National Tsing Hua University
Department of Electrical Engineering
EE429200 IC Design Laboratory, Fall 2021

Lab 12: APR Flow with IC Compiler
(CTS, Routing, Verification and Power Analysis)
Assigned on Dec. 2, 2021
Due day in Dec. 9, 2021

Objective

In this lab, you will learn:

1. How to use IC Compiler to perform a basic APR flow
2. Post-layout simulation
3. Power analysis with PrimeTime

Demo checklist:

- ☐ Acceptable DRC errors & LVS clean
- ☐ Post-layout simulation result
- ☐ Power, timing, utilization result with your new utilization APR flow
- ☐ Compare your power-estimation result between ICC and Primetime
- ☐ Questions:
  1. Why should we use tie cells to connect constant pins? Can we directly connect constant pins to power/ground ports?
  2. What's the function of corefiller?

Action Items

I. Clock Tree Synthesis.

1. In the ***icc_run/***, invoke the GUI of ICC by

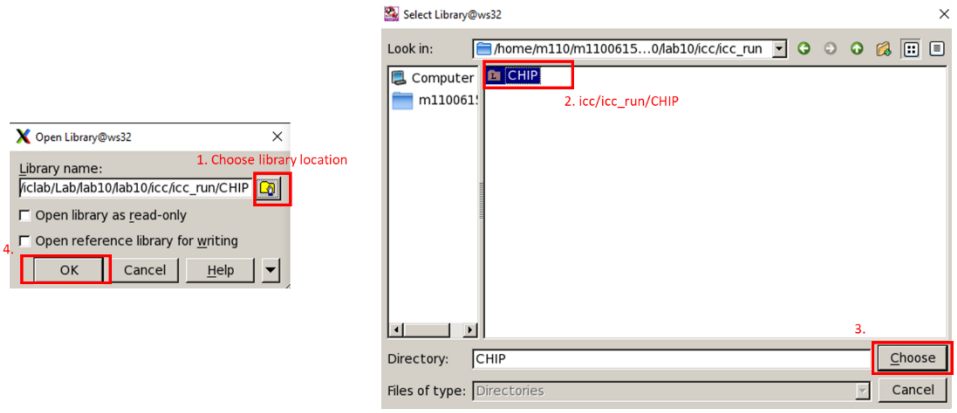
   ```
   $ icc_shell -64 -gui
   ```

   Set link library and target library:

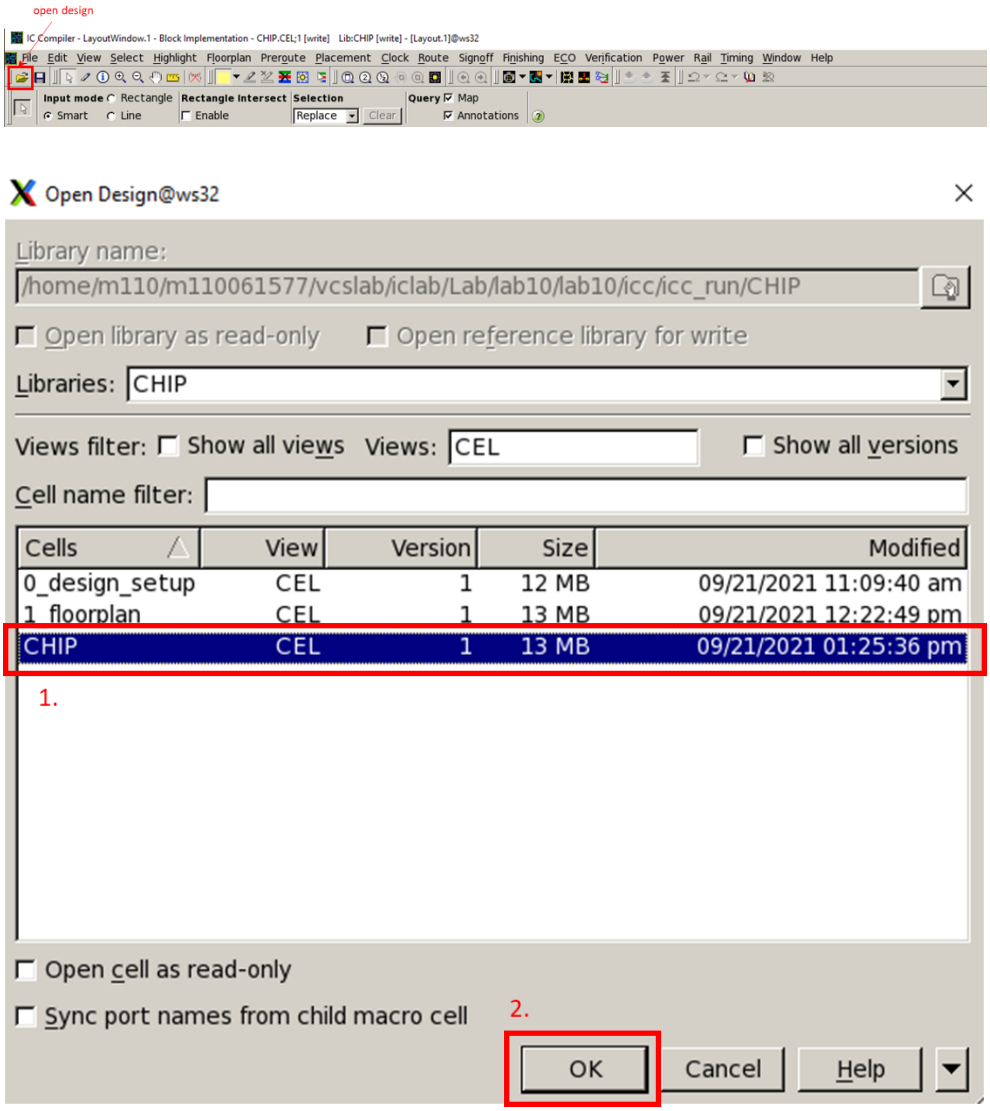
   ```
   $ source setup.tcl
   ```

2. Open the library and design from the previous step.

   "File->Open Library"



Click open design icon, then choose open design “CHIP”.



3. Execute the command. Since it has been reset to false when ICC is initialized.

```
$ set power_cg_auto_identify true
```

4. Add tie cells for constant signals 1'b1 (tie high) and 1'b0 (tie low) by executing:

   $ source -echo ../pre_layout/design_data/add_tie.tcl

   How many tie-high and tie-low cells are instantiated?

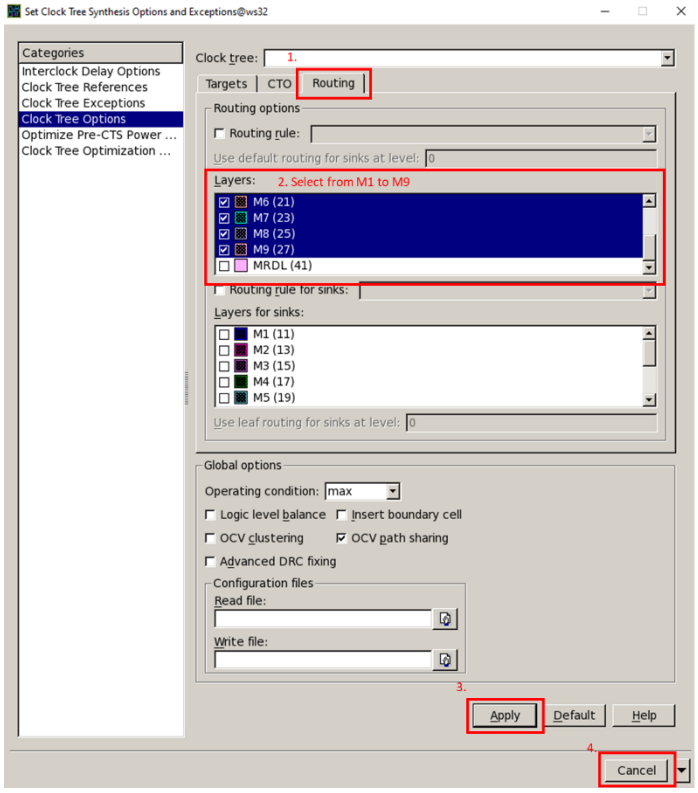
5. Identify clock gating by executing

   $ identify_clock_gating

6. Set available metal layer.

   "Clock->Set Clock Tree Options…"

   Choose Routing tab and select the metal layers from **M1** to **M9 (exclude MRDL)**. Press **"Apply"** and then **"Cancel"** to leave.



7. Turn on hold time fixing by executing

   $ set_fix_hold [all_clocks]

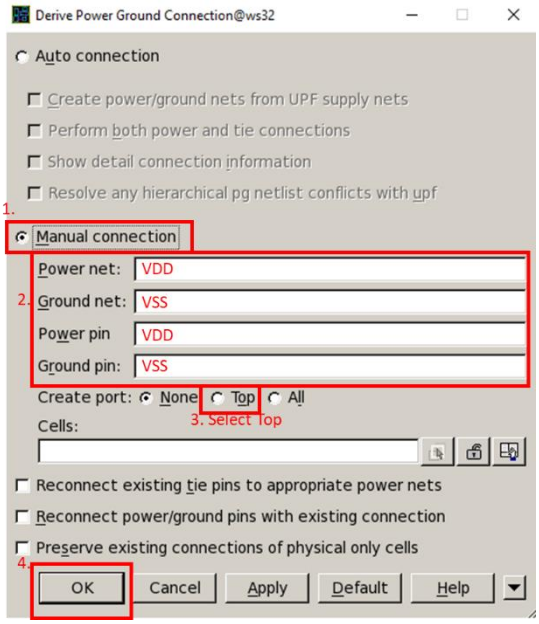
8. CTS optimization by executing. This step may take a bit longer.

   $ clock_opt -fix_hold_all_clocks -no_clock_route

9. Power/Ground connection.

   "Preroute->Derive PG Connection" (in Layout window)

Item	Content
Manual connection	Selected
Power net	VDD
Ground net	VSS
Power pin	VDD
Ground pin	VSS
Create port	Top



Note: This step **needs to be re-executed** whenever power nets are modified or new cells are added.

10. Check Timing:

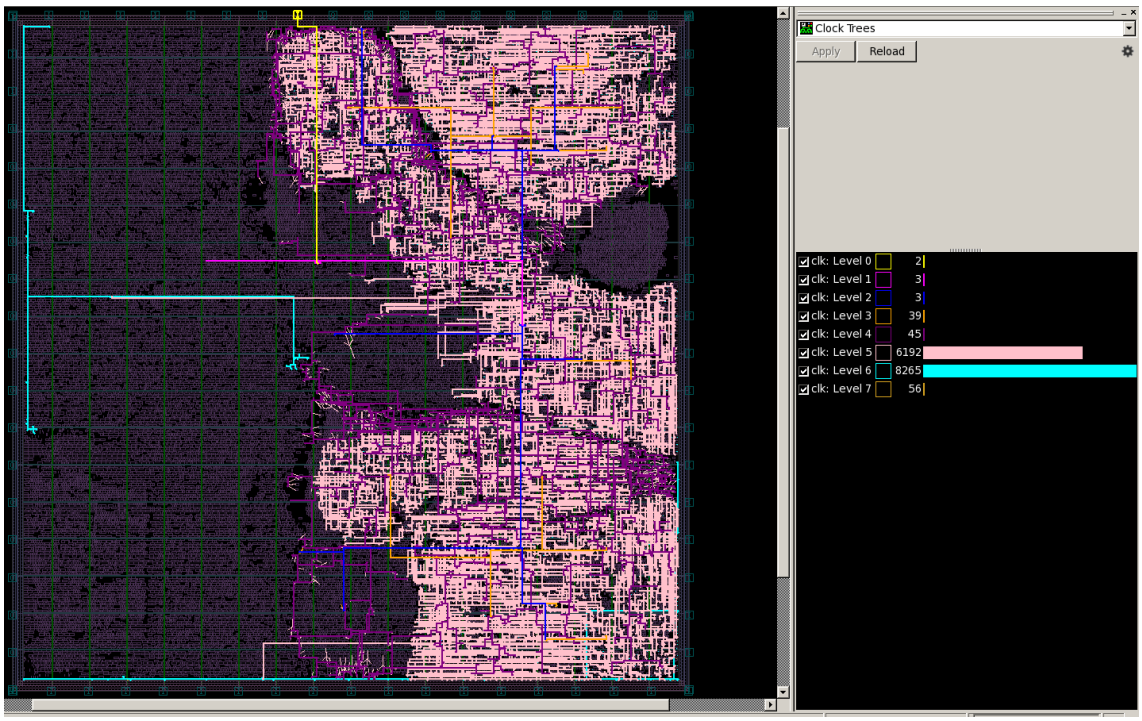
    Check setup time: $ report_timing

    Check hold time: $ report_timing -delay_type min

    Note: The operating conditions and libraries are **different** for setup and hold time checking.

11. View the clock tree by selecting **View->Visual Mode** in the Layout Window.

    Choose the option **Clock Trees** in the menu on the right. Press **Reload** button to select how many levels of the clock tree you want to see and click **Apply** to execute.



12. Save your design.

    "File->Save Design" and click **"Save all"**

    Or you can save your design with command line interface.

    $ save_mw_cel CHIP

13. Save your design as new file for backup.

    "File->Save Design" and check **"Show advanced options"**.

Item	Content
Save as	Enable
Save as name	4_cts

    Or you can save your design with command line interface.

    $ save_mw_cel -as 4_cts

II. Routing

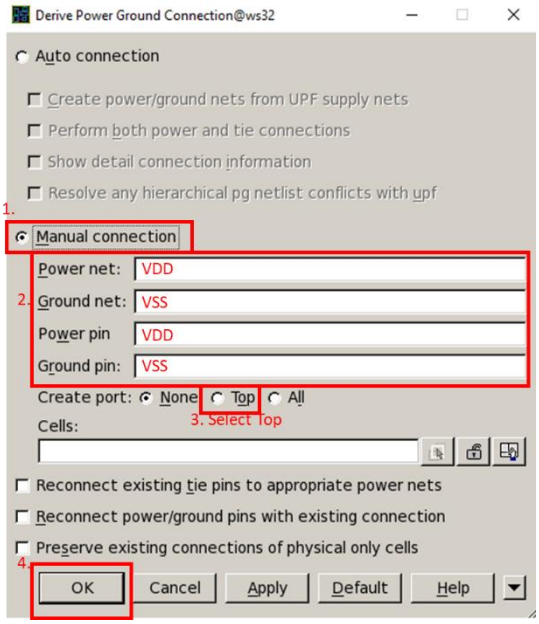
1. Check routability of the design by selecting

   "Route->Check Routability…" and pressing **"OK".** Make sure there is no error.

2. Special routing for the Power/Ground nets of the standard cells. (Note that the cells won't be modified from now on.)

“Preroute->Derive PG Connection” (in Layout window)

Item	Content
Manual connection	Selected
Power net	VDD
Ground net	VSS
Power pin	VDD
Ground pin	VSS
Create port	Top

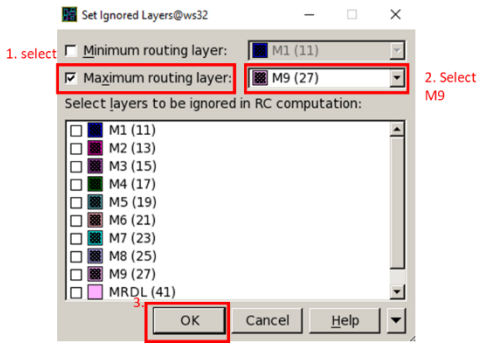


Note: This step **needs to be re-executed** whenever power nets are modified or new cells are added.

3. Routing setup.

(1) **“Route->Routing Setup->Set Ignored Layer”**

Item	Content
Maximum routing layer (selected)	M9 (avoid using MRDL)



(2) Fix antenna violations.

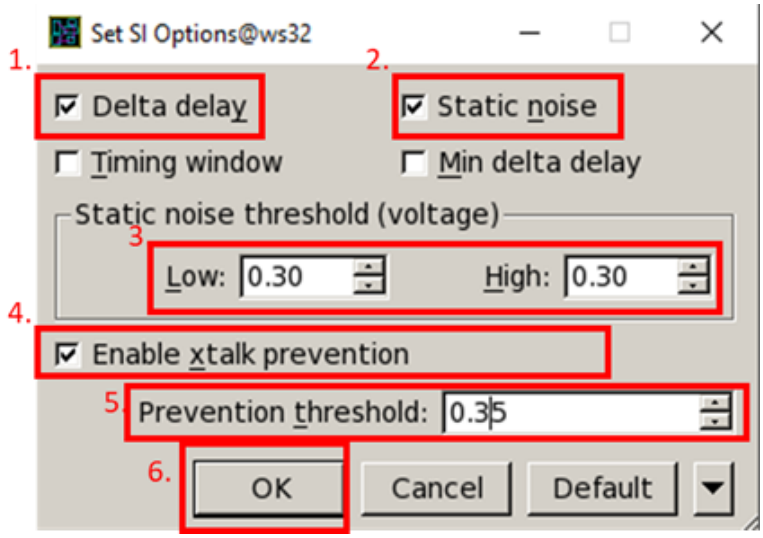
Define library-specific antenna rules by executing (only one command)

```
$ source -echo
/usr/cadtool/cad/synopsys/SAED32_EDK/tech/milkyway/saed32nm_ant_1p9m.tcl
```

(3) Signal integrity setting.

"Timing->Set SI Options"

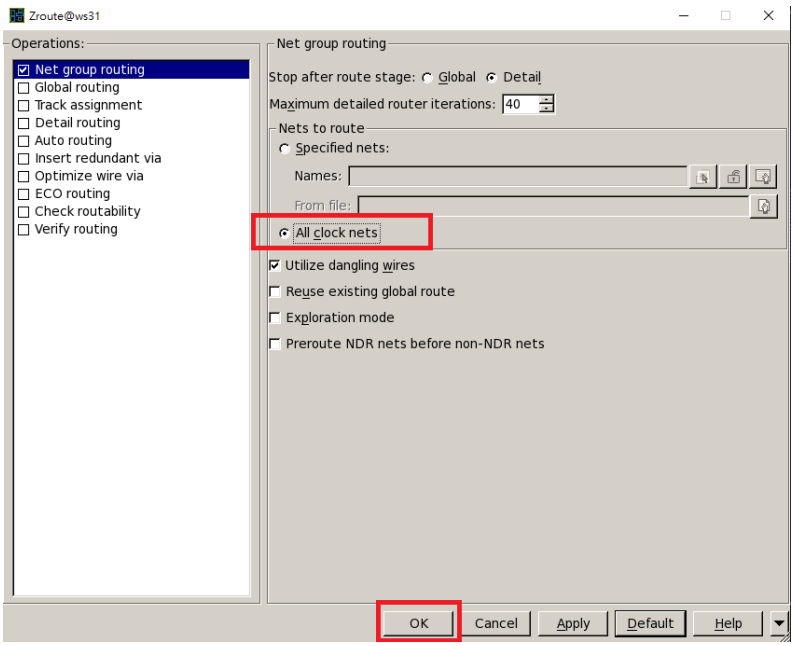
Item	**Content**
Delta delay	Enable
Static noise	Enable
Static noise threshold - Low	0.30
Static noise threshold - High	0.30
Enable xtalk prevention	Enable
Prevention threshold	0.35



4. Clock route.

"Route->Net Group Route"

Item	**Content**
Net to Route	All clock nets



When finished, make sure **0 DRC error** and you can check the layout for the clock routing.

5. Perform auto routing.

   "Route>Auto Route…" -> "OK"

6. Detail route by executing

   $ route_opt -stage detail -xtalk_reduction

When finished, make sure **0 DRC error** and you can check the layout for the signal routing.

7. Report the timing and power.

   $ report_timing

   $ report_timing -delay_type min

   $ report_power

   If timing is not met keep, execute the following command

   $ route_opt -incremental

   Answer the following questions:

   (1) Is there any negative slack for setup time? If so, is this acceptable? Why? -

   (2) Is there any negative slack for hold time? If so, is this acceptable? Why?

   (3) Compare the power numbers estimated in the placement stage.

Total power: _____(mW)

Cell Leakage Power: ____(mW)

Total Dynamic Power: _____(mW)

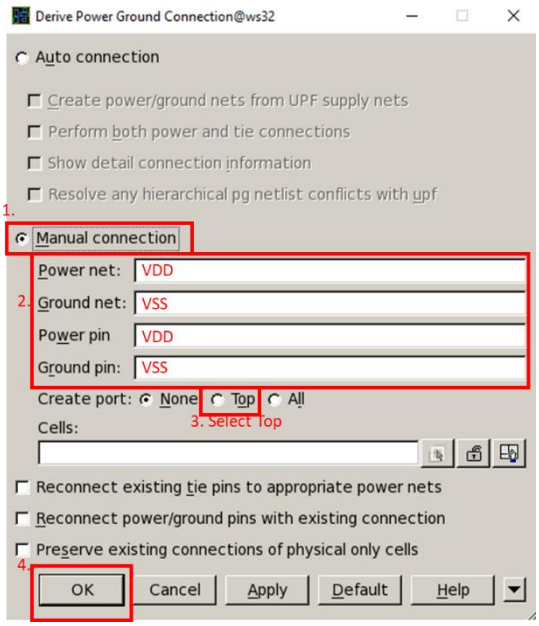
Clock network consumes _____% of the total power.

	Routing	Placement
Total Power		
Cell Leakage Power		
Total Dynamic Power		

8. Power/Ground connection.:

"Preroute->Derive PG Connection" (in Layout window)

Item	**Content**
Manual connection	Selected
Power net	VDD
Ground net	VSS
Power pin	VDD
Ground pin	VSS
Create port	Top



Note: This step **needs to be re-executed** whenever power nets are modified or new cells are added.

9. Save your design.

“File->Save Design” and click **“Save all”**

Or you can save your design with command line interface.

```
$ save_mw_cel CHIP
```

10. Save your design as new file for backup.

“File->Save Design” and check **“Show advanced options”**.

Item	Content
Save as	Enable
Save as name	5_route

Or you can save your design with command line interface.

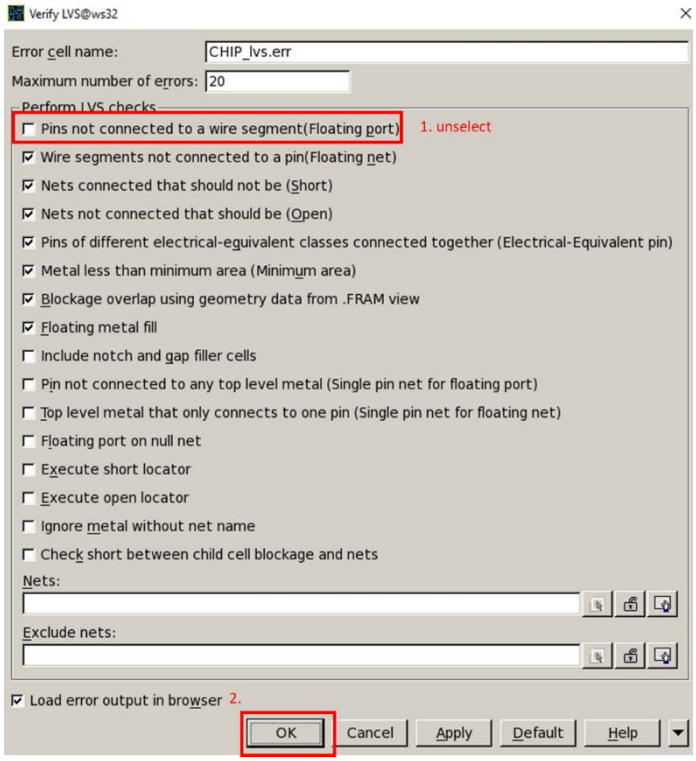
```
$ save_mw_cel -as 5_route
```

III. Verification and Data Export

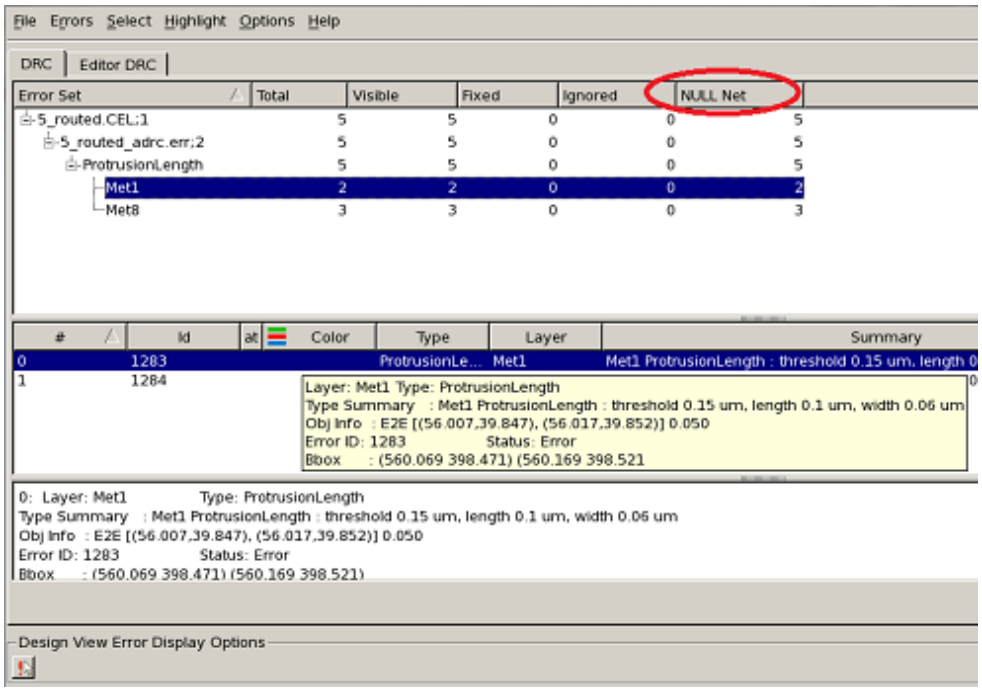
1. Verify DRC & LVS.

In ICC (version 2013 or later), the DRC result is included in the LVS report.

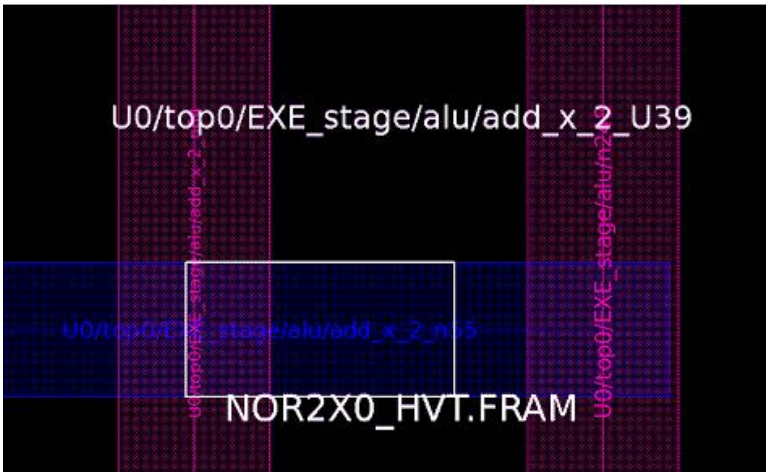
“Verification->LVS…”



You may find some DRC errors as follows:



Since these errors are all on null nets, which can be ignored. By clicking the error in the middle row, you will find its corresponding position in the Layout Window:



Note: (Optional) If the DRC errors can't be ignored (on other types of nets), they should be cleaned either by ICC's ECO Route: **"Route->Detail Route"**

Item	**Content**
Incremental routing	Enable
Initial DRC from input cells	Enable

Press **"OK"** to run automatic DRC fixing; or by modifying the layout manually with your own knowledge of VLSI and Electronics. (If this happens, good luck!) The DRC here is not final (e.g. no DRC rules for core filler and

metal filling). Usually we will use advanced DRC tool, e.g. ***Calibre***, for the final DRC checking on the exported GDSII layout.

2. Core filler insertion by executing

   $ source -echo ../pre_layout/design_data/addCoreFiller.tcl

3. Save your design.

   "File->Save Design" and click **"Save all"**

   Or you can save your design with command line interface.

   $ save_mw_cel -design CHIP

4. Save your design as new file for backup.

   "File->Save Design" and check **"Show advanced options"**

Item	Content
Save as	Enable
Save as name	6_corefiller

   Or you can save your design with command line interface.

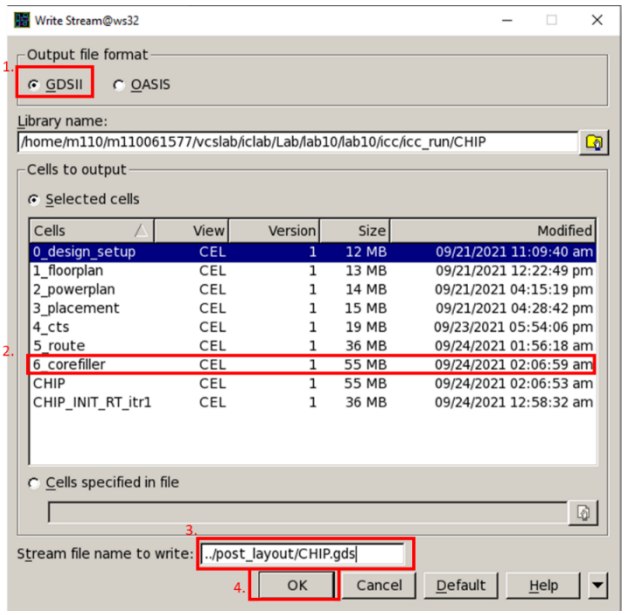
   $ save_mw_cel -as 6_corefiller

5. Export the GDSII layout.

   Type the following command in icc_shell terminal:

   $ set_write_stream_options -map_layer /usr/cadtool/cad/synopsys/SAED32_EDK/tech/milkyway/saed32nm_1p9m_gdsout_mw.map -child_depth 20 -flatten_via

   "File->Export->Write Stream"



Note: This is the layout which will be processed and verified by other tools in a normal backend flow. But we won't involve these operations in the following lab.

6. Export the netlist and the corresponding timing data in SDF (Standard Delay Format) file format for post-layout netlist simulation.

   (1) Netlist:

   ```
   $ write_verilog -diode_ports -wire_declaration -
   keep_backslash_before_hiersep   -no_physical_only_cells -
   supply_statement none ../post_layout/CHIP_layout.v
   ```

   (2) SDF:

   ```
   $ write_sdf -version 1.0 -context verilog ../post_layout/CHIP_layout.sdf
   ```

7. Export the design constrain (SDC) and parasitic information (SPEF) for power simulation.

   (1) SDC:

   ```
   $ write_sdc ../post_layout/CHIP_layout.sdc -version 1.9
   ```

   (2) SPEF

   Extract parasitic component first:

   ```
   $ extract_rc
   ```

   Then export SPEF:

   ```
   $ write_parasitics -output ../post_layout/CHIP_layout -format SPEF –
   compress
   ```

IV. Core Utilization Confirm:

1. Get core area and die area with the commands:

   ```
   $ get_attribute [get_core_area] bbox
   ```

   ```
   $ get_attribute [get_die_area] bbox
   ```

   The core utilization is the cell area reported by DC (the synthesis area report is in *./syn/report/*) over core area reported by ICC:

$$\boldsymbol{core\ utilization} = \frac{\boldsymbol{Total\ cell\ area\ (from\ DC)}}{\boldsymbol{core\ area\ (from\ ICC)}}$$

   What's your final utilization?

2. Try higher utilization (raise the core utilization from 0.5 to 0.8):

   To achieve better performance (high score in final project), try to do the APR flow again with higher utilization setting. This time, rather than using the GUI again, we can use the scripts to finish the flow.

(1) Close ICC.

"File->Close Design," "File->Close Library," and then type **exit** in the **icc_shell** terminal **(make sure to close library before you leave)**

(2) In the **Linux terminal**, type the following command:

```
$ cd ../
$ mv icc_run icc_run_utl50
$ mkdir icc_run
$ cd icc_run
$ icc_shell -64 -gui
```

(3) In the **icc_shell terminal**, type the following command:

```
$ source ../../ref_scripts/run.tcl
```

Note: You can use the scripts recorded by yourself or the reference scripts provided in *ref_scripts/*. 0~6 Tcl files in *ref_scripts/* are equivalent to all the steps in lab10~12.

Note: Make sure you have read these scripts before you use them, and remember to modify the related settings first.

3. Show TA your power, timing, and utilization result. Compare the result (utilization=0.8) to your original one (utilization=0.5).

V. Post-layout Simulation

1. In the *post_sim/* directory, read these files (post_sim.f, test_post_sim.v) and understand how the simulation environment is configured.

   Note: In the testbench, we need to annotate the SDF file to the CHIP instance (line79~81), the SDF file records the delay information of each gates and nets, and is exported by ICC in the previous step.

2. Run the simulation by execute the following command, and then check whether the simulation result is correct or not

   ```
   $ ncverilog -f post_sim.f
   ```

3. Use **nWave** to check the waveform.

VI. Power Analysis

1. Power analysis with **post-layout waveform**.

   In lab 9, we have learned how to execute time-based power analysis with the gate-level netlist and the gate-level simulation waveform. Now, let's analyze power consumption for the post-layout netlist with post-layout simulation

waveform.

To run the power analysis, you need to prepare the following files:

- Post-layout netlist(*.v, generated by ICC)
- Design Constrain file(*.sdc)
- Waveform (*.fsdb), actual waveform by post-layout simulation
- SPEF file (Standard Parasitic Exchange format, generated by ICC)

In lab09, we don't feed SPEF file to PrimeTime because we don't have any physical information about wiring. The SPEF file records parasitic information of the layout netlist, which provides important parasitic information to PrimeTime.

(1) Open PrimeTime and execute the provided script. Type the following command in **Linux** terminal:

```
$ cd ../primetime/post_sim_waveform/

$ pt_shell
```

(2) Type the following command in **pt_shell** terminal:

```
$ source pt_px.tcl
```

Note: You may find some error when **read_sdc** due to missing nets. We ignore these warnings since the missing nets do not affect the functionality.

Note: You may find some error when **check_power** due to design rule cost violation. We ignore these errors in the lab and final project although they may reduce the reliability of power estimation.

2. Power analysis with **pre-sim waveform**.

Sometimes, we cannot actually run the post-layout simulation for some reasons (the runtime is too long and undoable, the post-layout waveform is too large to be read by PrimeTime if your design is very very large…).

A substitution is using **pre-sim waveform** as the waveform fed into PrimeTime for power analysis, although the estimated power may not be as precise as using post-sim waveform. However, the error is small enough, so it's acceptable.

(1) Generate pre-sim waveform.

Type the following command in **Linux** terminal:

```
$ cd ../../
$ cd ./presim_for_pt
$ ncverilog -f presim.f
```

(2) Go back to PrimeTime directory and open PrimeTime

```
$ cd ../
$ cd ./primetime/pre_sim_waveform/
$ pt_shell
```

(3) Type the following command in **pt_shell** terminal:

```
$ source pt_px.tcl
```

Note: For the convenience of reading the VCD/FSDB file and setting strip path for the command "**read_vcd**," we wrap the RTL designs (in *./source/*) in the CHIP_syn_app.v module (in *./icc/icc_run/pre_layout/design_data/*). You can check this by open the .f file (*./presim_for_pt/presim.f*).

Note: When we want to use pre-sim waveform for power analysis in PrimeTime, we have to add the argument **"-rtl"** in "**read_vcd"** command (line 21 in *pt_px.tcl*) to specify the VCD/FSDB file comes from a RTL simulation.

3. Power Estimation Comparison.

Compare your power reports estimated in ICC and in primetime.

	ICC	Primetime (pre-sim waveform)	Primetime (post-sim waveform)
Total Power			
Cell Leakage Power			
Total Dynamic Power			

Congratulations!! You have finished all the labs in this semester!!

Appendix

Here is the file list of the lab package. Please check if anything missing!

Directory	**Filename**	**Description**
icc/icc_run	setup.tcl	Environment setup
icc/pre_layout/dc	top_pipe_syn.sdc	Synthesis SDC

icc/pre_layout/dc	top_pipe_syn.v	Synthesized
icc/pre_layout/design_data	add_tie.tcl	Tie cells insertion
icc/pre_layout/design_data	addCoreFiller.tcl	Core filler insertion
icc/pre_layout/design_data	CHIP_syn.f	Filelist for simulation
icc/pre_layout/design_data	CHIP_syn.sdc	SDC for ICC
icc/pre_layout/design_data	CHIP_syn_app.v	Netlist wrapper
icc/pre_layout/design_data	get_pnr_netlist.bat	Batch file for merging netlist
icc/pre_layout/design_data	io_pin.tdf	Pin Constraint
sim	instruction.txt	CPU simulation pattern
sim	presim.f	Filelist for pre-simulation
sim	test_top.v	Testbench for the CPU
sim	top.f	Filelist for the CPU
source	top.v	Top module for this CPU
source	top_pipe.v	Pipelined design for this CPU
source	ID_stage.v	RTL for ID stage
source	IF_stage.v	RTL for IF stage
source	controller.v	RTL for control signal
source	regfile.v	RTL for register file
source	ID_EXE.v	RTL for ID to EXE flip flop
source	EXE_stage.v	RTL for EXE stage
source	alu.v	RTL for alu module
source	CPU_define.v	Definition file
source	dsram.v	RTL for memory
source	PC.v	RTL for program counter
source	IF_ID.v	RTL for IF to ID flip flop
syn	.synopsys_dc.setup	DC setup file
syn	*.tcl	Synthesis scripts
syn	*.log	Log file
syn	run_dc.bat	Batch file for running synthesis
syn/report	*.out	Report files
syn/report	*.log	Log files
syn/netlist	top_pipe_syn.ddc	Netlist & constraints can be read by DC
syn/netlist	top_pipe_syn.saif	Switching activity interchange format (for power)
syn/netlist	top_pipe_syn.sdc	Synopsys design constrains format
syn/netlist	top_pipe_syn.sdf	Standard delay format

syn/netlist	top_pipe_syn.v	Netlist in verilog