EECS 627 Tutorial (Winter 2001)

Matthew Guthaus (EECS627 TA)
Office: 2001 EECS
764-8033

These files are available in /nfs/ds.eecs/ecad4/eecs627/tutorial. Feel free to distribute it as long as my name remains attached.

Tool Setup

To setup Cadence's Silicon Ensemble and Verilog, follow the instructions at the ECAD Support Page:

http://www.eecs.umich.edu/eecs/dco/ecad/cadence.html

This tutorial is based on SE 5.3, IC 4.45, Synopsys 2000.05, and NCSU CDK 1.2.

Verilog Design & Simulation

Make a synthesizable Verilog design that functions correctly. Be sure to follow good synthesis guidelines discussed in class. You can run the simplefsm example by executing the following command line:

```
verilog topfsm.v simplefsm.v
```

You will see output similar to the following:

```
                  26 count_out=0000 data_out=1

                  75 count_out=0000 data_out=0

                  76 count_out=0001 data_out=0

                 126 count_out=0002 data_out=0

                 175 count_out=0002 data_out=1
<SNIP>
                2726 count_out=0036 data_out=0

                2775 count_out=0036 data_out=1

                2776 count_out=0037 data_out=0

L28 "topfsm.v": $finish at simulation time 2800
0 simulation events (use +profile or +listcounts option to count)
CPU time: 0.1 secs to compile + 0.0 secs to link + 0.1 secs in simulation
```

```
End of VERILOG-XL 2.7.s020   Sep 28, 1999  13:25:16
```

Synopsys

Set up Synopsys

Copy the config file by typing:

```
    cp /nfs/ds.eecs/ecad4/eecs627/config_files/synopsys/025leda.synopsys_dc.setup
    ~/.synopsys_dc.setup
```

and change the designer name in the file to your own name. Put this file in your home directory and/or your design directory. Synopsys will try to source the file in your home directory first, then in the current directory for project specific modifications. You don’t have to worry about this now.

Running Synopsys scripts

The executable for synopsys with a Graphical User Interface (GUI) is called design_analyzer. This should be in your path if you followed the setup above. This tutorial, however, will use the Design Compiler command line interface. You can use this in the interactive mode by going to:

    Setup->Command Window

This will spawn a new window that has a command prompt. This is the same interface as dc_shell, the script shell that is available from the unix prompt. You can cut and paste scripts into the command line to debug them or you can include your scripts by simply typing

```
    design_analyzer> include <scriptname>
```

You can also watch how the menu commands correspond to script commands in the COmmand Window. This is useful for writing scripts via the cut and paste method. The rest of the tutorial will use the command line interface. After your scripts are written and debugged it is easier to run non-interactive scripts with the command line:

```
    dc_shell -f <scriptname>
```

Getting help on Synopsys

There are three places to look for help on Synopsys:

1. You can use the Synopsys On-Line Documentation (SOLD) for help on almost everything. This is a hypertext document that has many examples and explanations of the error messages.
2. If you are using a command on the command line (discussed later), you can get help by typing “help <command>”. This will give you a listing similar to a man page for each command.
3. If you set up the MANPATH correctly above, you can also type “man <command>” at a unix prompt for descriptions of every command.

Reading Verilog files

Read in your Verilog files using analyze and elaborate. The following script will read the modules specified in module_list (with a .v appended).

```
/* RTL modules */
module_list = { module1, module2, module3 }

/* Clean up and create the design library */
sh "rm -rR ./CORE_ELAB"
sh "mkdir CORE_ELAB"
define_design_lib CORE -path ./CORE_ELAB

/* Analyze and elaborate */
foreach (file_name, module_list)
{
analyze -work CORE -format verilog file_name + ".v"
 if (dc_shell_status != 1) {
sh echo ’Analyze Error:’ file_name
quit
}
elaborate -update -work CORE -arch "verilog" file_name
}
/* Makes sure all needed designs are in memory */
link
```

The link command ensures that all necessary designs are read in. You can read the simple example verilog by executing the command:

```
analyze -format verilog  {"simplefsm.v"}
elaborate simplefsm -arch "verilog" -update
link
```

Creating clocks

Define the clock for the design. This can be done like the following:

```
clk_period = 50
clk_high_time = 0.5 * clk_period
clk_skew = 0.03 * clk_period     /* 3% of clock period as skew */
clk_name = "clk"
create_clock -name "gclk" -period clk_period -waveform {0 clk_high_time} clk_name
set_clock_skew -uncertainty clk_skew -propagated clk_name
set_dont_touch_network find(port, "clk")
```

Double click on the functional view of the design in the Synopsys window. You will get a view of the module. You should see a square wave on the clock port.

Constraining

Now that you've set the clock frequency, you need to set an input and output delays on the ports. This can be done using the set_input_delay and set_output_delay commands. You will also want to set a load (set_load) on the output ports.

As a default, in my scripts I set a high delay for all inputs (except the clock) and outputs. This is a safe-guard for ommitted constraints. The delay is strict so that a timing error will probably show up in the report files.

```
/* Generic delay - make it very strict */
set_input_delay 0.8 * clk_period -clock gclk { all_inputs() - clk_name }
set_output_delay 0.8 * clk_period -clock gclk all_outputs()
```

Now set the input and output constraints like this:

```
/* inputs */
in_delay = 0.68 * clk_period
set_input_delay in_delay -clock gclk  find(port, "data_x")
set_input_delay in_delay -clock gclk  find(port, "data_y")

/* outputs */
out_delay = 0.75 * clk_period
count_delay = 0.65 * clk_period
set_output_delay  out_delay -clock gclk  find(port, “data_out”)
 set_load 3.5 find(port,”data_out”)
set_output_delay count_delay -clock gclk  find(port, “count_out*”)
set_load 1.5 find(port,”count_out*”)
```

Compiling

Now you're ready to compile the design. Most of the time, you will want to use “-map_effort medium” for the compile option. Another option that you may turn on is “-boundary_optimization.” This allows the inputs/outputs of any submodules to be modified if it helps synthesis. Run these commands:

```
current_design simplefsm
set_wire_load tsmcwire
compile -map_effort medium
```

This will load the target library, allocate resources, and try to meet your timing constraints. It will take a while to run. You will see Synopsys try to match appropriate Designware library components to your design. After it meets timing, it fixes design rule errors (max_fanout, max_transition, max_capacitance), and then minimizes the area. The result will be a gate level netlist that, hopefully, meets the timing. (It will - I tested it!).

Generating timing reports

To check that your compilation has met timing, you will need to generate reports for the simplefsm. Here are the most useful reports that you will want to generate:

```
/* these are the constraints you set */
report_constraints -all_violators -verbose > report.constraints
/* these are the end points of the worst case paths */
report_timing -path end -delay max -max_paths 20 -nworst 2 > report.paths
/* these are the full paths of the worst case paths */
report_timing -path full -delay max -max_paths 5 -nworst 2 > report.full_paths
/* this is the wire and logic area */
report_area > report.area
```

If there are timing problems, you have several options

- Lessen the constraint
- Compile with high effort or boundary_optimization
- Change the architecture

Repeat this methodology until you meet your synthesis goals.

Output

Now you can output your design. For place and route, output a Verilog netlist. For synthesis in later hierarchy, output a Synopsys database (.db) file. Here are some examples:

```
write -hierarchy -format verilog -output simplefsm.gate.v
write -hierarchy -format db -output simplefsm.db
```

You can also output a timing file (SDF) to include in your Verilog simulations. Do this by typing

```
write_timing -output  "simplefsm.sdf"  -load_delay cell -format sdf -context verilog
```

Back-annotation

You can back-annotate in your Verilog top file by uncommenting the $sdf_annotate line in the topfsm.v file. Re-run all your verification!

```
verilog -v /nfs/ds.eecs/ecad4/eecs627/lec25sc25.rev6334/lec25dscc25/verilog/lec25dscc25.v
topfsm.v simplefsm.gate.v
```

You will see several warnings like “Warning! Too few module port connections.” Ignore these. They are because Synopsys doesn’t hook up the QBAR output of the flip-flops all the time.

Flat (non-hierarchal) standard cell physical design

The primary tool for flat standard cell design using the Cadence products is Silicon Ensemble Deep Sub-Micron (SEDSM). This tool takes a library and a hierarchal netlist as input. Optionally, it can take a timing information, but this isn't supported by our flow yet. It converts the hierarchal netlist to a flat netlist. Then you can choose the aspect ratio, power plan, and place and route the entire netlist. This is only acceptable for smaller (<10,000-15,000) cells that require no block (memories,analog, etc) placement. It should also be noted that the input netlist should also include the pad cells from the I/O library. SEDSM will create the pad frame for you as well.
Make sure SEDSM is set up according to the ECAD Support page. Try running Silicon Ensemble DSM by typing "sedsm -m=72 &" at the unix prompt. If you have memory problems during routing, you can try in ncreasing the memory size in the command line option. If this doesn't work, then you must follow the hierarchal design flow presented later.

Getting help on Cadence

The documentation tool for Cadence is called Openbook. It should be in your path if you set up your dot files according to the above description. So simply type "openbook" at a unix prompt and a new hypertext window will open up. You can follow the links to "Floorplanning/Place and Route." This will have documentation on how to use Silicon Ensemble. There is also information on Verilog-XL in the documentation.

Reading the library

Begin by reading in the library exchange format (LEF) files located at

```
/nfs/ds.eecs/ecad4/eecs627/lec25sc25.rev6334/lec25dscc25/lef/lec25dscc25.lef
/nfs/ds.eecs/ecad4/eecs627/lec25sc25.rev6334/lec25ioscp25/lef/lec25ioscp25.lef
```

using the File->Import->LEF... command. Make sure to load them in the shown order.

Reading your design

Now read your gate level Verilog netlist by selecting File->Import->Verilog. Select your source files and click on the "Add" button. Also, make sure to include the library Verilog files at:

```
/nfs/ds.eecs/ecad4/eecs627/lec25sc25.rev6334/lec25dscc25/verilog/lec25dscc25_stub.v
/nfs/ds.eecs/ecad4/eecs627/lec25sc25.rev6334/lec25ioscp25/verilog/lec25ioscp25_stub.v
```

The list on the right displays the files to be read in to Silicon Ensemble. Specify the top module of your design in the corresponding field.
For the parameters, change all the vdd!'s to VDD and gnd!'s to VSS. These are the names used in the library. Also specify "clk" as a special net. Click on the Variables button and specify the bus delimiters as "[]" since this is the default output of Synopsys. Now click "OK" to procede. It will take a few minutes for the Verilog to be parsed.

Initializing the floorplan

First, select Floorplan->Initialize to start your design floorplan. This is where you specify the aspect ratio, cell placement density, etc. Change the IO distance to core to 50u on the left/right and top/bottom. This will give room for a power ring. Also select to flip and abut adjacent rows. Press “OK.” You should see the cell rows in red and the IO cells (if you have pads included) in red.

Power routing

Create a VDD and VSS ring around your design using the Route->Plan Power menu. This pops up a new window that lets you add rings and stripes to the design. Make sure that you left space around the standard cell rows when you initialized your design. Select Add Rings from the window. Specify the core ring width to be 9um for M1 and M2 and the names as VDD and VSS. Cllick OK and SEDSM will make two rings around your design.
Now click on Add Stripes. Add a 9um M2 stripe with a spacing of 5um between VDD and VSS by filling in the blanks. You can specify the number of stripes using a constant (Number of sets) or with a stride spacing (Set to Set Spacing). Depending on the design size, choose the appropriate number and press OK.

Placement

Now it is time to place the IO pad cells or just metal ports for a higher level of your design. You can give a specified order of the IOs or have SEDSM choose a random even spacing. You will get better results if you plan based on the floorplan of a hierarchal design. For a flat design, random is probably sufficient. Select Place->IOs and choose random placement. Press OK.
Now select Place->Cells. Check the box next to Pin Placement and press “OK”. This will perform a general placement of your design using QPlace. I am still working on getting Timing Driven placement working. This requires converting the Synopsys .lib file to Cadence .tclf format. Make sure to select these special options in by clicking on the variables button:

```
QPLACE.LLC.IGNORE.LAYER.1 "FALSE"
QPLACE.LLC.IGNORE.LAYER.2 "FALSE" ;
QPLACE.LLC.IGNORE.LAYER.3 "FALSE" ;
QPLACE.LLC.IGNORE.LAYER.4 "FALSE" ;
QPLACE.LLC.IGNORE.LAYER.5 "FALSE" ;
QPLACE.LLC.PREWIRE.KEEPOUT TRUE ;
```

Otherwise, QPlace will put cells directly beneath your power stripes. Since these are in M2 and the cell ports are in M1, they will all be unaccessable and produce routing errors. Press OK to perform the placement. After you perform cell placement, it is advisable to replace the IO pins.This will produce better results.

Importing the timing library

Before you do special net routing or anything for timing extraction, you must import the CTLF (compiled timing library format) file. Do this by executing:

```
FINPUT CTLF CTLFFILE /nfs/ds.eecs/ecad4/eecs627/lec25sc25.rev6334/tlf/
lec25dscc25.ctlf ;
FINPUT CTLF CTLFFILE /nfs/ds.eecs/ecad4/eecs627/lec25sc25.rev6344/tlf/
lec25ioscp25.ctlf ;
```

This is the timing information for the IO and standard cell library so make sure you have both of these libraries loaded as described before or you will see warnings. You will see warnings that there is no timing information for the VDD pads, VSS pads, corner pads, ERC cells, ad cell, etc. These are ok, because they don't have timing associated with them really. Note that importing this file takes a while.

Special net routing

You can route your clock first by selecting Route->Clock Route. More is described in the SEDSM manual. Select Route->Connect Ring. Make sure Follow Pins routing is selected and that blocks and stripes will be connected. Press OK. This will connect the standard cell power stripes in M1 and the M2 power stripes you added in the power planning to the power rings.

Routing

Perform a global routing by selecting Route->WRoute. Make sure that global and final route is selected and that auto search-and-repair is on. Now press OK. Your design is now routed.

Checks

Once you're done and everything is routed, run Verify->Geometry and Verify->Connectivity. For the Geometry, specify to allow different cell violations. There are some errors in the .lef file that are between metal blockages and metal1. These aren't valid so you can ignore them. Metal blockages aren't really in the design. These are SE's own DRC/LVS operations, however, don't trust them! For final tape out, you should use Dracula and Diva to do DRC and LVS, respectively. There also may be a few errors near the edges the standard cell rows where there are unconnected VDD and VSS stripes. But because the cell rows are flipped and abutted, these are OK.

Extraction

You can create an SDF file by using Report->Delay. This will allow you to simulate your design with physically extracted parameters. Make sure to select "GE/SE" instead of "Pearl." Pearl requires you to set up a GCF constraints file first. This GCF file is also required for timing driven placement and routing. I'm not supporting this currently, but you're welcome to try it out.
To balance your clock networks manually, you can export an RC extraction of your nets using Report->RC. You can then iterate to balance the nets by adding/subtracting load cells to different clock tree branches. Note that this contributes to the timing convergence problem, because when you replace your design, the placement and, hence, the load of your clock nets will change. This can be tricky!

Back-annotation

Use the SDF file generated and export a Verilog file from SEDSM using File->Export->Verilog. Then do a back-annotation like before. You can use the exported Verilog or your structural Verilog -- they shouldn't be any different. Change the $sdf_annotate command to use the new file and re-run all your simulations. You can also look into doing timing analysis in SEDSM using Report->Timing Analysis. Refer to the manual section on the Affirma Pearl Timing Analyzer User Guide for more information.

Command line interface

You can also use script files or the command line interface to perform all these actions. Scripts can be executed using File->Execute. The following script file shows a generic flat design:

```
 # Sample Silicon Ensemble script for EECS 627
 # Matt Guthaus
 # 11/9/99

 # Read the LEF library
 INPUT LEF FILENAME "lec25dscc25.lef" REPORTFILE "importlef.rpt" ;
 FINPUT CTLF CTLFILE "/nfs/ds/ecad4/eecs627/lec25sc25.rev6334/tlf/lec25dscc25_TT.ctlf" ;

 # Read the Verilog design
 SET VAR INPUT.VERILOG.POWER.NET "VDD";
 SET VAR INPUT.VERILOG.GROUND.NET "VSS";
 SET VAR INPUT.VERILOG.LOGIC.1.NET "VDD";
 SET VAR INPUT.VERILOG.LOGIC.0.NET "VSS";
 SET VAR INPUT.VERILOG.SPECIAL.NETS "VDD VSS";
 INPUT VERILOG FILE "lec25dscc25_stub.v ../synopsys/encrypt_logic/v1_syn/encrypt_logic.gate.v1.v" LIB
"cds_vbin" REFLIB "cds_vbin"
 DESIGN  "cds_vbin.encrypt_logic:hdl" ;

 # Initialize the floorplan
 FINIT FLOOR;
 FINIT FLOORPLAN  rowu 0.85 rowsp 0 blockhalo 2000 f a 1 abut xio 5000 yio 5000 ;

 # Make power/ground rings around the design
 BUILD CHANNEL ;
 CONSTRUCT RING
 NET "VSS"
 NET "VDD"
 LAYER M1 CORERINGWIDTH 900 SPACING CENTER BLOCKRINGWIDTH 0
 LAYER M2 CORERINGWIDTH 900 SPACING CENTER BLOCKRINGWIDTH 0 ;
 ADD STRIPE    NET "VSS" NET "VDD" DIRECTION Vertical LAYER M2 WIDTH 900 SPACING 500
COUNT 1 ALL ;
 DISPOSE CHANNEL ;

 # Place the IOs
 IOPLACE AUTOMATIC STYLE EVEN ;

 # Place the standard cells
 SET VAR QPLACE.LLC.IGNORE.LAYER.1 "FALSE" ;
 SET VAR QPLACE.LLC.IGNORE.LAYER.2 "FALSE" ;
 SET VAR QPLACE.LLC.PREWIRE.KEEPOUT TRUE ;
 QPLACE NOCONFIG ;

 # Refine the random pin placement
 SET VAR QPLACE.PLACE.PIN "refine" ;
 QPLACE PIN NOCONFIG ;

 # Save your design!
 SAVE DESIGN "NTQP" ;
 FLOAD DESIGN "NTQP" ;
 # Connect the standard cell power/ground stripes to the rings.
 CONNECT RING NET "VSS" NET "VDD" STRIPE BLOCK ALLPORT IOPAD ALLPORT IORING FOL-
LOWPIN ;

 # Do any special net routing here (e.g. clk)

 # Perform a Warp Route of the entire chip
 SET VAR WROUTE.FINAL TRUE ;
 SET VAR WROUTE.GLOBAL TRUE ;
 SET VAR WROUTE.INCREMENTAL.FINAL FALSE ;
 WROUTE NOCONFIG ;

 # Save your design!
 SAVE DESIGN "NTWR" ;
 FLOAD DESIGN "NTWR" ;

 # Export the LEF block
 OUTPUT LEF BLOCK FILENAME "NTWR.lef" ;
```

Hierarchal standard cell physical design

The last line of the previous script exports a LEF block. This can be used in higher level designs to manually do a hierarchical placement. First, however, you must add the following line after the MACRO statement in your LEF file (note the space before the semicolon):

```
CLASS BLOCK ;
```

Then add the following line after the SITE statement in your LEF file:

```
CLASS CORE ;
```

This allows SE to recognize the LEF component as a block and not a core cell. The site class tells SE that the block is to be placed in the core and not the pad frame. When you include your library LEF file, also include the block LEF files. Then, when you read your Verilog, the Verilog hierarchical flattener will stop at leaf cells OR blocks. Be sure to include the Verilog of the block as well or else SE will produce an error.This strategy for hierarchical design also applies to placing analog and memory blocks. If you have a really big block, you might want to make a "_stub.v" file like I did with the library. This is just the module port definitions with nothing inside. It will speed up importing the Verilog considerably.

Placing blocks

After initializing your floorplan, but before you place standard cells, select Place->Blocks to place your macro blocks. Then select Floorplan->Update Core Rows. This removes the rows that are coveredby the placed blocks. You cannot place a block only design. There must be at least one standard cell in the design. Don't ask me why, but this is stated throughout the SE manual. After you have placed your blocks in this manner, you can place the place the standard cells and route like normal.

Creating the pad frame

After your design is nearly ready, you need to make a top level module that includes the IO cells for the pad frame. These cells are described at:

file:/nfs/ds/ecad4/eecs627/lec25sc25.rev6334/lec25ioscp25/databook/databook.html

Be sure to include the corner cells and power/ground like this:

```
cdvdd p0.PAD(vdd));
cdvss p1(.PAD(vss));
cornerbl c0(); // bottom left
cornerbr c1(); // bottom right
cornertl c2(); // top left
cornertr c3(); // top right
```

When you read your libraries in SE, also refer to the IO library at:

/nfs/ds.eecs/ecad4/eecs627/lec25sc25.rev6334/lec25ioscp25/lef/lec25ioscp25.lef

The Verilog library and _stub.v files are located at:

/nfs/ds.eecs/ecad4/eecs627/lec25sc25.rev6334/lec25ioscp25/verilog/lec25ioscp25.v

and

/nfs/ds.eecs/ecad4/eecs627/lec25sc25.rev6334/lec25ioscp25/verilog/lec25ioscp25_stub.v

The Floorplan->Initialize command will differentiate the pad frame cells and the standard cells & blocks in your core. Place the IO cells before you place the core cells.

Connecting power

To connect the power, ground, and ESD protection rails in your pad frame, issue the command;

<COMING SOON>

Exporting GDSII

<COMING SOON>

Full custom physical design

For more (and better) information on full-custom design processes, take a look at the NCSU CDK totorials at:

http://www.ece.ncsu.edu/cadence/tutorial.html

They have tutorials onn design capture, simulation, and design methodology. This tutorial isn’t intended for full-custom design. It is intended for standard cell design. However, there is some brief intro information on setting up the NCSU CDK at the University of Michigan. Good luck!

Set up Cadence

Set up Cadence exactly like described on the ECAD Support Page. However, you must add this environment variable setting:

```
setenv CDS_SITE /nfs/ds.eecs/ecad4/eecs627/local
```

You must also copy the dot files that set up the NCSU Cadence Design Kit (CDK) like this:

```
cp /nfs/ds.eecs/ecad4/eecs627/config_files/NCSU_CDK/* ~
cp /nfs/ds.eecs/ecad4/eecs627/config_files/NCSU_CDK/.* ~
```

These will set up your Cadence environment for use with the TSMC 0.35u and 0.25u design kit from North Carolina State University. You can now enter “icfb &” to start Cadence.

Getting help on full custom physical design

I did some basic schematic entry and DRC/LVS, but I do not know much about full custom design in Cadence. If there are problems, first consult the manual (Openbook) then come talk to me. I’ll try to help you, but cannot guarantee anything. The NCSU CDK comes with no guarantee or suppoort as well, however, there is documentation available at:

```
file:/nfs/ds.eecs/ecad4/eecs627/local/doc/index.html
```

This includes a link to the TSMC design rules as well.

Creating a design library

On startup, the NCSU CDK scripts will start the library manager. However, to create a new library, you MUST USE THE COMMAND INTERFACE WINDOW (CIW). This is the small rectangular window at the bottom of the screen. The File->New->Library command in the CIW uses the NCSU CDK scripts to attach the library to the proper process library.

Enter a design library name, and select to attach it to an existing techfile. You can optionally give a path for the library. By default, it will be put in the directory you started Cadence in. Select either of the TSMC processes (as long as it matches any standard cell design you will do). These are listed in their drawn minimum feature size. The 0.40u is actually 0.35u and the 0.30u is actually 0.25u. Press OK and a new library should appear in the Library Manager.

Creating schematics

To create a schematic, select File->New->Cell View from the Library Manager. Select your design library and enter a cell name and enter “schematic” for the view name. Select Composer-Schematic for the Tool and press OK. You will see a new cell and cell view appear in the Library Manager.

To edit the schematic, double click on it in the Library Manager. Cadence will open the Composer program and the NCSU CDK will open a Component Browser window. There are many simple digital and analog schematics provided with the CDK. You can use these or create your own.

There are pop-up descriptions when the mouse is over an icon. Use this to figure out the commands in Composer. It's pretty intuitive. If you need more help, see the manual in Openbook.

Creating layout

Create another cell view and name it "layout." Select Virtuoso as the tool. After you have done this, double click on the layout view and Virtuoso will open in layout editor mode. Two new windows will open. One is the Layer Select Window and the other is the Layout Editor. Create your inverter according to the schematic you made. The layout rules for the processes are available on MOSIS's web site at http://www.mosis.org.

Design Rule Checking (DRC)

Select Verify->DRC from the editor window. Press OK. A DRC check of the current design is performed and errors are highlighhted in the editor window. There are more options in the DRC menu that you may want to look at when performing DRC on entire designs. To examine the errors, select Verify->Markers->Explain. Then, when you click on an error in the editor, it will describe the DRC violation marker text window.

Layout Verses Schematic (LVS)

Run Verify->Extract to extract a netlist from your layout. Then select Veify->LVS from the editor window. Select "Browse" for the left netlist selection and pick your design's schematic. On the right netlist, select "Browse" again, but select the extracted netlist you created. Press OK. LVS will run and errors will be displayed in the CIW.

Parasitic extraction

Coming soon to a Sun workstation near you...