Tutorial for Encounter Place+Route

##Encounter Setup

After each step, make sure to check the encounter.log file for errors in place and route. Keep in mind that the step may appear to complete successfully even if there were errors.

1. There are two sample files, *top.conf* and *top.tcl*, in /data/eehpc0/software/tutorials-scripts/encounterScript/layout/. Copy those two files to your layout directory, renaming them so that top is replaced with your design's top-level module name.

2. **top.conf** is the initial script that loads the necessary files to Encounter.

* Change top to your top-level design module name in *rda_Input(ui_topcell)*

* Add the .vclef files for Standard Cell Libraries and SRAMs you generated to *rda_Input(ui_leffile)**

* Add the Synopsys libraries for any SRAMS you generated to *rda_Input(ui_timelib)*

3. Open **top.tcl**. Note the *TODO* markers in the file. You will be running the script in parts so we can modify the top.tcl in parallel to running Encounter.

* Change *DesignName* to your top module name

* Observe the script structure.

* In your script, you will see many IF statement with STEP's inside. We will set the constant STEP to various strings to perform different stages of layout.

* If we do "load", it will only run what is inside of that IF statement.

* If we do "one", it will run all of the IF statements with "one" as an argument (load, floor plan, power plan, pre place, pre time, pre powerr).

* Finally, "big" will run all of the commands in the script.

* When you are more confident the design will place and route correctly, you can start using "one" instead of "load", "fp", etc.

* Additionally, instead of having to run steps over and over, the script saves your design in .enc files which can be loaded via **File->Restore Design**. To see where the design is saved, search for "saveDesign" in the top.tcl.

4. In a separate terminal, go to your layout directory and type: encounter

* Wait for the GUI to load. If it fails to load, please make sure you set your account up correctly. See (Account Setup)[Account_Setup.md].

* You can use either the GUI or the command line in the terminal to perform synthesis. You will mostly be using the command line, but the GUI is used to visually check for errors and look at your design.

* Two terminals are recommended because if you open up a file from within Encounter, the GUI will freeze until that file is closed.

Encounter Place+Route

Step one

1. load

* To load the top.conf, from the terminal where you spawned Encounter, type:

    1. set STEP load

    2. source top.tcl

* You should see multicolor three-dimensional lines if everything loaded correctly.

* TROUBLESHOOTING

    * Make sure that synthesis ran correctly without error.

    * Make sure that you modified the top.conf file as specified above.

2. fp

* In the GUI, go to **Floorplan->Specify Floorplan**

    * Adjust the core size and hit apply.

        * Generally, it is preferrable to keep the height and width equal, but in certain circumstances you may want a different aspect ratio. You can either change the *Aspect Ratio* and let the tool generate dimensions based on the *Core Utilization* or use the *Dimension* option to manually specify. If you use the latter, make sure the resulting *Core Utilization* is desired.

    * After hitting apply, make sure the resulting size, dimensions, and core utilization are what you want. You may need to iterate.

    * On your first pass, you may need to lower the utilization to 80-90% depending on how much area is lost due to the createObstruct required when placing SRAMs; more on that later. Ideally, the final layout would have a value close to 100%.

* In top.tcl around line 42, set **corewidth** and **coreheight** to the values you generated in the GUI.

* If you have any standard cells to place, you need to create **placeInstance** lines in the rtl.tcl.

    * placeInstance relative/path/to/module lower_left_x lower_left_y

* In the terminal, type

1. set STEP fp

2. source top.tcl

* You should now see a square or rectangle within the GUI. Additionally, if you placed any SRAM, they should show up as teal or grey boxes.

* TROUBLESHOOTING

* If you preplaced standard cells (e.g. SRAM), make sure that they appear as teal/grey boxes of the expected size and that they fit inside the core. If not, look at the troubleshooting steps of the [Advantage Tutorial](Advantage_Tutorial).

* Also double-check that any blocks you placed are not overlapped.

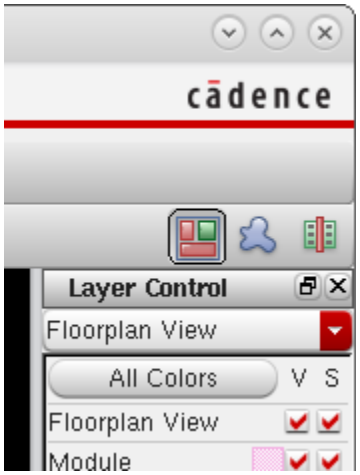
* TIPS/TRICKS

* Use the ruler tool or place the cursor and note coordinates in the lower right of the GUI for determining positionings.

* To clear the ruler, press Shift+K

* At this and later stages, it is useful to look at the design under various views: Layout, Amoeba, Physical.

* ![Encounter Views](images/encounter_views.png)



> **Figure 1 - Various Views of Encounter**

* To avoid violations later and minimize the area lost to *createObstruct* rectangles, placing SRAM side-by-side is a useful technique. The distance between them should be 1 or fewer units.

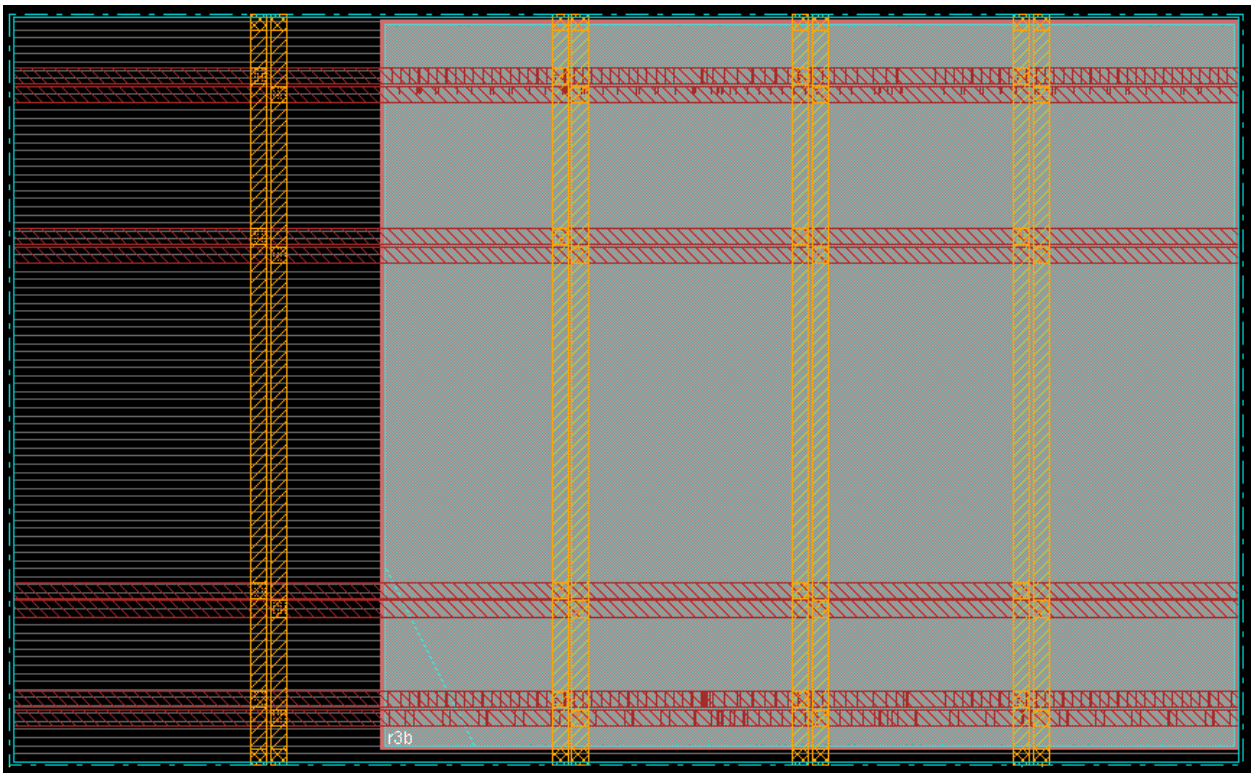
3. powerplan

* Looking at your design, determine where you want your vertical/horizontal VDD/GND rails.

* You want vertical rails at regular intervals. How many and how far apart depends on the design; you may need to experiment.

* The only time you need horizontal rails will likely be if you have SRAM memories. For these, it is recommended to place two sets of VDD/GND pairs near each other near the edges of the SRAM. See Figure 2.

* ![Horizontal VDD/GND Rails for SRAM]



> **Figure 2 - Horizontal VDD/GND Rails for SRAM**

* In top.tcl (line 176), go to vertical stripes and horizontal stripes

1. For vertical stripes (metal 6):

* **start_x** [where you want first vertical stripe relative to x axis] - 0 on the x-axis is the left edge

* **set_to_set_distance** [distance between the next vertical rail]

* **number_of_sets** [number of vertical rails]

* You can copy/paste this code again if you want to more precisely place many vertical stripes. Note that the command is one line of code, extended using the '\' character.

2. For horizontal stripes (metal 5):

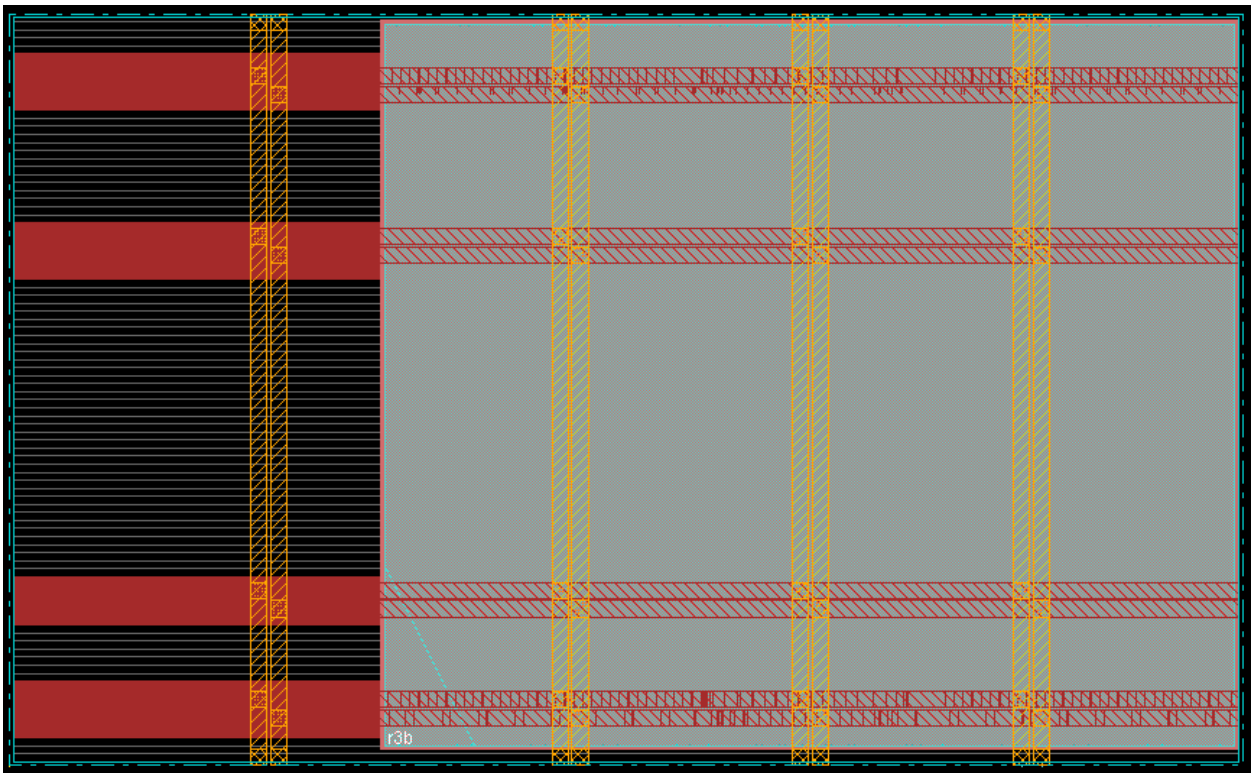
* Same thing, adjust *start_y*, *set_to_set_distance*, and *number_of_sets*. Similarly, 0 on the y-axis is the bottom edge.

* Unless necessary, as in when you have SRAMs, it is best to avoid horizontal stripes because they require additional work to keep place and route from causing design errors when you have both horizontal and vertical stripes

3. If you have SRAMs, you need to write one or more *createObstruct* commands. By creating obstructions over the horizontal rails that are not on top of the SRAM, you will eliminate all kinds of violations, which show up as white Xs. These rectangles are areas where Encounter is not allowed to place standard cells, which causes the violations arising from having both horizontal and vertical power and ground rails.

* createObstruct lower_left_x lower_left_y upper_right_x upper_right_y

* ![Horizontal VDD/GND Rails for SRAM]



> **Figure 3 - createObstruct over Horizontal VDD/GND Rails for SRAM**

4. You may also need to either use *addHaloToBlock* to create a zone around the cell where Encounter cannot place standard cells or use *createObstruct* on only the side of the SRAM that has violations.

* See the rtl.tcl for an example of *addHaloToBlock*.

* Once you have all the settings saved, type:

1. set STEP powerplan

2. source top.tcl

* TROUBLESHOOTING

* Check that the VDD/GND rails

* fit within the dimensions of the design,

* do not overlap each other, and

* are the right number and locations you expected. In some cases, the *number_of_sets* does not seem to really indicate the number of sets. Even if you take out the *extend_to design_boundary*, you may still get more sets than you want. For safety, set the *stop_x* or *stop_y* to the correct index.

* If you are trying to add a halo to a block, ensure that the addition of the halo does not place the cell outside the dimensions of the core. If you have other issues, *createObstruct* could be used to a similar effect.

4. preplace

* TROUBLESHOOTING

* Looking at the GUI is helpful to visually locate violations as they are marked with a white X. To get a list of them, go to **Tools->Violation Browser**. This is most useful if only a few errors are present, so eliminate the major issues first.

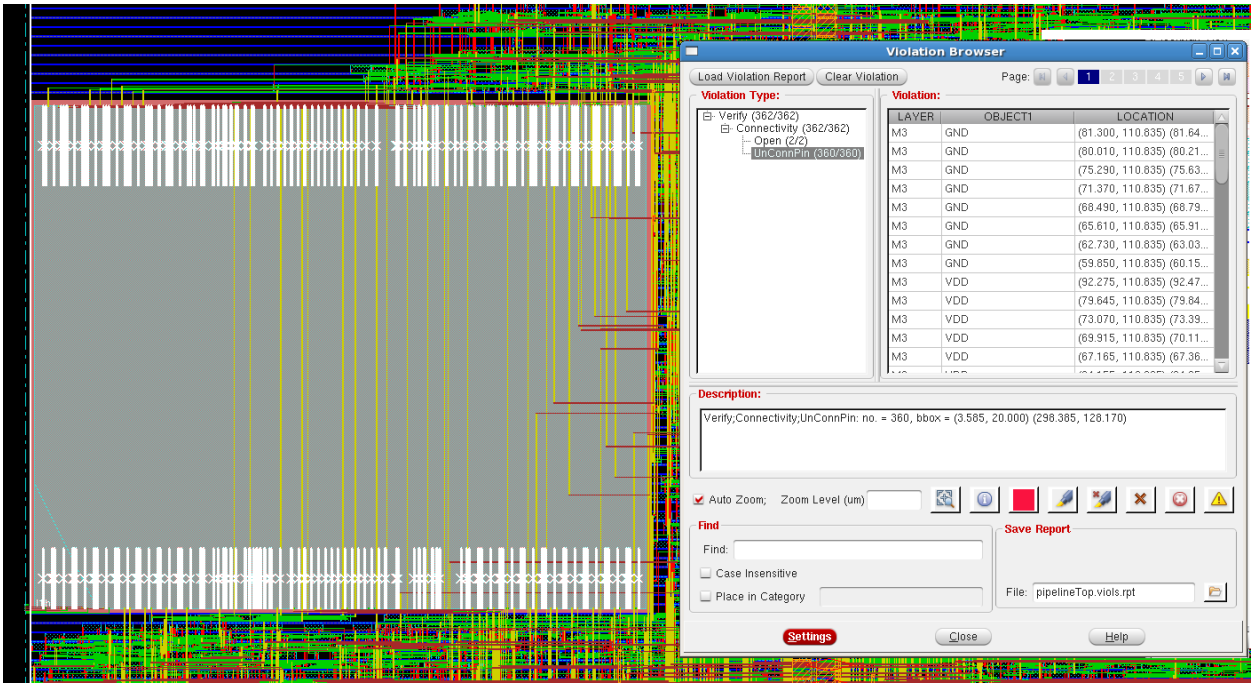
* If you have *lots of violations in the design*, the core width/height may be too small. Usually the floor plan utilization underestimates the percentage used, so that's why I usually start at 90% and gradually decrease the height/width. Keep in mind that if you use other libraries (e.g. SRAM), the floor plan utilization will not be useful since it will **not** estimate the area occupied by other cells correctly.

* If you have *a few violations in the design, there are probably issues with the VDD/GND rails or placement of blocks from other libraries. Encounter may try to place a standard cell where you do not want it to be. You may need to adjust the placement of the rails/library blocks and/or use the *createObstruct* command.

* If this is not the reason, the Violation Browser will help provide more information for personalized troubleshooting.

* If you have *a lot of violations on the top/bottom of an SRAM* similar to Figure 4 below, this means that you have unconnected VDD/GND pins. Adding horizontal rails as described above usually fixes this, but the steps below can be used to verify and fix in a more fine-tuned manner.

* ![Violations on top/bottom of SRAMs]



> **Figure 4 - SRAM violations due to unconnected VDD/GND pins**

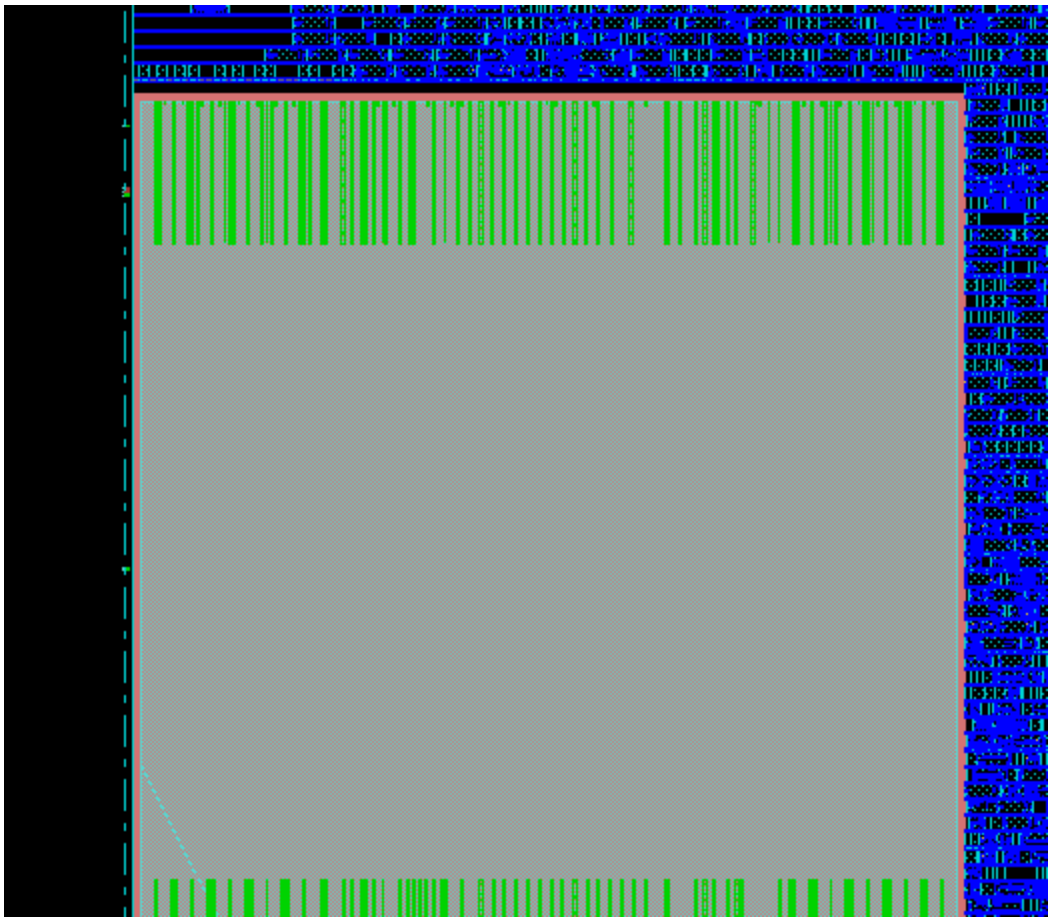
1. Change the view to "Physical View" on the right side of the GUI if not already.

2. Click on the **All Colors** button to bring up a window

3. On the *Objects* tab, turn off *Violations*. On the *View Only* tab, turn on *Instance Pin* and turn off *Net*

4. You should then see something like below. The long green bars are VDD/GND pins of the SRAM

* ![SRAM VDD/GND pins]

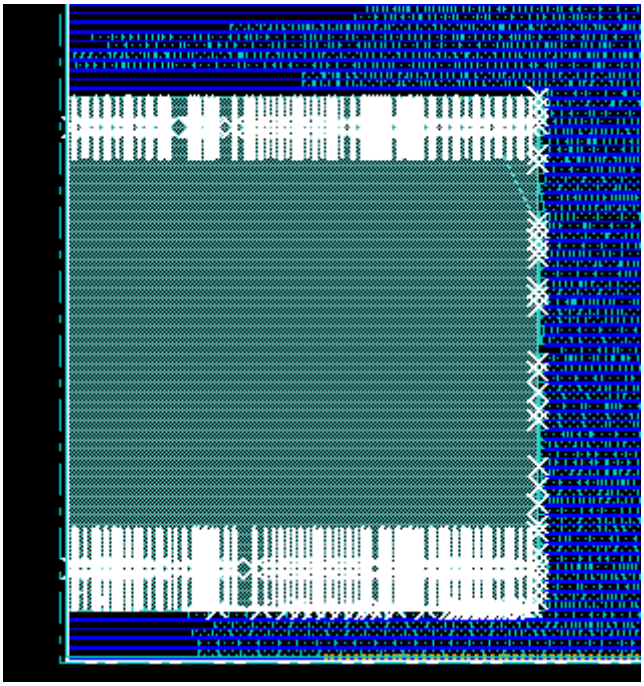


> **Figure 5 - SRAM VDD/GND pins**

5. To get the pins connected, you need to go back to the *powerplan* step and pass a horizontal stripe on M5 to connect these bars to the VDD/GND rails.

* If you have *a lot of violations on the sides of an SRAM* similar to Figure 6 below, this is likely because Encounter placed other standard cells too close. If you zoom in, they will probably say "Spacing violation, ...". The image below also has the issue described above due to the unconnected GND/VDD pins.

* ![Violations on all sides of SRAMs]



> **Figure 6 - SRAM violations due to minimum spacing requirements**

* This requires you to either create a halo around the SRAM or write a createObstruct for specifically the area causing issues.

* If you have *sporatic short violations*, check to see if they are shorts of the same signal to itself. These can be ignored as they can be fixed by hand at a later date before fabrication.

5. pretime

6. prepower

* In top.tcl (line 407) under prepower, set the speed and gate switching

* {clk 1170 0.2} refers to a clock frequency of 1170 MHz with 20% of gates switching every cycle

* This affects the power consumption calculated, the speed and activity factor

Step two

1. place

2. prectsopt

Step three

1. cts via GUI

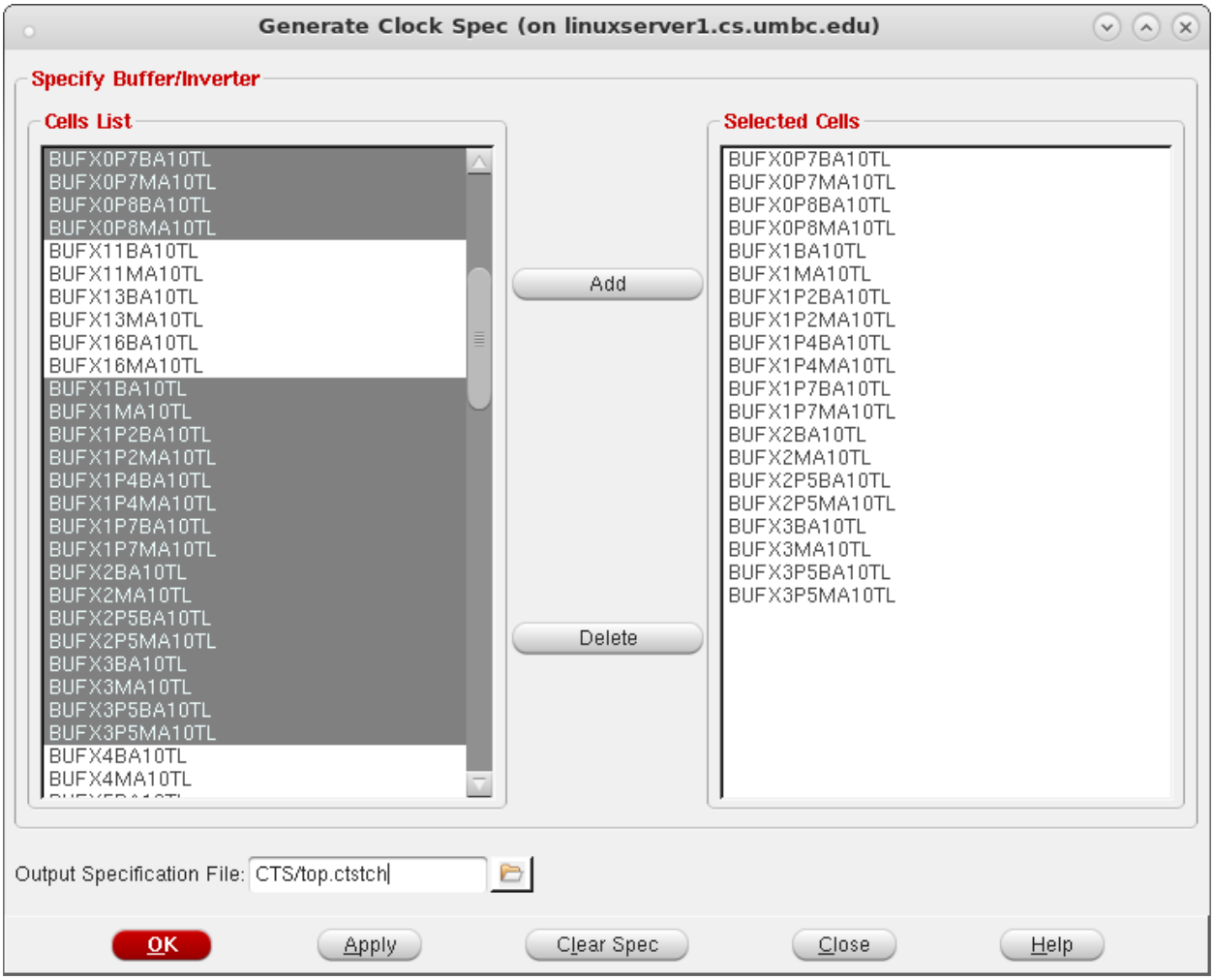
* If you look at the top.tcl, you'll see that step *three_bad* is cts, or clock tree synthesis. The script does not perform the behavior correctly, so you will need to perform this step through the GUI as described below.

* Go to **Clock->Synthesize Clock Tree**

* Press **Gen Spec...**

* Add the twenty cells that fit this regular expression: BUFX\[0123\]\[PBM\]. In otherwords, only those that start with BUFX and a number from 0-3. See screenshot below for the list, or refer to the list in the template top.tcl. Set the **Output Specification File** to CTS/top.ctstch, where top is the name of your design's top module. This is the required location for subsequent steps in the TCL script.

* ![Clock Tree Synthesis Screenshot]



> **Figure 7 - Clock Tree Synthesis Screenshot**

* Click *OK*

* Click *Apply* and wait up to several minutes for it to complete. In the meantime, you can watch the status in the terminal.

* When done, click *OK*.

2. postctsopt

* Type:

    * set STEP postctsopt (set STEP three)

    * source top.tcl

Step four

1. route

* Type:

    * set STEP route

    * source top.tcl

* For basic designs, you should not need to modify anything in this script.

* Clock tree synthesis is performed in step three, which adds additional routing. This step could result in errors from the width/height being too small.

2. postroute

* Type:

    * set STEP postroute

    * source top.tcl

3. postpower

* Type:

    * set STEP postpower

    * source top.tcl

Step export

* Type:

    * set STEP export

    * source top.tcl

* TROUBLESHOOT

    * If you did not name your .ctsch appropriately, this step will fail.

Step wirereport

* Type:

    * set STEP wirereport

    * source top.tcl

##Post Power Analysis

The area is specified by the floorplan width/height provided and the speed was set in the rtl.tcl script in the synth folder. The power numbers are calculated based on speed and area specificiations.

In the layout/postpower folder that is created, the resultant power and area numbers are given. Skimming through the files provides the power numbers.