1. Introduction

In the previous lab session we generated the synthesized netlist of the cruise control circuit of a vehicle. We now intend to use this netlist along with the standard cell libraries to place and route the circuit on a die. The same standard cell library that was used to generate the synthesized netlist will be used here too.

You must observe here that we do not use Custom Layout design as we practiced in the first 8 lab sessions. You will notice in this lab that the cruise control circuit will use many more standard cells than the 16 bit pipelined adder that you previously designed. It will be extremely difficult to manually draw the layout and perform routing on the standard cells. Hence we use a library with predefined standard cells (defined in terms of area, delay and power consumption) to generate the layout of the circuit by using automatic place and route procedures. The disadvantage with this automatic flow is that sometimes we do not get the desired level of optimization.

2. Starting Soc Encounter

Open the *".cshrc"* file in your home directory. Add "*source /usr/local/bin/setup.soc*" to the file and save it.

Execute "*source .cshrc*"

3. Place and Route with Soc Encounter

- Create a new folder called "*cru_con/* ". We intend to save all the required files in this folder.

- Make a copy of the synthesized verilog netlist that you generated for the Cruise Control circuit using *Design Vision* in this folder.

- Download the following files from the lab website into the same folder.
    a) *iit018_stdcells.lef*: This file provides the spacing rules for the given technology and internal cells. It also defines the corner cells (of pad frame) and I/O pads.
    b) *iit018_stdcells.tlf:* TLF (Timing Library Format) is an ASCII representation of the timing and power parameters associated with any cell for a particular technology. A TLF library is used as an input to Cadence timing tools.
    c) *iit018_stdcells.lib:* The *lib* file again stores the timing and power parameters of the library standard cells in an open source format called *liberty*.
    d) *encounter.conf*: This is the configuration file that we will use to load the library details into *encounter*.

- Modify the Configuration file *encounter.conf*

  Open the file *enocunter.conf* using any text editor. Make the following required changes.

  Replace *cru_con* on line 10 of the file (*set my_toplevel cru_con)* with the name of the top level module in your synthesized verilog netlist

  Do the same on line 21 of the file (*set rda_Input (ui_topcell) cru_con*).

  Also on line 19 replace *cru_con.vh* with the name of the verilog netlist. You should save it in the same folder.

  Note: On lines 31 and 40 you note that the timing library format and library exchange format points to a file called the *iit018_stdcells.tlf* and *iit018_stdcells.lef* respectively. Hence make sure a save a copy of these files in the very same folder.

- Make sure you are currently in folder *cru_con/* and type *encounter* at the prompt. Make sure there is no "&" after the command because running encounter in the background locks up the shell which is required by the tool.

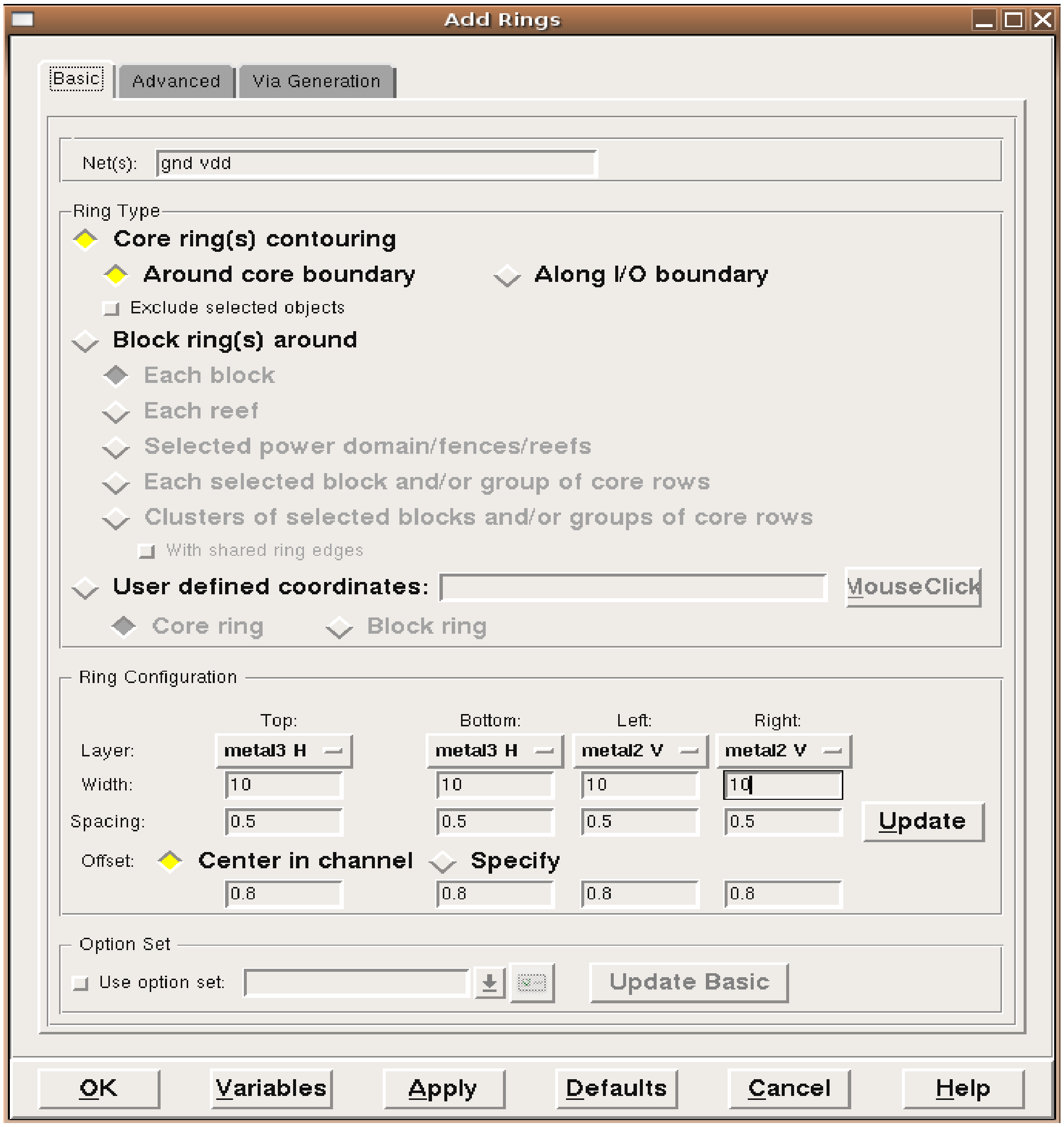
- The above command starts Soc Encounter. You will notice that the terminal where you execute this command gets a "encounter 1 > " command prompt. We can actually execute the required place and route operation commands on the terminal. But for now we shall concentrate on how to perform Place and Route using the GUI.

- First we import the design into encounter. On the menu bar select "*Design > Import Design* ". Select *Load*. From the files choose the *encounter.conf*. Observe that the timing library (tlf), lef file and the verilog netlist along with its top cell details are loaded. Now click *Ok.*

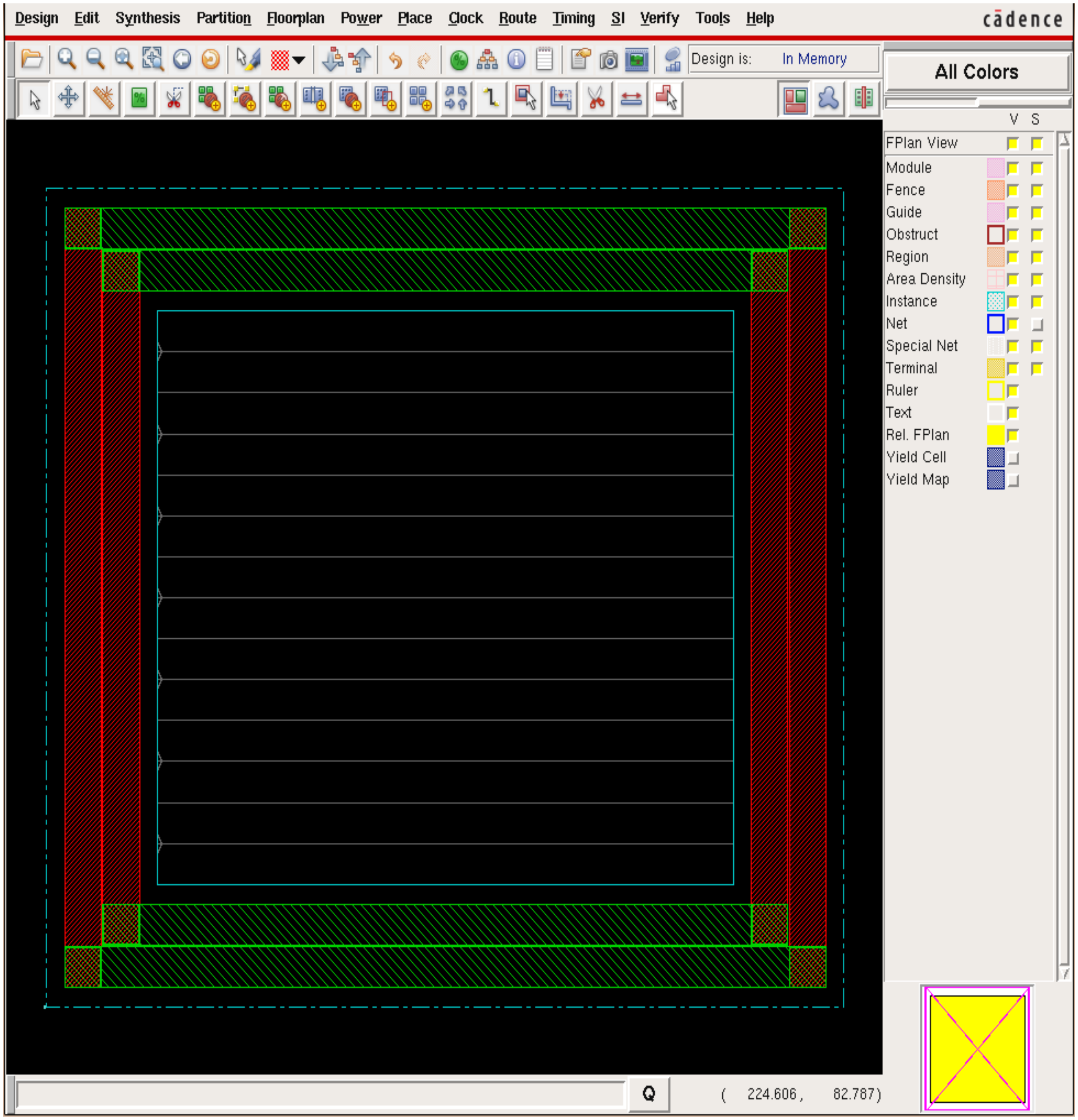
- The GUI is filled with a rectangle indicating the die area on which the circuit will be placed and routed. Next we specify the Floor plan details. Select "*Floorplan > Specify Floorplan*". We next specify the design dimensions by specifying the core size using Aspect Ratio of 0.9 and Core Utilization of 70%. (The lesser the core utilization the more sparse the design will be and the routing congestion will be lesser too but the obvious disadvantage is that it will lead to an increase in circuit area...which in the chip world means more costly).

- Also specify the Core Margins by filling in 30 for each of the four "*Core to Left*", "*Core to Top*", "*Core to Right*" and "*Core to Bottom*". This will leave sufficient space for the power railings to surround the die. Click *Ok*.

- Next we place the power rails around the die. Select "*Power > Power Planning > Add Rings*". Choose *metal3* in the Ring Configuration for the *Top* and *Bottom* Layers and let the *Left* and the *Right* remain at the metal2. Also modify all the metal *width* to 10 and change the Offset to *Center in Channel.* The details are shown below.



- Click *Ok*. After the power rails are placed the die figure in the GUI looks as follows.



- Next we need to add Power Stripes through the die to create a power grid. Select "*Power > Power Planning > Add Stripes*". Let the *Direction* in the *Set Configuration* remain *Vertical*. Change the *Width* of the Stripes in the *Set Configuration* to *5*.

- Retain the other default values, except in the sub section of *First/Last Shape* change the *X from left* to around half the size of the width of the die (around 60- 80). Click *Ok*. This creates Vertical stripes in metal2 at 80um from the left edge.

Note: All measurements are in um.

- To see the physical view with all standard cells and nets, click on the right most of the three icons along the top right corner of the floor-plan. It should read physical view.
- Next we route the power grid entirely through the die. Choose "Route > Special Route". A window pops up. Retain all the default values set in it. Click *Ok*.
- Now we next place the standard cells into the die. Select "*Place > Standard Cells and Blocks*". Retain the *Run Full Placement* and *Include Pre-Placement Optimization*. Click *Ok.*
- Observe the terminal (the same one with the encounter shell). It goes through a number of iterations before the placement is complete. Before the iterations begin the details of the design are mentioned in the terminal, one of them being the number of standard cells. **Note down the number of standard cells used in your design**.
- After the placement is complete we have to route the required wires to complete the design. Choose "*Route > Nanoroute > Route*". In the *Concurrent Routing Features* subsection choose *Timing Driven* and then click *Ok*. This procedure performs the global and detailed routing of the design.
- All the routing details are displayed on the terminal and also saved in the latest "encounter.log*" file generated. Make a note of the total wirelength and the total number of vias used in Detail Routing
- After the routing goes through completely save the design by choosing "Design > Save Design As > SoCE". The design will be stored as a "*.enc" file. Give it any appropriate name of your choice.
- Also extract the SPEF (Standard Parasitic Exchange Format) file which stores the parasitic information of the nets used in the layout. This file can later be used to generate more accurate timing analysis results. Choose *"Timing > Extract RC"* and enable the tag *Save SPEF. Just remember that SPEF contains the parasitic information which can be used to do post layout static timing analysis.*

Report Requirements:

1. Note the total wirelength that is used for the Detail Routing along with the number of vias that are used. Also note the number of standard cells your design used.
2. Print out a copy of the layout generated by performing a screen dump. (Tools > Screen Capture > Dump to GIF file).