Chip Assembly Lab
ECEn 451
Dr. David A. Penry

In this lab you will use the Cadence tools to perform the top-level assembly of a chip. Just for reference (you don't need to know this to do the lab), the chip is a test chip for a serial interface block

Steps to follow
----

1. Create a new directory for this homework assignment. Change to that directory. Use the following command to untar the tarball for this lab which is available on Blackboard: The tarball contains the following directories and files:

   ```
   tar xzvf HWAssembly.tgz
   ```

   This command will create the following files:

HWAssembly	A Cadence library of cells you will need; the cells are:
spi_top	A serial port interface
tap_top	JTAG test access port
dbg_top	Debug controller (connects the JTAG to the SPI)
whole_chip	The top-level of the chip's design hierarchy

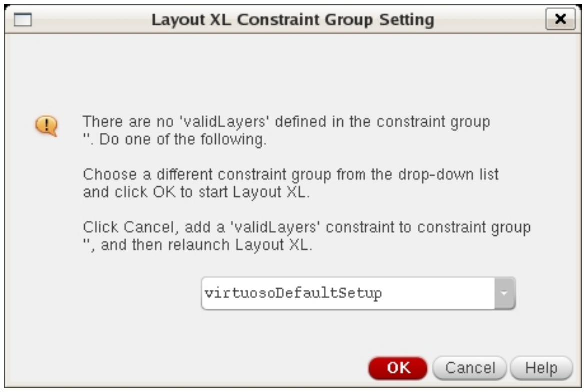
   Do not use libraries from previous labs; do all your work in the HWAssembly library. Note that this lab uses a different process (the AMI 0.6u process)

HWassembly.do	Commands to set up the router
cds.lib	cds.lib file defining paths to libraries

2. Prepare the initial layout

   1. Open the schematic for the whole_chip cell. This cell represents the top level of the entire chip's hierarchy. You'll see that there are four blocks inside the cell: the three logic blocks and the pad ring (or "pad frame")

   2. Start a layout session connected to the schematic. The idea here is to start the layout tool in such a way that it understands what the connections are in the schematic. To do this, you do **not** open the layout tool in the normal way. Instead, from the schematic window, choose Tools->Design Synthesis->Layout XL. You will see a dialog box entitled "Startup Option". Chose "Create New" and click on OK. You'll see another dialog box which asks about the cell view name to create. It should be "layout" by default, so you can just click on "Create New File". You may also get a dialog box that looks like:

      There are no 'validLayers' defined in the constraint group ''. Do one of the following.

      Choose a different constraint group from the drop-down list and click OK to start Layout XL.

      Click Cancel, add a 'validLayers' constraint to constraint group '', and then relaunch Layout XL.

      virtuosoDefaultSetup

      If you do get such a dialog box, change the drop-down menu box to read "virtuosoDefaultSetup" and press OK.

      You should now see an empty layout window.

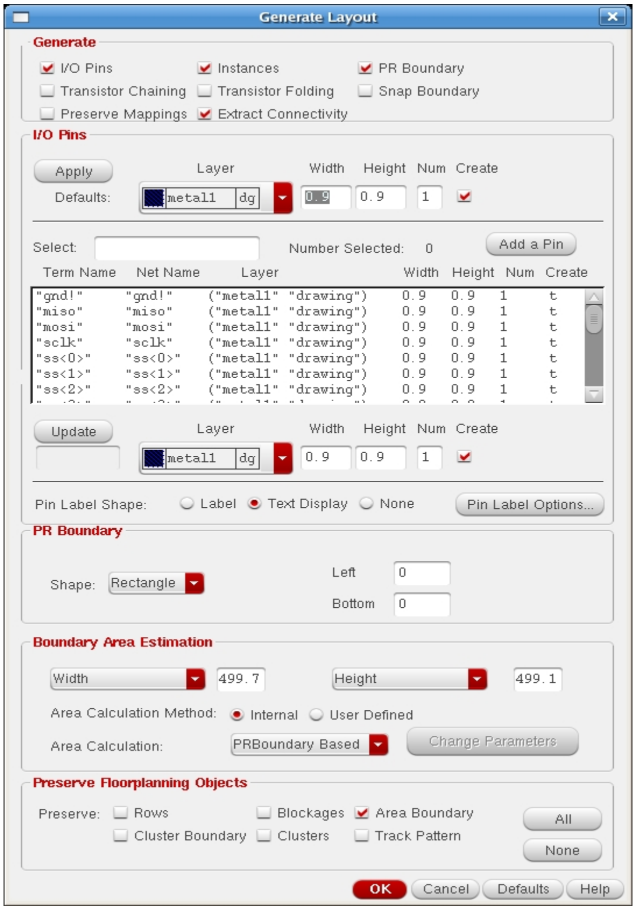
NOTE: If you quit your layout session and later reopen it, you'll need to reopen the layout in the same way: by opening the schematic and choosing Tools->Design Synthesis->Layout XL. However, in the Startup Option dialog box, chose the "Use existing cell" (or something like that) option.

3. Generate an initial bounding box and layout. The bounding box is used to show later tools how big the design is intended to be. The initial layout will include all of the modules and pins. To create it, select Connectivity -> Generate -> All from Source. A dialog box will appear. Change the entries in the following fashion:

   1. Set the design boundaries (i.e. how big the chip will be). Go to the "B oundary Area Estimation" portion of the dialog box and change the two pulldown menus on the right to read "Width" and "Height". Enter a width of 4499.7 and a height of 4499.10 in the boxes to the right of each pulldown menu.

   2. Select "Text Display" in the Pin Label Shape radio buttons

   3. The dialog box should look like: (NOTE: the width/height boxes aren't big enough to show the whole number)



☑ I/O Pins ☑ Instances ☑ PR Boundary
☐ Transistor Chaining ☐ Transistor Folding ☐ Snap Boundary
☐ Preserve Mappings ☑ Extract Connectivity

Defaults: Layer: metal1 dg; Width: 0.9; Height: 0.9; Num: 1; Create: ☑

Select: ; Number Selected: 0

```
Term Name  Net Name  Layer                 Width  Height  Num  Create
"gnd!"     "gnd!"    ("metal1" "drawing")  0.9    0.9     1    t
"miso"     "miso"    ("metal1" "drawing")  0.9    0.9     1    t
"mosi"     "mosi"    ("metal1" "drawing")  0.9    0.9     1    t
"sclk"     "sclk"    ("metal1" "drawing")  0.9    0.9     1    t
"ss<0>"    "ss<0>"   ("metal1" "drawing")  0.9    0.9     1    t
"ss<1>"    "ss<1>"   ("metal1" "drawing")  0.9    0.9     1    t
"ss<2>"    "ss<2>"   ("metal1" "drawing")  0.9    0.9     1    t
```

Layer: metal1 dg; Width: 0.9; Height: 0.9; Num: 1; Create: ☑

Pin Label Shape: ○ Label ● Text Display ○ None

Shape: Rectangle; Left: 0; Bottom: 0

Width: 499.7; Height: 499.1

Area Calculation Method: ● Internal ○ User Defined

Area Calculation: PRBoundary Based

Preserve: ☐ Rows ☐ Blockages ☑ Area Boundary
☐ Cluster Boundary ☐ Clusters ☐ Track Pattern

   4. Press OK. The generated layout appears. The blue rectangle is the boundary of the chip; the red rectangles are each of the cells in the chip: 3 internal logic cells and the padring. Note that the padring is as big as the chip.

3. Perform floorplanning. Unfortunately, the floorplanning tool doesn't do what we want in this case, so you'll need to do some things by hand. Here's how you do it:

   1. Turn on the display levels ('e' key) so that you can see the inside of the blocks.

   2. Move the pad ring (that's the block whose layout looks like a squared ring) inside the purple rectangle. The easiest way to do this is to edit its properties ('q' key) and change its location to coordinates 0,0.

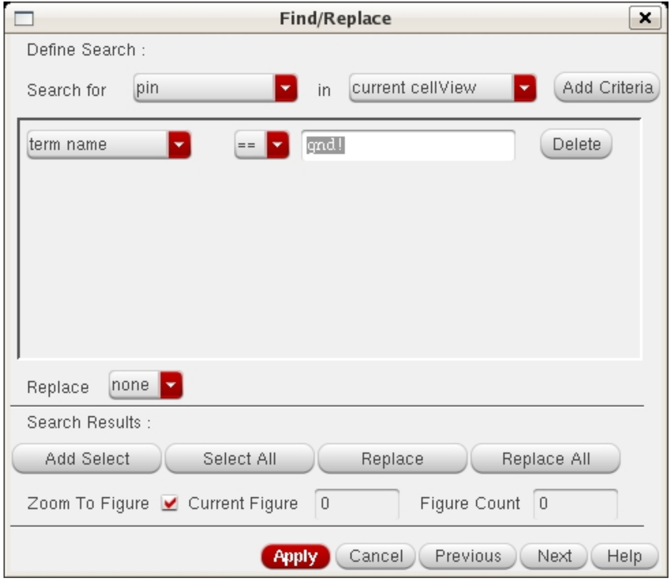
   3. Rotate each of the other blocks. In our design methodology, we're going to have metal 2 run horizontally and metal 1 and 3 run vertically. As they are currently oriented, the metals go in the right direction. So, you need to rotate each of the internal blocks (but not the pad ring!) by 90 or 270 degrees. The easiest way to do this is to edit the instance properties of each ('q' key) and change the orientation (the drop-down box at the bottom).

   4. Now, move each of the blocks inside of the pad ring. Make sure you don't overlap the circuitry. You'll notice that a bunch of yellow lines appear as you drag the blocks; those yellow lines indicate connections that will have to be made between the pins of blocks. Try to place your blocks so that you won't have to route around or over other blocks and so that connections don't cross each other much. You may need to change the rotation of the blocks to achieve this.

      Also, make sure there's a decent amount of separation (at least 100-150 microns) between blocks and between the blocks and the pad ring.

   5. Move the layout pins onto the pads of the design. The pins are tiny and hiding near the origin. However, Cadence can put them where we need them. Select Place->Pin Placement. Go to the “Pin Optimization” tab and press the “Optimize” button.

      The pin optimization routine doesn't get vdd! and gnd! quite right, which isn't surprising because they're “global” signals which have many connections throughout the pad ring. You'll need to move them to the correct locations. Move vdd! to the wide “tongue” of metal sticking towards the interior of the chip in the center of the top side of the pad ring. Move gnd! to the corresponding tongue on the bottom side of the pad ring. To find these pins in all that huge design, use the “Find/Replace” function (press 'S' to get it), select “pin” in the “Search for” drop-down menu, press Add Criteria, and add the pin name you want. Click on “Zoom to Figure” and press Apply. Here's an example of the dialog box:



Search for pin in current cellView

term name == gnd!

Replace none

Zoom To Figure ☑ Current Figure 0 Figure Count 0

      Note that the vdd! and gnd! pins will seem to change connection lines as you move them and won't connect to a pad directly.

   6. **Now would be a good time to save your layout and make a backup copy you can return to if things go wrong later. Save and then use the Library Manager to copy the layout view to a “layout_after_placement” view.**

4. Add a power grid. This is often done by scripts written by a project CAD team; we will do it manually.

   1. Place a horizontal pair of metal2 wires, each 28.8 microns wide, across the top and bottom of the chip interior. Do not run them all the way into the pad ring.

    2. Connect the wide metal2 wires to the power and gnd pad elements which are at the top and bottom of the chip. The pads can be recognized easily because they don't have as much circuitry as the signal pads. Power and gnd should be delivered from the pads on wide metal1 wires the size of the “tongue” sticking out from the pad; connect these wires to the metal2 wires you just drew through a large array of vias. (19 by 19 should fit. To create an array of vias, select Create Contact ('o'), set the Contact Type to M2_M1, and select 19 rows and 19 columns.) Make sure you have a wide metal2 gnd bus and wide metal2 vdd bus on both the top and the bottom.

    3. Draw an array of vertical 1.8 micron-wide metal3 wires between the metal2 power busses you've created and connect them to the appropriate busses using arrays of vias. vdd and gnd wires should be interleaved with adjacent wires at 102 microns center-to-center spacing. This is a real pain to do one by one; a much easier way is to draw in one wire with its vias and then copy it using array copy. To do an array copy, select what you want to copy and then hit the copy key ('c'). (If the Copy dialog box doesn't appear, press F3.) Set “Array Columns” to the number of copies (17 or 18 will probably be what you want), Snap Mode to horizontal, and Delta-X to 204. Then press the ApplyXY button twice.

    4. Slide the 3 already-routed sub-blocks left or right until their metal3 vdd and gnd wires line up with the array you just created. Be sure that you're lining up the right wires and aren't connecting vdd to gnd! If you've rotated the blocks by 270 degrees, the leftmost metal3 power/gnd wire within the block will be gnd; if you rotated the blocks by 90 degrees, the rightmost metal3 power/gnd wire within the block will be gnd.

    5. Double check that the metal3 grid you drew and the metal3 lines in each of the sub-blocks coincides. If they don't and you can't fix it by sliding, you've drawn your metal3 grid incorrectly.

    6. Run DRC. Fix any errors. DRC will take several minutes to run; you can watch its progress in the virtuoso window. If you get really strange DRC errors, it may mean that you have lined up the vdd/gnd wires of the sub-blocks backwards and have accidentally shorted vdd to gnd.

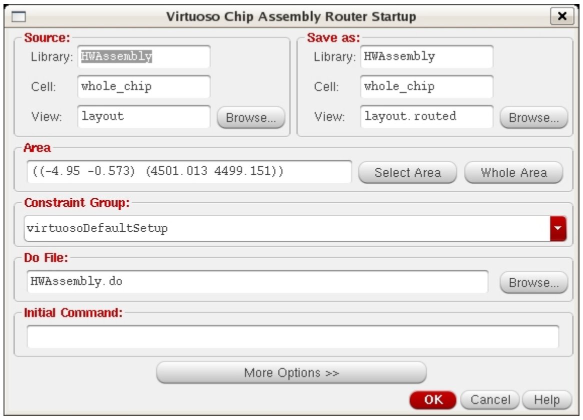
    7. **Copy your layout cell view to a layout_before_routing backup cell view using the Library Manager so you won't have to do the power grid again if something goes wrong in routing. Your final grid will look something like:**

5. Route the top-level signals. We will use the “Chip Assembly Router”, a typical routing tool.

    1. Change the cellview of each of the blocks (padframe, dbg_top, tap_top, spi_top) to the abstract view. This can be done by editing the instance properties. Abstract views have nothing but metal in them; we use these views because the router is only going to work with metal and would run much more slowly if it had all of the layers in its database.

    2. Select Windows->Toolbar->Chip Assembly Router to add a “Start Router” button to the tool bar at the top of the Layout XL window.

    3. Press the Start Router button. A dialog box will appear. Enter “HWAssembly.do” in the Do file box and press OK. The dialog box looks like:



Source:
Library: HWAssembly
Cell: whole_chip
View: layout

Save as:
Library: HWAssembly
Cell: whole_chip
View: layout.routed

Area: ((-4.95 -0.573) (4501.013 4499.151))

Constraint Group: virtuosoDefaultSetup

Do File: HWAssembly.do

Initial Command:

4. Wait for several minutes for the router to start up. You might wish to look at the Chip Assembly Router tutorial, starting at the "Using the Router" portion to get an idea for what the various windows will look like.

   Between executing commands, the router window sometimes redraws itself over and over. We don't know why it does this, but it usually will stop this after a few iterations. If it does it for more than 5 minutes, you probably will need to kill the router process and start it over. This problem seems to occur less if you make sure you don't put other windows over the router window and don't minimize the router window. Note that the router window will also redraw during routing; this is because it shows you the new routes as it goes along and is normal behavior.

   1. Go to Autoroute -> Pre Route -> Pin Escape and press OK in the dialog box to perform an initial routing step for "easy" wires.

   2. Go to Autoroute->Global Route->Local Layer Direction and choose Layer Panel on the dialog box. Click OK. This tells the router the preferred routing directions for each metal layer.

   3. Go to Autoroute->Global Route->Global Router and click on OK in the dialog box. Wait while the router runs 3 passes of the global router (basically a planner for routing.

   4. Go to Autoroute->Detail Route->Detail Router and click on OK in the dialog box. Wait while the router runs up to 25 passes of the detailed router.

   5. Go to Autoroute->Clean and press OK in the dialog box to run a cleanup router

   6. Go to Autoroute->Post Route->Remove Notches to remove notches from wires

   7. If any nets are still unconnected (there's a status line below the layout which says "Unconnects"), you'll have to route them manually; if there are any conflicts (again on the status line), you'll need to resolve them manually as well. The buttons on the left side of the screen allow you to edit routes. If you have large numbers of unrouted wires or conflicts or cannot figure out how to fix them, quit the router without saving, and change your floorplan by moving the blocks around or rotating or flipping them. Be careful to line up the metal3 vdd/gnd wires properly if you do so. Then try the router again.

      NOTE: the router uses completely different commands from everything else in Cadence. This is because Cadence bought the company that made the router and didn't make the router consistent with their own products. This is standard procedure for CAD companies.

   8. If all went well, quit and save from the router and you'll have the routing back in your layout in a new window (with a new cell view name: layout.routed)

5. Copy the layout.routed view to the layout view using the Library Manager. You will need to close your layout
)

window before you do this.

    6. Open the layout cellview again using Layout XL.

    7. Change the cell views of the four cells you changed in step 1 back to layout. **If you forget this step, DRC will fail.**

6. Run DRC and LVS. (Look back at the instructions for earlier labs which tell you how to do this.) Fix any errors you find. There will likely be a few DRC errors; the router sometimes places vias right up against the noDRC boundary (the dotted blue line) within the pad ring and thus DRC can't see all of the metal around the via. If this happens, just move the via inward towards the center of the chip.

   LVS errors indicate either that not all nets were routed or that you've not lined up the metal3 vdd and gnd wires properly.

7. Optional: there is a long-standing tradition of adding artwork such as logos or initials to layouts. This artwork should be added in a metal layer but must be designed so as to not cause DRC or LVS to fail. Making DRC pass is a matter of drawing taking into account the design rules; the easiest way of making LVS pass is to connect every chunk of metal to gnd.

Deliverables

Provide the following in your report:

1. Names of team members (teams of up to two, please)
2. A plot of the "extracted" view of the chip; because this view has only metal and poly in it, it is a bit easier to read than a full-chip plot.
3. DRC output and LVS output.

You must provide hard-copy; electronic submissions will not be accepted.

Be aware that if you choose to work in a team that you are both responsible for knowing how to use all the tools. There will be exam questions which cover tool usage.