LUNDS UNIVERSITET
Lunds Tekniska Högskola

Physical Placement with Cadence SoCEncounter 7.1

Joachim Rodrigues

Department of Electrical and Information Technology
Lund University
Lund, Sweden

November 2008

Address for correspondence:
Joachim Rodrigues
Digital ASIC Group
Dept. of Electrical and Information Technology
Lund University
Box 118,
S-221 00 Lund, Sweden

Telephone:	+46 46 22 24868
Fax:	+46 46 129948
e-mail:	joachim.rodrigues@eit.lth.se

Contents

1 Introduction and Disclaimer

This manual is intended to guide an ASIC designer through the basic designs steps from netlist to tapeout. The reader is guided by using the SoCEncounter GUI. However, this tutorial is not complete and was never intended to be so. The design steps considered in this manuscript are presented in Figure 1. Several digital ASIC's have been successfully produced (in our digital ASIC group) according to the presented design flow. For a more detailed description the reader is referred to the SoC Encounter manual.

Although every precaution has been taken in the preparation of this manual, the publisher and the authors assume no responsibility for errors or omissions. Neither is there any liability assumed for damages or financial loss resulting from the use of the information contained herein.

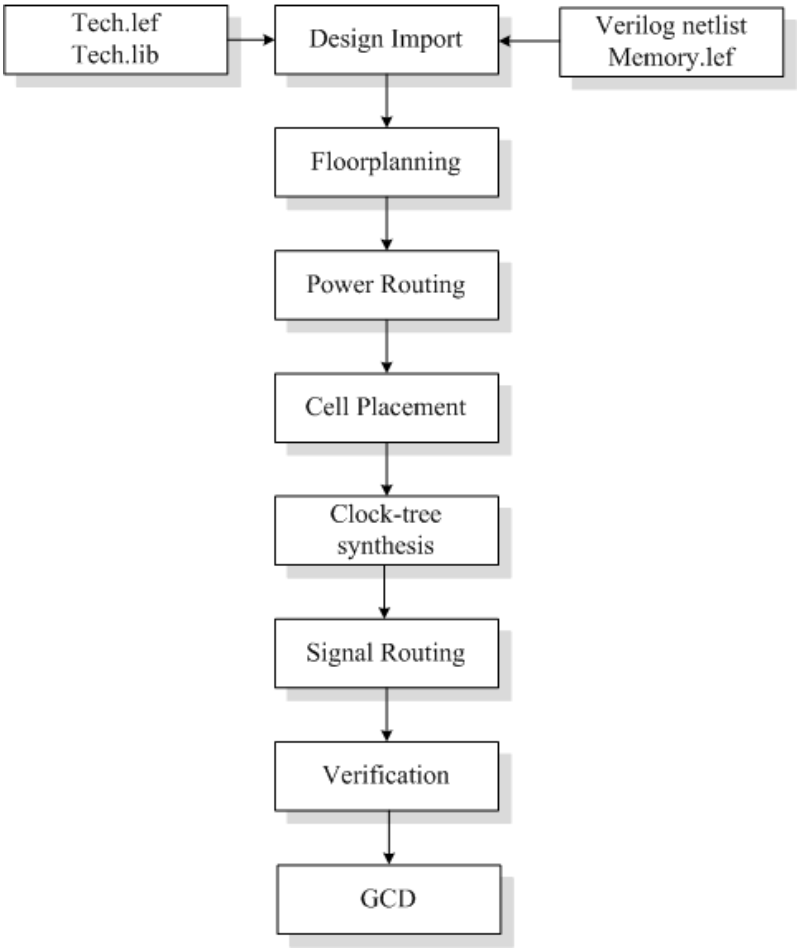

Figure 1: Basic Encounter design flow.

2 Physical Placement Basics

This section introduces the terminology used in an ASIC back-end flow, as depicted in Figure 1. For a fully developed flow it is recommended to include, RC extraction, signal integrity check, cross-talk and IR-drop analysis. However, this is out of the scope of this tutorial. Each task in Figure 1 will be described in more detail in the sections that address a specific task in the design flow.

2.1 Floorplanning

Floorplanning of a design is the first step after import of the technology files and gate-level verilog netlist. In this step the *floorplan* needs to be prepared for power supplies, placement of IO-pads, hard macros (RAM), and standard cells. The designer is required to determine the die size by arrangement of the IO's on the pad frame and the *pad-frame to core distance*. Moreover, the core rows need to be interrupted where the hard macros should be accommodated.

2.2 Power Planning

The task where the power supplies for the core rows and macros are specified is referred to as *power planning*. The designer needs to design the power supply rails around the core and blocks by choosing metal layers as well as physical location and width of the supply lines.

2.3 Cell Placement

In this task the information in the gate-level netlist is used to place the standard cells on the floorplan. The standard cells may be arranged by providing timing information generated during RTL synthesis.

2.4 Clock Tree Synthesis

After *cell placement* the location of registers is known and the clock tree(s) can be synthesized. The constraints for clock tree synthesis are obtained from the clock tree specification file (.cts). The clock buffers will be placed in the gaps of the floorplan. Furthermore, due to the nature of CTS the clock tree will be routed in this step ahead of any other signal. This guarantees that the clock tree will be routed in the most efficient way, and thereby clock skew will be reduced.

2.5 Power Routing

The power supplies specified during power planning need to be connected in this task. SocEncounter will create connections between the core rows and core rings, and from the core rows to the core supply pads.

2.6 Signal Routing

In this task the signals as specified in the netlist need to be routed to complete physical placement of the design. The tool will try to optimize timing by optimal routing, i.e., the signals will be assembled by wires that span over several layers.

2.7 Verification

This is the final step where the designer checks if everything is properly connected and if *antenna problems* need to be fixed.

2.8 GCD Export

In this step the layout needs to be exported in GCD format to be further analyzed with Cadence Virtuoso.

3 Technology and Design Files

Physical placement of a design requires the layout and timing information of the standard cells. Constraint driven placement may be utilized with constraints generated during RTL synthesis. Layout and timing information is provided from the technology vendors. The former is provided by the library exchange format (LEF) file. The information in the LEF file is the text version of the Virtuoso cellView abstract and includes layer names, layer widths, layer usage, external dimensions, cell pin and port as well as blockage description. A typical abstract of a 2-input 1 output standard cell is presented in Figure 3. The latter, timing information, is provided by the technology libraries (*.lib*), which are available as worst- and best-corner cases to model process variations.

The constraints set during synthesis may be transferred to Encounter by a Synopsys Design Constraints (SDC) file. The file is tool command language (Tcl) based and specifies the design intent, including the timing, power, and area constraints.

The gate-level netlist of the design which was generated during RTL synthesis needs to be provided in verilog format.

3.1 IO Assignment File

The physical location as well as the orientation of the IO's need to be specified in a IO assignement file. Beside the IO's which are already inferred with the RTL model, IO's for core and pad power as well as for ESD protection need to be specified.

A template file *ioplace.io* is available that specifies the pads for power supply, etc. location/orientation of the pads. Dependent on the location, the pads need to be oriented by specifying the direction of the pads. The names of the IO's as specified in the vhdl code are listed with their respective net name in the verilog netlist.

Open the *ioplace.io* file with an editor and complete the assignment with the pad names specified in the verilog netlist. The orientation of the pads, as presented in Figure 2 for *top, left, bottom, right* is *N, W, S, E*, respectively. The core power supplies should remain as specified in the template, i.e., in the center of *top* and *bottom*.

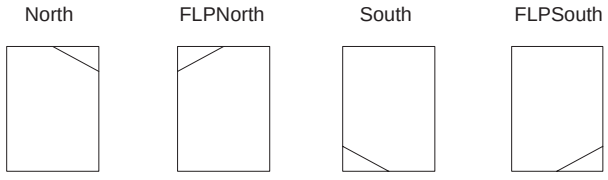

Figure 2: Orientation of cells.

4 SoCEncounter

This section explains how to start SoCEncounter and introduces a graphical interface to monitor the content of the SoCEncounter log file.

4.1 Starting Encounter

It is recommended to create a new directory before initializing the environment. The environment is set up by typing **inittde farall** in the shell and will generate a directory structure.
To start SoCEncounter change directory to *soc*. SoCEncounter is started with
encounter

The GUI, presented in Figure 4, pops up and the tool is ready to use. Functionality of the tool can be accessed through the pull-down menus, which are located on top of the encounter window. Below are some toolbar and floorplanning icons which are frequently required during placement and routing. On the right hand side are to switches to enable/disable visibility (V) and selectability (S) of floorplan components, e.g., signals or specific metal layers.

4.2 Log Viewer

The encounter log file is updated in real time and contains all executed commands as well as any warning or error messages generated by the tool. The content of the log file can be scrutinized with Log Viewer (LV) which is accessible through a pull-down menu in the GUI. With the LV it is possible to expand/collapse information of the log file, view color highlighted warning and error messages, execute string matching, etc. The LV window is activated by choosing

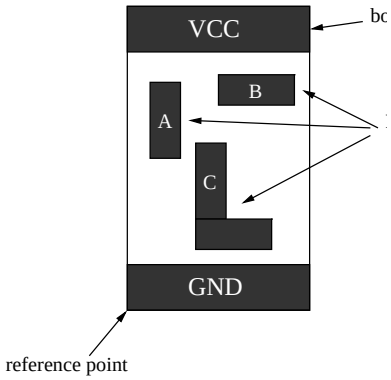

Figure 3: A typical abstract of a standard cell.

Tools → Log Viewer

Open the log viewer and confirm that SoCEncounter did start properly.

4.3 Technology and Design Import

The technology and design files need to be read by Encounter to generate a database. The technology files that accommodate physical and timing information are accessed by selecting files with the endings

- .lef
- .lib

The required *.lib* files model worst- (wc) and best (bc) process corners by max- and min-timing libraries, respectively. This is valid for the standard cells and IO's, as well as for any hard macro model.
The design files consist of the gate-level netlist, timing constraint file, and pad arrangement, having the endings

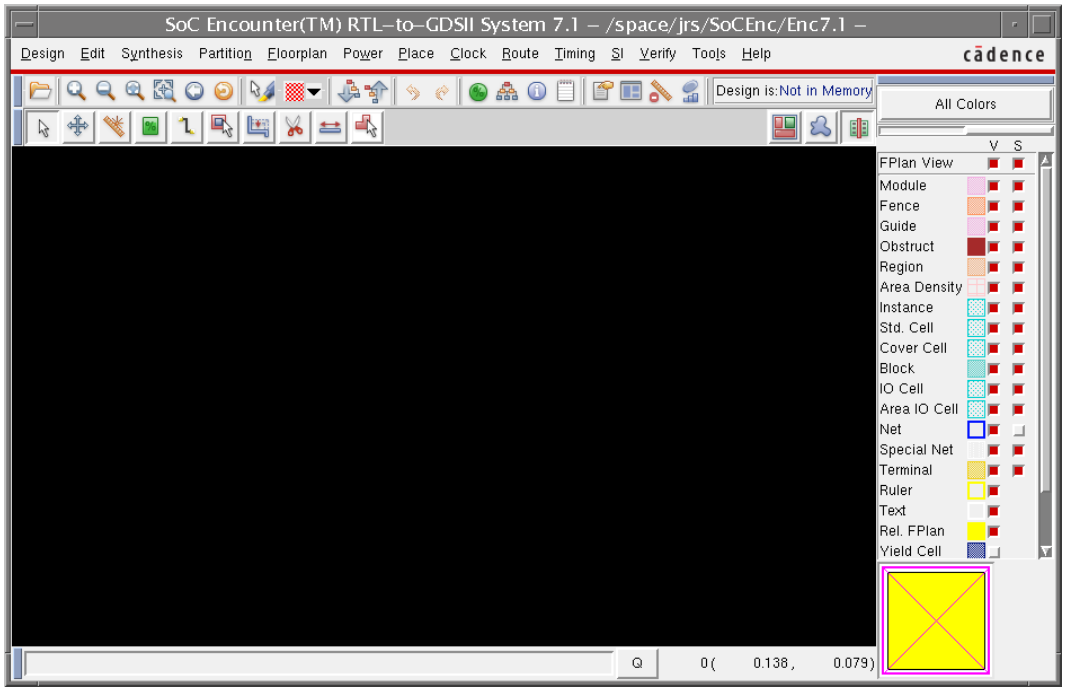

Figure 4: SocEncounter's GUI.

- .v
- .sdc
- .io

To import the required libraries choose

Design ⇒ *Import Design*

on the pull-down menus. The interface as shown in Figure 5 pops up and the files that contain the verilog netlist, maxmin timing libraries, lef information, constraints, and IO assignments need to specified. However, if a configuration file is available it can be *loaded* and completed with the specification of the io assignment file, netlist, and *memory.lef*. It is recommended to save these specifications in a config file, i.e., select *Save* and specify the name of the configuration file. Select the *Advanced* Tab, Select *Power*, and specify VCC and GND for Power Nets and Ground Nets, respectively.
Finally, select *OK* to complete this task. If everything was properly declared the initial floorplan and memories become visible. The design hierarchy can be viewed by choosing

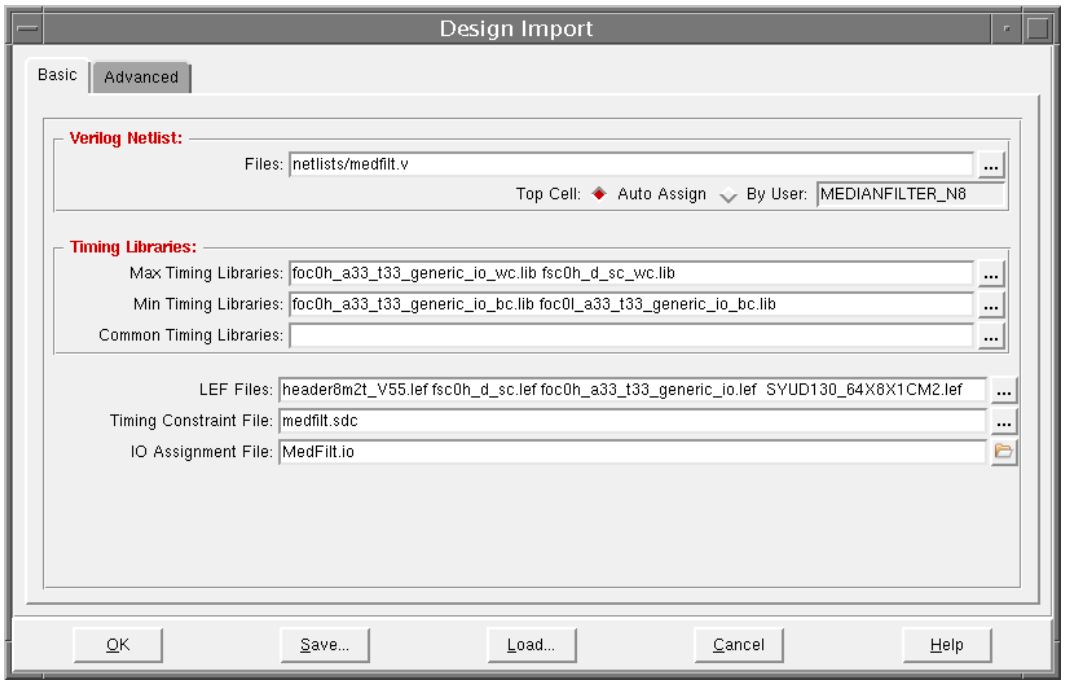

Figure 5: Design Import window.

Tools ⇒ *Design Browser*

The schematic becomes visible by selecting the *schematic* icon.

4.4 Floorplanning

In this design step you need to specify the dimensions of the ASIC core, arrangement of the core rows, distance between ASIC core and IO's, physical location of any hard macros and distance between these blocks and the core rows. On the pull-down menu select

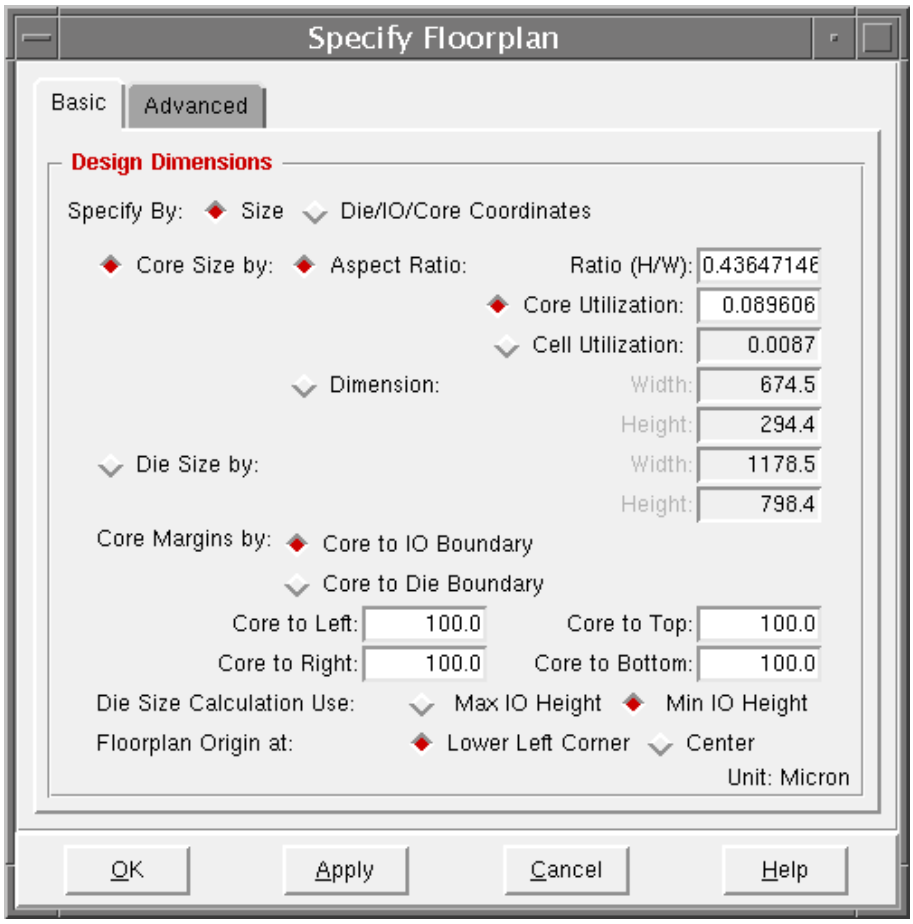

Figure 6: Interface that specifies the floorplan.

Floorplan ⇒ *Specify Floorplan*

which will open the Floorplan interface, presented in Figure 6. The floorplan can be either core- or pad limited dependent on the size of the design and the number of pads. The space between the core and pad frame need to be specified to make space for the power supplies. In order to improve timing the gates should be placed as dense as possible on the core rows. This density is indicated by the *core utilization*, located on the right hand side of the panel in Figure 6. By specifying the *Core to IO Boundary* on each side, i.e., left, top, right, bottom, a gap between the core and IOs will be introduced, and the *Core Utilization* is updated. It is recommended to increase both *Core to IO Boundary* and the value for the desired core utilization until *Core Utilization* reaches a value around 0.8. Thereby, 20 % of the core row space will be reserved for buffers in the clock tree (to be created later) and signal routing. The settings under the *Advanced* tab specify arrangement of the core rows and should remain unchanged. Signal congestion around hard macros can be reduced by specifying *block halos*. Choose

Floorplan ⇒ *Edit Floorplan*⇒ *Edit Halo*

and specify the halo as presented in Figure 8, e.g. 20 μm, for each side, and the halo be-

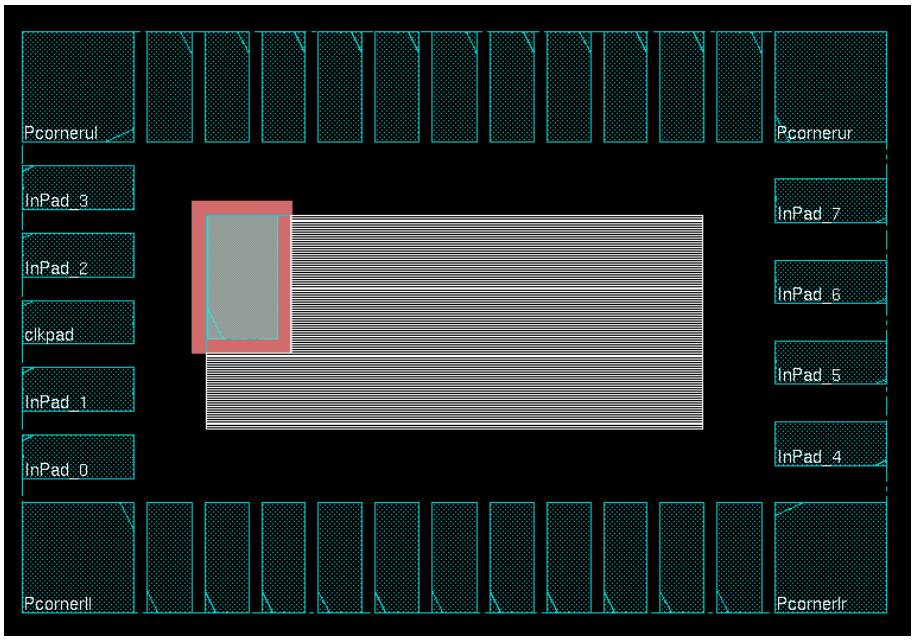

Figure 7: A floorplan with a RAM block placed on the left hand side.

comes visible as a reddish shadow. The hard macro, also referred to as block, needs to be placed on the core. Select the 2^{nd} floorplanning icon and move the block to the desired location on the core rows. The blue *flight lines* show the connections between the block and the logic. To change orientation of the block press the **r** key top open the *Flip/Rotate Selected Instances* interface and choose the desired orientation, e.g., R90. If the block is placed on its final destination the rows at the placement blockage needs to be cut.
The core rows underneath the block and covered by the block halo need to be deleted, type

cutRow

in the command shell, and zoom in to check if the core rows have disappeared. A floorplan where a block is placed on the left hand side is shown in Figure 7.
The IO's need to be placed in a certain distance to avoid a *design rule* violation. Measure the distance by activation the ruler tool and adjust the the IO-to-Core distance until the distnace betwenn the IOs is 10μm. Furthermore, it is necessary tp specify the global net connections, e.g., VCC and GND, need to be specified. On the drop-down menu select

Floorplan $\Rightarrow$ *Connect Global Nets*

Select *Pin* and specify *VCC* under *Pin name*, and select *Apply all*. Specify *VCC* as Global net and click on *Add to list*, and the connection will appear on the left hand side of the pane.

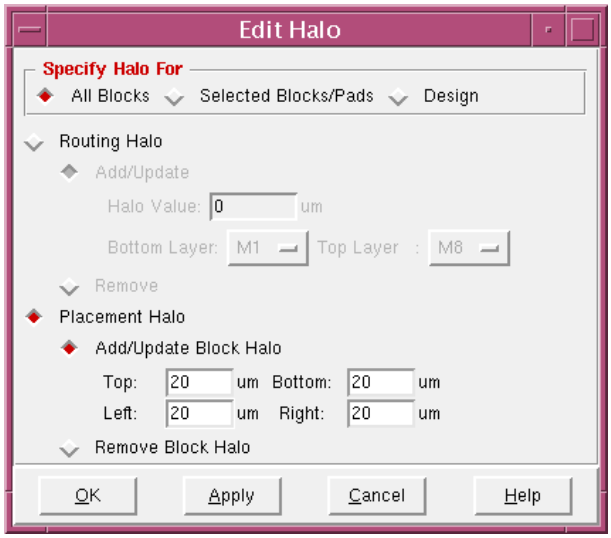

Figure 8: Block halo pane.

Select *Tie High* without changing the other settings and add it to the list. Iterate this two step for GND (Tie Low) and confirm that the global net connections match the list presented in Figure 9. Click on *Apply* and *check* and if no warnings appear close the window. Before you continue you should save the floorplan by choosing

Design ⇒ *Save*⇒ *Floorplan*

Thereby, it is possible to retrieve the floorplan settings for an eventual redesign.

4.5 Power Planning

The gates on the core rows and the blocks need to be connected to the IO supply pads. This connection is done by core and block rings that need to be setup during power planning. In this task the physical location and size of the power rings need to be specified. Furthermore, it is recommended to route power stripes above the core rows to assure a sufficient current propagation. To open the interface for power planning choose, presented in Figure 10, select

Power⇒*Power Planning* ⇒ *Edit Power Planning Option*

To start with the core ring you select *RING* as Object, and for the horizontal (H) and vertical (V)

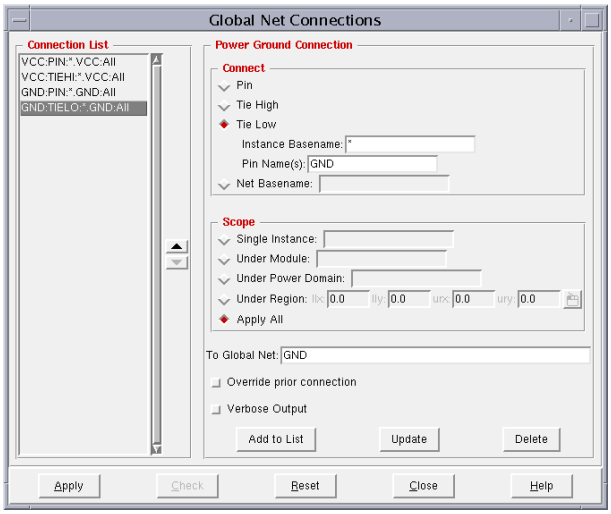

Figure 9: Global net specification.

layers select Metal5 and Metal6, respectively. Thereafter, select *Ring Style* under *Options* and assure that *Around Core Boundary* is selected. Under *Ring offset* specify 10μ, for all directions, and click on *Add/Modify*

To set the parameters for the *block ring* change to *block_rings* in the *Type field.*

A power mesh can be created by selecting *STRIPE* on the power interface. Choose *metal6* as HV-Layer and assure that *Core Ring* is selected under *Stripe Boundary.* To check your settings click on *Add/Modify* and the selected settings are displayed. Save theses settings by specifying a name in the *Power Planning Option Set* field, and click on save. All settings and commands are secured in a *.ppo* file, required to create the power lines.

In the next step the power rings will be created by choosing

Power $\Rightarrow$ *Power Planning*$\Rightarrow$ *Add Rings*

and *Add Rings* interface appears. Assure that the nets GND and VCC are prompted. The width

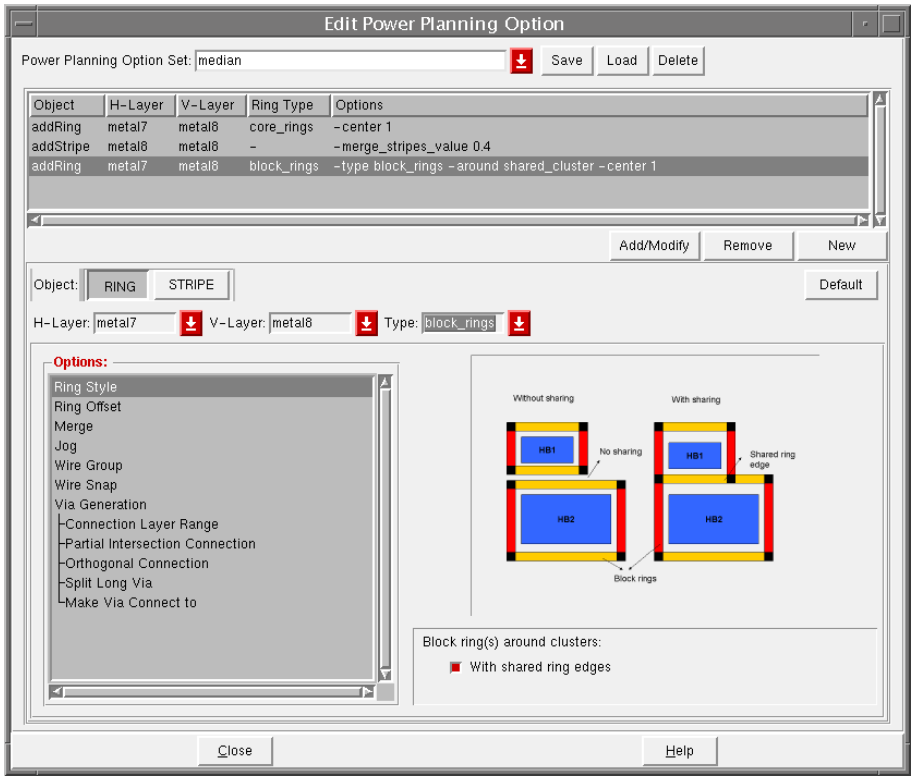

Figure 10: Interface for Power Planning.

of the rings need to be specified in the *Ring Configuration* field, presented in Figure 11. Select for the Top/Bottom and Left/Right layer **metal5 H** and **metal6 V**, respectively. Configure the width as 4.9 μm and spacing as 1 μm. In the *Option Set* field choose the previously created *.ppo* file, and by clicking *Update Basic* the settings will change accordingly. Change the selection in the *Ring Type* field to *Block rings(s) around each block*, and update the settings for *ring configuration*, e.g. 4.9 μm and spacing as 1 μm in all fields. The required power line width for a block can be retrieved from the block lef file.
The stripes are created by choosing

Power $\Rightarrow$ *Power Planning*$\Rightarrow$ *Add Stripes*

Specify the width and spacing of the stripes, e.g., 4.9 μm and 1 μm, and select the *.ppo* file. A stripe over a block is avoided by specifying a distance between core and *First/Last Stripe* in the *Add Stripes* pane. A layout with routed power rings and stripes is presented in Figure 12.

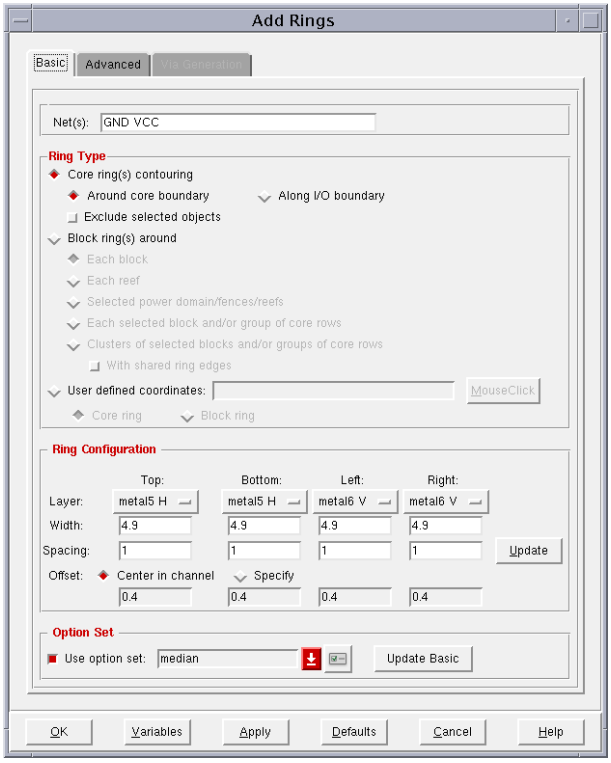

Figure 11: Add core ring pane.

Save the design as ∗PP.enc.

4.6 Cell Placement

The layout is now ready for cell placement and this can be influenced by information in the constraint file. Further parameters for cell placement may be set by selecting

Design ⇒ *Mode Setup*⇒ *Placement*

The standard settings in the *Placement Mode panel*, as selected in Figure 13, should be sufficient to achieve a satisfying placement result. To start placement with the provided settings choose

Place ⇒ *Standard Cells*

and select *Full Placement*, if not already selected. Depending on the size of your design this task requires some minutes. The standard cells become visible by changing from floorplan to layout view, using the switches presented in Figure 14.

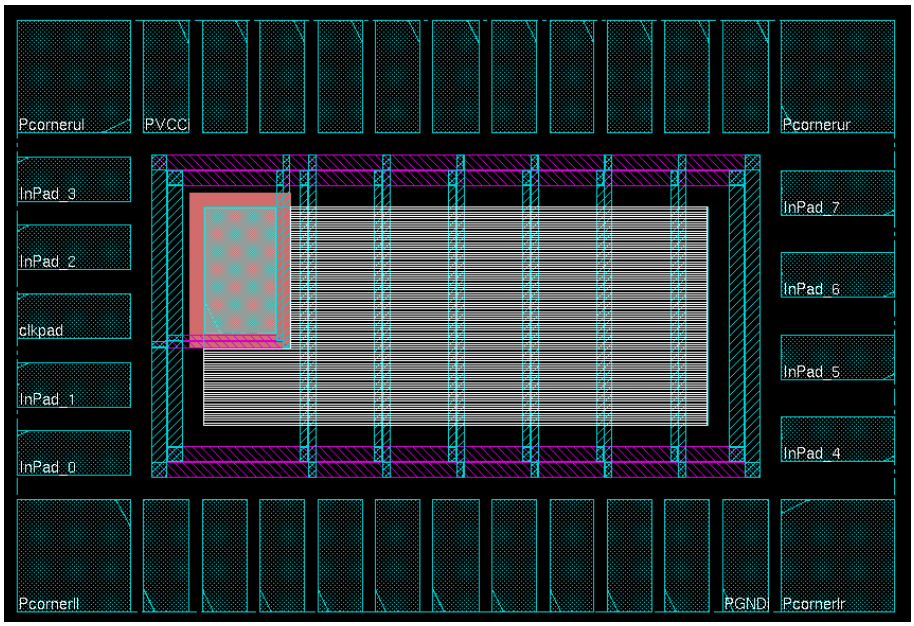

Figure 12: Layout with routed power lines.

Save the design as ∗Placed.enc.

The physical location of the registers is known by the tool after cell placement and the clock tree can be synthesized.

4.7 Clock Tree Synthesis

The physical location of the registers is known after cell placement and thus the clock tree can be synthesized. SocEncounter generates a clock tree by mapping the requirements in the clock specification file (.cts) and constraint file (.scf) to the physical facts. The clock tree is assembled by appropriate sized clock buffers that will be accommodated in the core row gaps.
Open the .cts file and specify the name for the *AutoCTSRootPin*, e.g., *clkpad/O*. The name can be retrieved by scrutinizing the verilog netlist.
Uncomment *routeCLKNet* which automatically routes the clock tree after clock tree synthesis. The parameters for clock skew and definition of clock buffer names remain unchanged. Save the

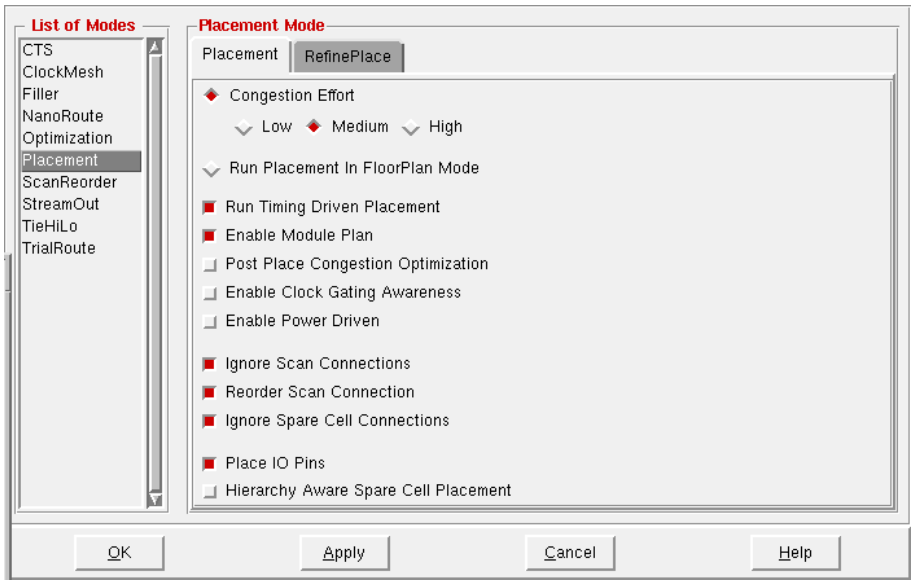

Figure 13: Setup for cell placement.

Figure 14: Switches for floorplan, amoeba, and layout view.

file and on the drop-down menu select

Clock ⇒ *Design Clock*

Select the .cts file that was previously updated and start clock tree synthesis (CTS) with **ok**. Dependent on the design size CTS requires some time to complete, and the clock tree as well as a trial route for signals becomes visible. The trial route information can be deleted with *delete-TrialRoute*, and the clock tree remains highlighted. Browse through the clock tree after selecting

Clock ⇒ *Clock Tree Browser*

Save the design as ∗CTS.enc.

4.8 Filler Cells

The gaps on the core and IO rows need to be filled with dummy cells referred to as core and IO filler, respectively. Core filler cells ensure the continuity of power/ground rails and N+/P+ wells in the row. Since filler cells will close any gap it is important to perform CTS before filler cell placement. Select

Place ⇒ *Filler*⇒ *Add Filler*⇒ *Select*

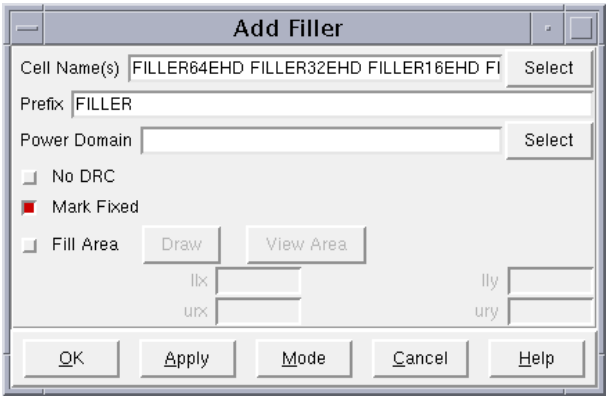

Figure 15: Specification of the core filler cells.

and add all available filler cells. The number in the name specifies the width of the filler cell. The left and right pane of Figure 16 shows some core rows before and after filler cell insertion. The gaps on the pad frame need to be filled with IO fillers which connect the pad-row power supply. The IO frame is filled by execution of the commands in the list below:

- addIoFiller -cell EMPTY16LB -prefix if16
- addIoFiller -cell EMPTY8LB -prefix if8
- addIoFiller -cell EMPTY4LB -prefix if4
- addIoFiller -cell EMPTY2LB -prefix if2
- addIoFiller -cell EMPTY1LB -prefix if1 -fillAnyGap

Click on redraw if the filler cells are not automatically visible. An example of an layout before and after core filler placement is presented in Figure 16.

4.9 Power Routing

In this task all connections to VCC and GND will be routed. In particular all pins (block, pad, standard cell) will be connected through the stripes and core/block rings to the power pads. Select

Route ⇒ *Special Route*

to open the *SRoute* pane presented in Figure 17. Deselect *Pad rings* since the used technology has already power rails on the IO cells, and start routing be clicking on *OK*. A layout that

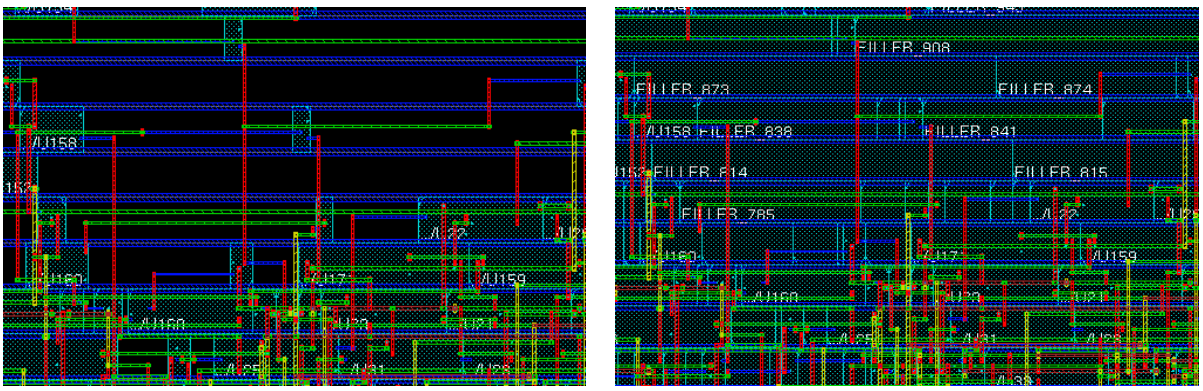

Figure 16: The gaps on the core rows to the left are closed with filler cells on the right.

has completed SRoute is presented in Figure 18. It can be seen that the *core rows* are connected to the *core ring*, which in turn is connected to the core supply pads.

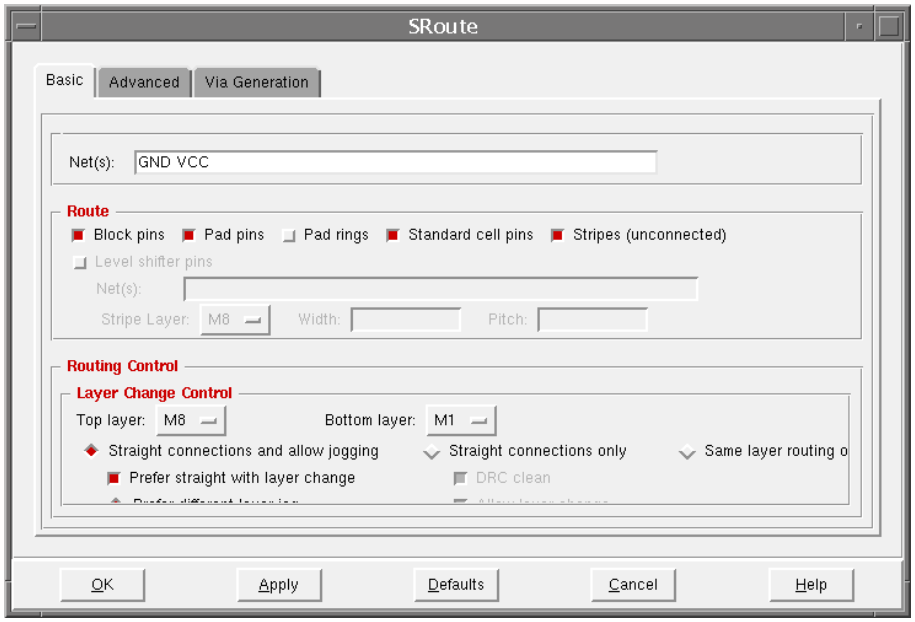

Figure 17: SRoute Pane

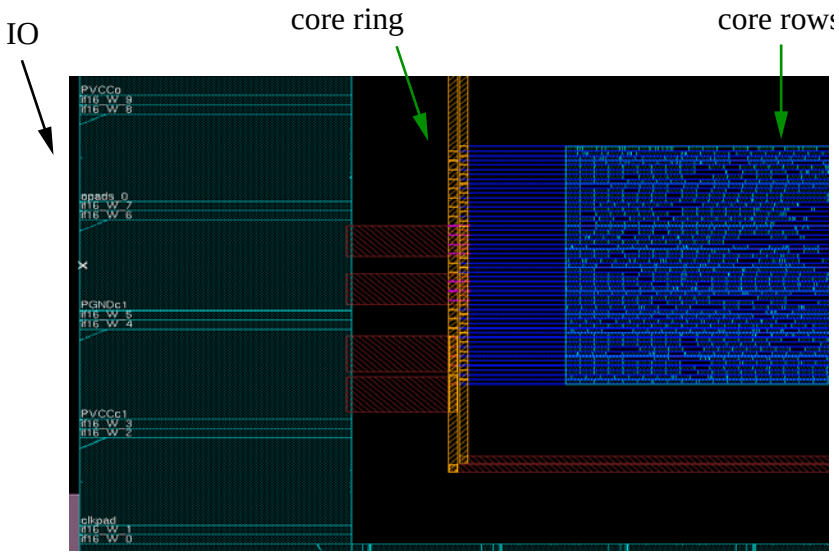

Figure 18: Core-ring to core-supply pad connection.

4.10 Signal Routing

After all block and cells are placed and the clock tree is routed the cells on the core rows need to be connected as specified in the netlist. This is accomplished by a routing tool named *NanoRoute*, which is incorporated into SoCEncounter. Select

Route ⇒ *Nanoroute*⇒ *Route*

to open the NanoRoute pane. Assure that *Global and Detailed Route* is selected. Furthermore, it is recommended to select *Fix Antenna*, and, if a *.sdc* file was provided *Timing Driven* placement may be selected. The time it takes to complete signal routing depends on the complexity of the design and the timing constraints that need to be met. Start routing by clicking on **ok**.
If the routing job has completed and the signals are not visible select *layout view*.

Save the design as ∗*Placed.enc*.

4.11 Antenna and Connectivity Verification

After the routing job has completed it need to be tested if everything was properly connected in the layout during power- and signal routing. Furthermore, it is necessary to check got antenna violations, which could be due to a long metal wire on a single layer. If it is possible that a charge builds faster than it can be dissipated a large voltage can be developed. If a transistor's gate is exposed to this large voltage then it can be destroyed. This is referred to as an antenna violation. On the drop-down menu select

Verify ⇒ *Verify Geometry*

and keep the standard settings. If any geometry violations were discovered white x's become visible on the layout. After completion select

Tools ⇒ *Violation Browser*

and highlight any violations that were detected. Repeat the verification for the *Process Antenna Violation*.

4.12 Script File

All the commands which were executed during a run of SocEncounter are dumped into a *encounter.cmd* file. This file needs to be cleaned from all excessive commands that were executed on the way. Open *encounter.cmd* in text editor and remove such commands. Save the *cmd* file with a different name and source it in the encounter shell. The entire place and route process should be automatically executed, and result in the same layout as done manually.