ECE 551

APR Tutorial

(Using IC Compiler)

(watch the video first)

What is IC Compiler?

- IC Compiler is an Auto Place and Route (APR) tool from Synopsys
- Once you have a netlist from design compiler how do you create the layout necessary to fabricate your digital circuit on a wafer?
- APR is much faster and much lower labor than laying out all those gates, flops, and interconnections by hand.

IC_Compiler Setup

- Ensure you have copied **.synopsys_dc.setup** from user **ece551** to your home directory*: (this should have already been done when you setup for the design_compiler tutorial)*

```
linux_promp% cp ~ece551/.synopsys_dc.setup ~/.
```

- Copy the IC_Compiler specific setup script (**apr_setup.icc**) from user **ece551.** This should be placed in the directory you intend to use as your work area for your APR run.

```
linux_promp% cp ~ece551/apr_setup.icc ~/ece551/my_apr_run/.
```

Launch ICC & Setup

- From a linux command window that is currently in your work area launch **IC_Compiler**

```
linux_prompt% cd ~/ece551/my_apr_run

linux_prompt% icc_shell –gui –shared_license
```

- Once **IC_Compiler** shell window is up source the setup script from within that command shell.

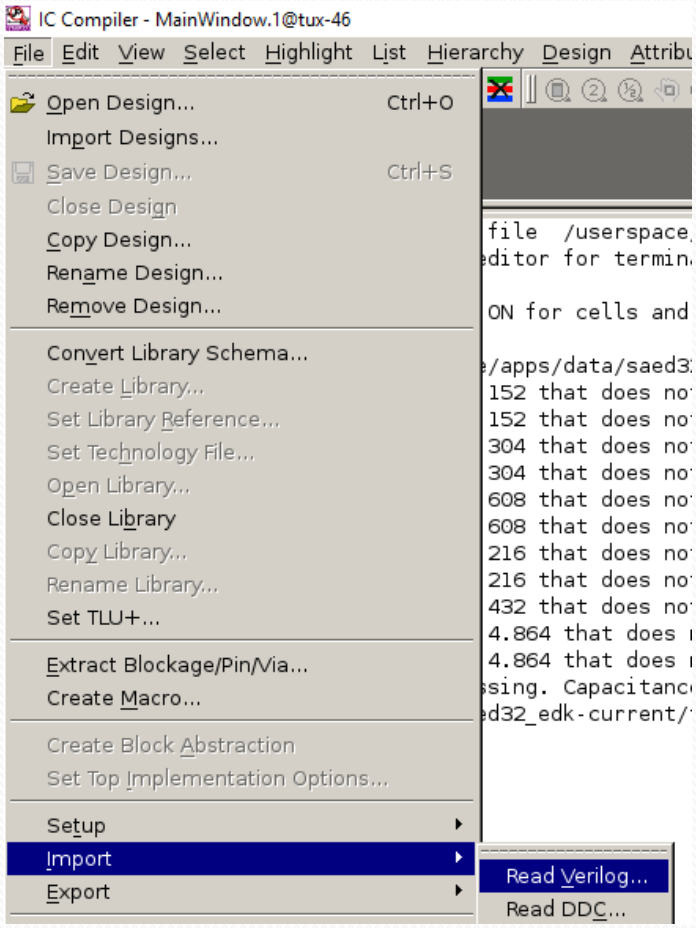
```
icc_shell> source apr_setup.icc
```

- This will setup some file pointers specific to our Synopsys 32nm process, and will create a "library" called **apr_lib** in your working directory. (if you already had a library setup in that directory named **apr_lib** you would have to delete or rename it).

Reading In Gate Level Verilog & Constraints File

- Next you read in your gate level netlist that was produced by **design_compiler**. Under the import menu of the **icc** window you will find verilog. Browse for your **.vg** file

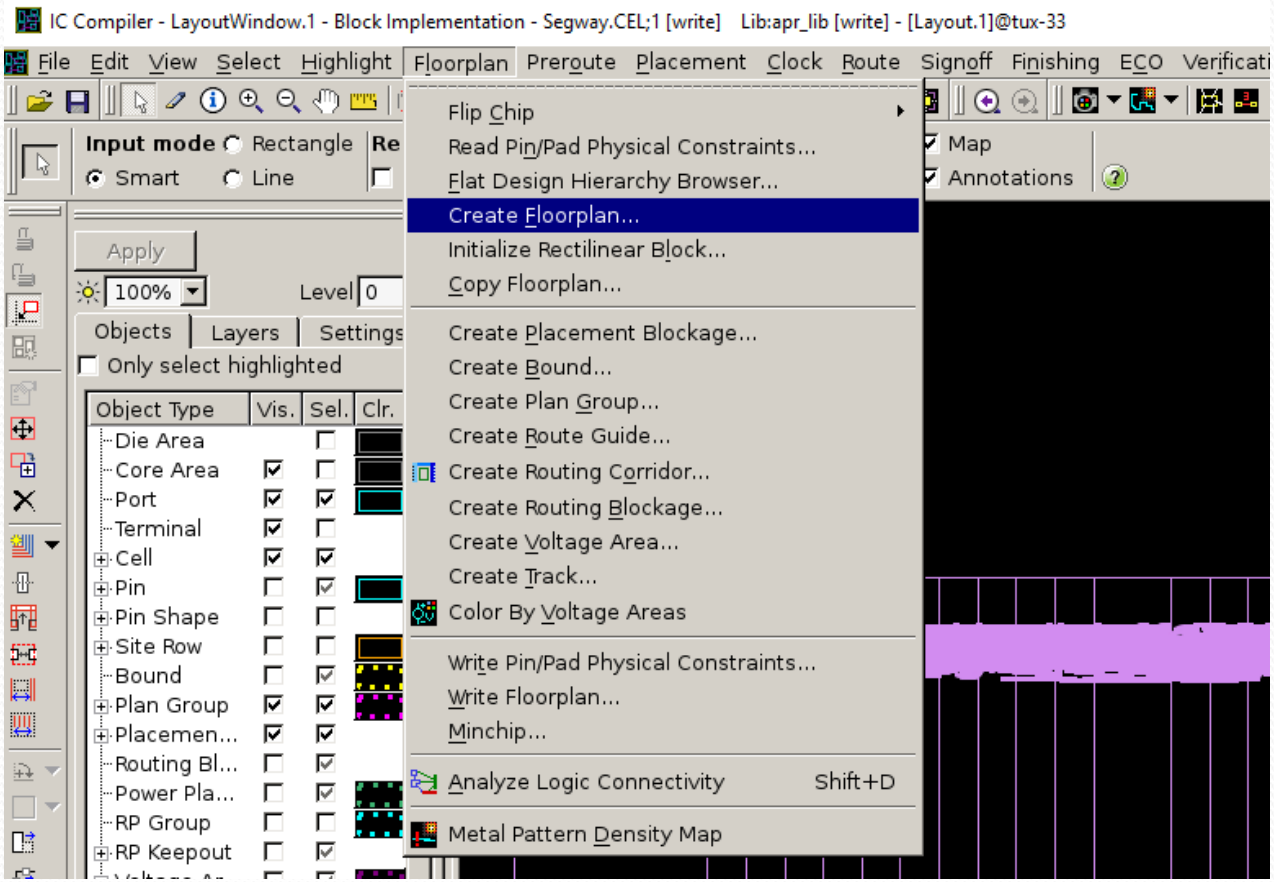
(**NOTE:** have to change filter to * since .vg is non-standard)



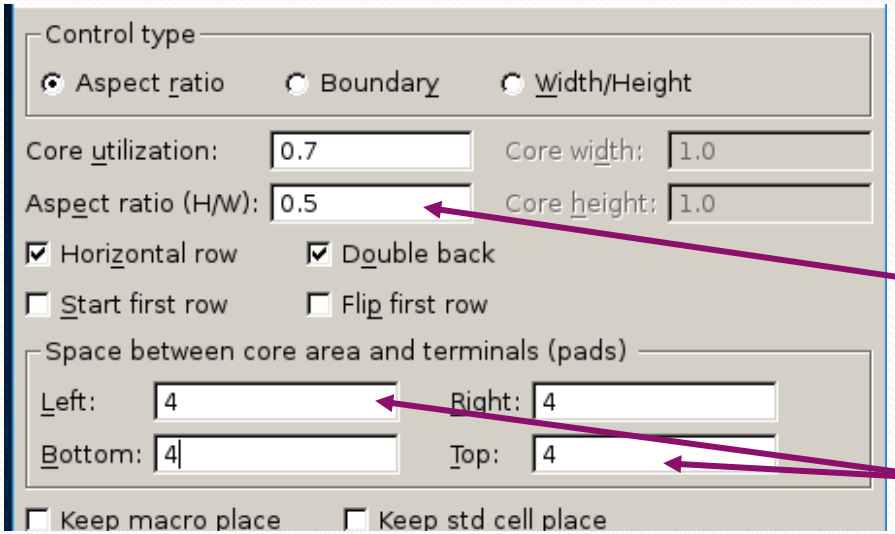
- You **also** need to import the **S**ynopsys **D**esign **C**onstraints file (**.sdc**) that you should have produced from your synthesis run.

Creating Floorplan

- A floor plan will define the outline of your APR block, the location of the interface pins, and the power gridding.
- In the banner menu under “Floorplan” you will find a “Create Floorplan” option.

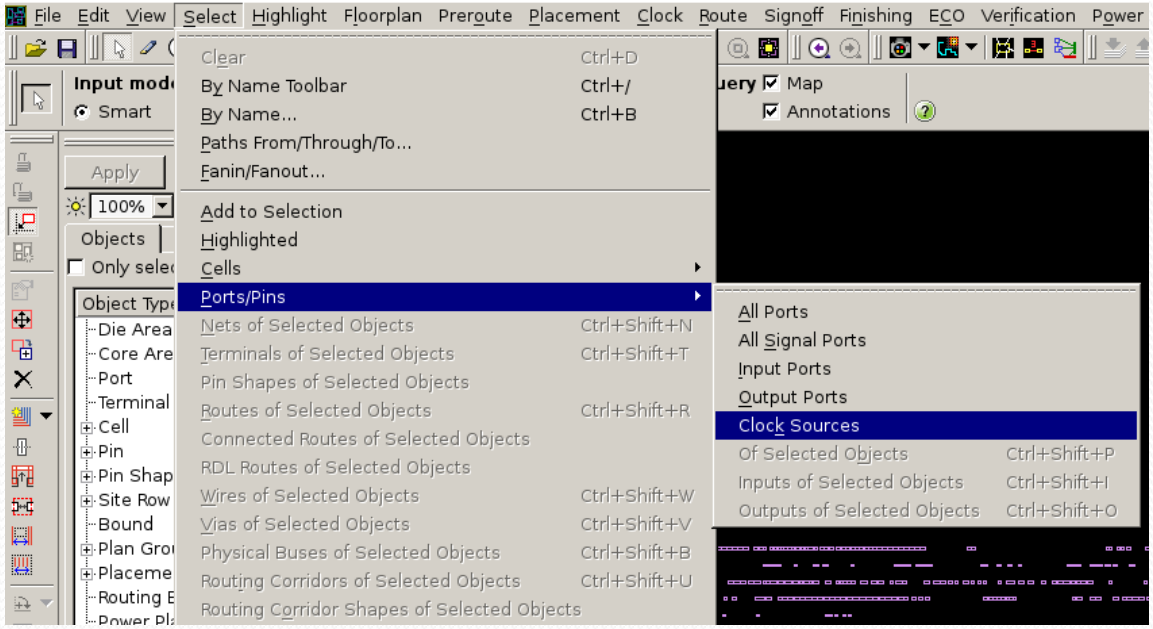


Creating Floorplan (continued)



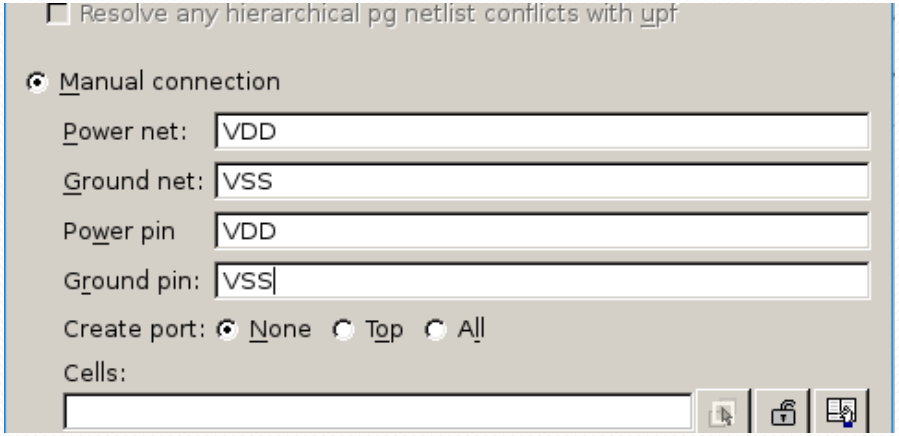
- We will choose a core utilization of 0.7 (not very aggressive)
- For aspect ratio choose 0.5 This will make the block twice as wide as it is tall.
- Give it 4microns all around so we have room for our power rings

- Locate your clock pin(s) via: **Select -> Port/Pins -> Clock Sources** as shown
- Move your clock pin to a more central boundary location.

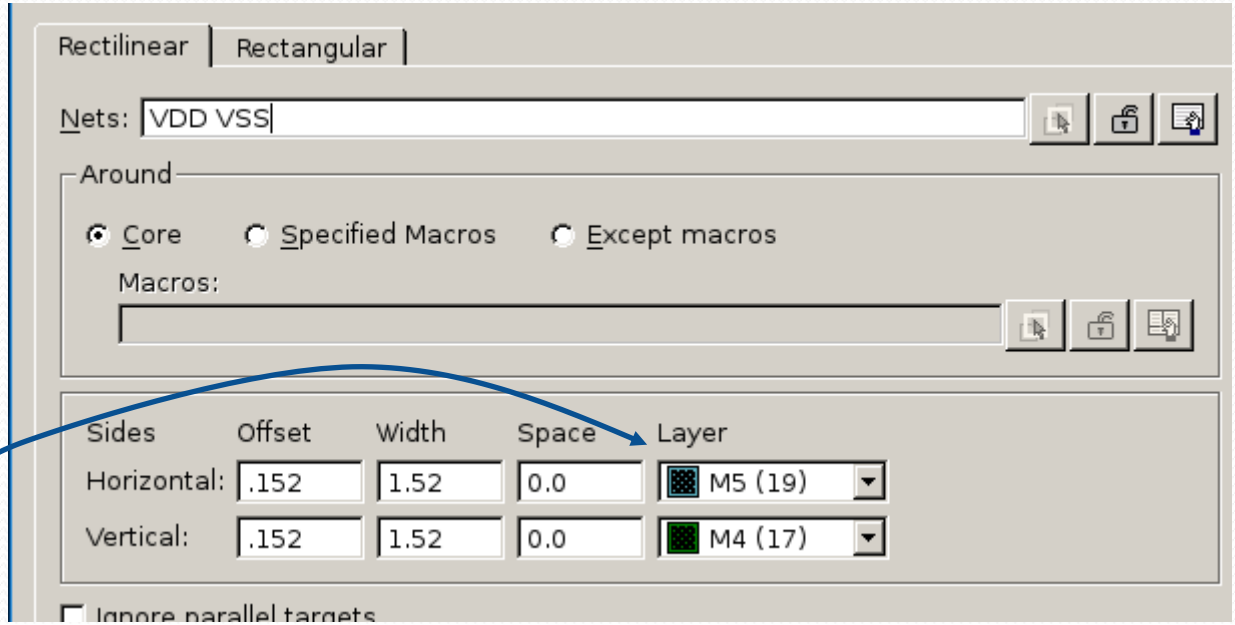


Power/Ground Rings

- Under **Preroute -> Derive Power Ground Connection**

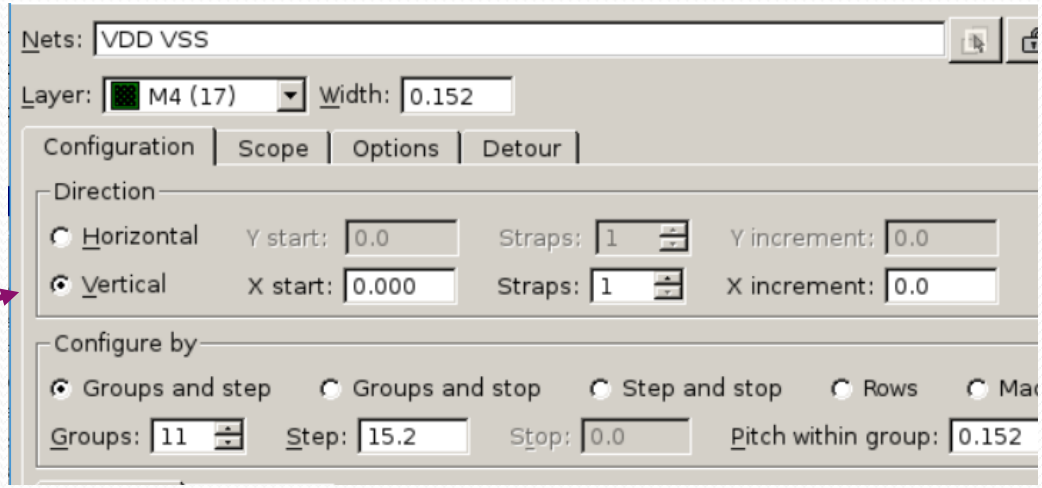


- Select **Manual connection** and fill in VDD and VSS as power and ground net names
- Under **Preroute -> Create Rings ...**
- Fill in the spacings as shown and fill in VDD and VSS as the nets for the rings.
- Note metal layers



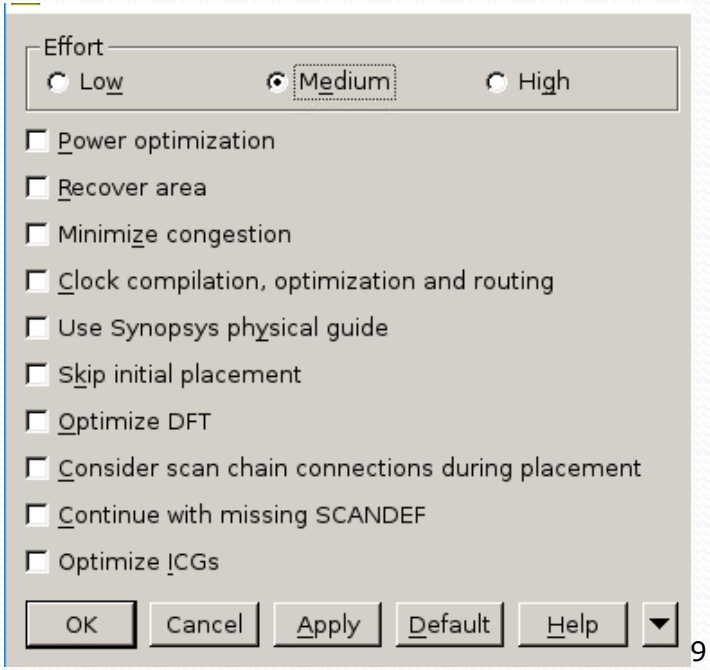
Power/Ground Strapping

- Under **Preroute -> Create Power Straps …**
- Number of groups may vary for your design. The width of my core was such that 11 groups worked nice at a 15.2 micron spacing. Remember to select vertical.



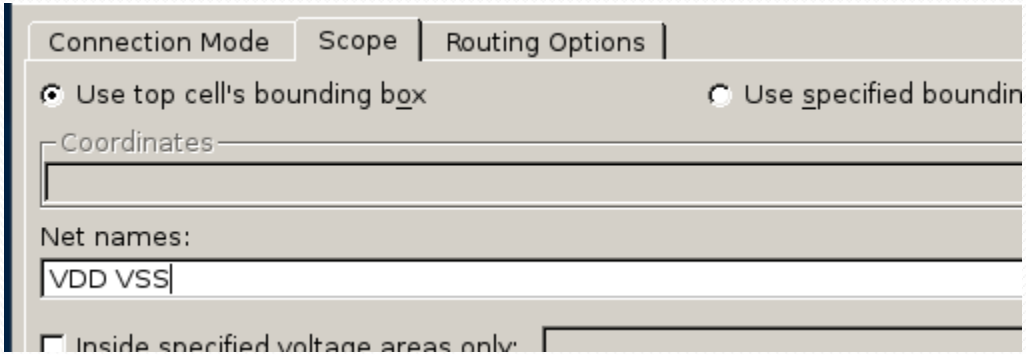
Cell Placement

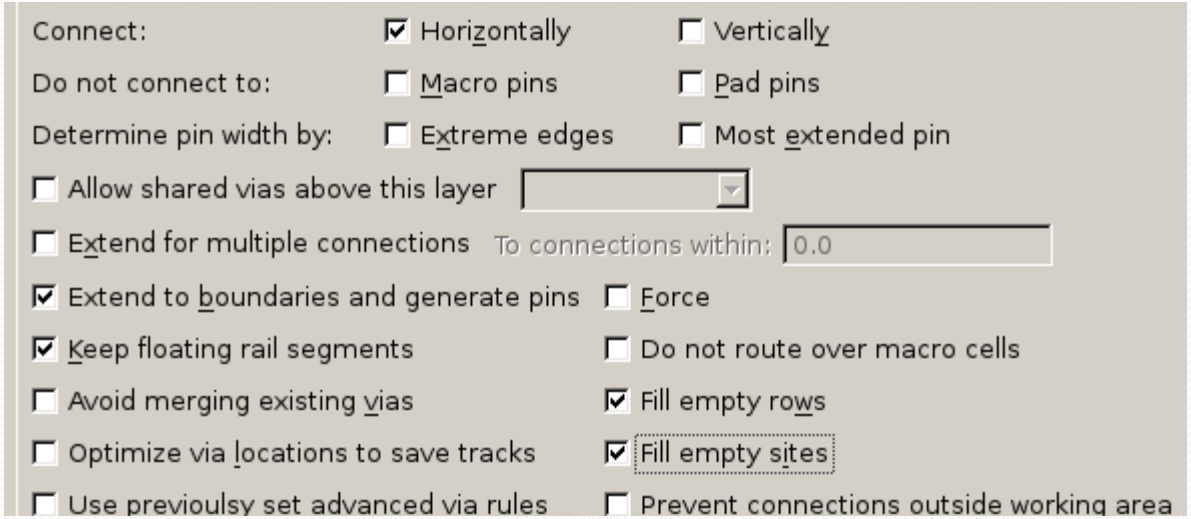
- Under **Placement -> Core Placement and Optimization**
- We essentially just take the defaults and wait while it minimizes its complex cost function associated with placement.



Completing M1 Power Rails

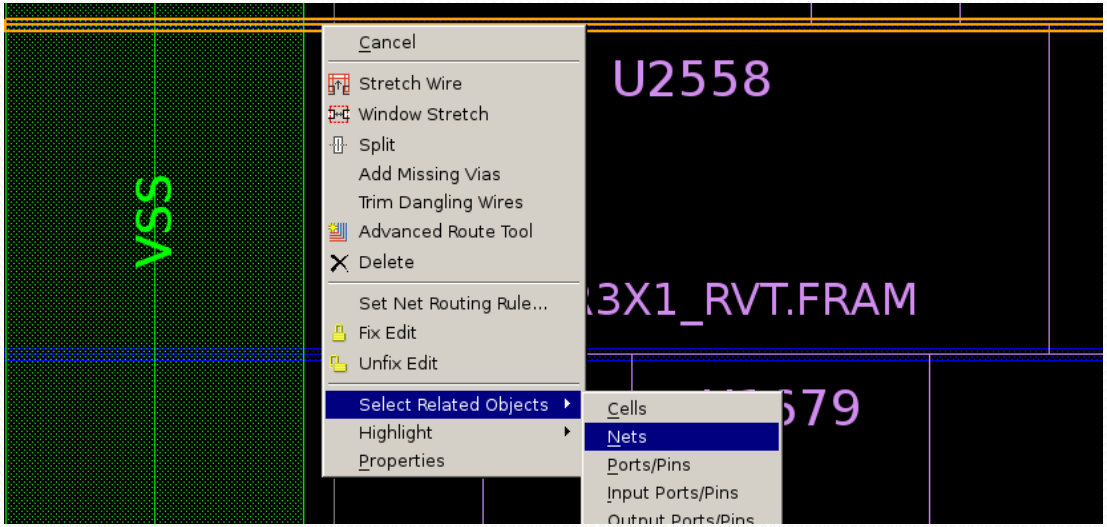
- Under **Preroute -> Preroute Standard Cells …**
- VDD and VSS are the power rails we are completing.
- Now under the "Routing Options" tab
- We want to select the options shown here.
- After completing this command you should see your M1 rails have been extended to the outer power ring but not necessarily via'd in.



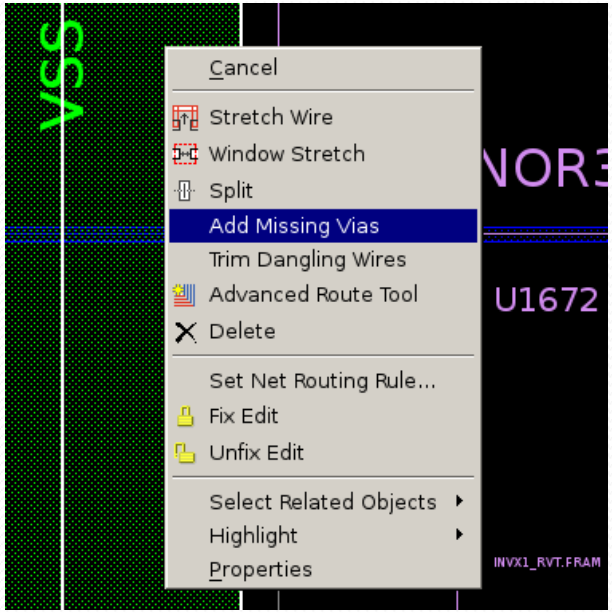


VIAing the Power Grid

- Not all the vias necessary to tie the power grid together are there.
- Select either a VSS rail, then right click and "**Select Related Objects -> Nets**"

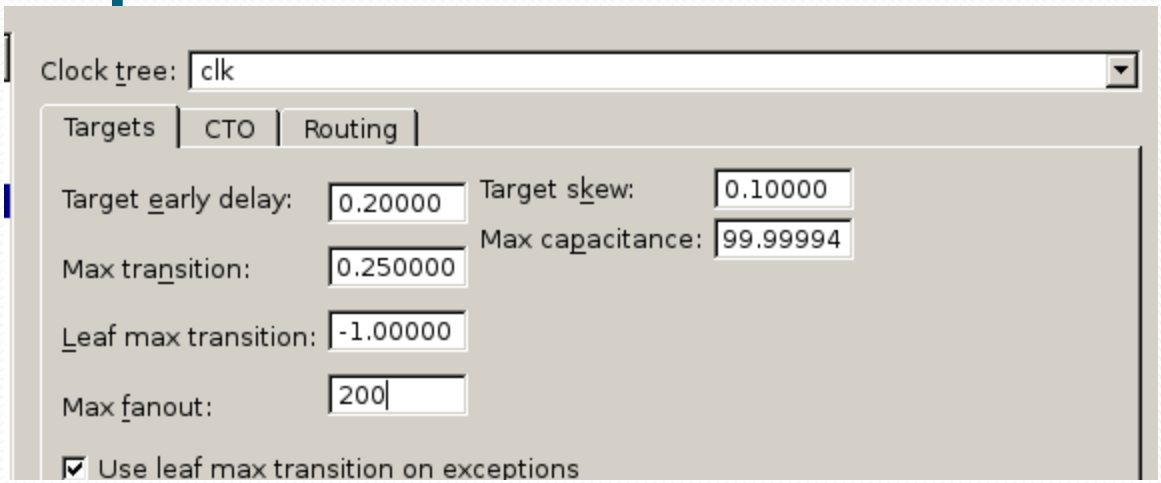


- Now that all VSS objects are selected right click and choose "**Add Missing Vias**".
- Now the entire power distribution grid should be tied together.



Clock Tree Options

- Under **Clock -> Set Clock Tree Options …**
- Select "clk" as the tree. You can try the values shown here, however, there is some "trial and error" to these settings.



- Now enter the following commands via the shell:

icc_shell> **clock_opt –only_cts –no_clock_route**
icc_shell> **report_clock_tree**

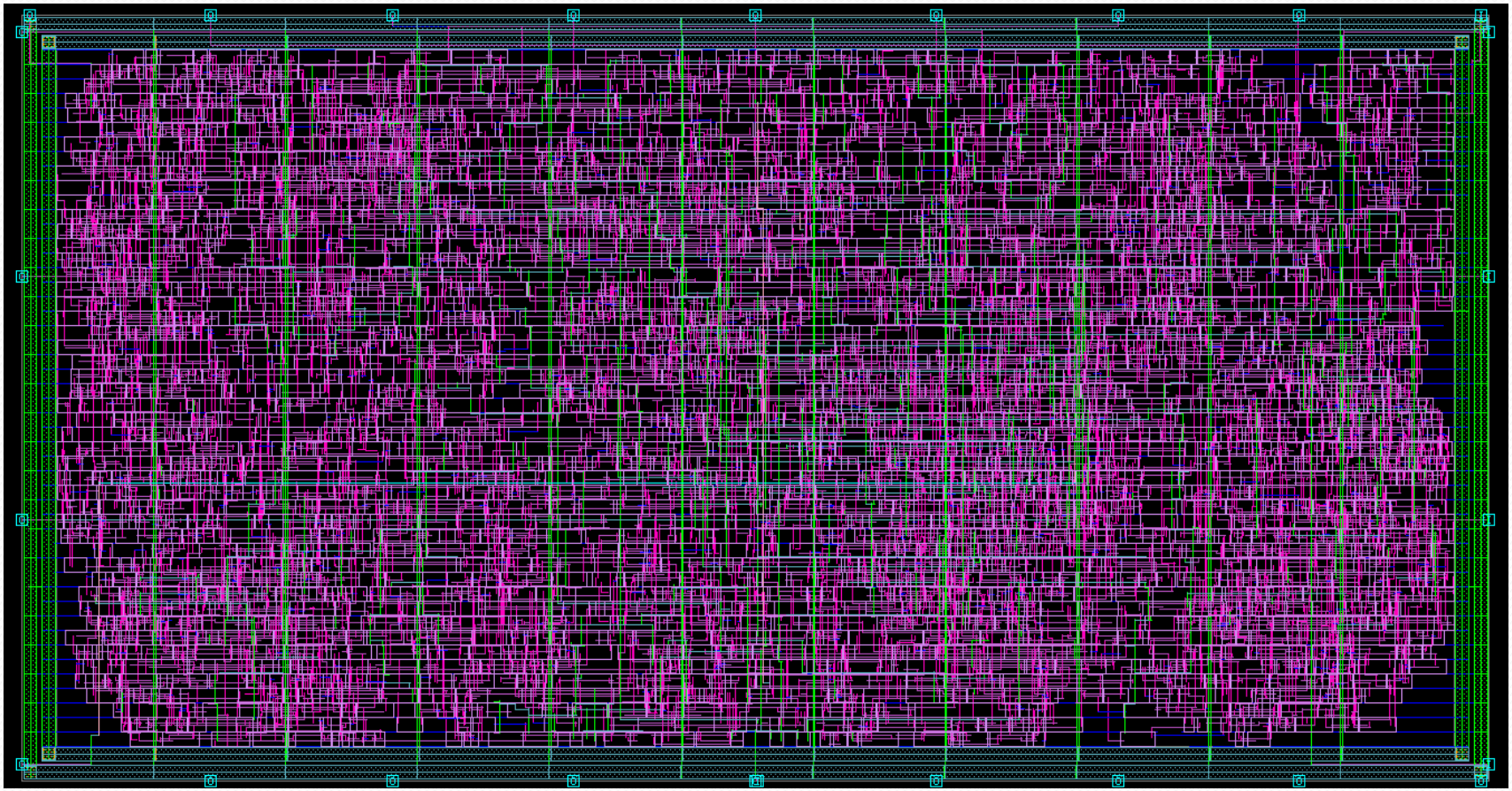
- Ideally the min and max clock paths are similar in delay and number of levels of buffering. We would like to see buffering of less than 4 levels for our size designs.

- Sometimes running the **clock_opt** command multiple times yields better results.
- When happy with results commit to them with:

icc_shell> **route_zrt_group –all_clock_nets –reuse_existing_global_route true**

Final Detailed Route

- Under **Route -> Core Routing and Optimization …**
- We just take all the default settings and let it rip.



Outputs of Completed APR

- Timing report (accurate now that is has parasitics accurately extracted)

icc_shell> **report_timing**

- SDF file (Standard Delay Format) for a full chip rollup of timing
- Parasitics file
- Stream file to have chip manufactured or to be imported by other CAD tool