

Chip Assembly

Using a Chip Assembly Router

Virtuoso Space-based Router (vsr)
Cadence Chip Assembly Router (ccar)

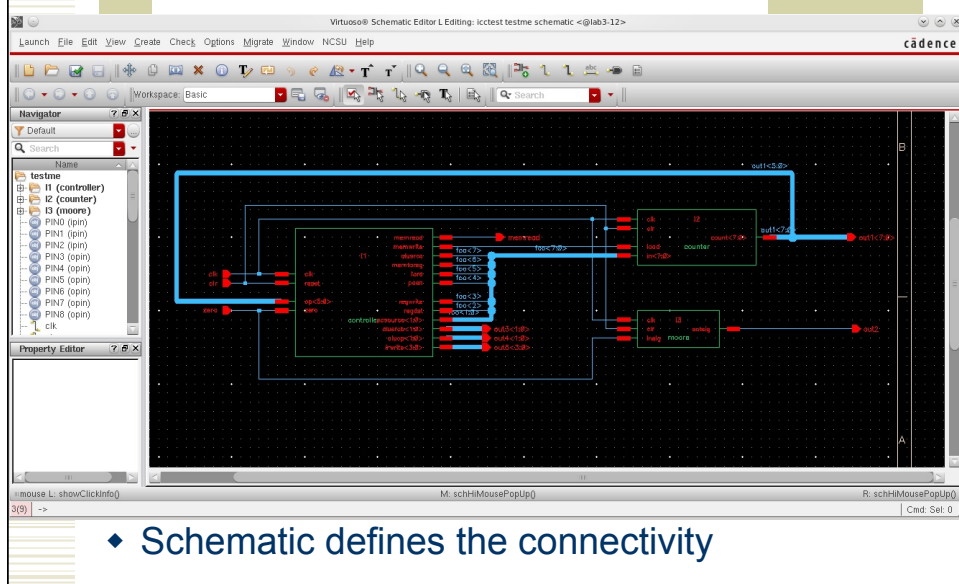
Yet Another Tool...

- ♦ This is a tool you can use to connect large blocks that have been designed separately
 - Like placed and routed blocks from Innovus, or memories, or custom register files, etc.
 - Also useful for wiring a fully-connected core to the pads
- ♦ Hand-placement, but automated wiring...
- ♦ Chapter 12 in CAD book...

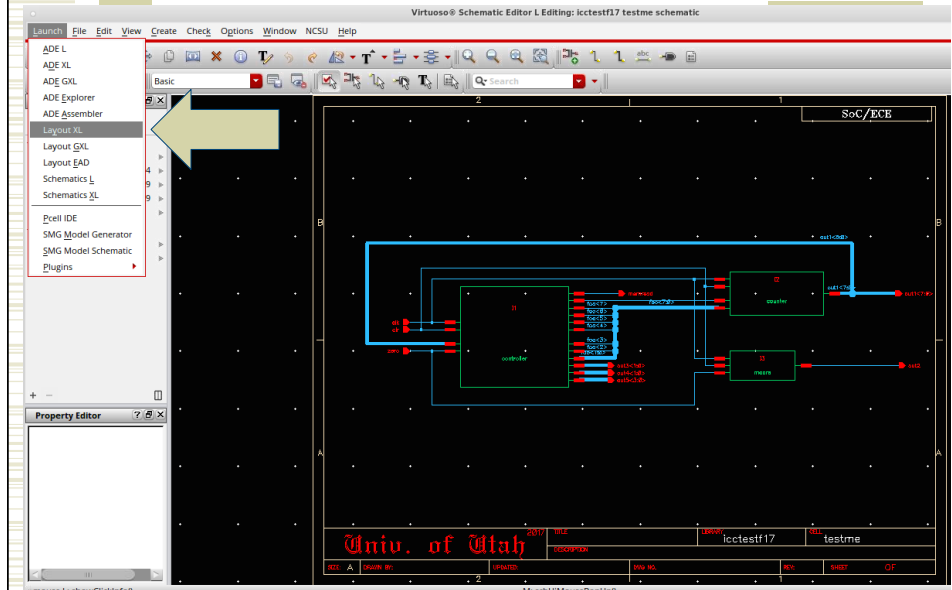
Outline

1. Start with a schematic to define connectivity
2. Then make a layout template for Layout-XL
3. Place blocks by hand and adjust floorplan
4. Wire vdd and gnd by hand
5. Use vsr or ccar for signal routing
6. Import back into Virtuoso for DRC, Extract, LVS, GDS, etc...

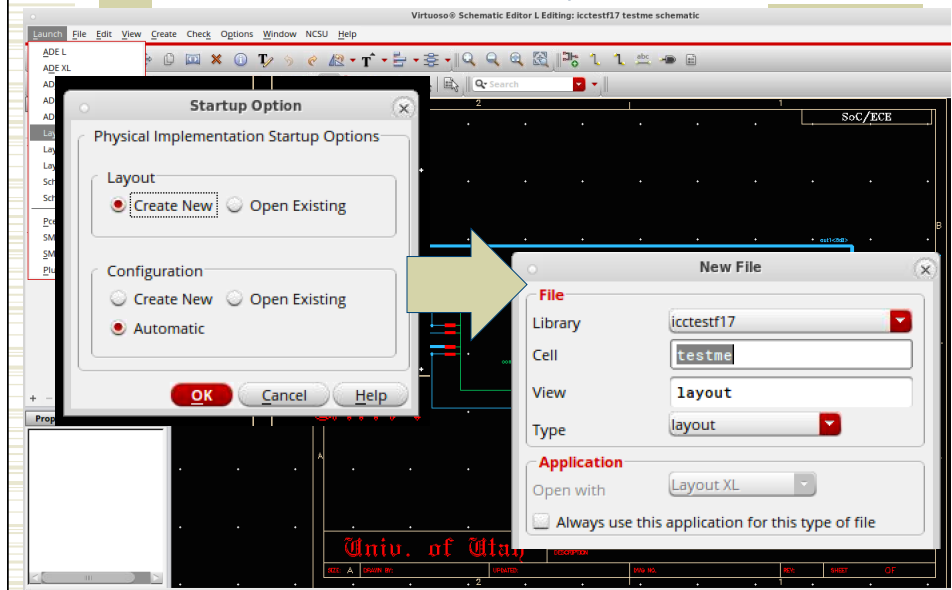
Start with connected macro cells



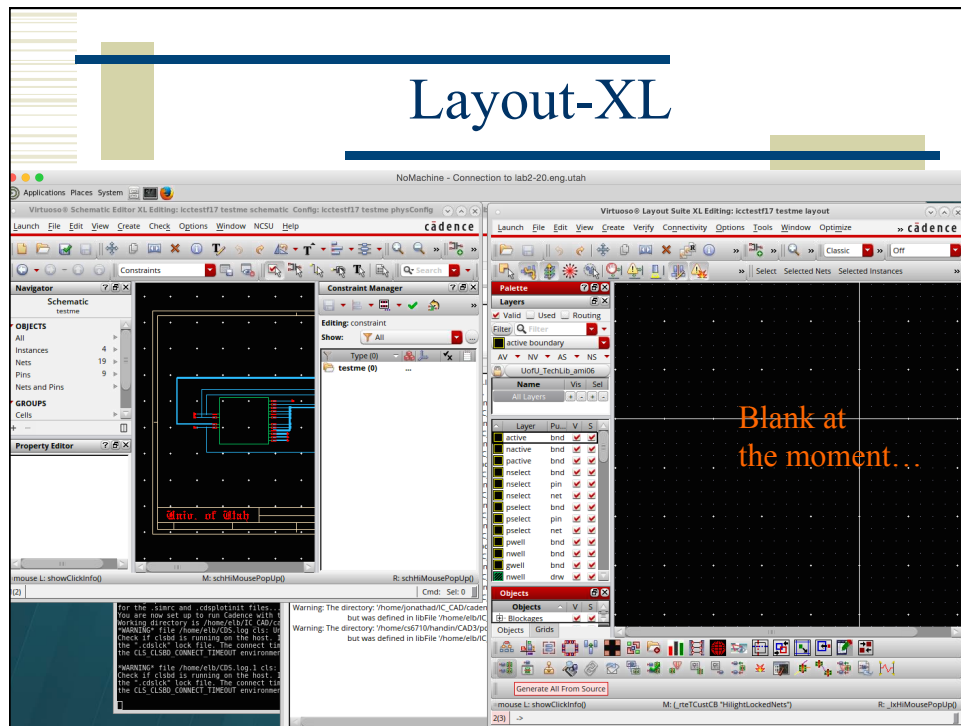
Launch Virtuoso-XL (Layout-XL)



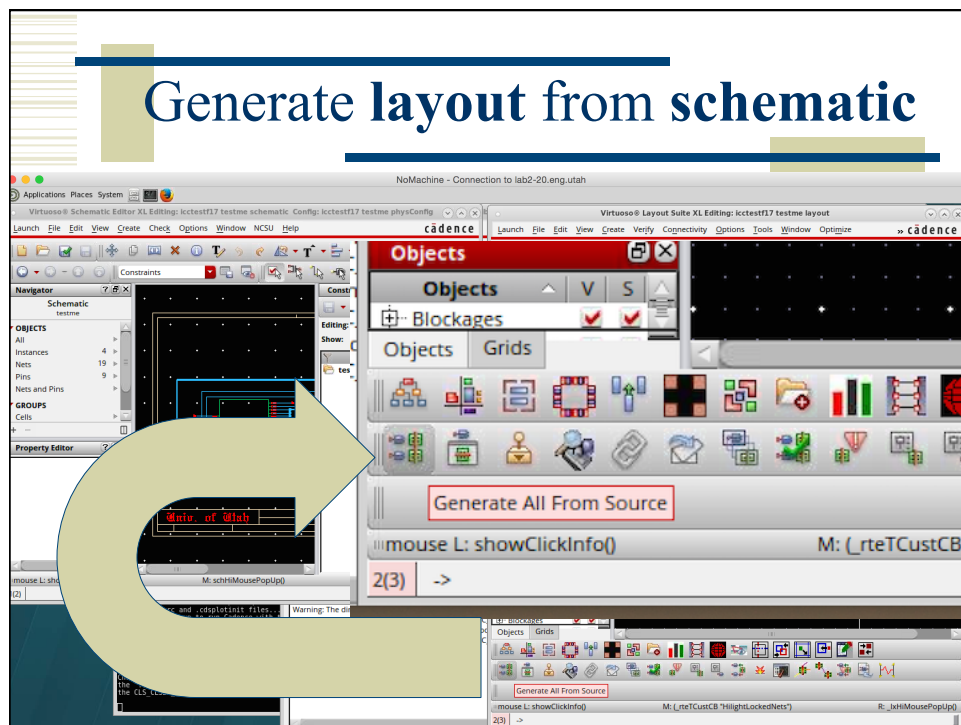
Launch Layout-XL



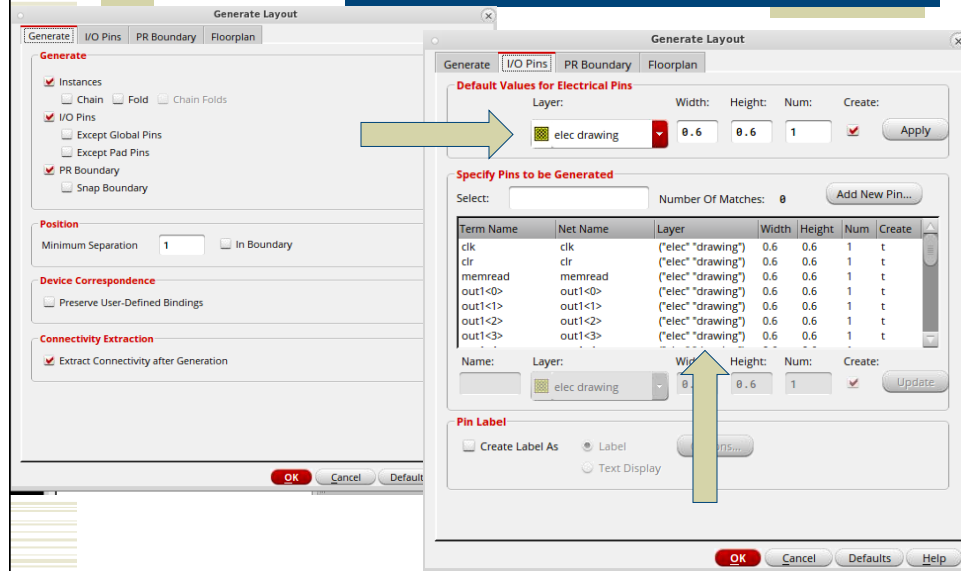
Layout-XL



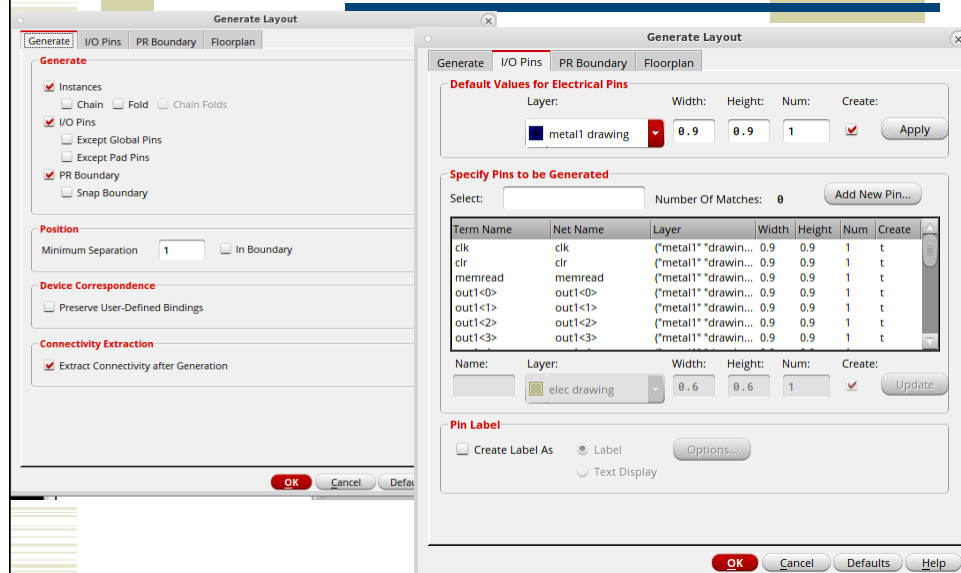
Generate layout from schematic



Generate layout from schematic



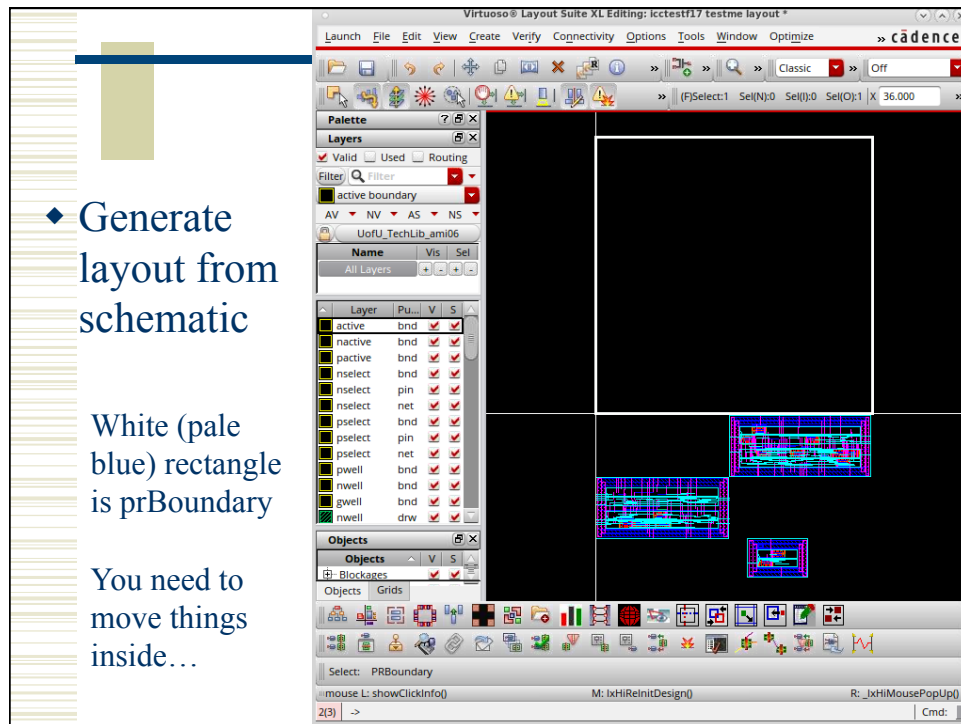
Generate layout from schematic



◆ Generate layout from schematic

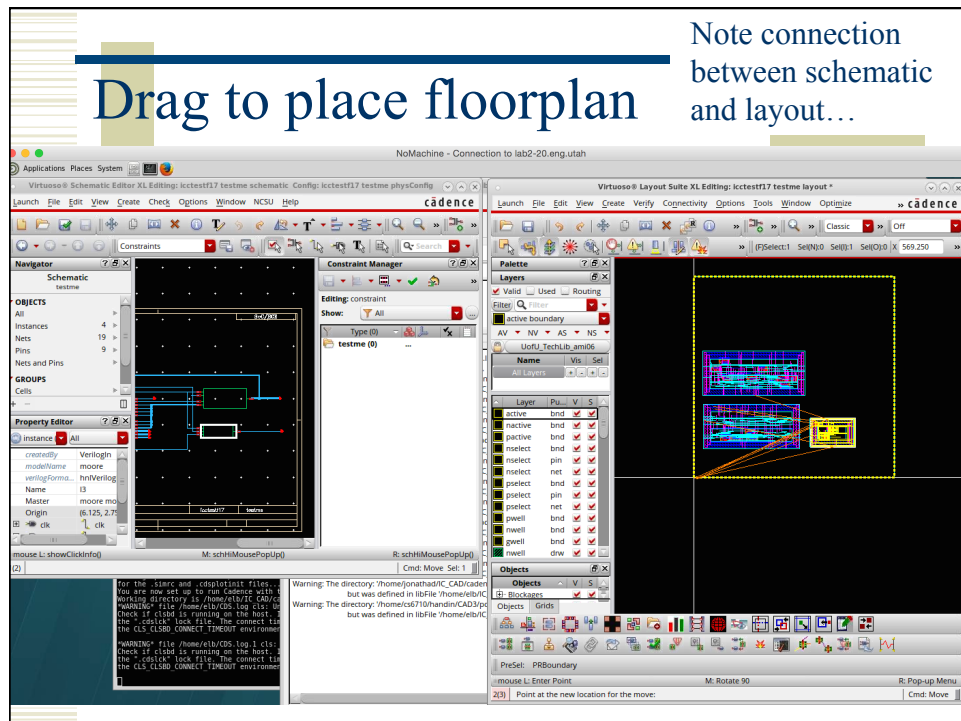
White (pale blue) rectangle is prBoundary

You need to move things inside...

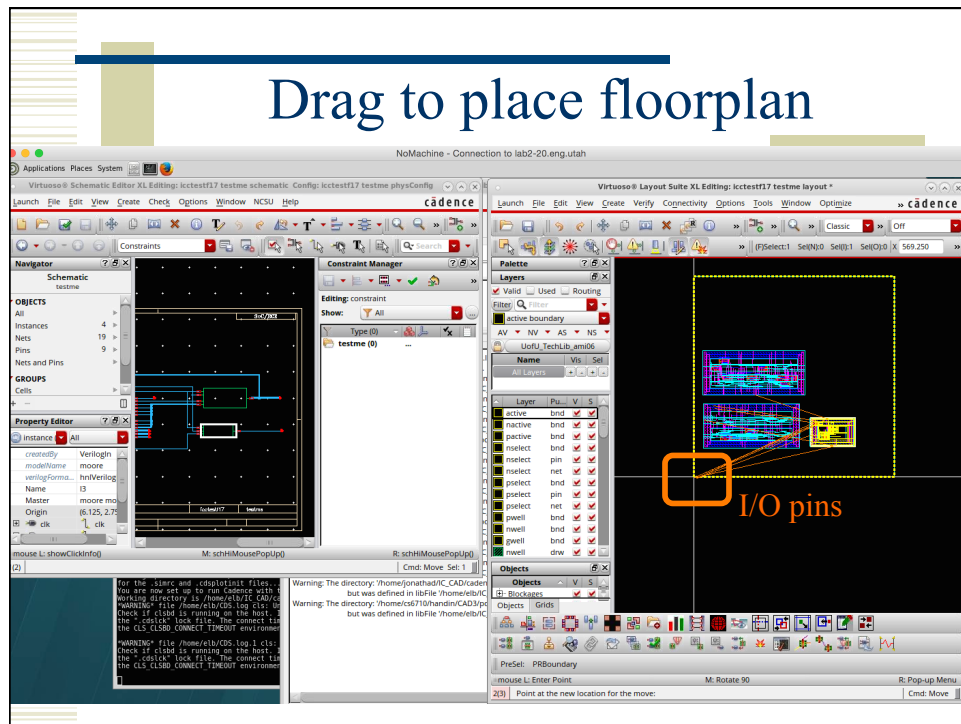


Drag to place floorplan

Note connection between schematic and layout...



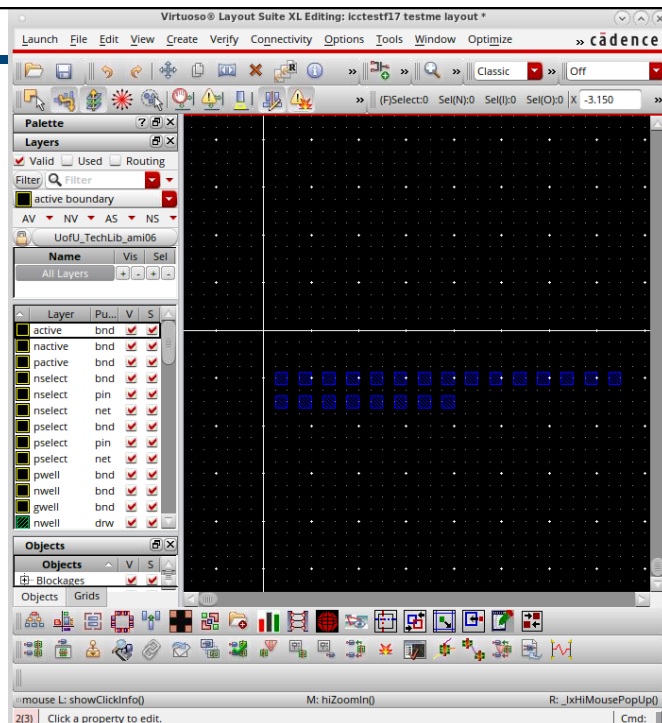
Drag to place floorplan



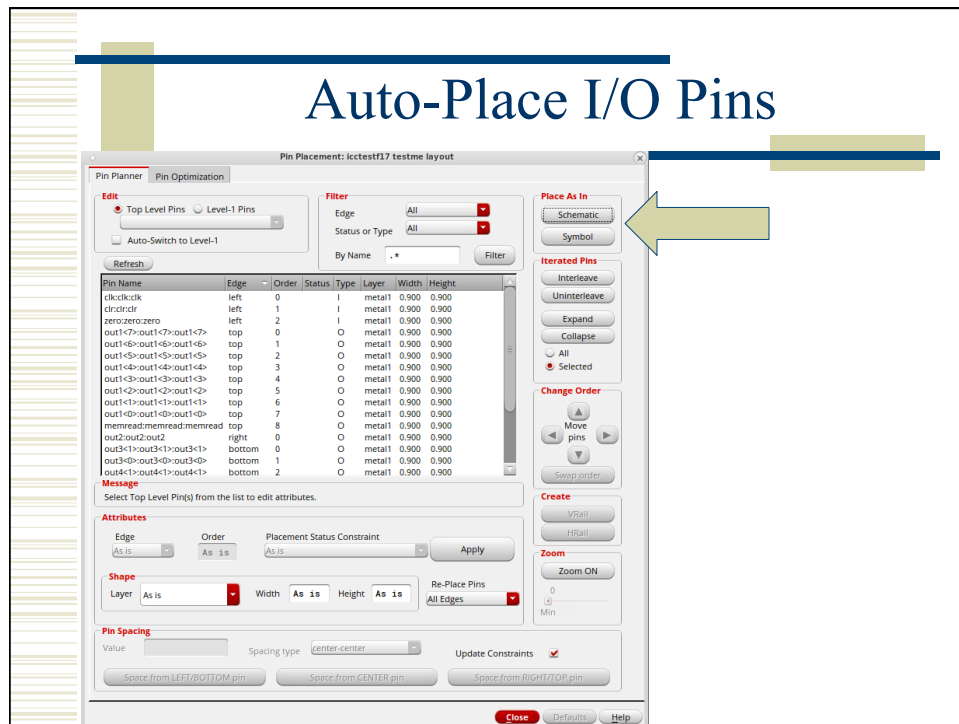
Place I/O Pins

One way is to select and drag by hand...

Also, remove vdd! and gnd! if they are there.

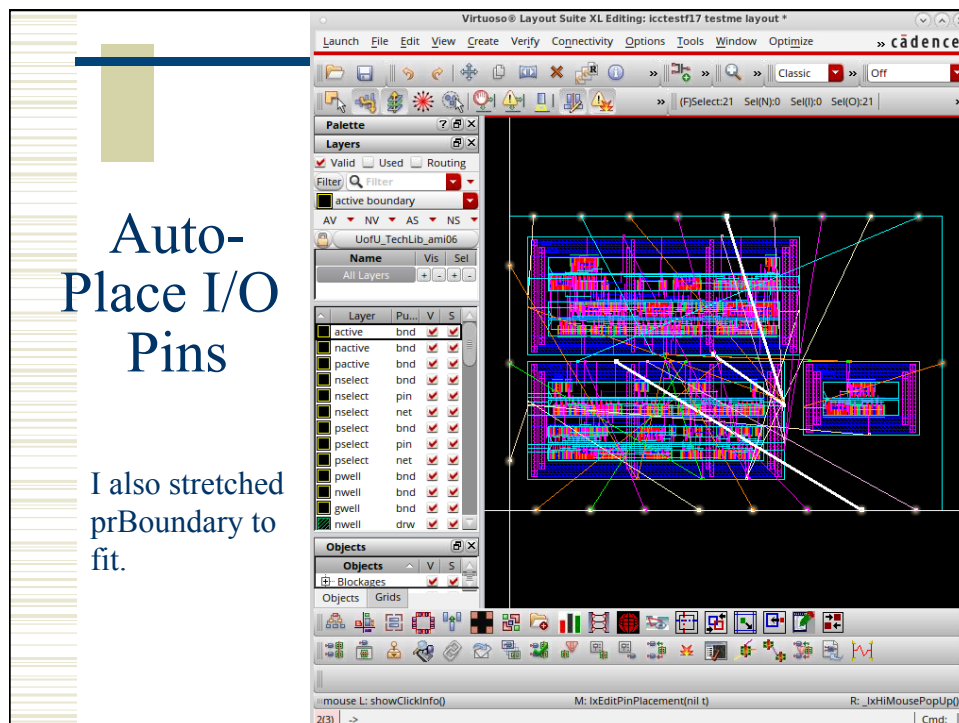


Auto-Place I/O Pins



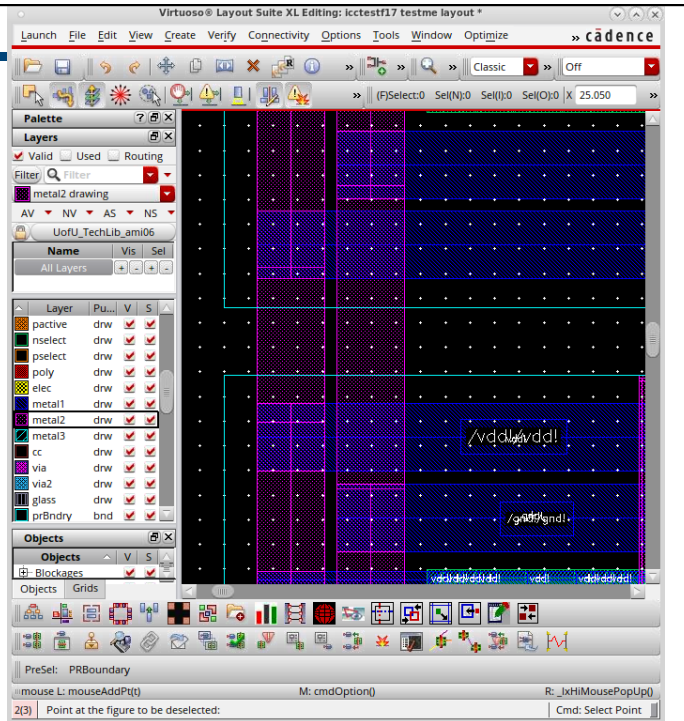
Auto-Place I/O Pins

I also stretched prBoundary to fit.

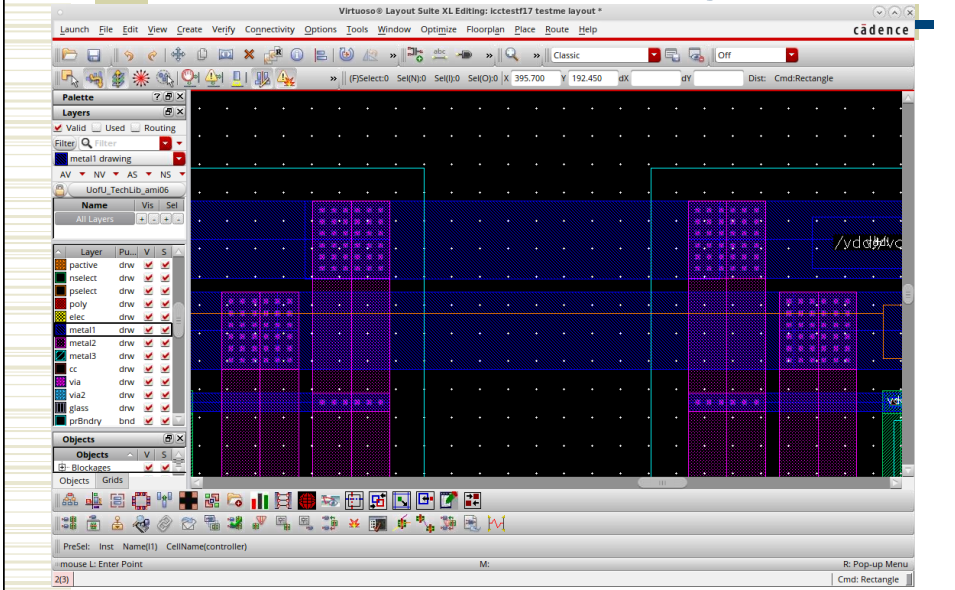


Connect
vdd! and
gnd!

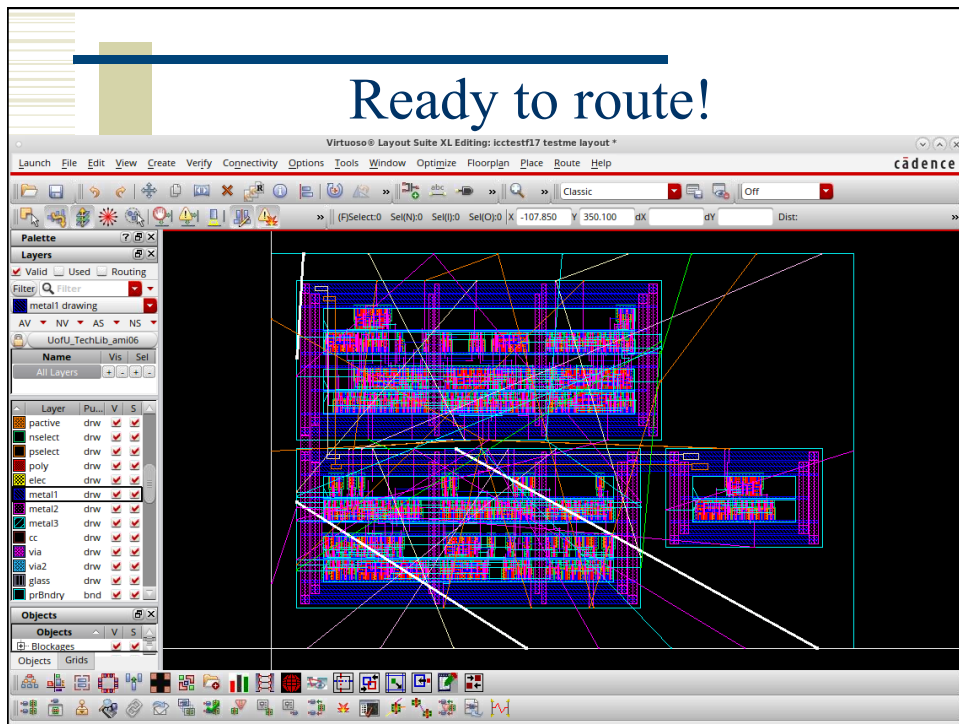
I use nice
big fat
rectangles...



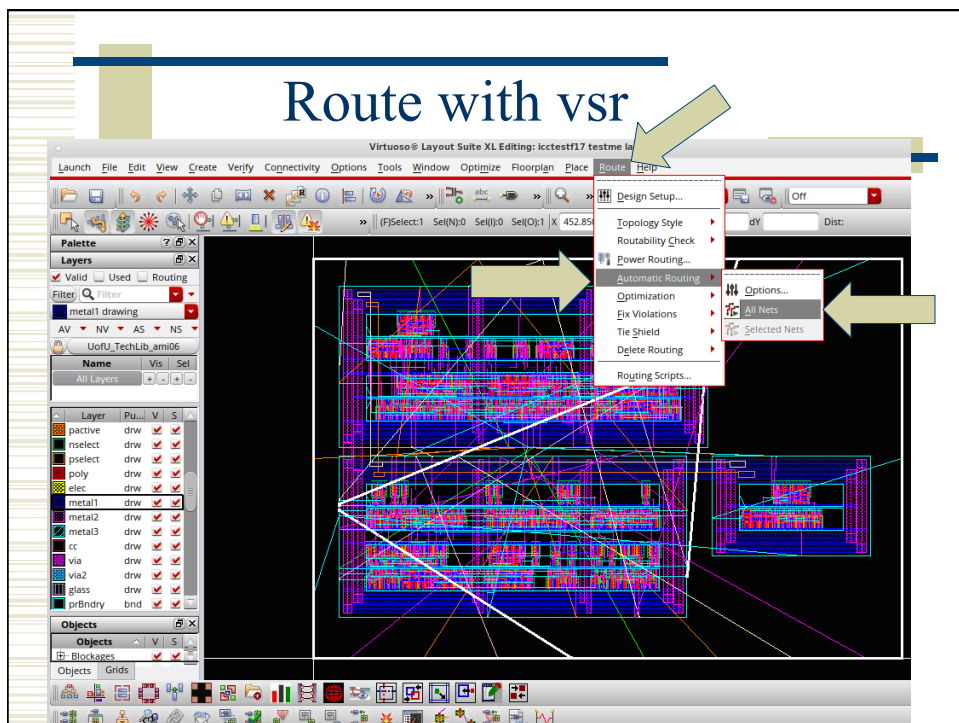
Connect vdd! and gnd!

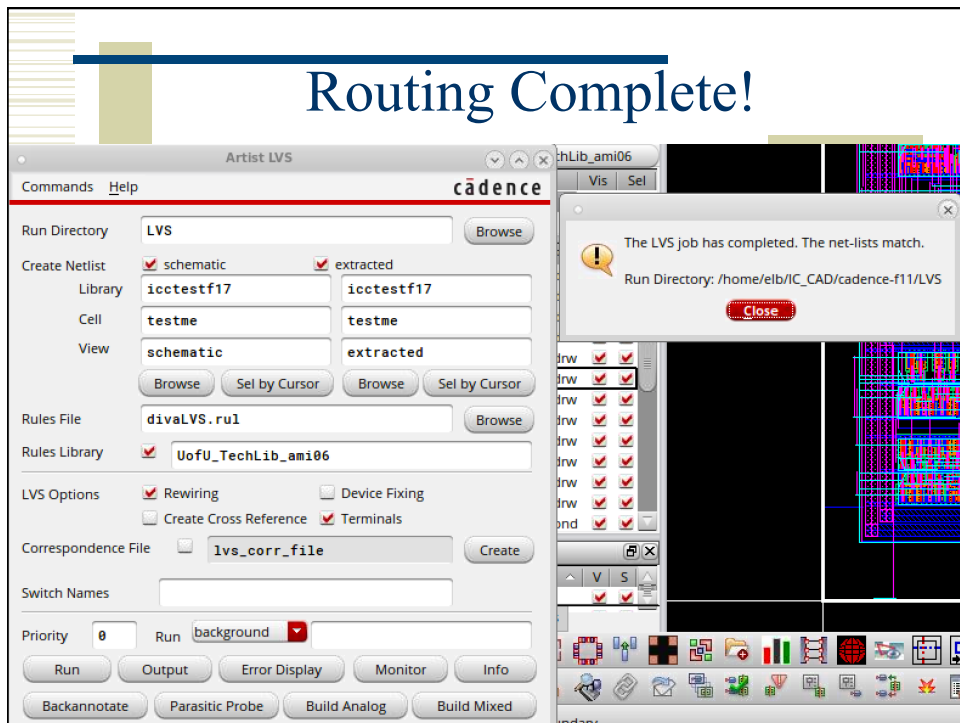
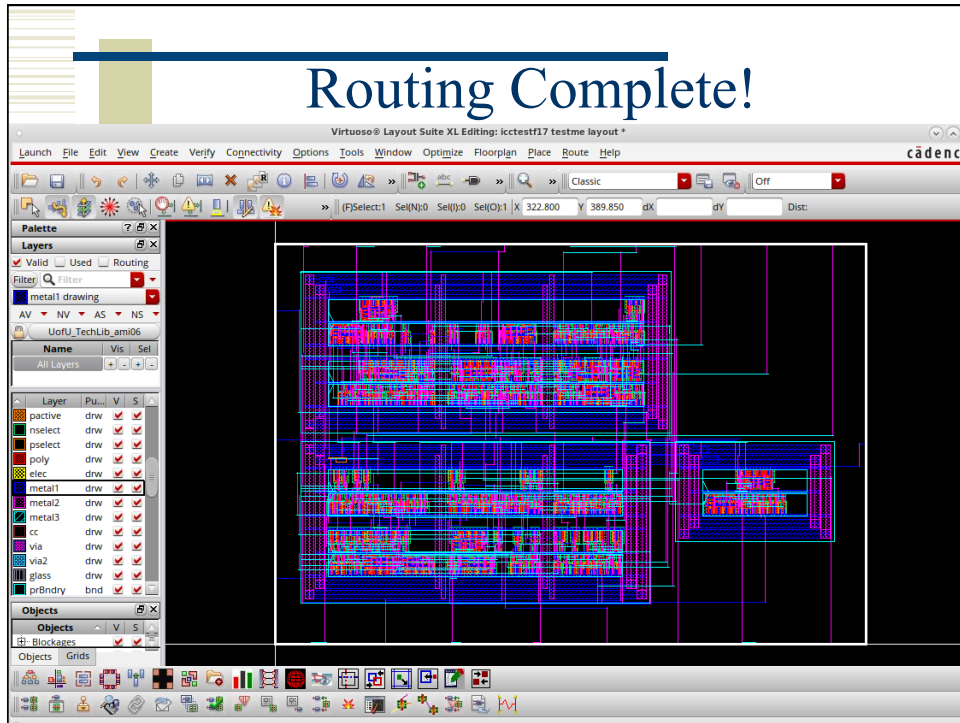


Ready to route!

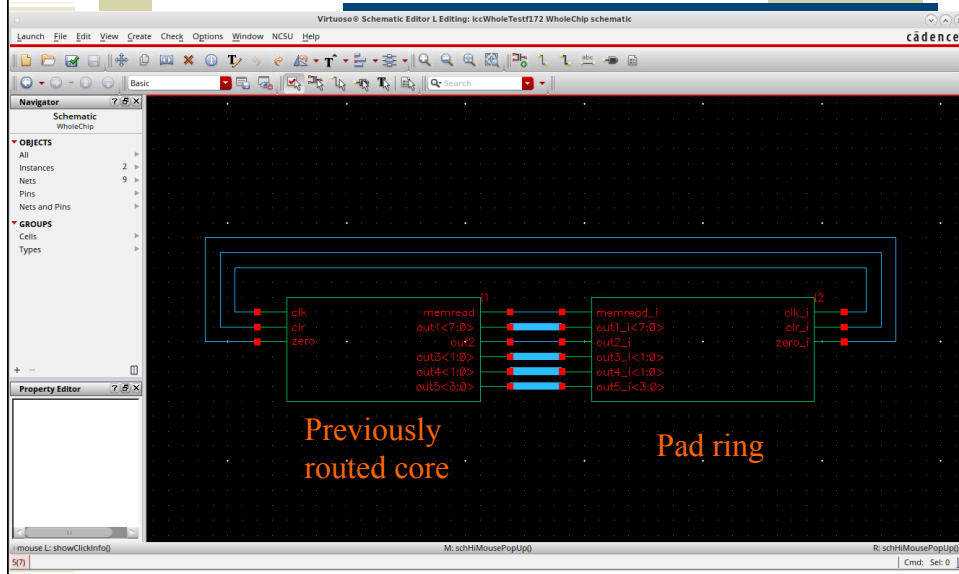


Route with vsr





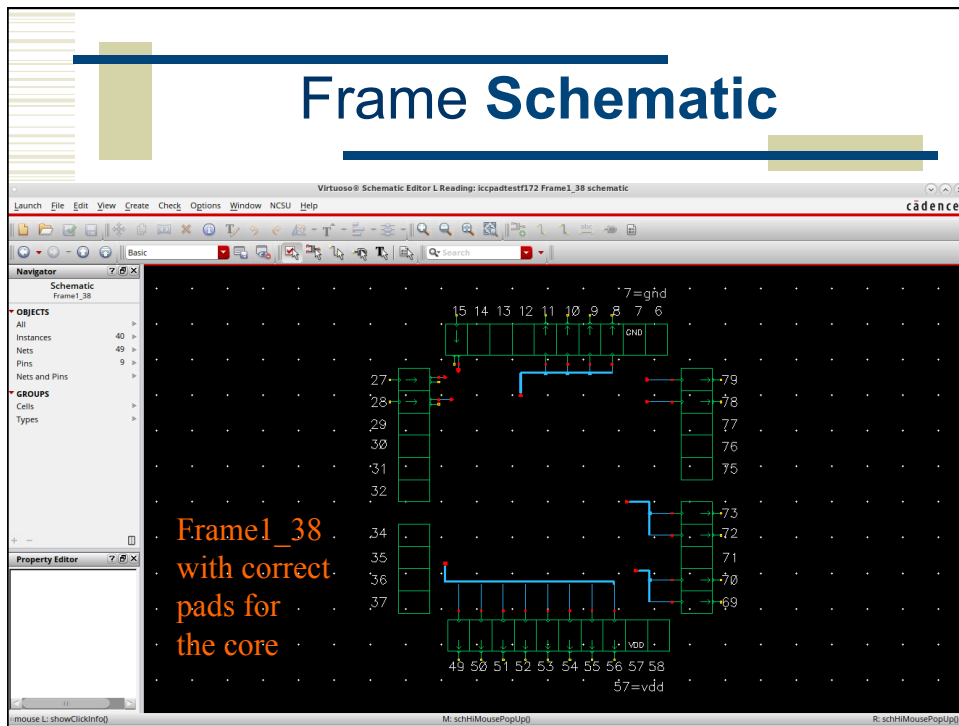
Pad Routing...



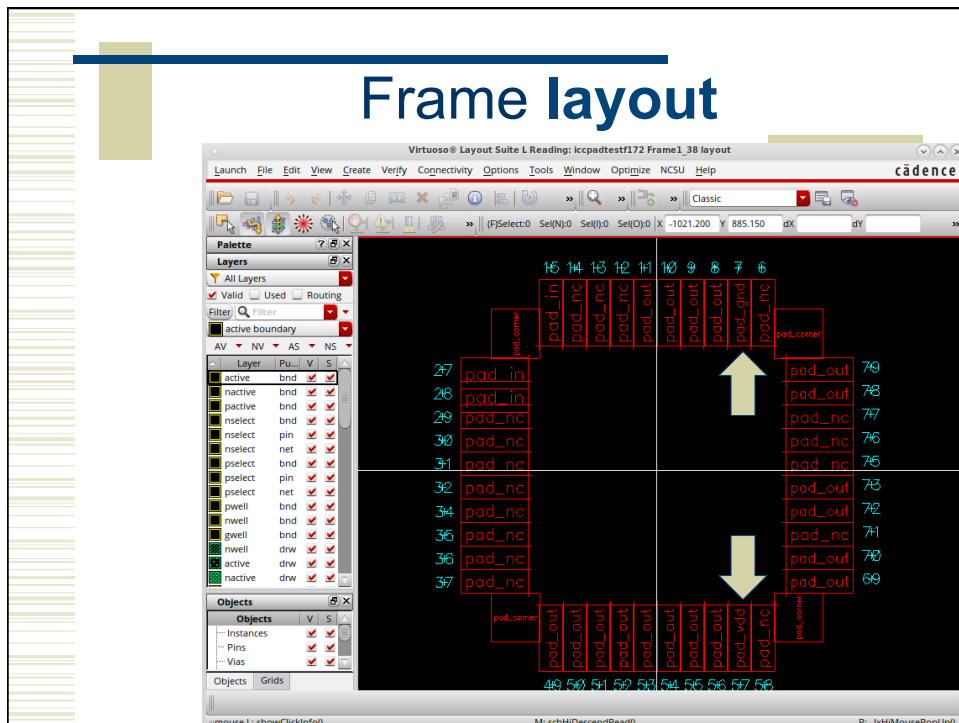
How to Use the Rings

- ◆ Copy the pad ring of your choice
 - `/uusoc/facility/cad_common/local/Cadence/lib/OA/UofU_Pads`
 - From `UofU_Pads` to your project directory
 - Leave the `pad_vdd` and `pad_gnd` where they are! (unless you're building your own test rig...)
- ◆ Select the other pads, use properties to change to the pad type you want
 - DON'T move them!
 - Use `pad_bidirhe`, `pad_out`, and `pad_in`

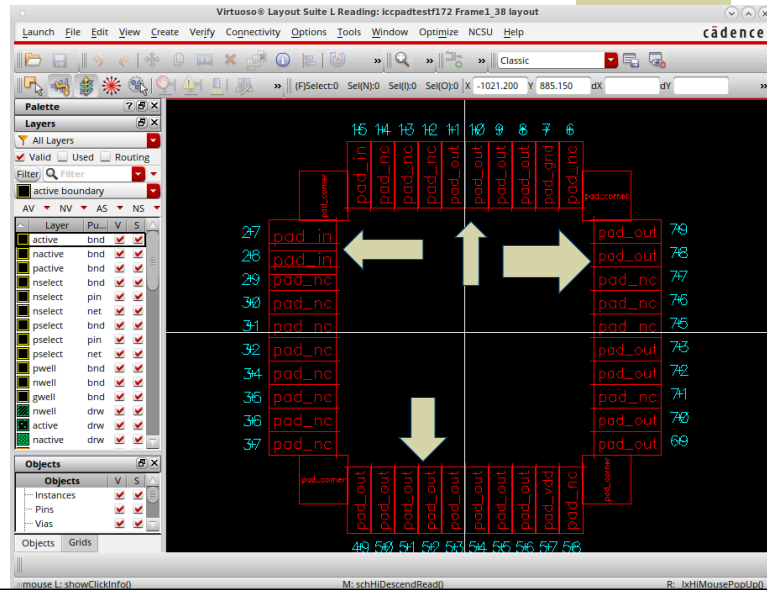
Frame Schematic



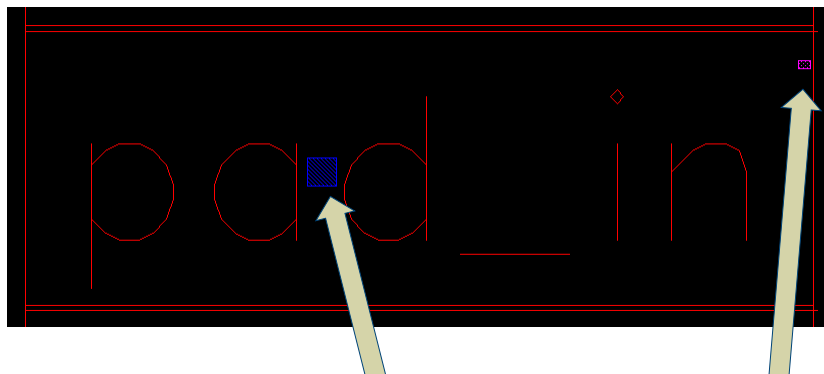
Frame layout



Frame layout

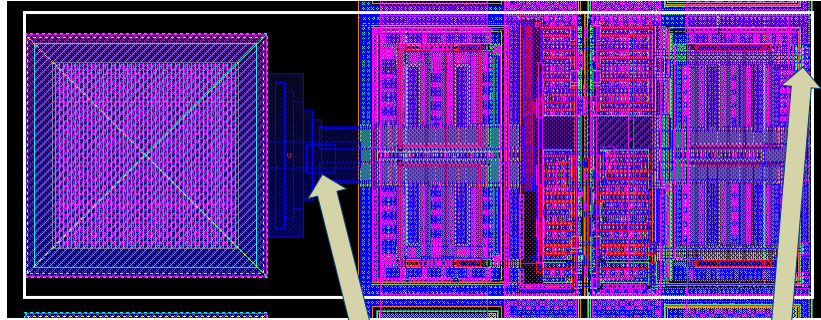


Pins



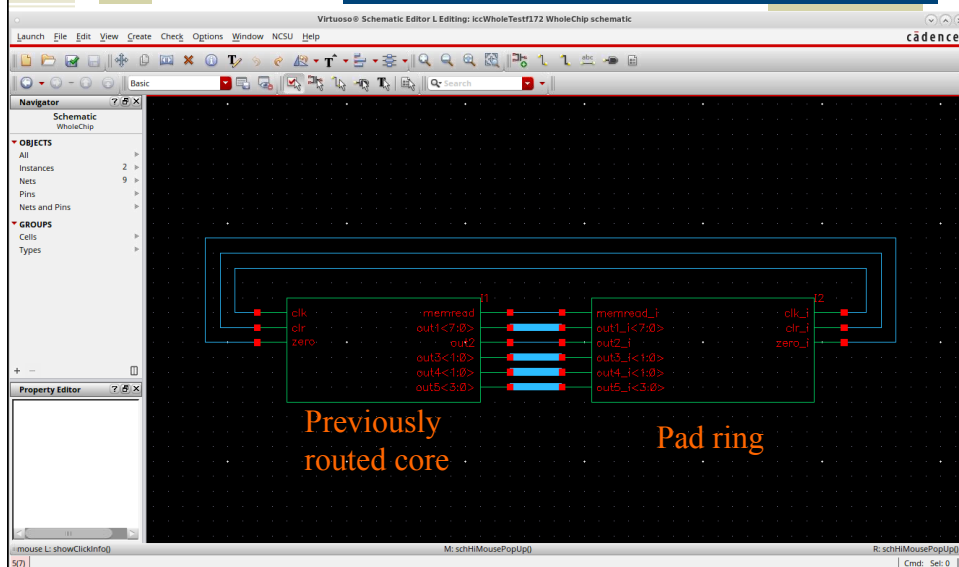
- ◆ Close up on layout pins. One for core connection, one for pad connection

Pins



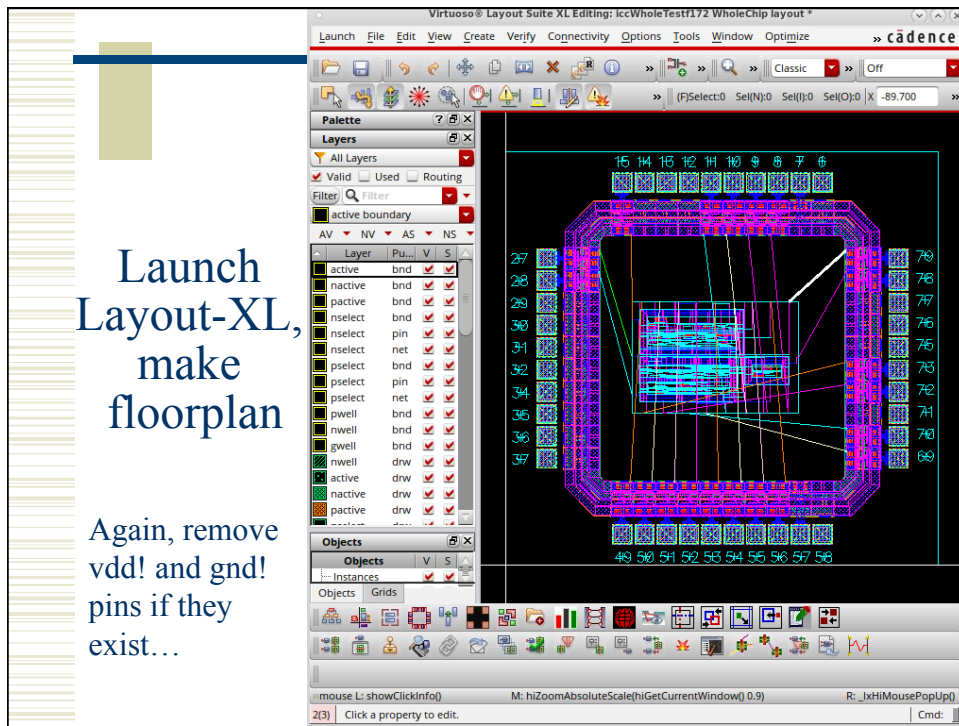
- ◆ Close up on layout pins.. One for core connection, one for pad connection

Pad Routing...

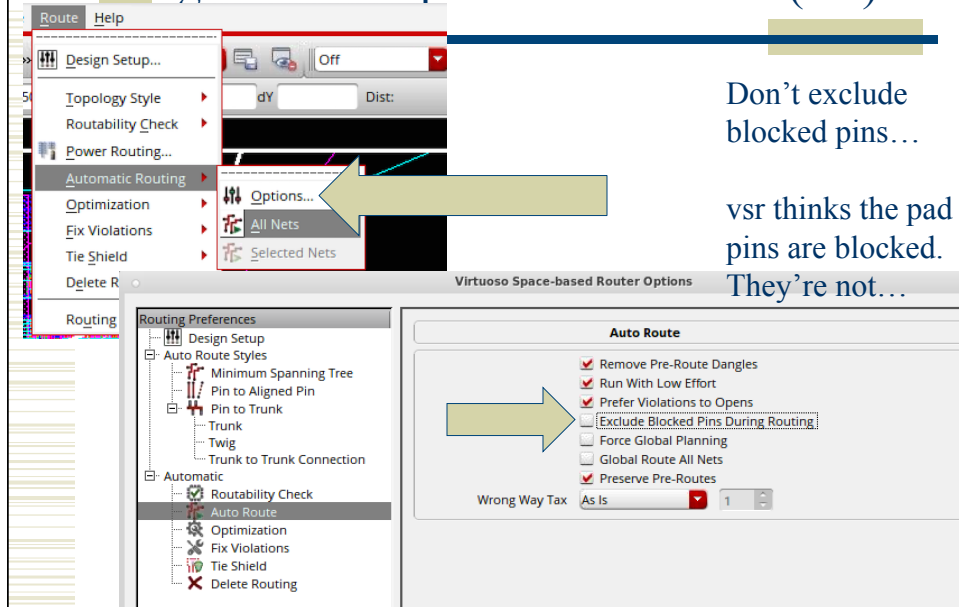


Launch Layout-XL, make floorplan

Again, remove vdd! and gnd! pins if they exist...



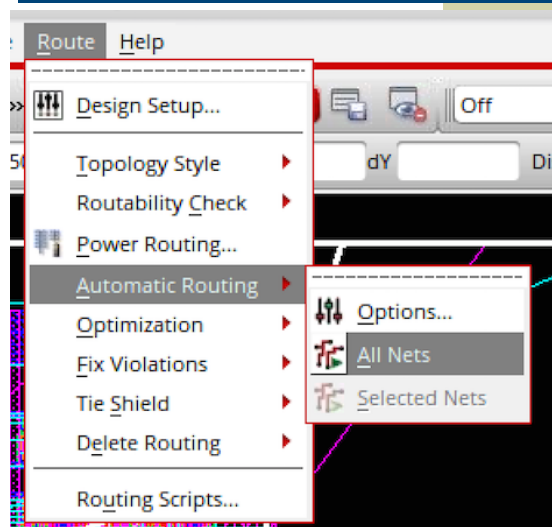
Using Virtuoso Space-based Router (vsr)



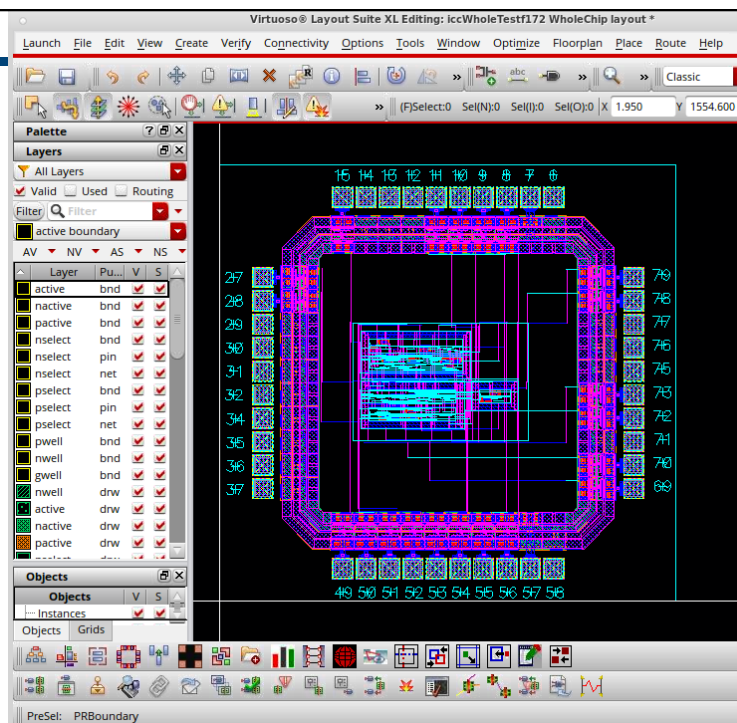
Don't exclude blocked pins...

vsr thinks the pad pins are blocked. They're not...

Fire up vsr

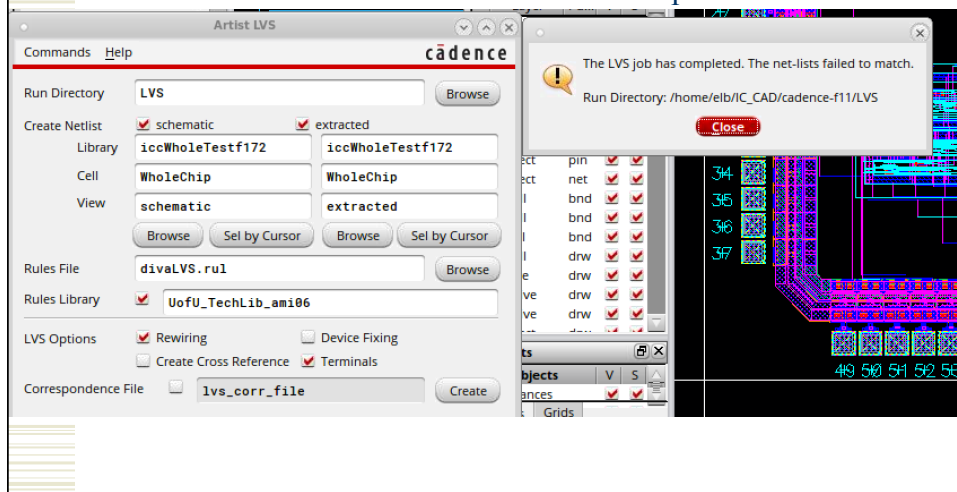


Routed!

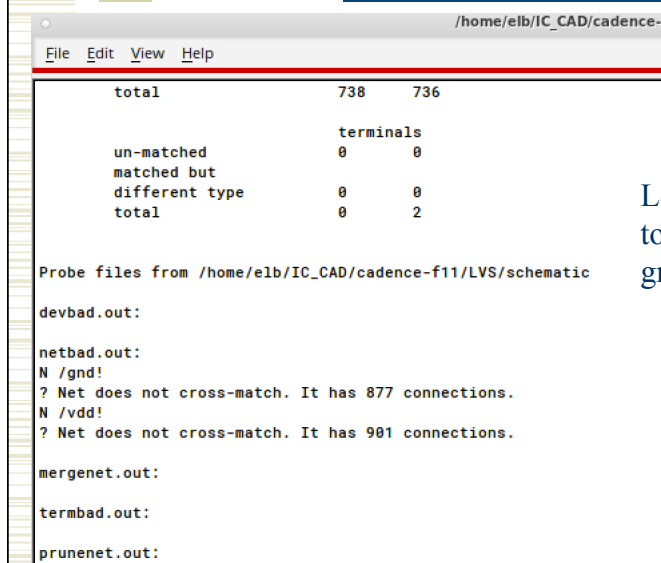


Routed!

Oops!!!!

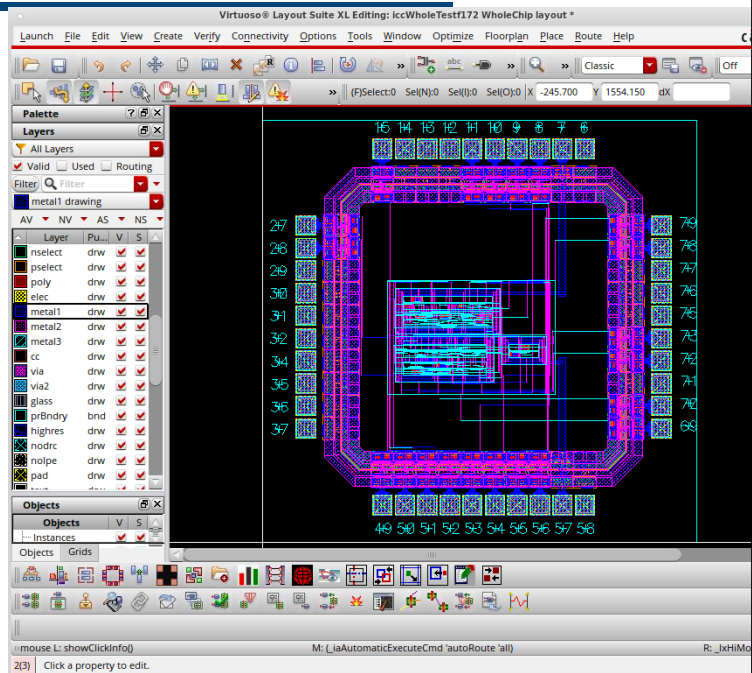


LVS...

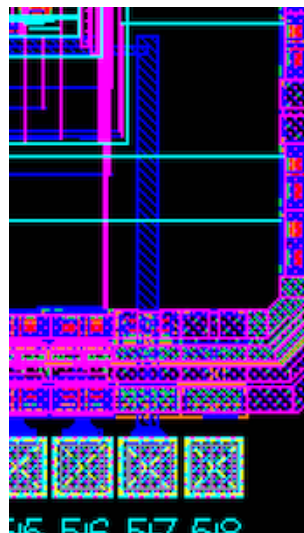
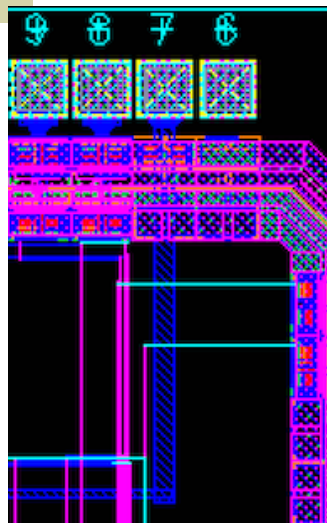


Looks like I forgot to connect vdd! And gnd!

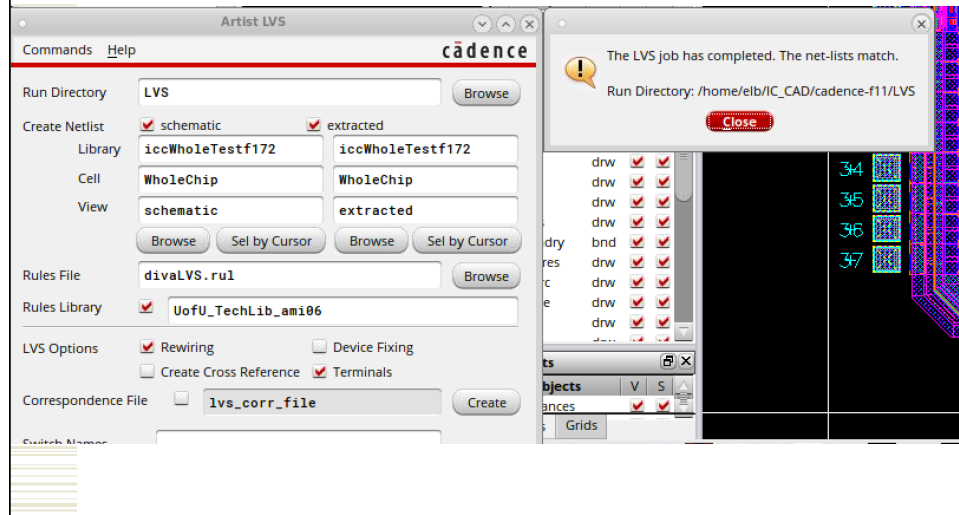
Connect
vdd!
and
gnd!
to pads



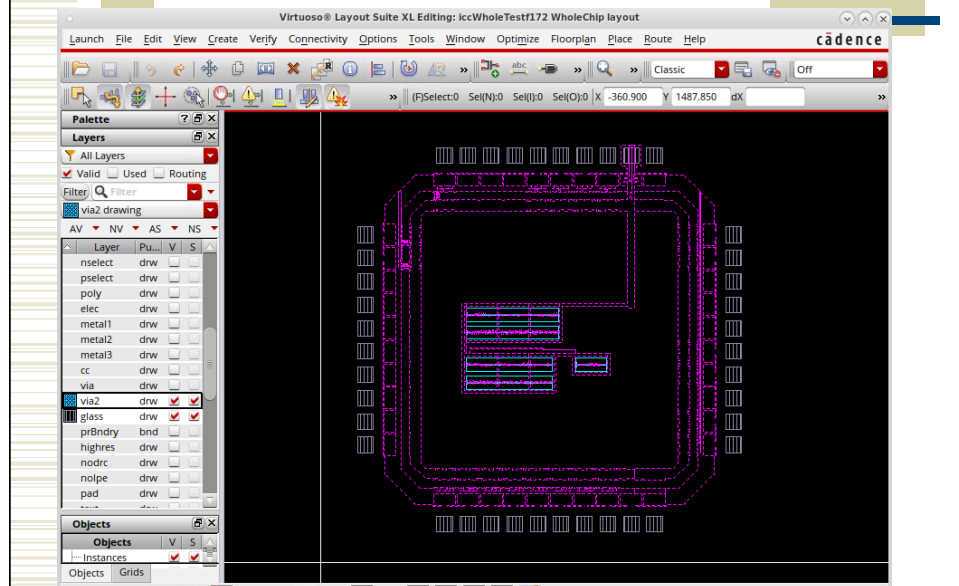
Power and ground pads...



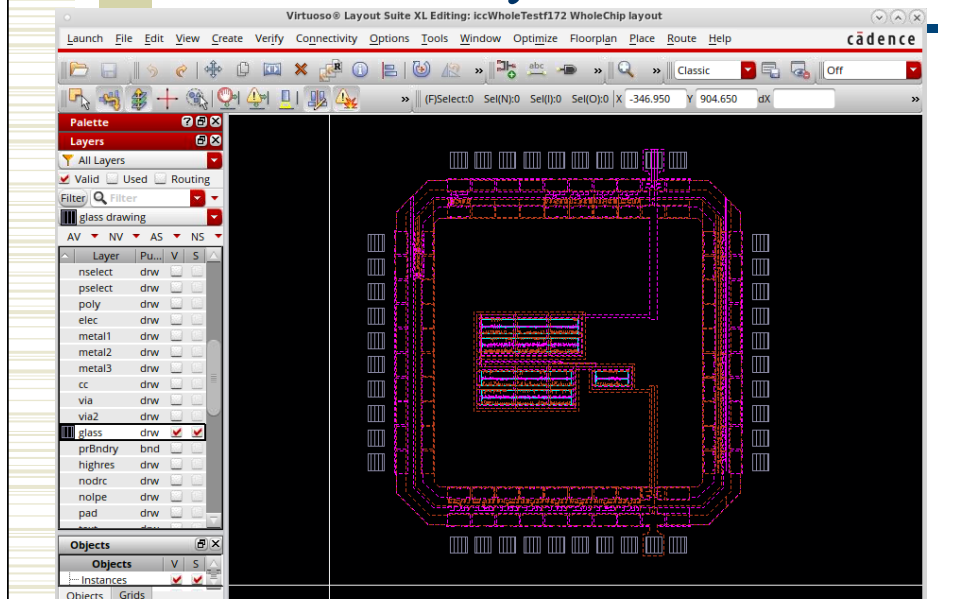
This time it worked!



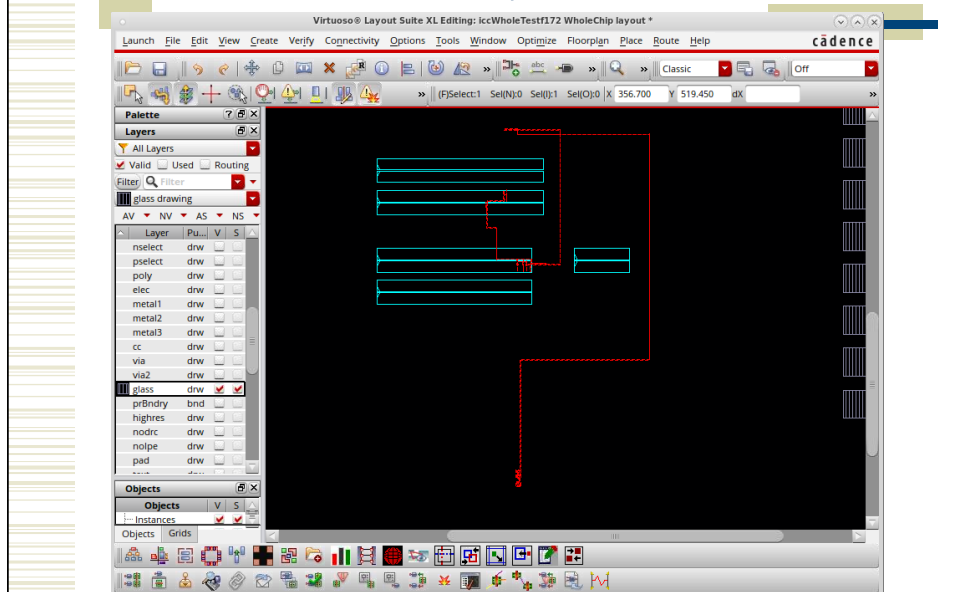
Connectivity -> Mark Net

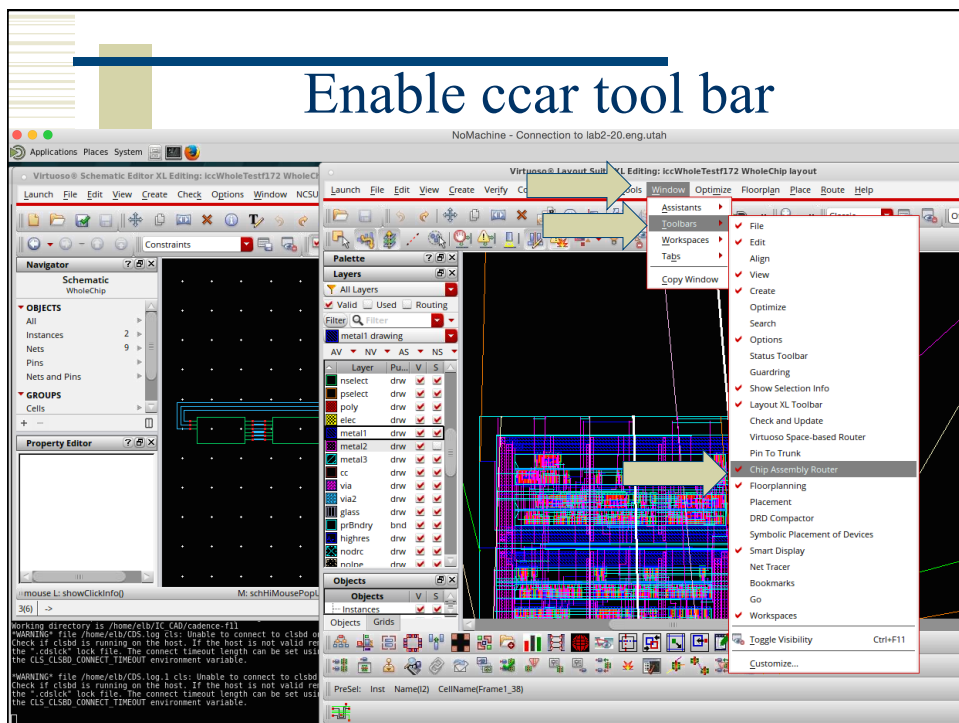
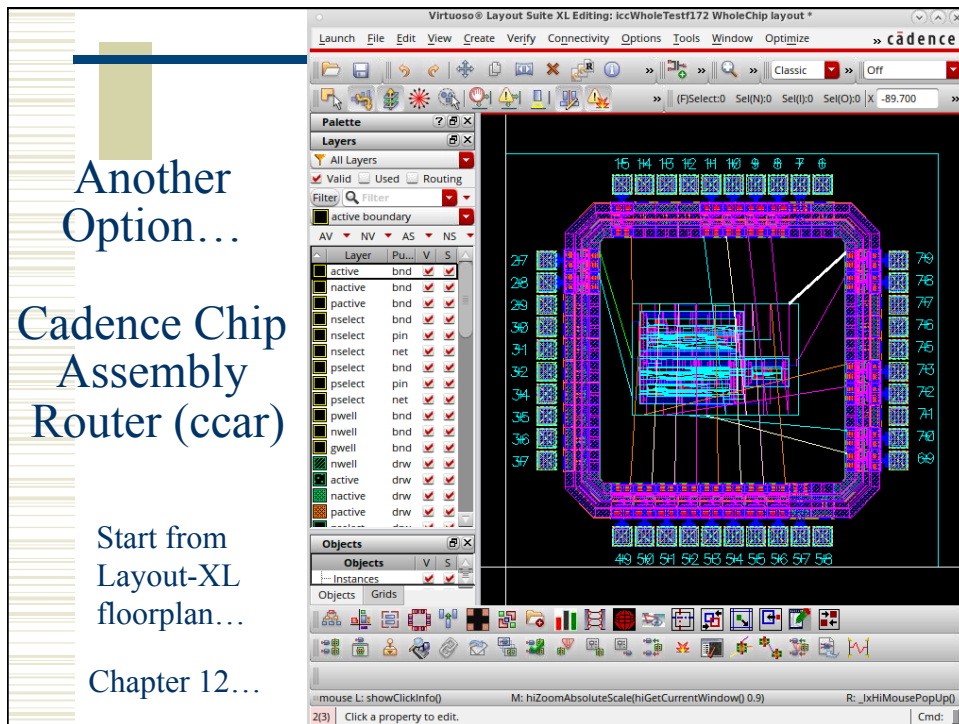


Connectivity -> Mark Net

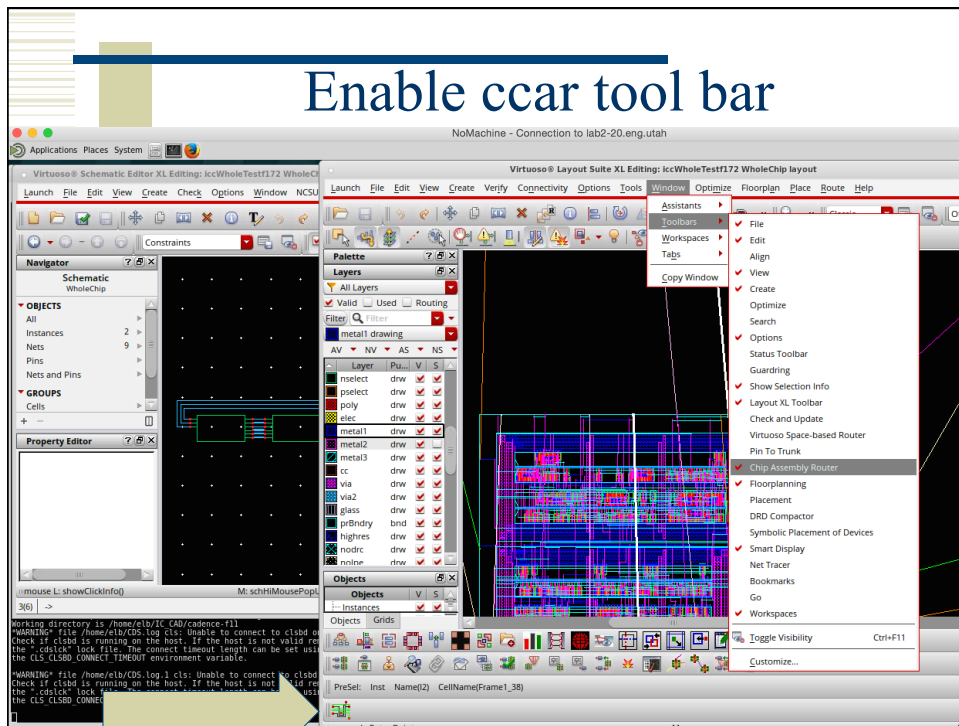


Connectivity -> Mark Net

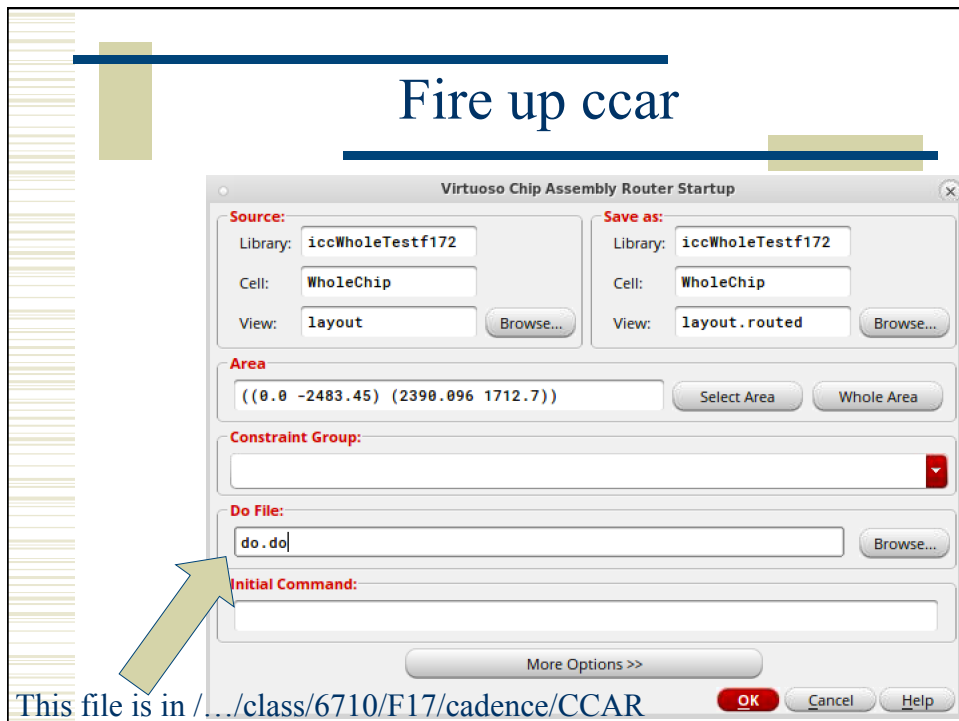




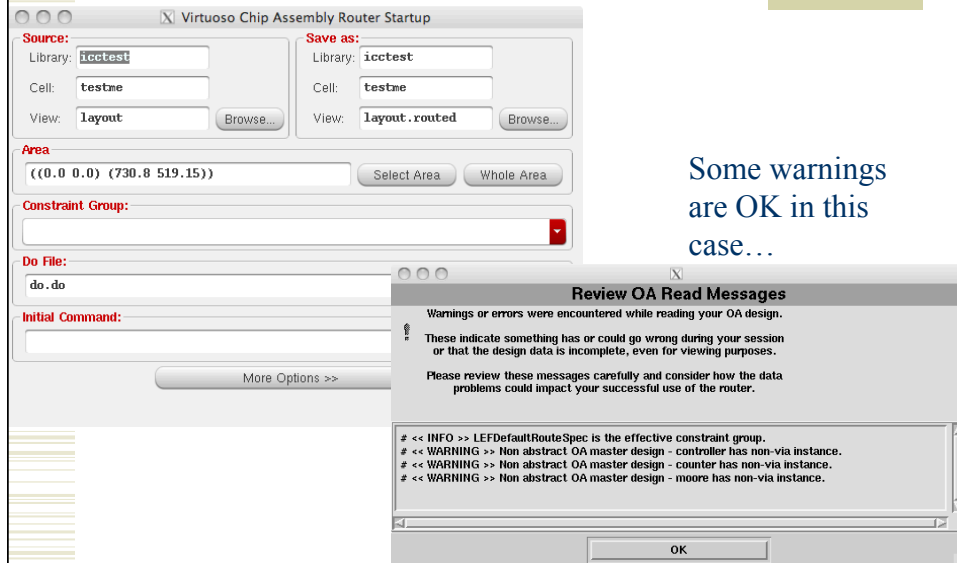
Enable ccar tool bar



Fire up ccar



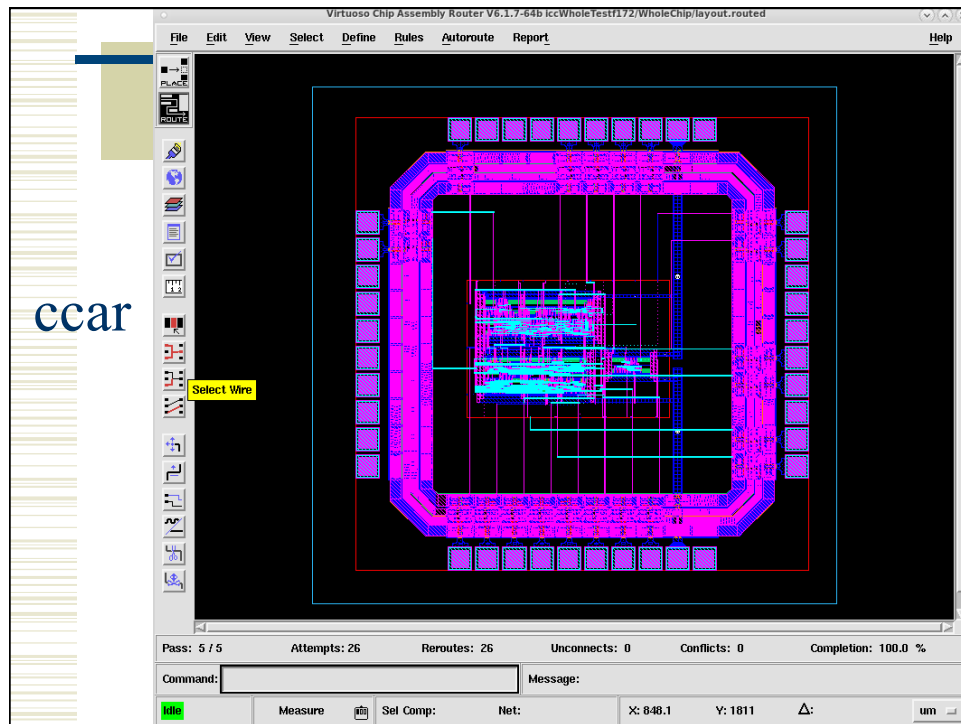
Export to vcar



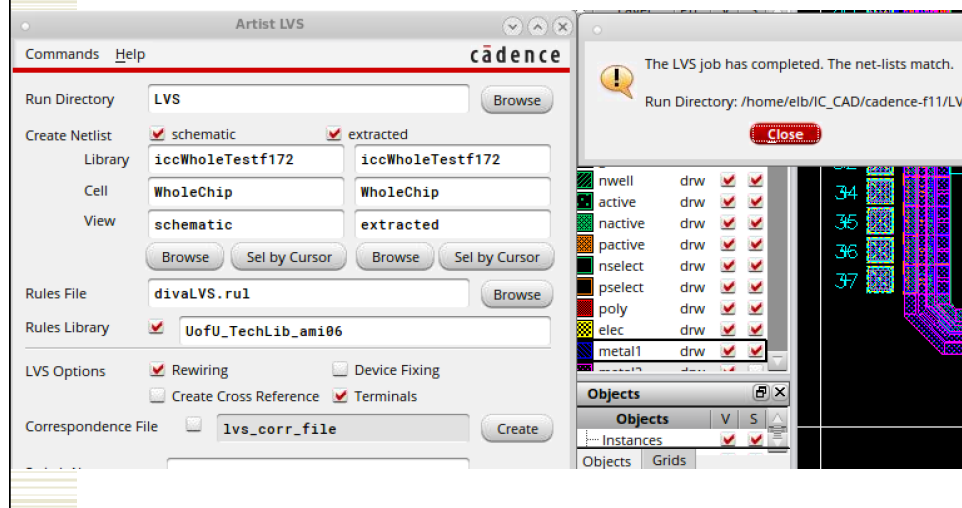
Chip Assembly Router

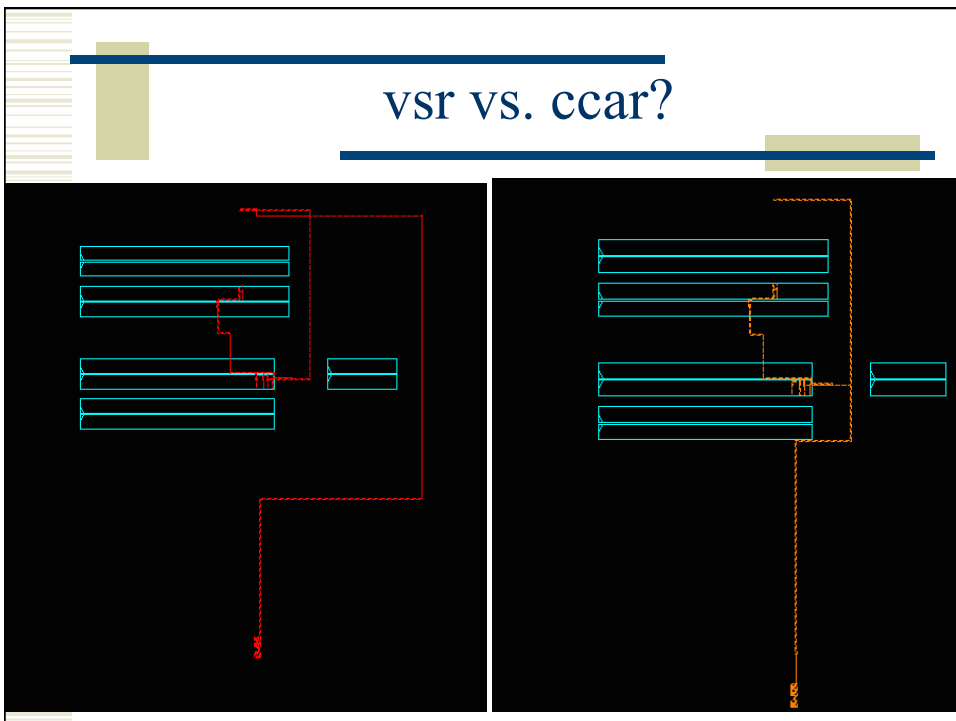
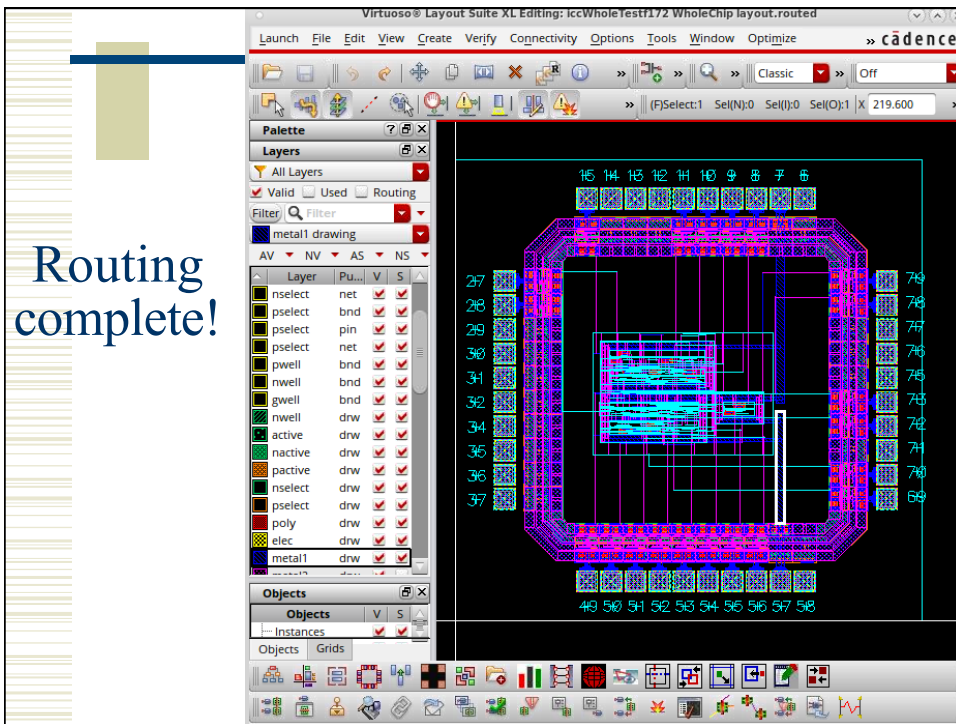
- ♦ Look in Section 12.1.2 in the CAD book
- ♦ Procedure is on page 380
 - Global Route Layer Direction setting
 - Global Route
 - Detail Route
 - Cleanup
 - Remove Notches

ccar

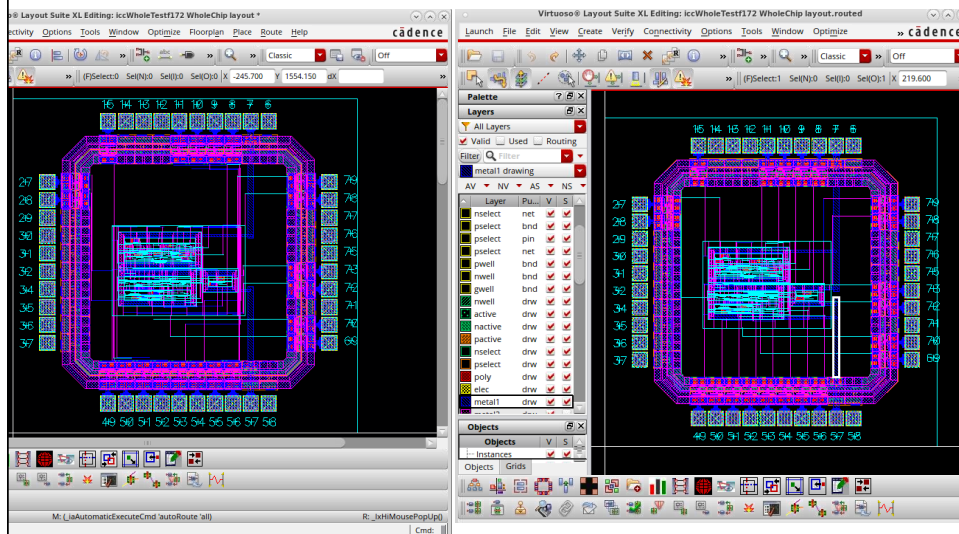


Routing complete!



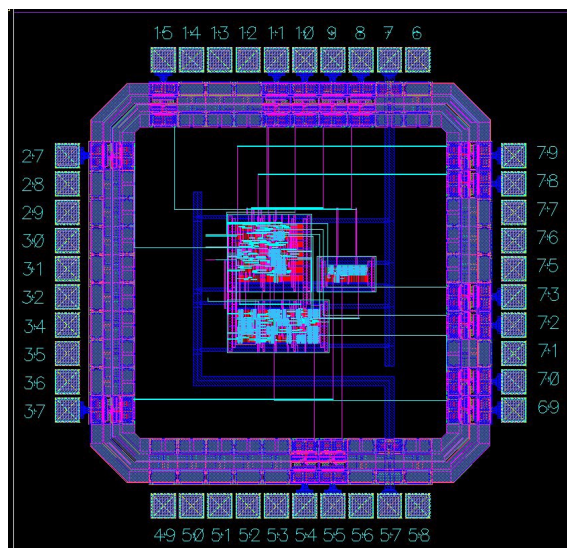


Either way you get a pad-routed chip!

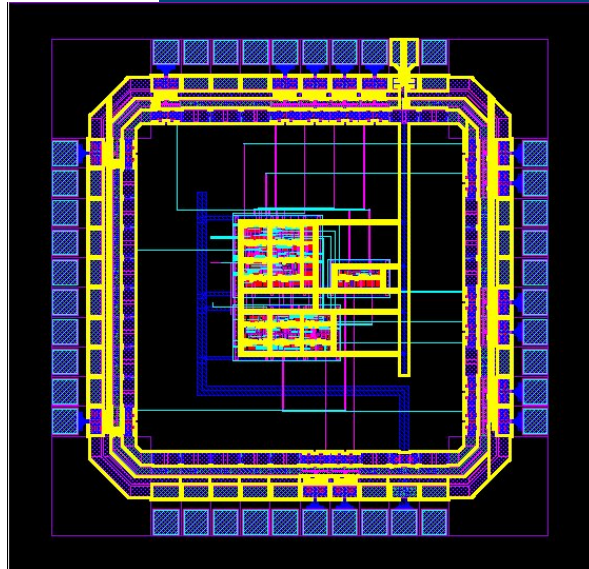


Another ccar example...

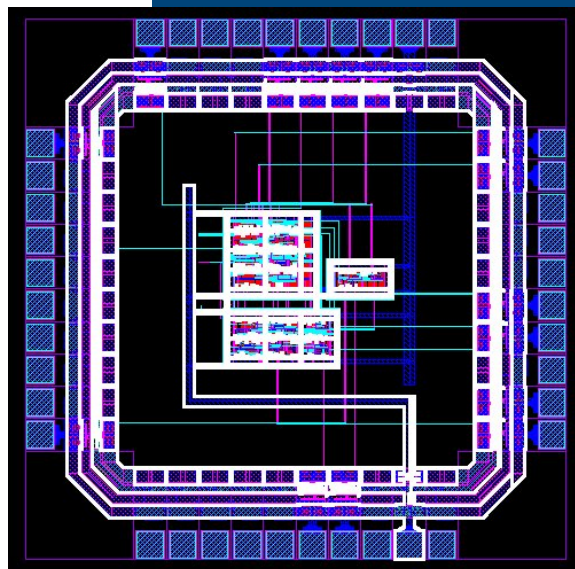
Note different
vdd! and gnd!
routing
strategy...



Check GND

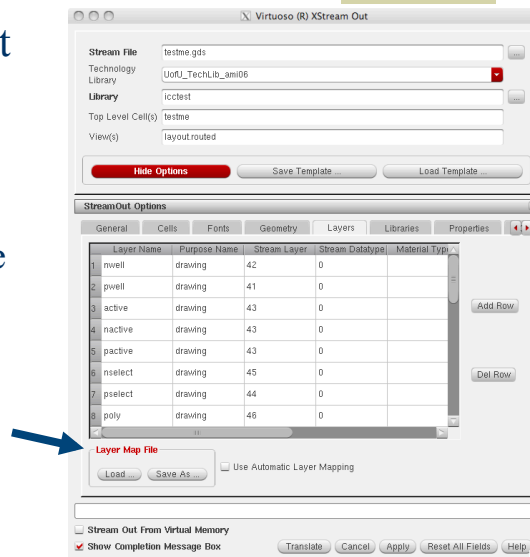


Check VDD



Now generate gdsII (stream)

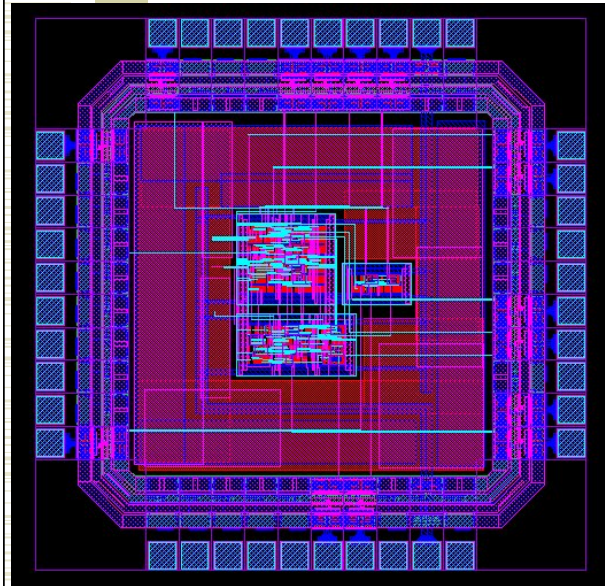
- ◆ The binary format that MOSIS wants
- ◆ Use **export->stream**
 - Make sure to load **stream4gds.map** as the Layer Map File



One Final Tweak

- ◆ If you're fabbing, before you generate your final fab-read gdsII (stream) file...
 - ...You need to add blocks of poly, M1, and M2 to meet the minimum density requirements
 - Take open areas of your chip and add large blocks of those layers
 - Remember to DRC and LVS to make sure you didn't mess anything up!

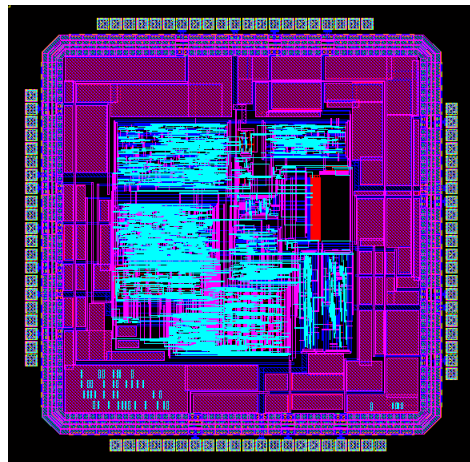
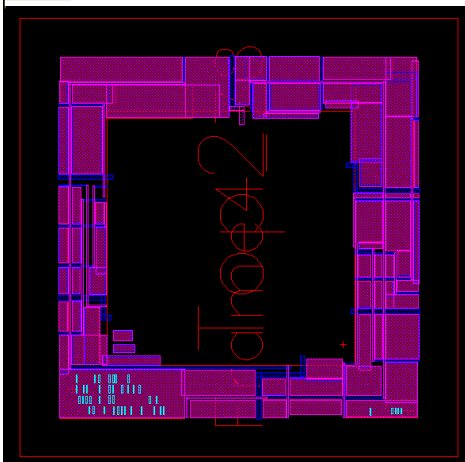
Add Fill



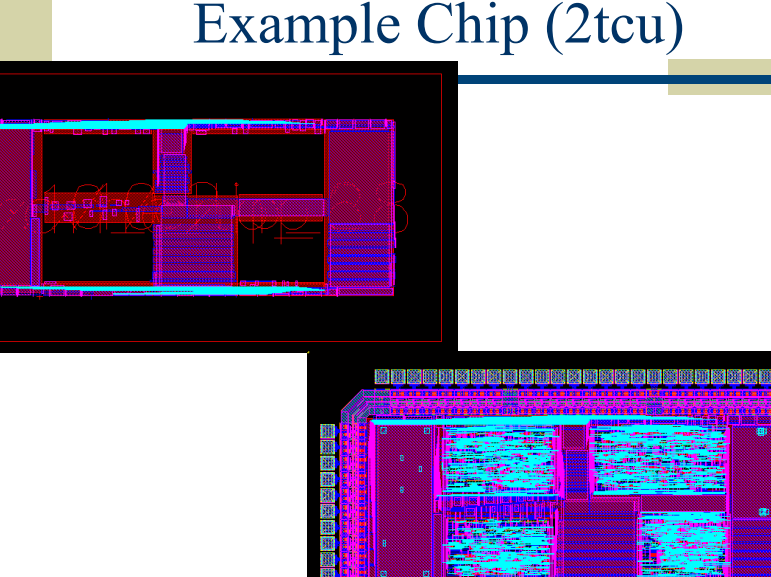
Fill in blank area
with big patches
of material...

Turning layer
visibility on and
off is a big help
in not messing up
here...

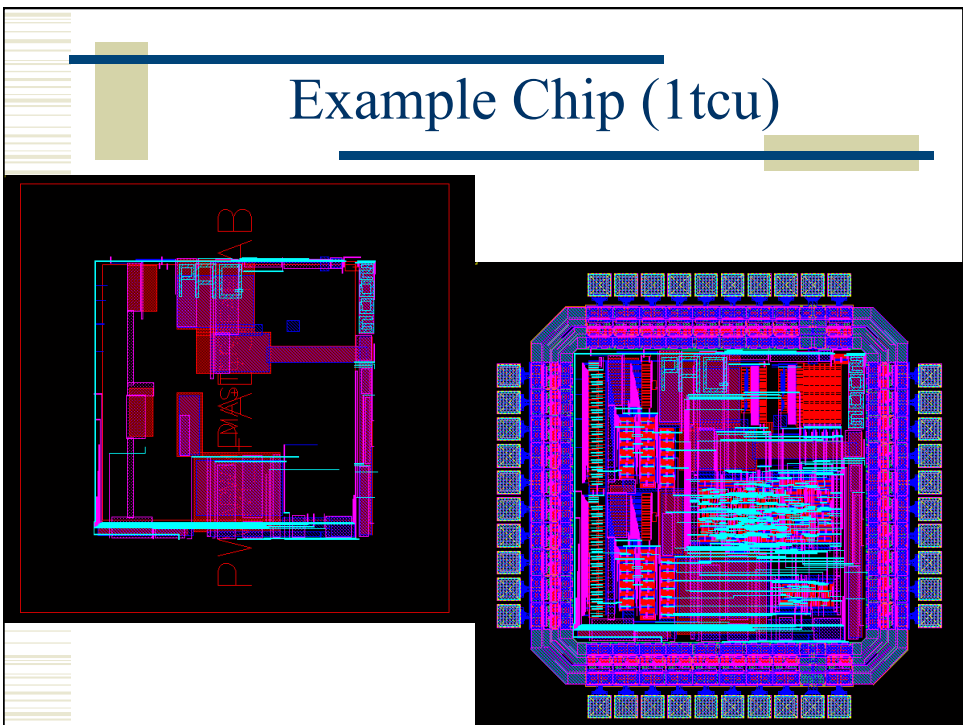
Example Chip (4tcu)



Example Chip (2tcu)



The top diagram is a high-level block diagram of the 2TCU chip. It shows a central processing unit (CPU) block, a graphics processing unit (GPU) block, and various memory blocks. The layout is symmetrical, with the CPU and GPU blocks positioned on the left and right sides, and the memory blocks in the center. The bottom diagram is a detailed physical layout of the chip, showing the intricate circuitry and components. It includes a central processing unit (CPU) block, a graphics processing unit (GPU) block, and various memory blocks. The layout is symmetrical, with the CPU and GPU blocks positioned on the left and right sides, and the memory blocks in the center.



Final Sanity Check

- ♦ If you can Import your GDS in to a new library, extract, and LVS against the “golden” schematic, you win!
- ♦ This makes sure that the GDS you’ll be sending to MOSIS is the same circuit as your schematic...

Summary

- ♦ A little bit of pain to make schematic of core-pads connectivity
- ♦ A little bit of pain to make pad ring layout
- ♦ Then, two choices for automatic core/pad routing
 - Virtuoso Space-based Router (vsr)
 - Cadence Chip Assembly Router (ccar)
 - Which one’s better? Nobody knows! They both work...