

	(congestion-driven, timing-driven)			
	(off, off)	(on, off)	(off, on)	(on, on)
slack	???	???	???	???
total cell area	???	???	???	???
total wirelength	???	???	???	???

(3) The difference(s) between the congestion-driven placement and timing-driven placement

(4) An explanation of why we insert filler cells

(5) Show your best result (including clock period, core area, total wirelength, slack, congestion- and timing-driven on/off setting, and their snapshots) to maintain a non-negative slack and no DRC violation. You can use the following equation to evaluate your results and choose the result with the smallest score as your best result.

$$\text{score} = \frac{\text{clock period}}{8} + \frac{\text{core area}}{74610} + \frac{\text{total wirelength}}{221415}$$

4. The File to be Handed in

Please package all the following items in one file named `CS6135_HW1_{STUDENT_ID}.zip` and submit it to eeclass.

(1) `CS6135_HW1_{STUDENT_ID}.tar.gz`

➢ An archive containing your post P&R design

```
$ tar -zcvf CS6135_HW1_{STUDENT_ID}.tar.gz HW1/
```

(2) `CS6135_HW1_{STUDENT_ID}_report.pdf`

➢ Your report

(3) `CS6135_HW1_{STUDENT_ID}_chip.jpg` (or .png)

➢ The final chip layout of your best result generated by IC Compiler (use print-screen)

5. Grading

✓ The completeness of your submitted report

✓ The quality (clock period, core area, and total wirelength) of your physical implementation

6. Procedures

Please follow the following steps to complete your physical implementation of the given design. Note that each figure shown below is just an example, so your result may not be the same as it.

To follow the procedures, you'd better have a basic understanding of CLI (command-line interface). If you are not familiar with how to operate shell and vim, please study the "Shell Tutorial.pdf" that has been uploaded to eeclass. If you have any additional problems, please contact TA.

A. Design Preparation

Step 1. Log in to the workstation (via Xming or X-Win32, using MobaXterm for ease)

Please download `HW1.tar.gz` from eeclass. Then, upload the tar file to the server nthucad.cs.nthu.edu.tw.

```
$ scp HW1.tar.gz {YOUR_ACCOUNT}@nthucad.cs.nthu.edu.tw:.
```

(`$` is a prompt, not a command, so don't type it. `scp` stands for secure copy.)

Next, log in to the server nthucad.cs.nthu.edu.tw (by using the command `ssh` in some sort of command windows such as putty, Cygwin, MobaXterm, terminal.app, etc), and then change your password by entering the command `yppasswd`.

(`ssh` stands for secure shell. `-XY` means to enable X11 forwarding to see GUI on Linux.)

```
$ ssh -XY {YOUR_ACCOUNT}@nthucad.cs.nthu.edu.tw
$ yppasswd
```

The server nthucad.cs.nthu.edu.tw is just a proxy server (or a so-called relay server), which is used to connect other servers on a private network. That means it only contains some basic services, so you need to log in to another workstation with Linux system (ic51, ic53, ic54) to run IC Compiler, e.g.,

```
$ ssh -XY ic51
```

Also, you could enter `lab_uptime` to see the loading information of each connectable server. You could choose the server which has fewer active users to log in.

```
nthucad:~> lab_uptime
--------users---load average-----------------users---load average---
ic27 (l): 0   0.02, 0.01, 0.00   ||   ic28 (l): 4   0.00, 0.01, 0.00
ic29 (l): 0   0.00, 0.00, 0.00   ||   ic30 (l): 4   0.00, 0.01, 0.00
ic51 (l): 2   1.22, 1.22, 1.23   ||   ic53 (l): 1   0.00, 0.01, 0.05
ic54 (l): 0   0.00, 0.01, 0.05   ||   ic55 (l): 0   0.00, 0.03, 0.00
ic56 (l): 0   0.00, 0.00, 0.00   ||   ic57 (l): 1   0.00, 0.00, 0.00
ic58 (l): 1   0.00, 0.00, 0.00   ||
last updated: 公曆 20廿一年 九月 廿八日 週二 廿時36分一秒
(l) Linux, (s) SunOS, (D) Shutdown
```

Step 2. Set up environment and invoke IC Compiler

```
$ source /tools/linux/synopsys/CIC/icc.cshrc # set ICC Env. Var.
$ tar -zxvf HW1.tar.gz                       # untar the archive
$ cd HW1/run/
```

You can take a look at "`.synopsys_dc.setup`" file in "`run/`" directory. Logic and timing libraries are specified in that file. The file is used to run logic synthesis on your RTL code, and we've done that for you, so you don't need to worry about it.

Before invoking IC Compiler with GUI, you might need to set the environment variable "DISPLAY" to your IP address and also make sure that you have already installed X server such as Xming or X-win32 in your computer. (Hint: Use MobaXterm in Windows or XQuartz in OS X at the beginning, and then you won't worry about it.)

```
$ icc_shell -gui                             # invoke icc in gui mode
```

(If you didn't set up a X server on your PC and forward X11 to the workstation properly, you will get the following error.)

```
Initializing...
Initializing gui preferences from file  /users/student/          /.synopsys_icc_prefs.tcl
Cannot initialize GUI.
Please set the DISPLAY environment variable before starting the application.
For this session, the tcl command 'setenv' can be used to set the value of the DISPLAY variable,
then the command 'gui_start' is used to start the GUI.
```

B. Design Setup

First, we need to create a library, import the given design and read in all required files. Each time you enter a command, it's very important for you to check the messages on the console to see whether it succeeded or failed.

Step 1. Create a new library

MainWindow > File > Create Library…

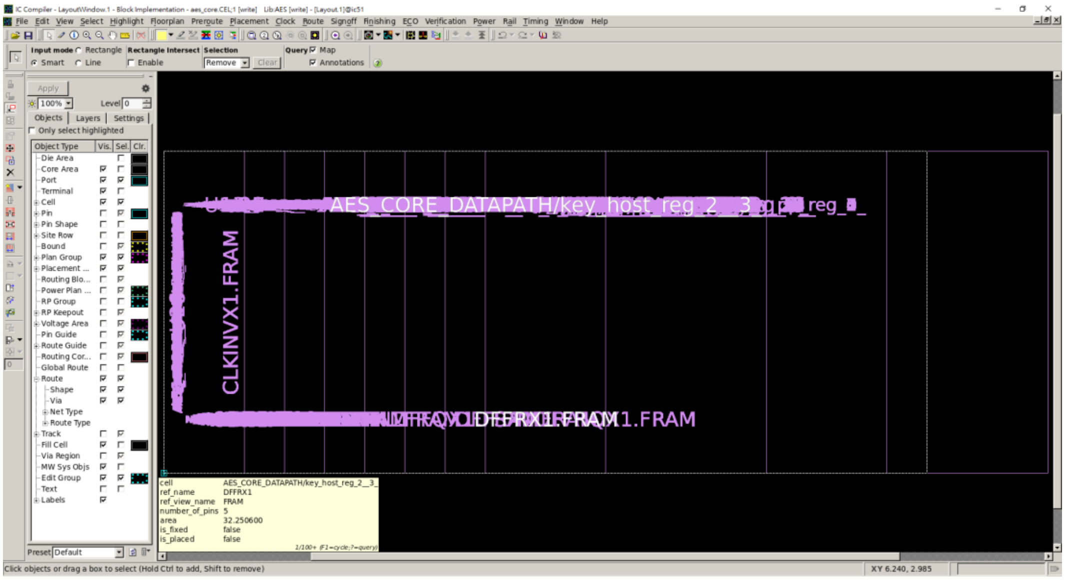
New library name	AES
Technology file	../tech/tsmc13_CIC.tf
Input reference libraries	../ref_lib/tsmc13gfsg_fram
Bus naming style	[%d]
Open library	Enable

Step 2. Import Design

MainWindow > File > Import Design…

Input format	verilog
Import verilog files	../design_data/aes_core_syn.v
Top design name	aes_core

Then it will open a layout window like the following figure.



Step 3. Set TLU+ RC model

MainWindow > File > Set TLU+…

Max TLU+ file	../tluplus/t013s8mg_fsg_typical.tluplus
Layer name mapping file between technology library and ITF file	../tluplus/t013s8mg_fsg.map

Step 4. Read in SDC file

MainWindow > File > Import > Read SDC…

Input file name	../design_data/aes_core_SYN.sdc
Version	Latest
Other	Default value

Step 5. Save design

MainWindow > File > Save Design… > Save All

MainWindow > File > Save Design…

Show advanced options	Enable
Save As	Enable
Save As Name	design_setup

P.S. Any time you want to restore a design

MainWindow > File > Open Library…

Library name	AES

MainWindow > File > Open Design > Your Save Design > Click OK

C. Design Planning

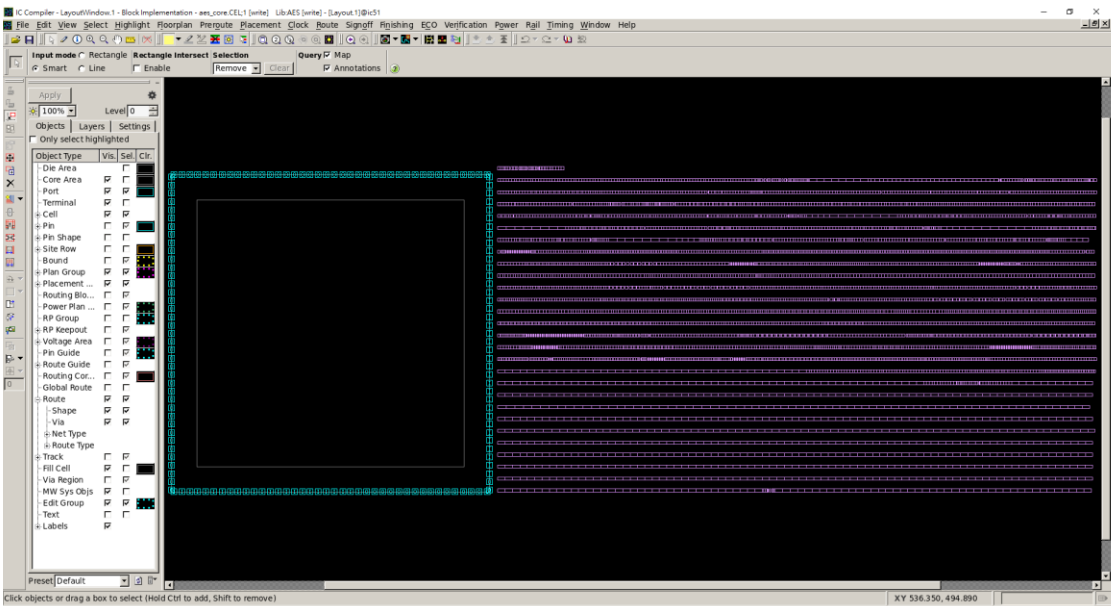
Next, we specify a region and put our design in it.

Step 1. Initialize floorplan

LayoutWindow > Floorplan > Create Floorplan…

Control type	Aspect ratio
Core utilization	0.7 (Depends on you, 0 < utilization < 1)
Aspect ratio (H/W)	1.0
Horizontal row	Enable
Double back	Enable
Start first row	Disable
Flip first row	Enable
Space between core area and terminals (pads)	Left: 30
	Right: 30
	Bottom: 30
	Top: 30

The initial floorplan should be like the following figure.



The square on the left is the core area which you will put all your standard cells in. All the standard cells are on the right-hand side.

Step 2. Save design

LayoutWindow > File > Save Design… > $\boxed{\text{Save All}}$

LayoutWindow > File > Save Design…

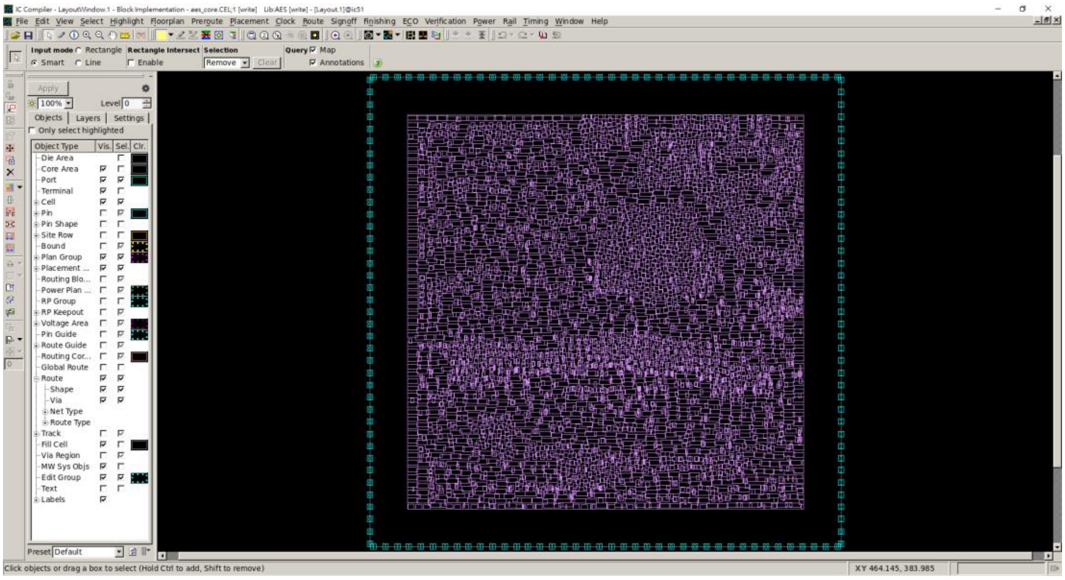
Show advanced options	Enable
Save As	Enable
Save As Name	die_init

Step 3. Global Placement and Legalization

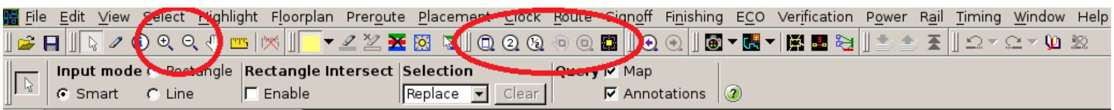
LayoutWindow > Placement > Coarse Placement…

Effort	Depends on you
Perform congestion driven placement	Depends on you
Perform timing driven placement	Depends on you
Others	Default

The following snapshot shows an example result of a finished global placement. As you can see, all the cells are moved into the core. However, you could see that most of them are overlapping with one another right now.



You can use the zooming utility to observe the layout.

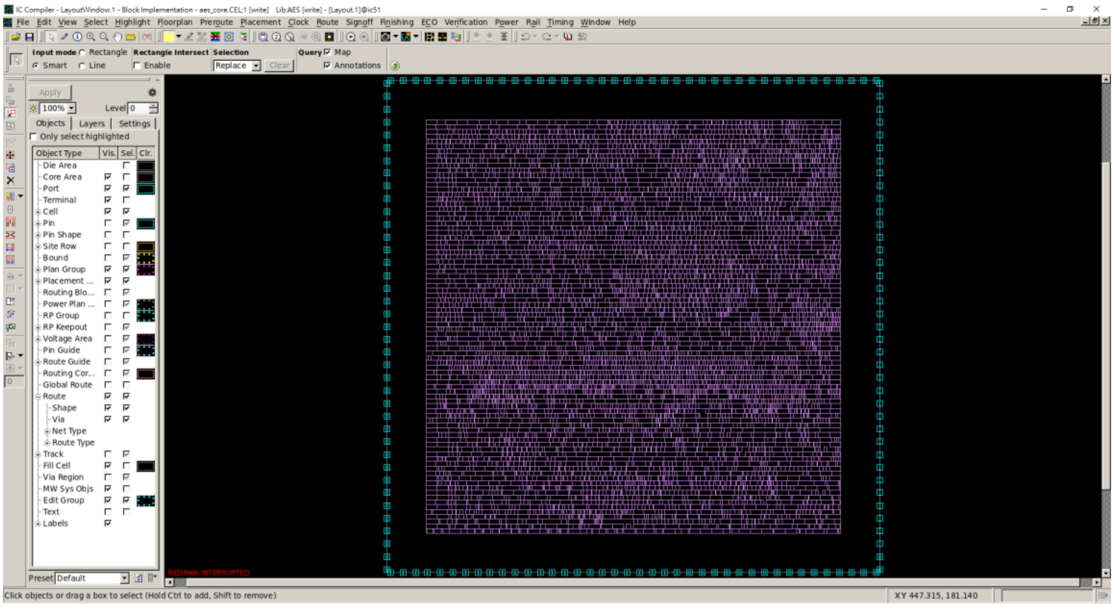


Now, since most of the cells are overlapping with one another, we need to do legalization to legalize the placement. (i.e., to remove all overlaps and move cells into rows.)

LayoutWindow > Placement > Legalize Placement…

Effort	Depends on you
Incremental	Enable

By now, you could see that all overlaps are resolved.



Step 4. Tie cell connection

This step is to connect the logic 1 to VDD, logic 0 to VSS and connect P/G nets of each standard cell together.

LayoutWindow > Preroute > Derive PG Connection…

Manual connection	Selected
Power net	VDD
Ground net	VSS
Power pin	VDD
Ground pin	VSS
Create port	Top
Other	Default value

Next, we need to build a power network, and then we can analyze the IR drop.

Step 5. Synthesize power network

LayoutWindow > Preroute > Power Network Constraints > Strap Layers Constraints…

layer	**METAL7**
Direction	Horizontal
Density (By strap number)	Max: 10, Min: 5
Width	Max: 5, Min: 2
PG spacing	Interleaving

Click $\boxed{\text{Set}}$

layer	**METAL6**
Direction	Vertical
Density (By strap number)	Max: 10, Min: 5
Width	Max: 5, Min: 2
PG spacing	Interleaving

Click Set and Close

LayoutWindow > Preroute > Power Network Constraints > Ring Constraints…

Power Ground nets	**VDD VSS**
Horizontal layers	**METAL7**
Vertical layers	**METAL6**
Ring width (enable)	Fixed: 2
Extend straps to	Core ring

Click Set and Close

LayoutWindow > Preroute > Create Virtual Power Pads…

Power or Ground net	**VDD**
Layer	Specified: **METAL7**

Click left mouse button at the left boundary and right boundary of the core.

Power or Ground net	**VSS**
Layer	Specified: **METAL7**

Click left mouse button at the left boundary and right boundary of the core.

Power or Ground net	**VDD**
Layer	Specified: **METAL6**

Click left mouse button at the left boundary and right boundary of the core.

Power or Ground net	**VSS**
Layer	Specified: **METAL6**

Click left mouse button at the left boundary and right boundary of the core.

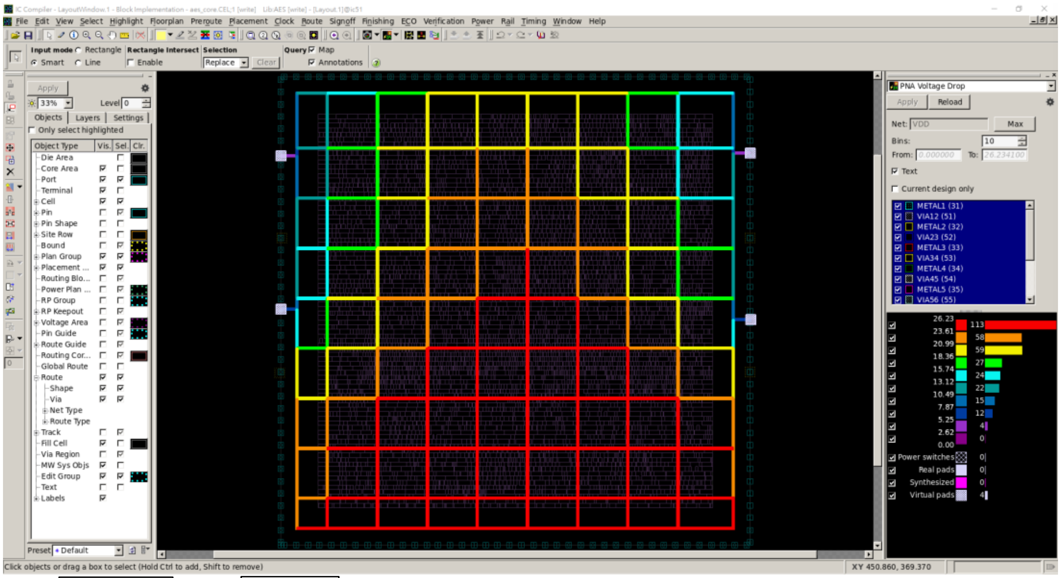
Click Close

LayoutWindow > Preroute > Synthesize Power Network…

By power network by nets	Selected
Nets	**VDD VSS**
Supply voltage	1.5
Target IP drop	10% of supply voltage
Power budget (mW)	100
Other	Default value

Click Apply and observe the IR drop map.

An example is shown in the following figure. Notice that if IR drop is too high, it might functionally fail. (E.g., 1V drops to 0.4V, so logic 1 will be considered as logic 0.) However, here we just built virtual P/G and let you know what IR drop is, so it's OK if you saw some of them are too high.



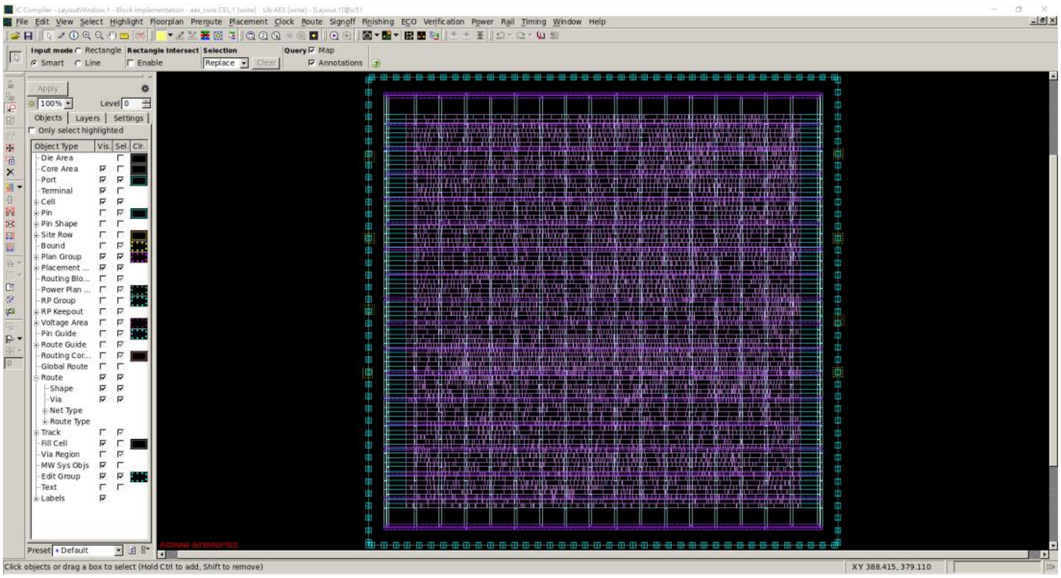
Click Commit and Cancel

By now, we have connected power rails to the P/G ring. Please zoom in the layout to see the VDD/VSS power rails.

LayoutWindow > Preroute > Preroute Standard Cells…

Routing Options	Selected
Extend for multiple connections(enable)	To connections within: 10
Keep floating rail segments	Disable
Do not route over macro cells	Enable
Fill empty rows	Enable

You can see the power network as follows.



Step 6. Save design

LayoutWindow > File > Save Design… > Save All

LayoutWindow > File > Save Design…

Show advanced options	Enable
Save As	Enable
Save As Name	design_planning

D. Placement Optimization (Detailed Placement)

In this step, you need to perform placement optimization to optimize the placement based on the power network that you just built. (Basically, it will take some time and try to swap or move some cells to get a better timing and quality.)

Step 1. Placement

LayoutWindow > Placement > Core Placement and Optimization…

Effort	Depends on you
Power Optimization	Depends on you
Clock compilation, optimization and routing	Depends on you

Step 2. Save design

LayoutWindow > File > Save Design… > Save All

LayoutWindow > File > Save Design…

Show advanced options	Enable
Save As	Enable
Save As Name	placement

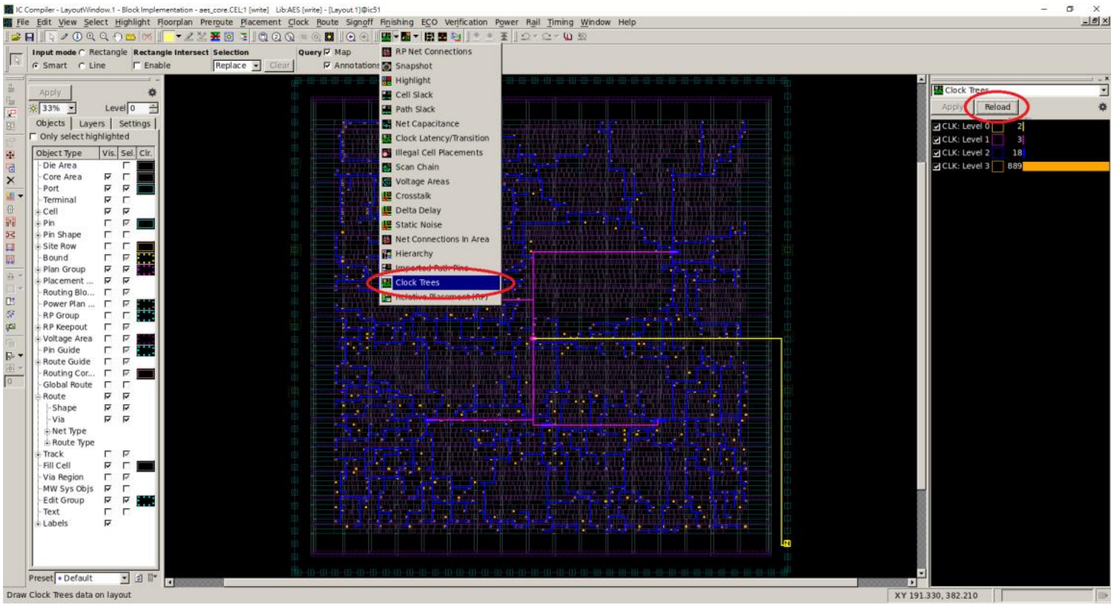
E. Clock Tree Synthesis (CTS)

Step 1. Set constraints and synthesize clock tree without routing

(Return value = 1 means success.)

```
icc_shell> set_fix_hold [all_clocks]
icc_shell> set_max_area 0
icc_shell> set physopt_area_critical_range 0.1
icc_shell> clock_opt -fix_hold_all_clocks -no_clock_route
```

Please check the clock tree topology. Basically, in this step, the tool will try to insert buffers to meet skew, latency and transition goals. (i.e., to solve potential timing problems.)



Step 2. Save design

LayoutWindow > File > Save Design… > $\boxed{\text{Save All}}$

LayoutWindow > File > Save Design…

Show advanced options	Enable
Save As	Enable
Save As Name	cts

F. Route (Clock Nets and Signal Nets)

Step 1. Analyze timing (setup and hold) and check routability

`icc_shell> report_timing`

Is there any timing violation (slack < 0)?

(slack = required time - arrival time. If slack < 0, it is failed.)

P.S. If it reports “No paths,” it might be due to a critical path was broken by an ICC utility called “dynamically break timing loop” during placement optimization. Thus, you need to restore your design_planning and redo placement optimization again. (For more information, please check http://bbs.eetop.cn/viewthread.php?tid=438372)

LayoutWindow > Route > Check Routability…

Is there any error?

Step 2. Use Zroute Mode to connect all clock and signal nets

```
icc_shell> route_zrt_group -all_clock_nets
icc_shell> route_zrt_auto
```

Step 3. Optimize routing results

`icc_shell> report_timing`

Is there any timing violation?

`icc_shell> verify_zrt_route`

Is there DRC violation (Total number of DRCs > 0)?

(DRC means Design Rule Check; if DRC > 0, it means some rules are not satisfied.)

P.S. If you have timing or DRC violations, you can use the following commands to fix them. If not, don't do it.

If you only have DRC problem:

```
icc_shell> route_zrt_detail -incremental true \
            -initial_drc_from_input true
```

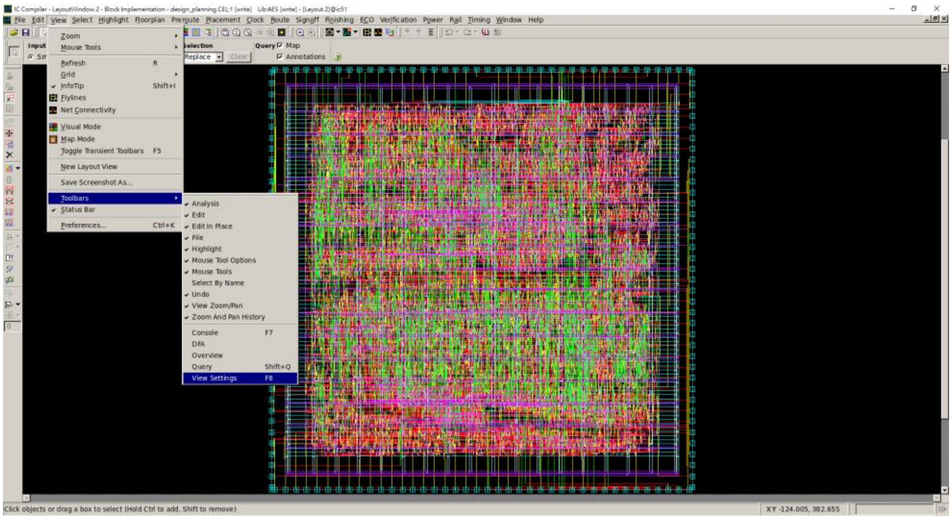
Not only DRC but also timing:

`icc_shell> route_opt -incremental`

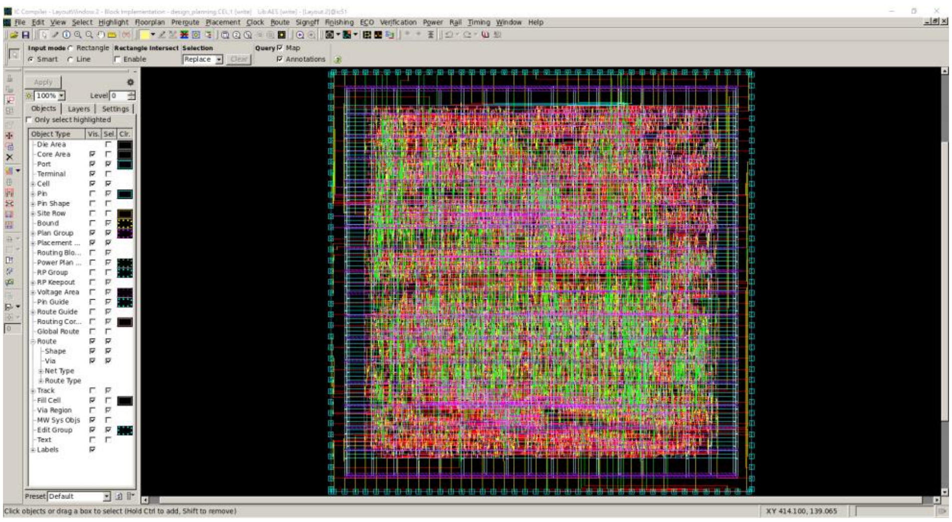
If it has serious violations:

`icc_shell> route_opt`

Open view settings to toggle which object(s) you want to highlight.



You can see the routing results as follows.



Step 4. Save design

LayoutWindow > File > Save Design… > Save All

LayoutWindow > File > Save Design…

Show advanced options	Enable
Save As	Enable
Save As Name	route

G. Insert Filler Cells

(Why do we insert filler cells? Please google it and write a reasonable explanation in your report.)

Step 1. Insert standard cell filler and connect P/G nets of cells

```
icc_shell> insert_stdcell_filler -cell_without_metal \
           "FILL64 FILL32 FILL16 FILL8 FILL4 FILL2 FILL1" \
           -connect_to_power {VDD} -connect_to_ground {VSS}
```

Step 2. Optimization

```
icc_shell> report_timing > timing.report
icc_shell> verify_zrt_route > route.report
```

(You always could change the .report file name as long as you remember to save your best result as the given name.)

If you have timing or DRC violations, you can use the following command to fix them and then report timing and verify the routing result again. If not, don't do it.

```
icc_shell> route_opt -incremental
```

Step 3. Save design

LayoutWindow > File > Save Design… > Save All

LayoutWindow > File > Save Design…

Show advanced options	Enable
Save As	Enable
Save As Name	post_route

Step 4. Report core area

```
icc_shell> report_area -physical > area.report
```

Step 5. Take a snapshot of your final layout

Save it as a graphic file "`CS6135_HW1_{STUDENT_ID}_chip.jpg`"

Step 6. Exit

```
icc_shell> exit
```

P.S. Using tcl

When the last time you exit the icc_shell, all your GUI operations are translated as commands and are stored in the file “command.log.” You could copy it into “apr.tcl” and modify it as you wish. Next time you can just enter source apr.tcl in icc_shell to complete the whole process.

```
icc_shell> source apr.tcl
```

P.S. If you have further interests in ICC or any tools related to IC design, you could consider taking TSRI training courses to have a better understanding of them. The link is: https://www.tsri.org.tw/main.jsp