

Cadence Auto-Layout Generation from Verilog code

Dr. L. G. Johnson

The following tutorial shows how to automatically generate layouts from Verilog code. This task requires the user to integrate various Cadence tools and acquire a better understanding of how these tools work together.

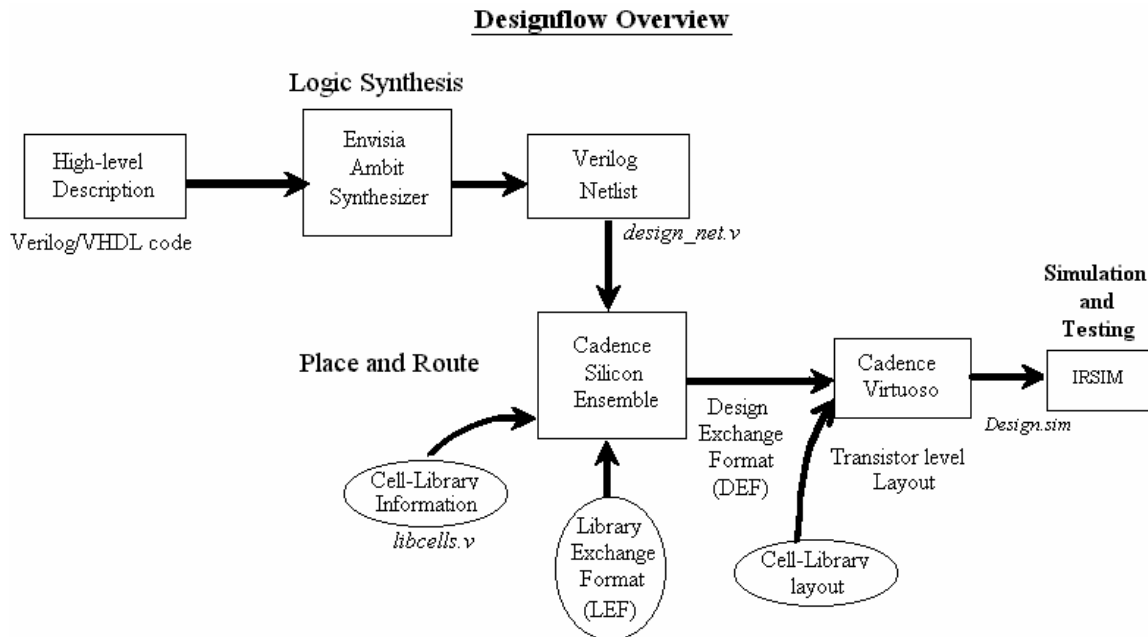
Cadence Tools:

The tools required for this tutorial and their function are given below:

- 1) Envisia Ambient Synthesizer (netlist generation).
- 2) Cadence Virtuoso (layout editor).
- 3) Cadence Silicon Ensemble (auto-place and route).
- 4) IRSIM (layout simulator).

Design Flow:

This tutorial follows the design flow as illustrated below.



The design flow may be divided into three parts.

- a. Logic synthesis: The Verilog code is compiled and synthesized into a Verilog netlist in the **ac_shell** using a **tel** script file. More information with examples is provided in "[Logic Gate Synthesis and Simulation](#)" on the Cadence page.
- b. Auto-Place and route: The Verilog netlist provides the connectivity information of the components (which belong to the cell-library) to the auto-place and route tool (Silicon Ensemble). Apart from this, Silicon Ensemble also requires the following files that need to be imported.

- 1) Library Exchange Format (LEF) file: contains library information for a class of designs. Library data includes layer, via, placement site type, and macro cell definitions. An elaborate description of the syntax and format is beyond the scope of this tutorial.
 - 2) Standard cell-library information file: This contains the behavioral description (in Verilog) of all the components of the library. Silicon Ensemble produces a file called Design Exchange Format (DEF). This contains the design specific information of the circuit and is imported into Cadence Virtuoso to be converted into a layout.
- c. Layout Simulation and Testing: The layout is converted into an IRSIM netlist and simulated.

Adder Design Example

A simple 8-bit adder has been taken as an example for this tutorial. The same procedure may be followed for any logic circuit.

Initial Setup:

- 1) Make a project directory (“adder8_proj” in this example) and run icfb in this directory. Use the following unix commands:


```
mkdir adder8_proj
cd adder8_proj
icfb&
```
- 2) Create a new library (“adder8_lib” in this example) and attach it to the AMI 0.6u cell library.

From the CIW, choose File→New→library...

In the name field of the Create Library window, type: adder8_lib

Click on “Attach to existing technology library” and use the pull down menu to select “AMI 0.60u C5N (3M, 2P, high_res)”

Click on “OK”
- 3) Use the Library Path Editor to make a relative path for the library and add the MSU cell library.

From the CIW, choose Tools→Library Path Editor...

In the Library Path Editor window, find the adder8_lib library and change the path to: ./adder8_lib

Scroll to the bottom of the Library Path Editor window. In the library column, enter: msu_lib

In the path column, type: /app1/cadence/msu/tutorial

In the Library Path Editor window, choose File→Save and then File→Exit

The cell-library layouts:

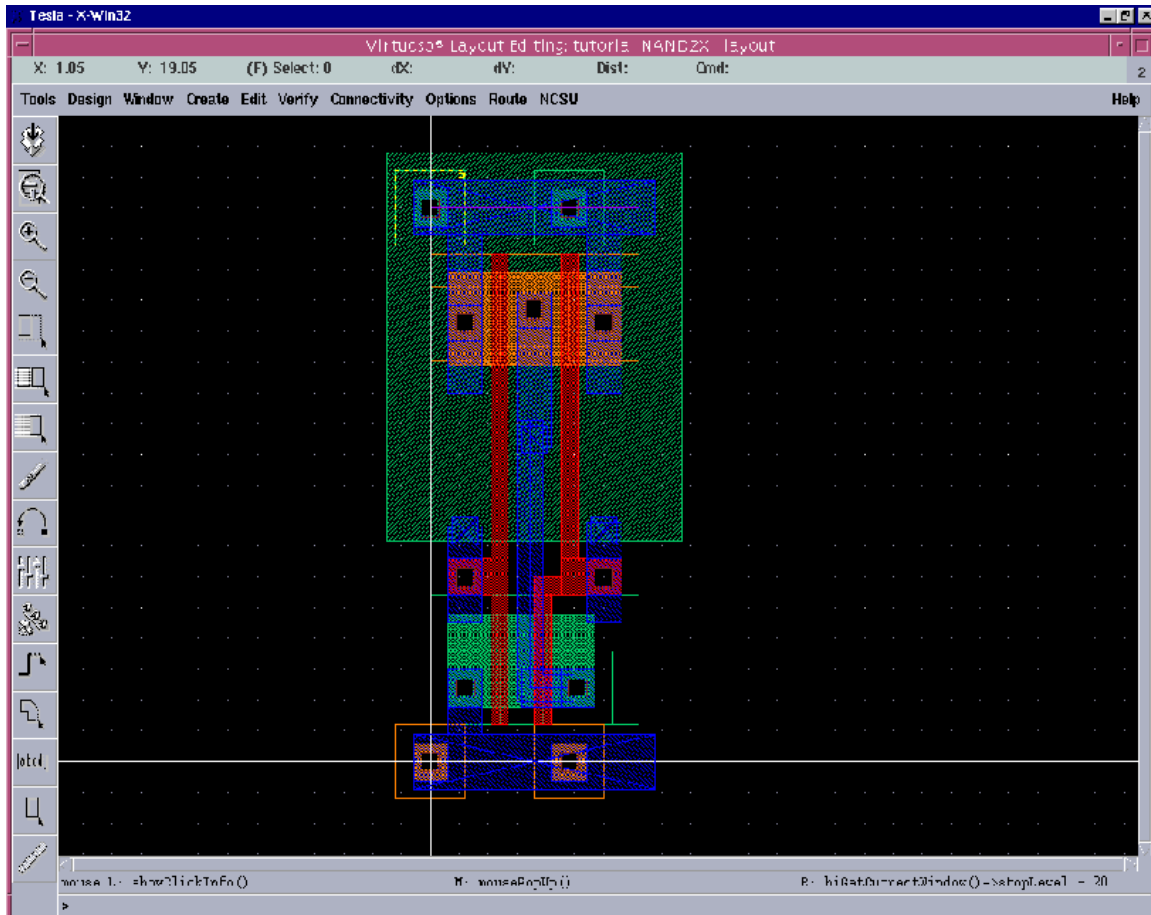
The cell-library layouts form the building blocks for larger layouts. By using a combination of a few types of gates, complex logic can be obtained. The MSU cell-library being used for this tutorial consists of the following logic gates:

INVX1 – Inverter.

NAND2X1 – 2 input nand gate.

NOR2X1 – 2 input nor gate.
DFFSRX1 – D flip flop with Set/Reset.
The rest of the layouts are mainly used for filling.

Example: NAND2X1 layout.



The initial setup is complete. It might be wise to open the library cells to make sure that the layouts can be viewed correctly. If there are errors, repeat the initialization process. Be sure that you have attached to the correct technology file (AMI 0.6u, not AMI 1.6u).

Logic Synthesis

1) Type the Verilog code for an 8-bit adder and save it as **adder8.v** in the project directory.

```
module adder8 (Cout, S, A, B, Cin);  
    output Cout;  
    output [7:0] S;  
    input [7:0] A;  
    input [7:0] B;
```

```

input Cin;
reg [8:0] SUM;
reg [7:0] S;
reg Cout;
wire [7:0] A, B;

always @(A or B or Cin)
begin
    SUM[8:0] = A + B + Cin;
    S = SUM[7:0];
    Cout = SUM[8];
end
endmodule

```

2) Type the following tcl script file and save as **adder8.tcl** in the project directory.

```

do_remove_design -all
read_alf ~/x/lgjohn/public/lib/msu.alf
read_verilog adder8.v
do_build_generic -all
set_wire_load_mode enclosed
do_optimize
write_ver -hier adder_net.v
quit

```

3) At the unix command prompt in the project directory, type **ac_shell**

4) In the ac_shell prompt, type **source adder8.tcl** . A netlist **adder_net.v** is generated.

Auto-Placement and Routing (Silicon Ensemble)

This tutorial uses a Cadence tool called Silicon Ensemble for auto-placement and routing.

Importing Library Exchange Format, Standard Cell Information and the verilog netlist

1) To start Silicon Ensemble, first **cd** into the project directory and type the following command:

seultra &

The Silicon Ensemble window opens.

2) Importing the Library Exchange Format (LEF) File into Silicon Ensemble.

a. In Silicon Ensemble window, click on *File -> Import -> LEF*.

b. In the **Import LEF** form, type the path in the “Selection” field:

/app1/cadence/msu/design_flow/cadence/dp_se/tech/

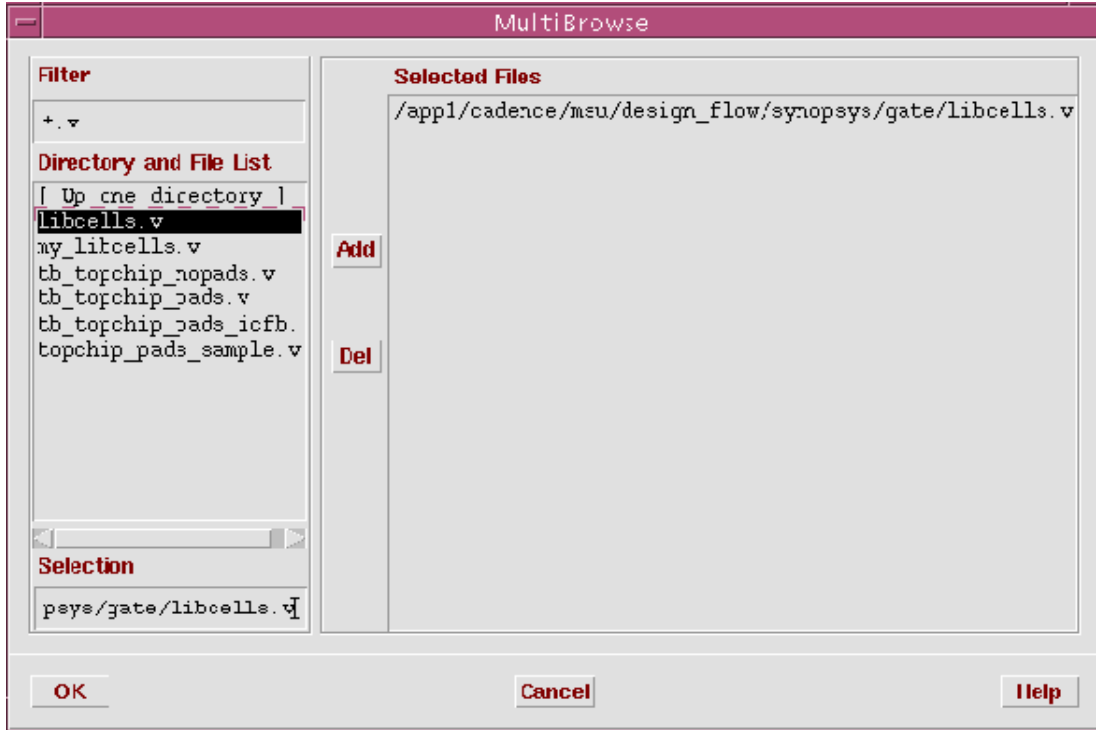
c. Select “jennings_ami06_pads_noqn.lef” and click OK.

3) Importing Standard Cell Information into Silicon Ensemble.

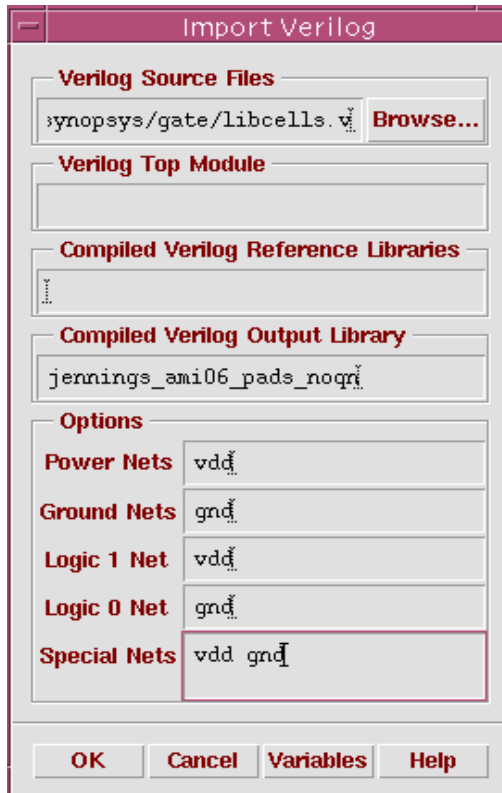
a. In Silicon Ensemble window, click *File -> Import -> Verilog...*

b. Click on ‘Browse...’ to choose the verilog source file (verilog functional description of the library components) and type the path in the “Selection” field:

- /app1/cadence/msu/design_flow/synopsys/gate/
c. Double-click on the file “**libcells.v**” and then OK in the File form.



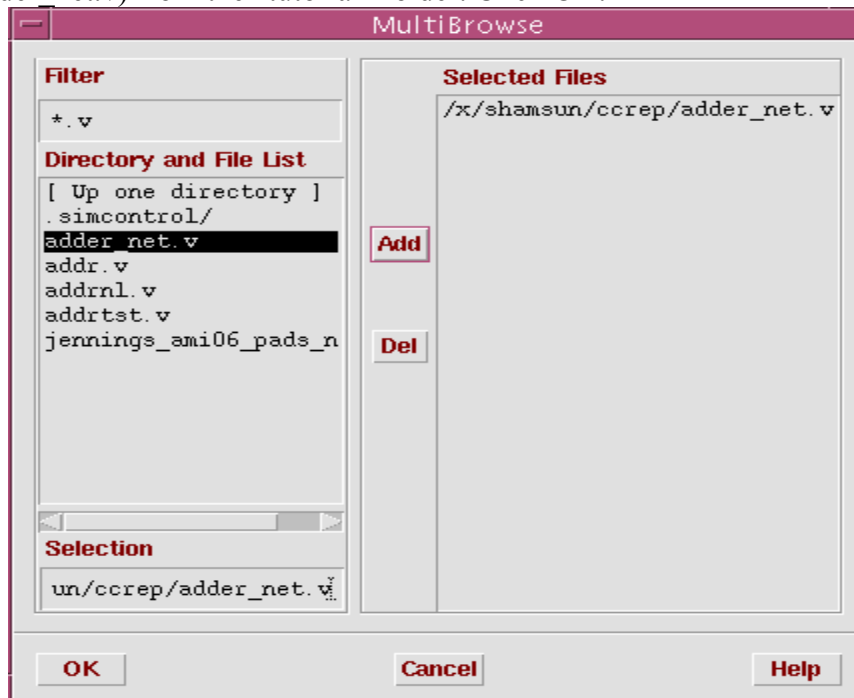
- d. Back in the Import Verilog form, fill in the information as shown below and click OK. This will import the standard cell information into Silicon Ensemble.



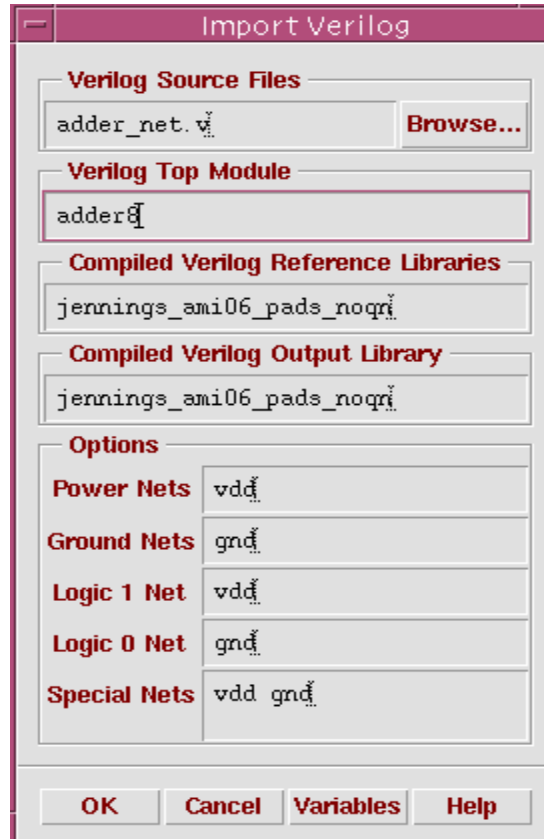
4) Importing Verilog netlist (8-bit adder netlist) into Silicon Ensemble

a. Click on *File -> Import -> Verilog...* again.

Click on 'Browse'. In the File form, delete the *libcells.v* from the selected files list by pressing the 'DEL' button, and add the 8-bit adder netlist (*adder_net.v*) from the "tutorial" folder. Click OK.

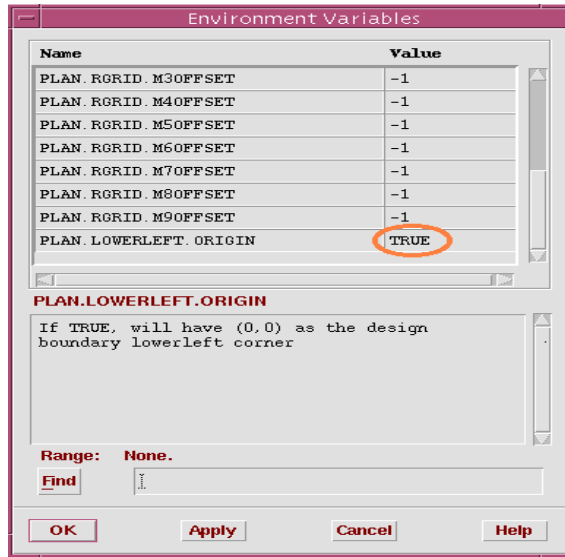


b. In the **Import Verilog** form, type “adder8” for the Verilog Top Module, and add ‘jennings_ami06_pads_noqn’ to the Compiled Verilog Reference Libraries. Click on ‘Yes’ when Silicon Ensemble asks if it is okay to overwrite the original content of the reference library . This will not be destroying the data, but appends it.

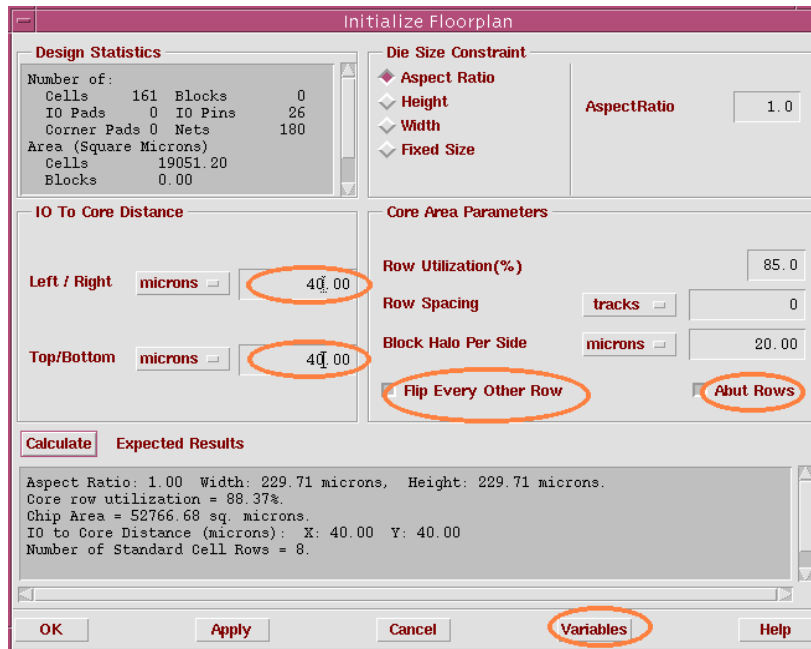


Floorplanning

- 1) In the main window, click on *Floorplan -> Initialize Floorplan*
- 2) In the **Initialize Floorplan** form, click on the Variables button. Another window, the **Environment Variables** form, will pop up.
- 3) In the Environment Variables form, change the variable PLAN.LOWERLEFT.ORIGIN to 'TRUE'.
Click OK in the Environmental Variables form.

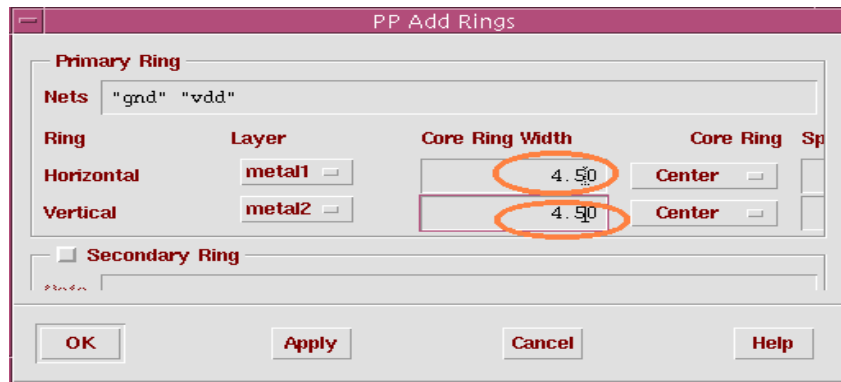


- 4) In the Init Floorplan form, set IO to Core Distance to 40.00 microns for both Left/Right and Top/Bottom.
- 5) Select the 'Flip every other row' and 'Abut Rows' boxes and click OK in the Initialize Floorplan form.



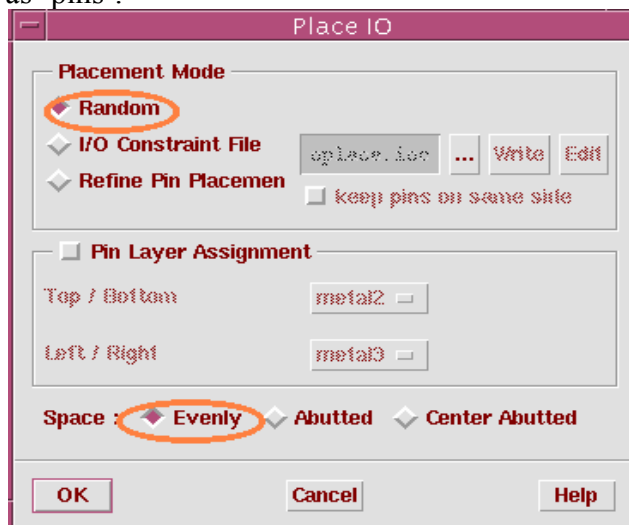
Adding Supply Rings

- 1) In the main window, click on *Route -> Plan Power*.
- 2) In the **Plan Power** window, click on the **Add Rings** button.
- 3) In the **PP Add Rings** window, change both the horizontal and vertical Core Ring Width to 4.50. Core Ring Width refers to the width of the supply rings that are between the I/O and Core sections.
- 4) Click on **OK** in the **PP Add Rings** form, and click on Close in the **Plan Power** form.



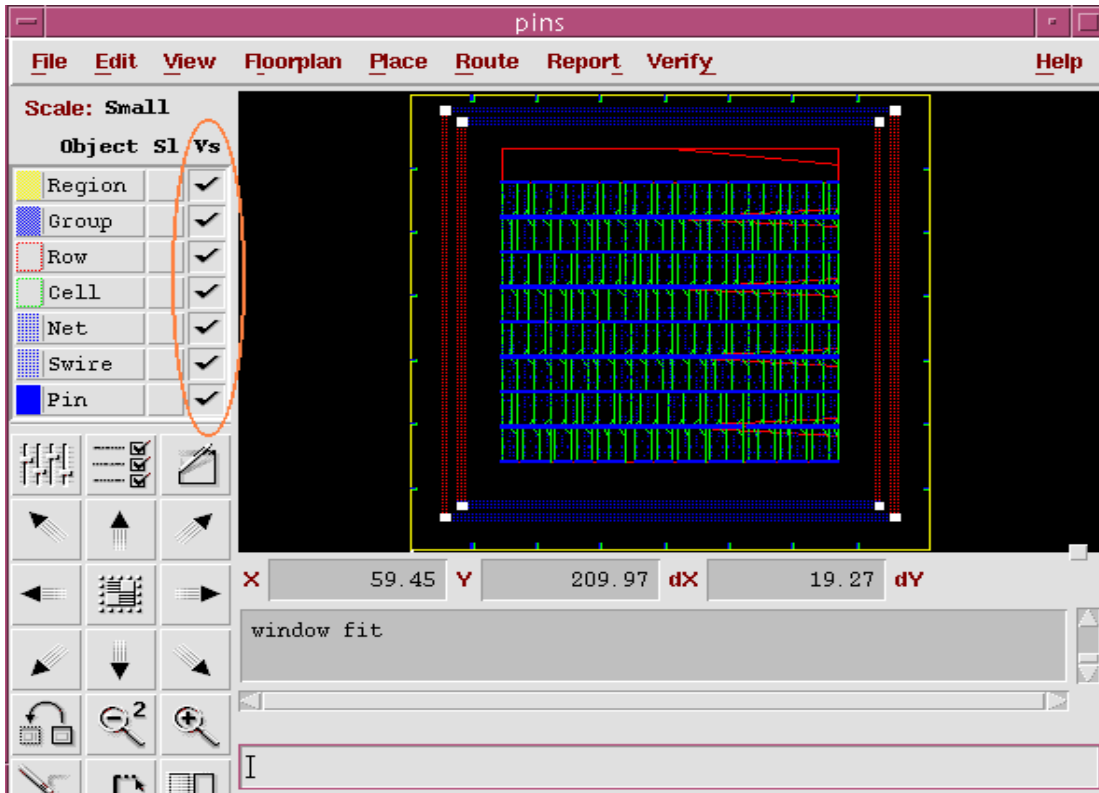
Adding Pins

- 1) To add top-level pins to the layout, click on *Place -> Ios...* in the main window. Choose random placing mode, and space evenly. Then click on **OK**. This will place top-level pins evenly around the perimeter of the layout area.
- 2) Save the design as 'pins'.



Placing Cells (Qplace)

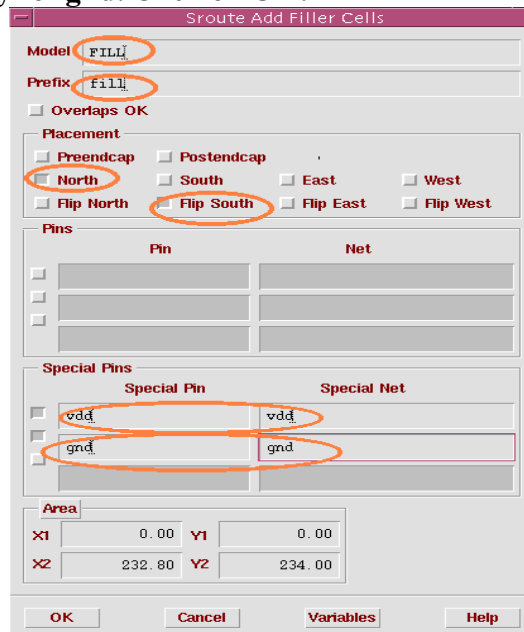
- 1) In order to place cells onto the layout, click on *Place -> Cells* in the main window.
- 2) In the **Place Cells** window, uncheck all the boxes, then click on **OK**.

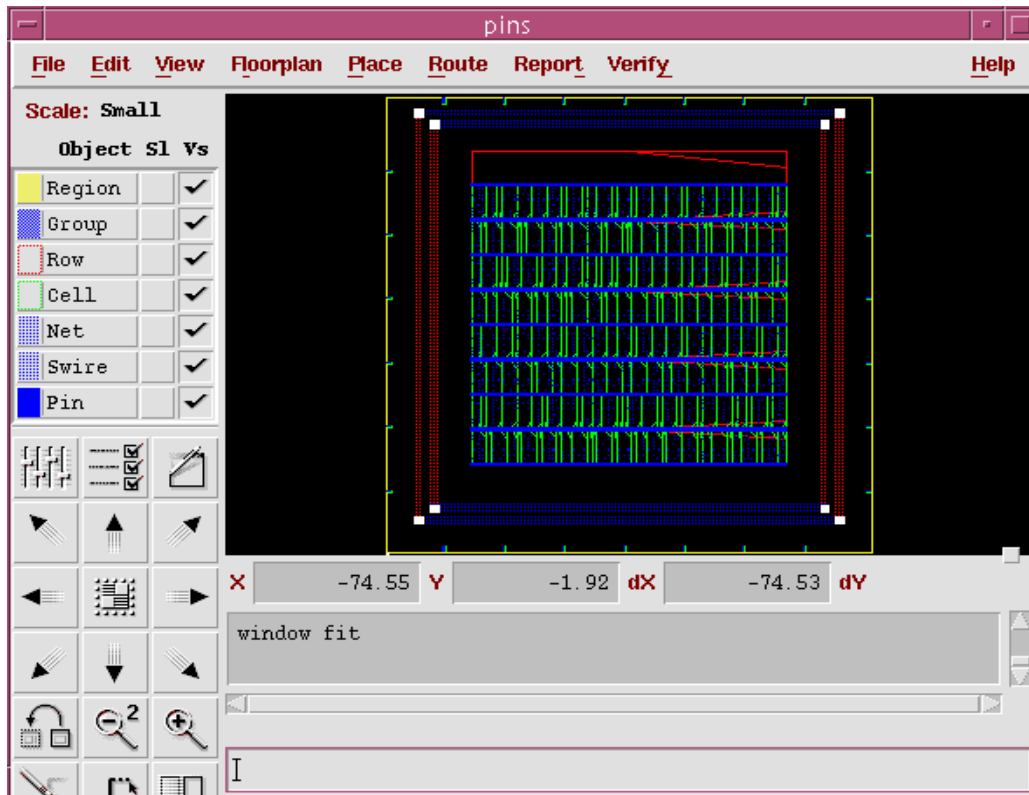


Note: To view nets, special wires, pins, cell boundaries etc., select all the appropriate Vs (visible) fields.

Adding Filler Cells

- 1) Click on *Place -> Filler Cells -> Add Cells*.
- 2) In the **SROUTE Add Filler Cells** form, type in "FILL" for **Model**, and "fill" for **prefix**.
- 3) Select the **North** and **Flip South** buttons. In the **special pins** section, add one entry for **vdd** and one entry for **gnd**. Click on **OK**.



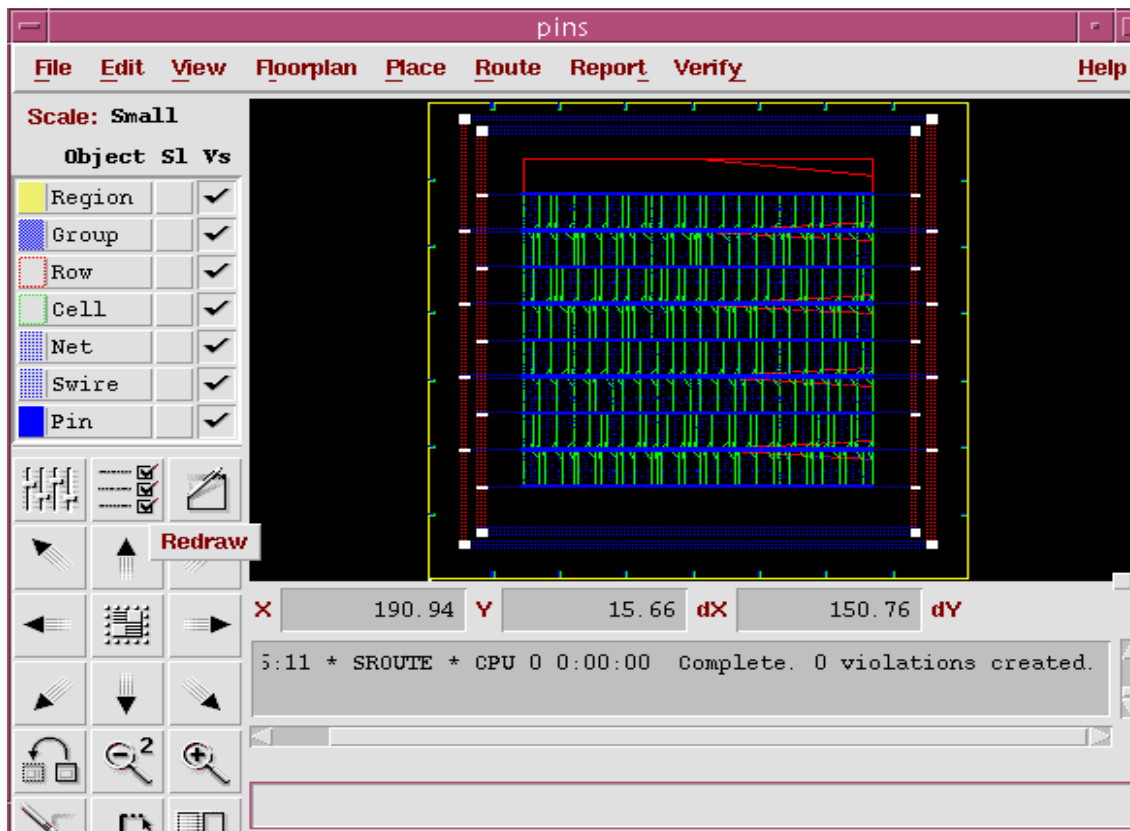
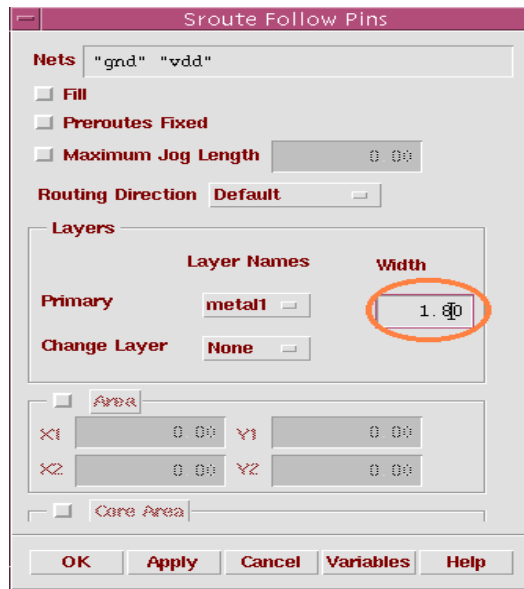


Layout with Filler Cells

Note: The purpose of Filler cells is to provide n-well continuity for the standard cells.

Routing Power

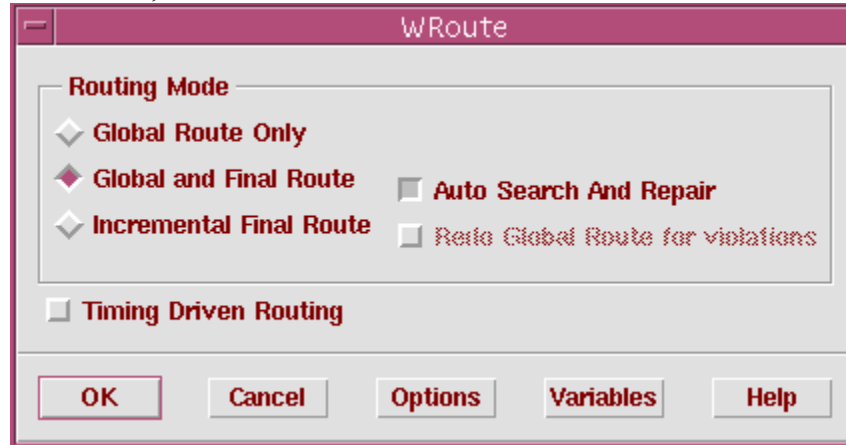
- 1) Click on *Route* -> *Route Power* -> *Follow Pins*.
- 2) In the **Layers** section of the **Sroute Follow Pins** form, set **Width** to 1.80.
Click on **OK**.



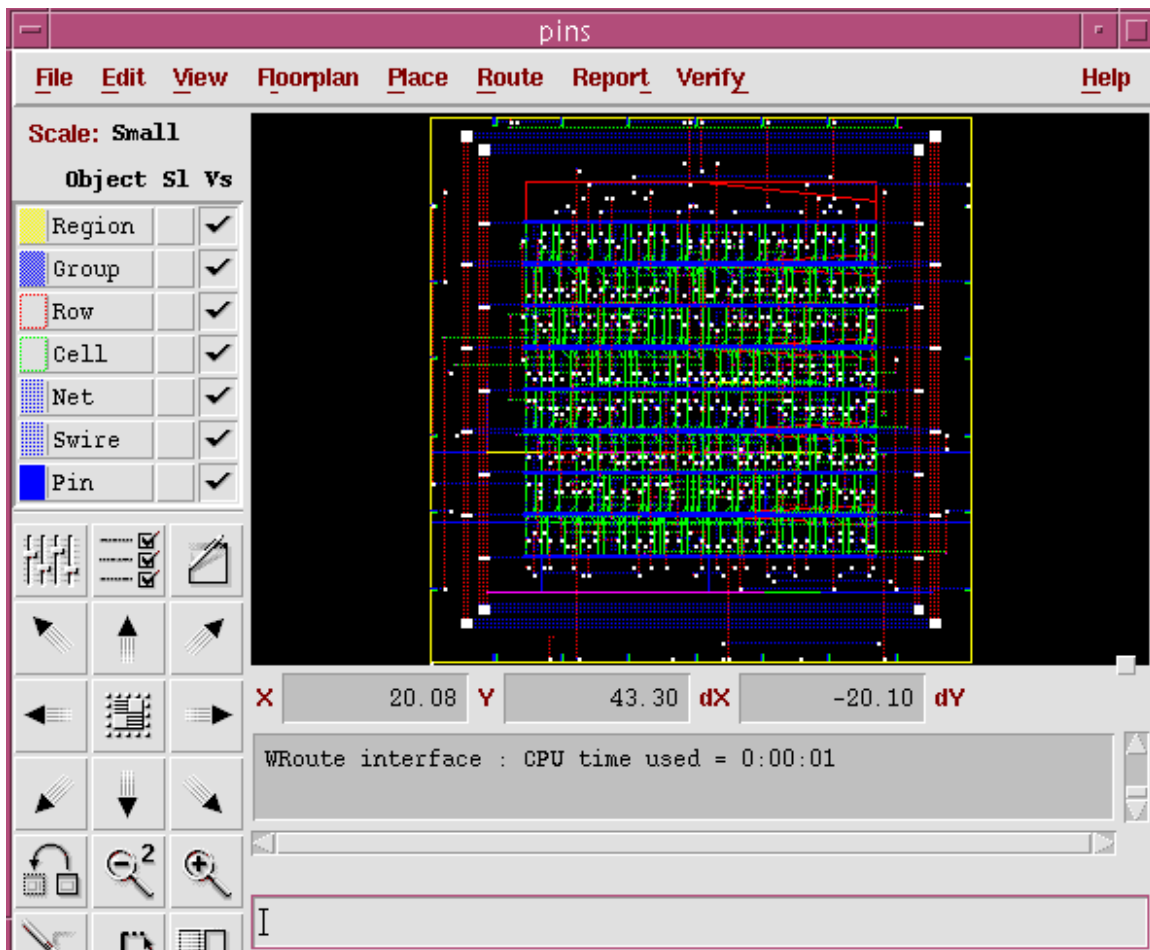
Post-SRoute (Follow Pins)

Wroute

1. Click on *Route* -> *Wroute*.
2. In the **Wroute** form, click on **OK**.

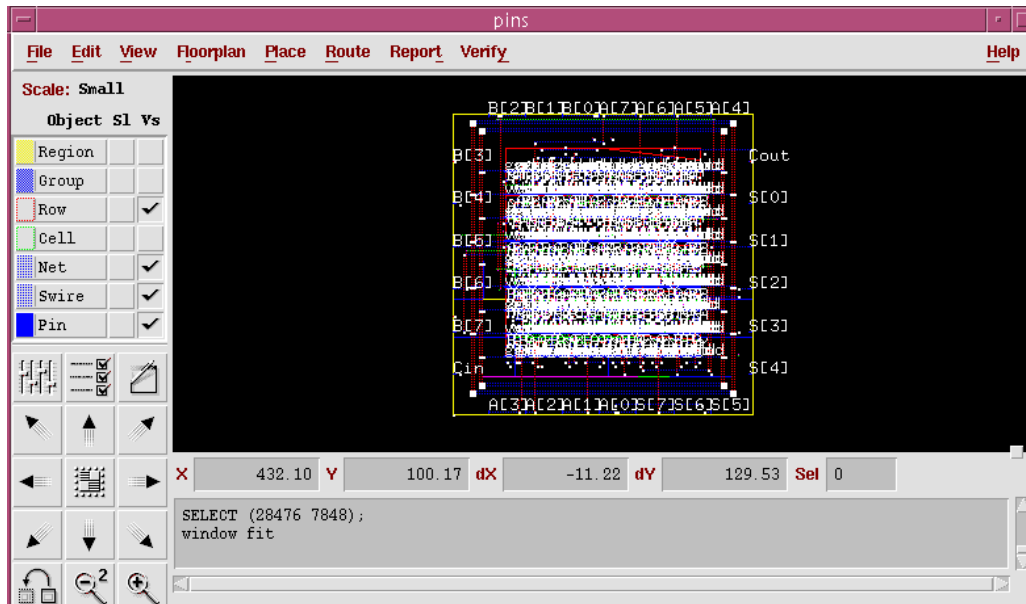


The interconnect routing of the design is complete.



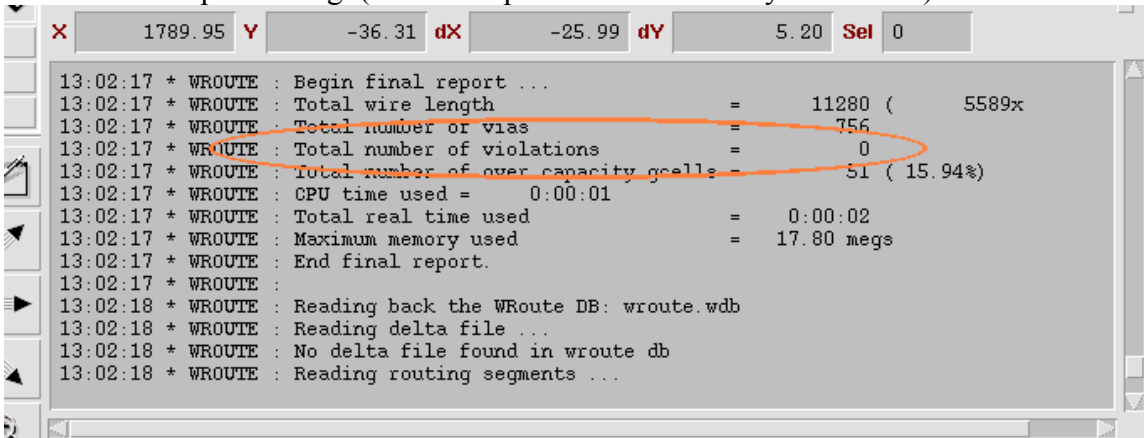
Checking Pin Names

- 1) The names of the routed pins in the standard cells using the **Display Options** form in *View -> Display Options...*
- 2) In the **Names** section of the **Display Options** form, set **Pins** to On. Click OK. The pin names will now be visible.



Checking for Violations

- 1) Violations may appear as 'X' marks on the design. Be sure that there are no violations created during the routing.
- 2) The number of violations created is reported in Wroute step. Fix the violations before proceeding. (This example does not have any violations.)



Total Number of Violations Reported by Silicon Ensemble

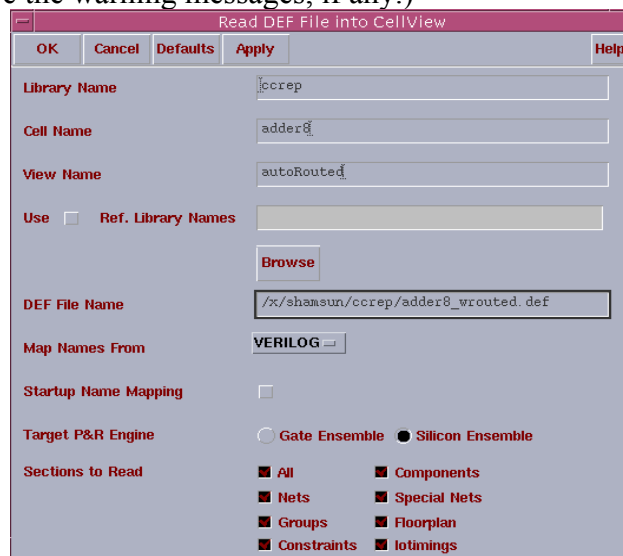
Export to DEF Format

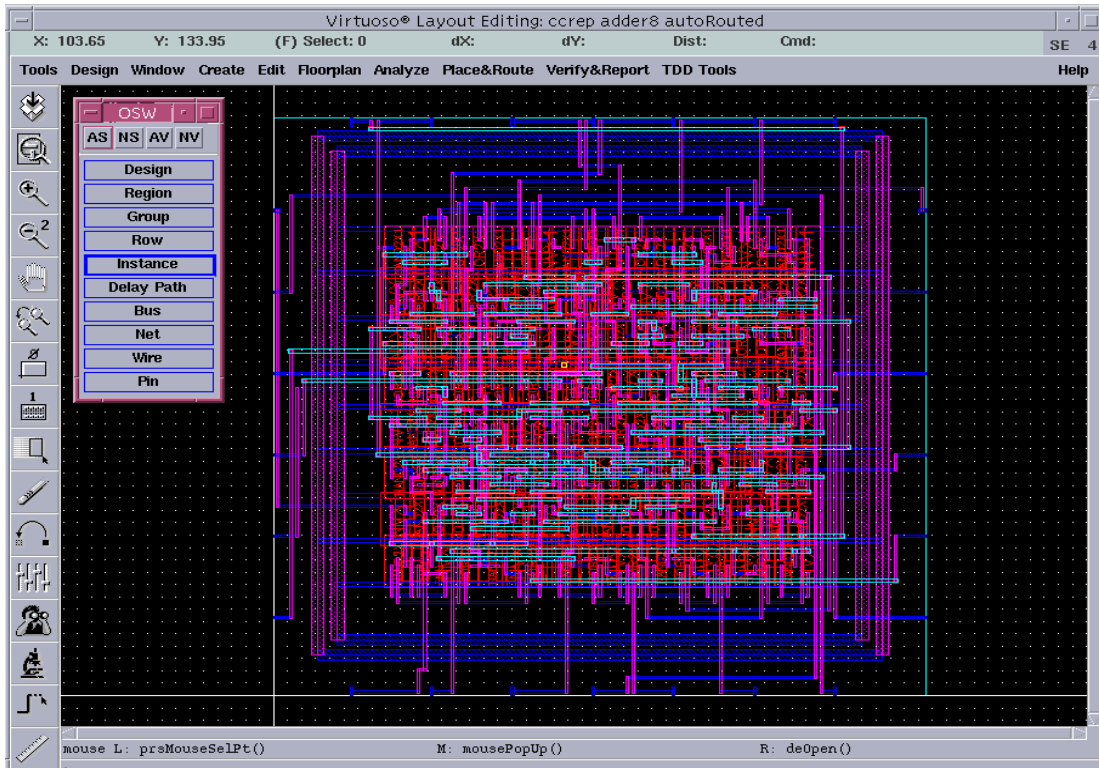
- 1) Click on *File -> Export -> DEF*.
- 2) Type “**adder8_wrouted.def**” for the DEF file name.
- 3) Select the **All** button. Click OK. A DEF file (adder8_wrouted.def) is created in the “tutorial” folder.



Importing DEF into Virtuoso

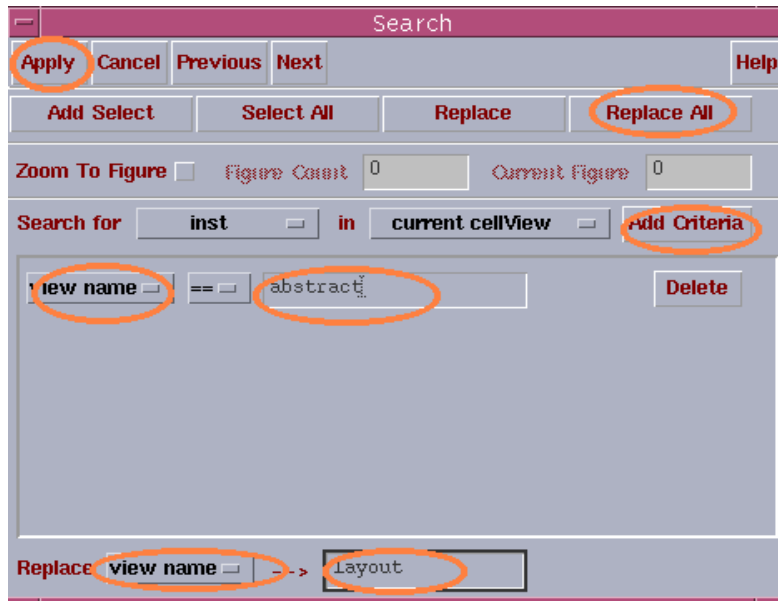
- 1) Type “**icfb &**” to launch ICFB.
 - 2) In the CIW, click on *File -> Import -> DEF*.
 - 3) Enter “**tutorial**” for Library Name, “**adder8**” for Cell Name, and “**autoRouted**” for View Name.
 - 4) Enter “**../tutorial/adder8_wrouted.def**” for DEF File Name.
 - 5) Select the **Silicon Ensemble** button.
- Click OK. (Ignore the warning messages, if any.)



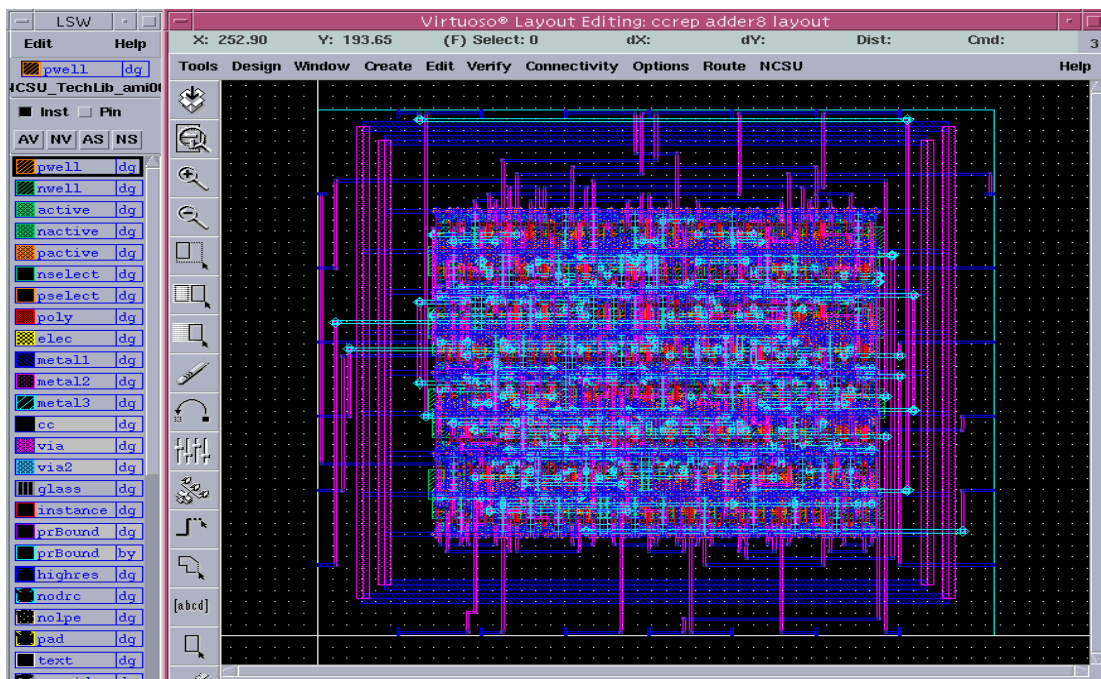


adder8 autoRouted view

- 6) Open the autoRouted view of adder8.
- 7) In the autoRouted view click *Design* -> *Save* to ensure that if anything goes wrong, the autoRouted view can be used as back up.
- 8) Click on *Tools* -> *Layout*. This changes the tool from abstract-editing mode to layout-editing mode. Click on *Edit* -> *Search...*
- 9) In the Search form, click on **Add Criteria** then choose **view name** = abstract. In the **Replace** with **view name**, type layout.
- 10) Click on Apply, then Replace All.
- 11) Close the search form, then click on *Design* -> *Save as...*
- 12) Save the design in the same library and cell, but change the view to 'layout'.
- 13) When the layout editing window is closed, click NO to save any changes (so that the autoRouted view is not over-written.)
- 14) To open the layout view from the CIW, click *File*->*Open...* and select **adder8** as cell-name and **layout** as view name. The layout opens with an Object Selection Window (OSW) box. To open the Layer Selection Window (LSW), click *Tools*->*Layout*. The window appears similar to the one shown in the next page.



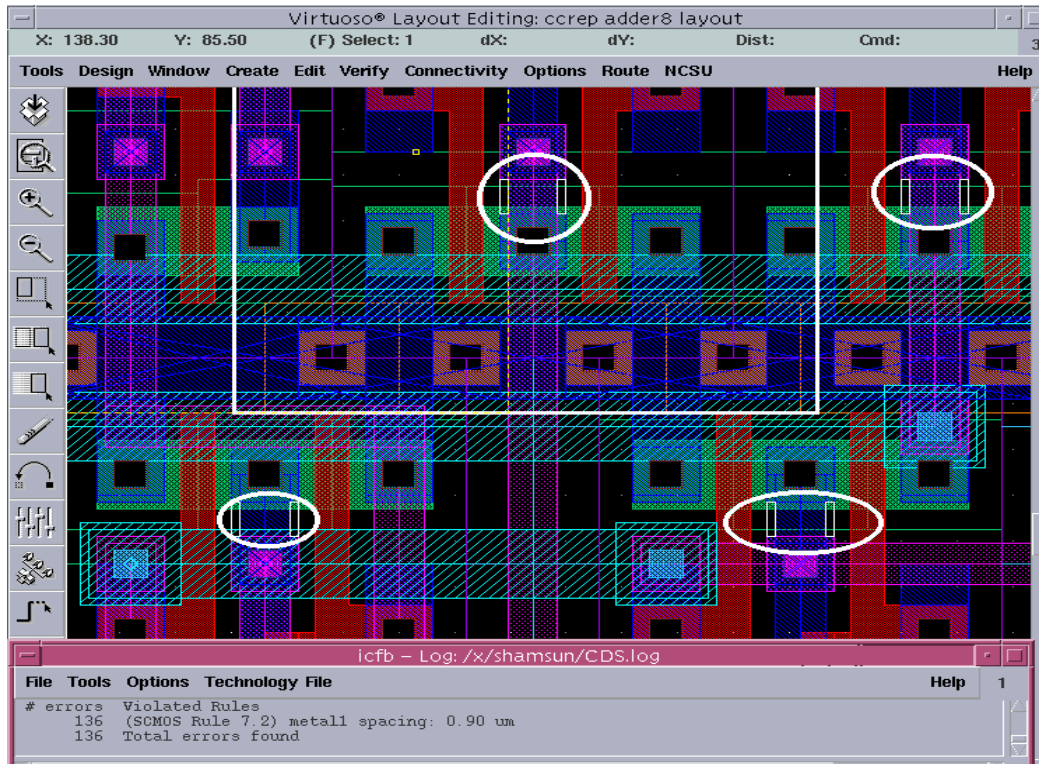
Search Form



adder8 Layout view

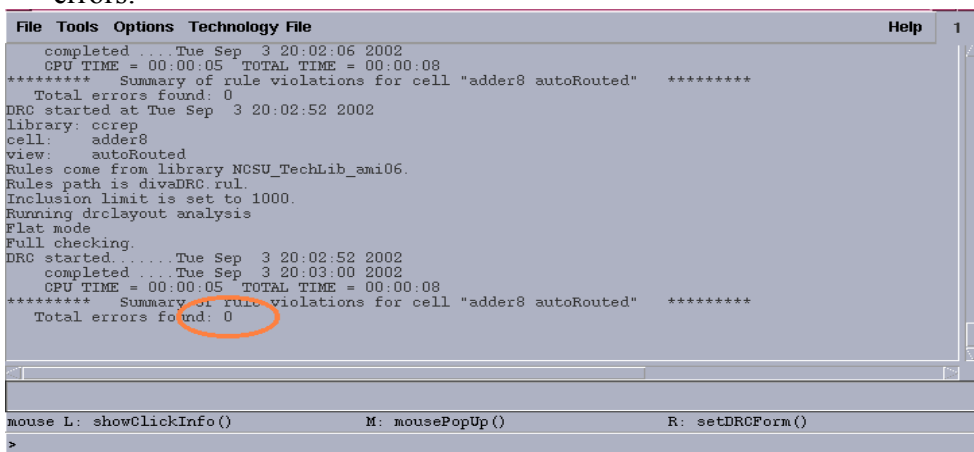
Checking for DRC (Design Rule Check) Errors

Silicon Ensemble does not take care of DRC errors as the lower modules are encapsulated during routing. Thus DRC errors similar to the ones shown may occur.



These errors occur in regular patterns and can be eliminated by minor changes in the lower level cells.

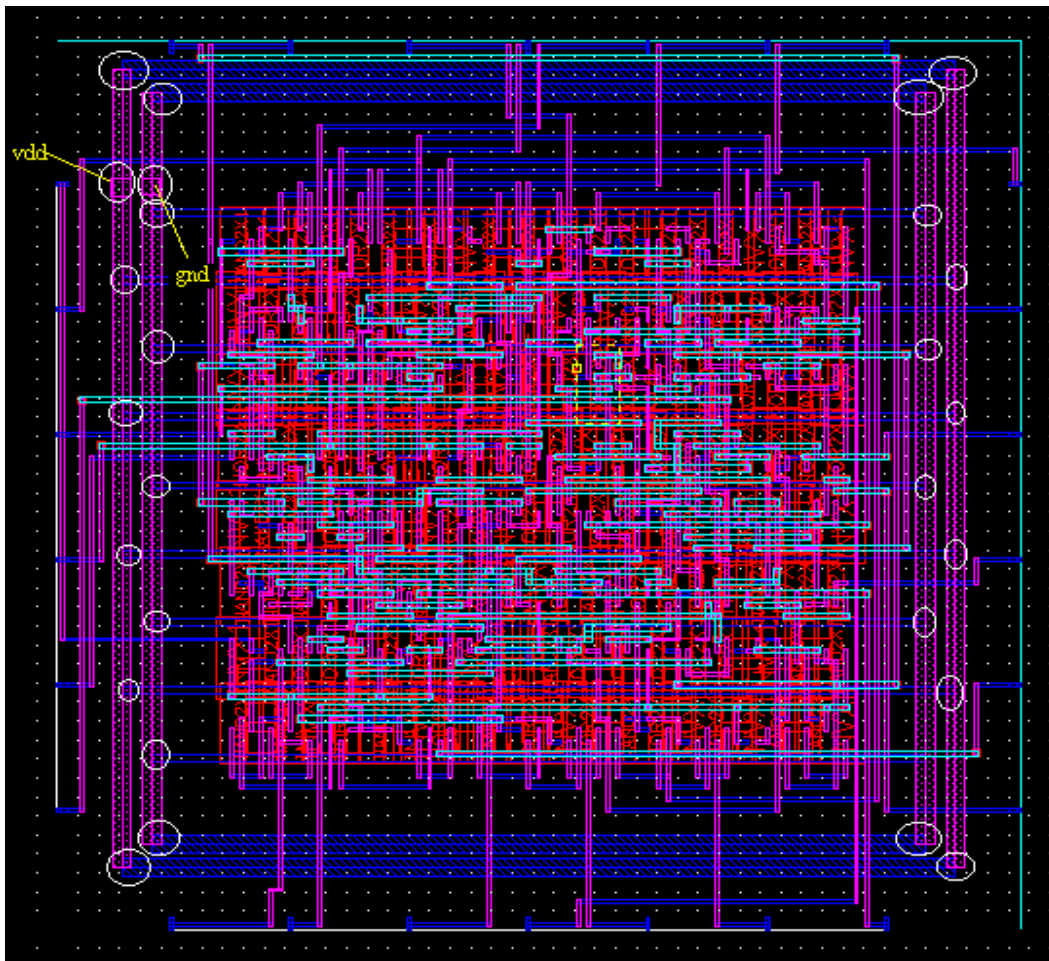
1. Left click near one of the DRC errors and press 'x' in the keyboard to go to the lower level cell.
2. Stretch/modify the appropriate layers (in this case- metal1) and save.
3. Press 'b' to go back to the auto-layout and verify for DRC again. There is a large decrease in the number of errors. Repeat the above steps until there are zero errors.



c. Simulation of layout using IRSIM:

Spectre simulation of large layouts is very time consuming. A convenient alternative is a switch level simulation using IRSIM, which models the transistors as voltage controlled switches. IRSIM is much faster than spectre, but less accurate.

Important Note: The Auto-router does not connect the “vdd” and “gnd” pins. This needs to be done manually in the layout view by putting M2_M1 contacts in the encircled areas. Label the outer and inner supply rings as “vdd” and “gnd” respectively.



- 1) After following the above note, extract the layout and create netlist.
- 2) Using the Spectre to IRSIM netlist parser program(sp2sim.c), convert the spectre netlist to IRSIM netlist. The IRSIM netlist is **Design.sim**.
- 3) Type the example IRSIM test file as given below and save it as **adder8.cmd**.
More tests may be added.

Sample test file:

```
vector Av A_{7:0}
vector Bv B_{7:0}
vector Sv S_{7:0}

w Cout Sv Bv Av Cin

|test1: carryin=0 0+0=0 carryout=0
l Cin
set Av 00000000
set Bv 00000000
s 5
assert Sv 00000000
assert Cout 0

|test2: carryin=0 170+85=255 carryout=0
l Cin
set Av 10101010
set Bv 01010101
s 5
assert Sv 11111111
assert Cout 0

|test3: carryin=1 254+0=255 carryout=0
h Cin
set Av 11111110
set Bv 00000000
s 5
assert Cout 0
assert Sv 11111111

|test4: carryin=1 64+16=81 carryout=0
h Cin
set Av 01000000
set Bv 00010000
s 5
assert Cout 0
assert Sv 01010001

|test5: carryin=0 255+255={carryout(256) + 254}=510 carryout=1
l Cin
set Av 11111111
set Bv 11111111
s 5
assert Cout 1
assert Sv 11111110

|test6: carryin=1 255+255={carryout(256) + 255}=511 carryout=1
h Cin
set Av 11111111
set Bv 11111111
s 5
assert Cout 1
assert Sv 11111111

exit
```

4) Invoke irsim by using the command:

```
$irsim /x/lgjohn/public/scmos0_6um.prm /path/Design.sim -/path/adder8.cmd
```

NOTE that there is a "-" before the .cmd file's path.

7) The simulation output is produced instantly with all matches.

Incase of mismatches, there may be errors in the logic/Verilog code. The auto-place and route tool faithfully lays out the logic implemented and is not responsible for logical errors. The Verilog code needs to be verified thoroughly before starting auto-layout generation.