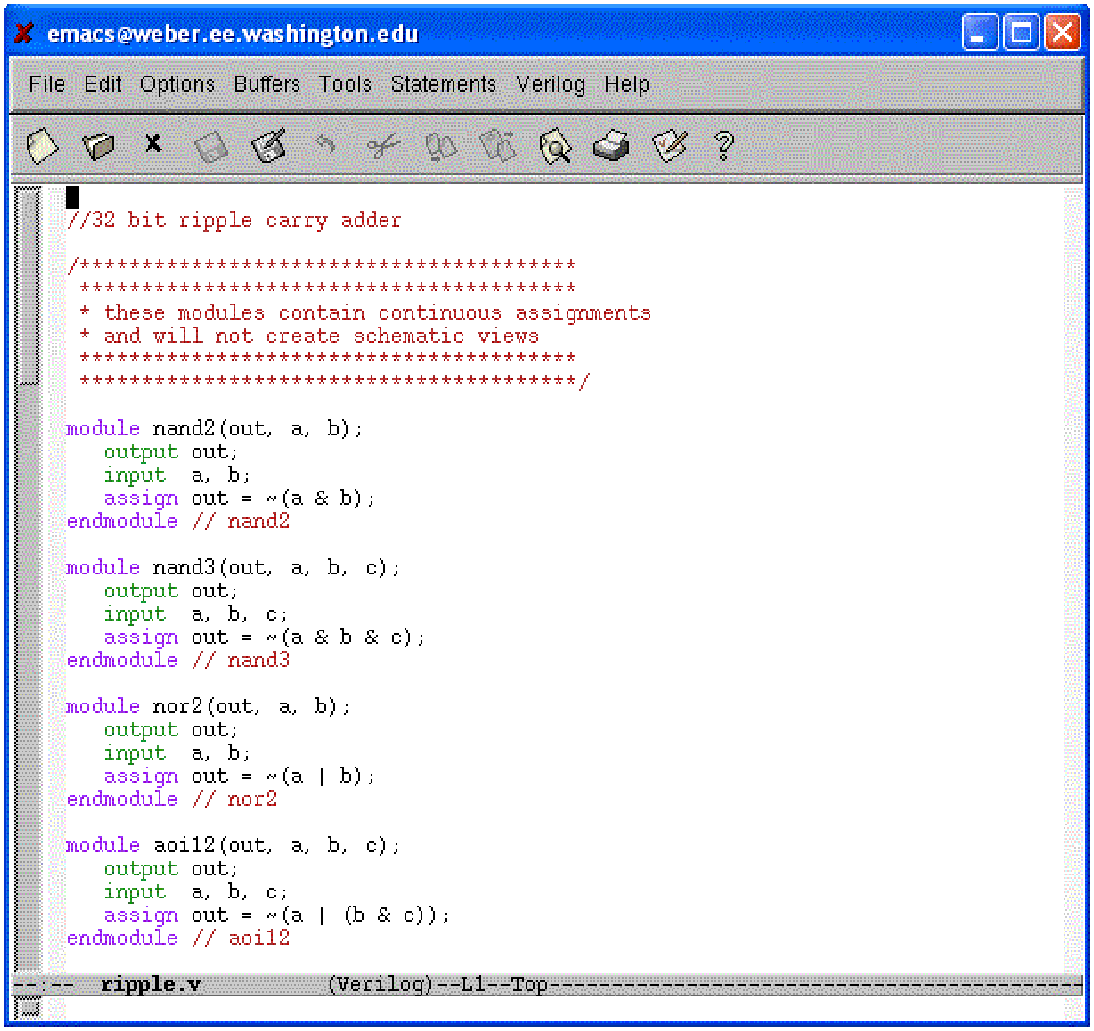
Importing Verilog to Cadence Schematic

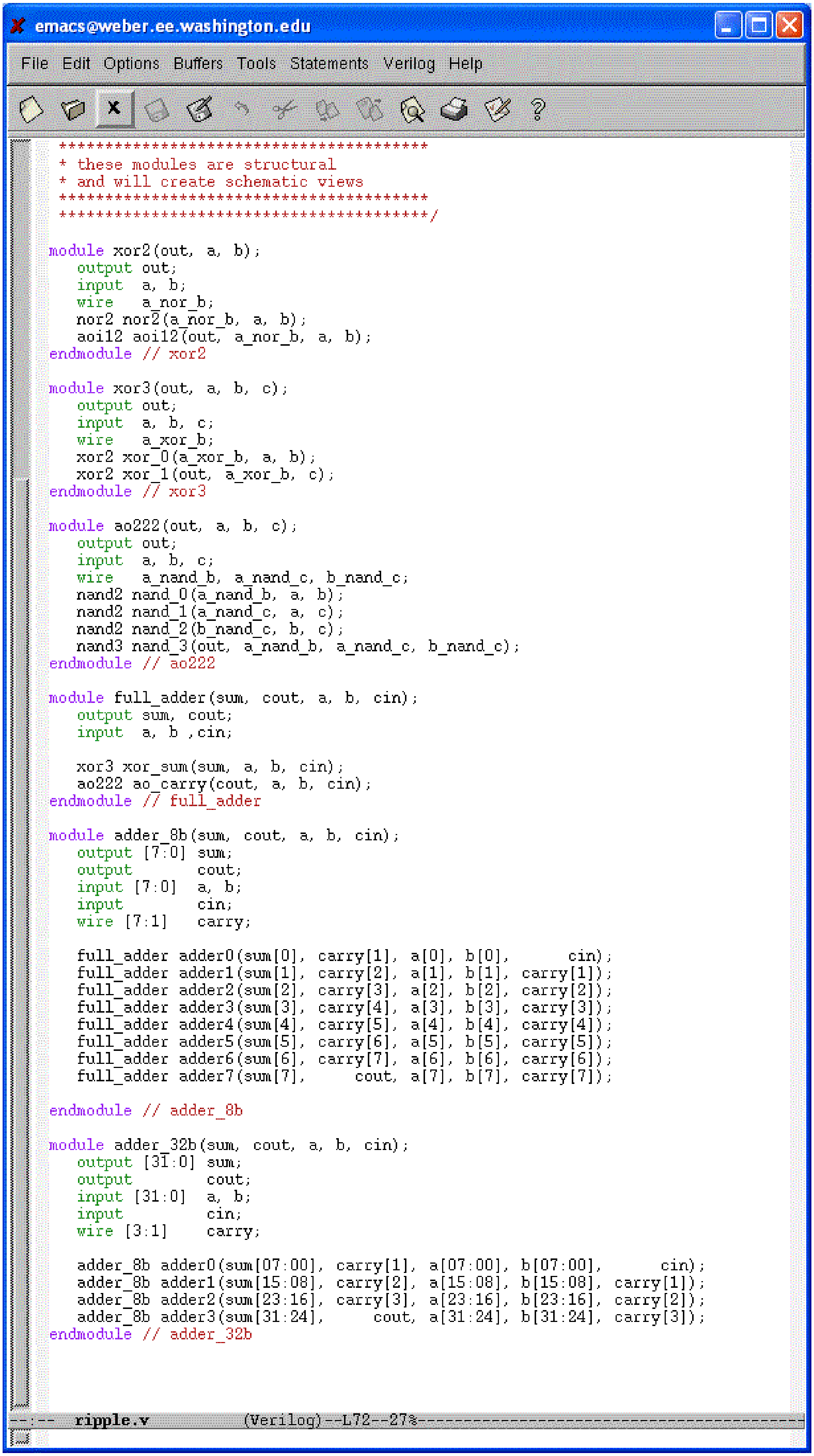
To create a Cadence schematic from structural verilog, you must write all of your verilog code calling modules in your cell library. The following is an example of the library needed to implement a 32 bit ripple carry adder.


emacs@weber.ee.washington.edu
File Edit Options Buffers Tools Statements Verilog Help

```
//32 bit ripple carry adder

/*****************************************
 *****************************************
 * these modules contain continuous assignments
 * and will not create schematic views
 *****************************************
 *****************************************/

module nand2(out, a, b);
   output out;
   input  a, b;
   assign out = ~(a & b);
endmodule // nand2

module nand3(out, a, b, c);
   output out;
   input  a, b, c;
   assign out = ~(a & b & c);
endmodule // nand3

module nor2(out, a, b);
   output out;
   input  a, b;
   assign out = ~(a | b);
endmodule // nor2

module aoi12(out, a, b, c);
   output out;
   input  a, b, c;
   assign out = ~(a | (b & c));
endmodule // aoi12
```

--:-- ripple.v (Verilog)--L1--Top--

When these modules are imported into cadence, they will not produce, schematic views. The schematics for behavioral models must be created manulally. The following modules form the 32 bit adder and are all defined in terms of the the cell library. Be careful when you name your nets because these nets will become the spice nets -- and spice is case insencitive.

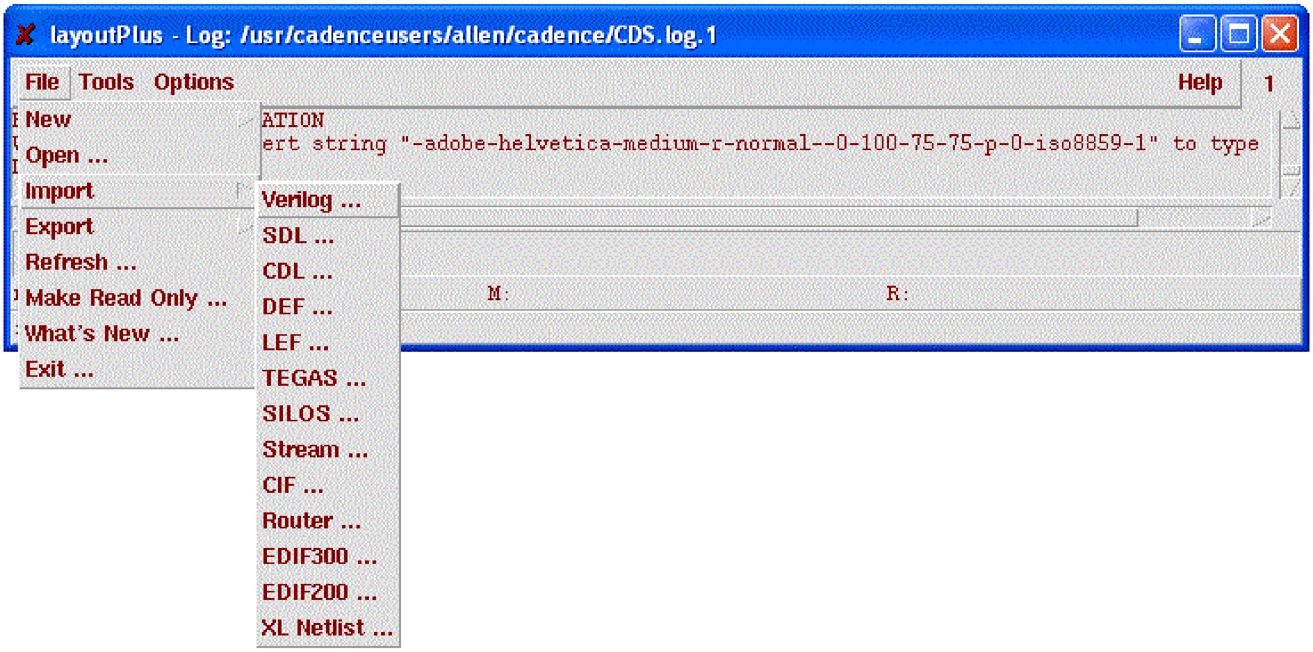

emacs@weber.ee.washington.edu
File Edit Options Buffers Tools Statements Verilog Help

```
 ******************************************
 * these modules are structural
 * and will create schematic views
 ******************************************
 ******************************************/

module xor2(out, a, b);
   output out;
   input  a, b;
   wire   a_nor_b;
   nor2 nor2(a_nor_b, a, b);
   aoi12 aoi12(out, a_nor_b, a, b);
endmodule // xor2

module xor3(out, a, b, c);
   output out;
   input  a, b, c;
   wire   a_xor_b;
   xor2 xor_0(a_xor_b, a, b);
   xor2 xor_1(out, a_xor_b, c);
endmodule // xor3

module ao222(out, a, b, c);
   output out;
   input  a, b, c;
   wire   a_nand_b, a_nand_c, b_nand_c;
   nand2 nand_0(a_nand_b, a, b);
   nand2 nand_1(a_nand_c, a, c);
   nand2 nand_2(b_nand_c, b, c);
   nand3 nand_3(out, a_nand_b, a_nand_c, b_nand_c);
endmodule // ao222

module full_adder(sum, cout, a, b, cin);
   output sum, cout;
   input  a, b ,cin;

   xor3 xor_sum(sum, a, b, cin);
   ao222 ao_carry(cout, a, b, cin);
endmodule // full_adder

module adder_8b(sum, cout, a, b, cin);
   output [7:0] sum;
   output       cout;
   input [7:0]  a, b;
   input        cin;
   wire [7:1]   carry;

   full_adder adder0(sum[0], carry[1], a[0], b[0],      cin);
   full_adder adder1(sum[1], carry[2], a[1], b[1], carry[1]);
   full_adder adder2(sum[2], carry[3], a[2], b[2], carry[2]);
   full_adder adder3(sum[3], carry[4], a[3], b[3], carry[3]);
   full_adder adder4(sum[4], carry[5], a[4], b[4], carry[4]);
   full_adder adder5(sum[5], carry[6], a[5], b[5], carry[5]);
   full_adder adder6(sum[6], carry[7], a[6], b[6], carry[6]);
   full_adder adder7(sum[7],     cout, a[7], b[7], carry[7]);

endmodule // adder_8b

module adder_32b(sum, cout, a, b, cin);
   output [31:0] sum;
   output        cout;
   input [31:0]  a, b;
   input         cin;
   wire [3:1]    carry;

   adder_8b adder0(sum[07:00], carry[1], a[07:00], b[07:00],      cin);
   adder_8b adder1(sum[15:08], carry[2], a[15:08], b[15:08], carry[1]);
   adder_8b adder2(sum[23:16], carry[3], a[23:16], b[23:16], carry[2]);
   adder_8b adder3(sum[31:24],     cout, a[31:24], b[31:24], carry[3]);
endmodule // adder_32b

--:--  ripple.v         (Verilog)--L72--27%----------------------------------------
```

After you have tested and debugged your verilog, you may import it into a Cadence schematic. Open Cadence with

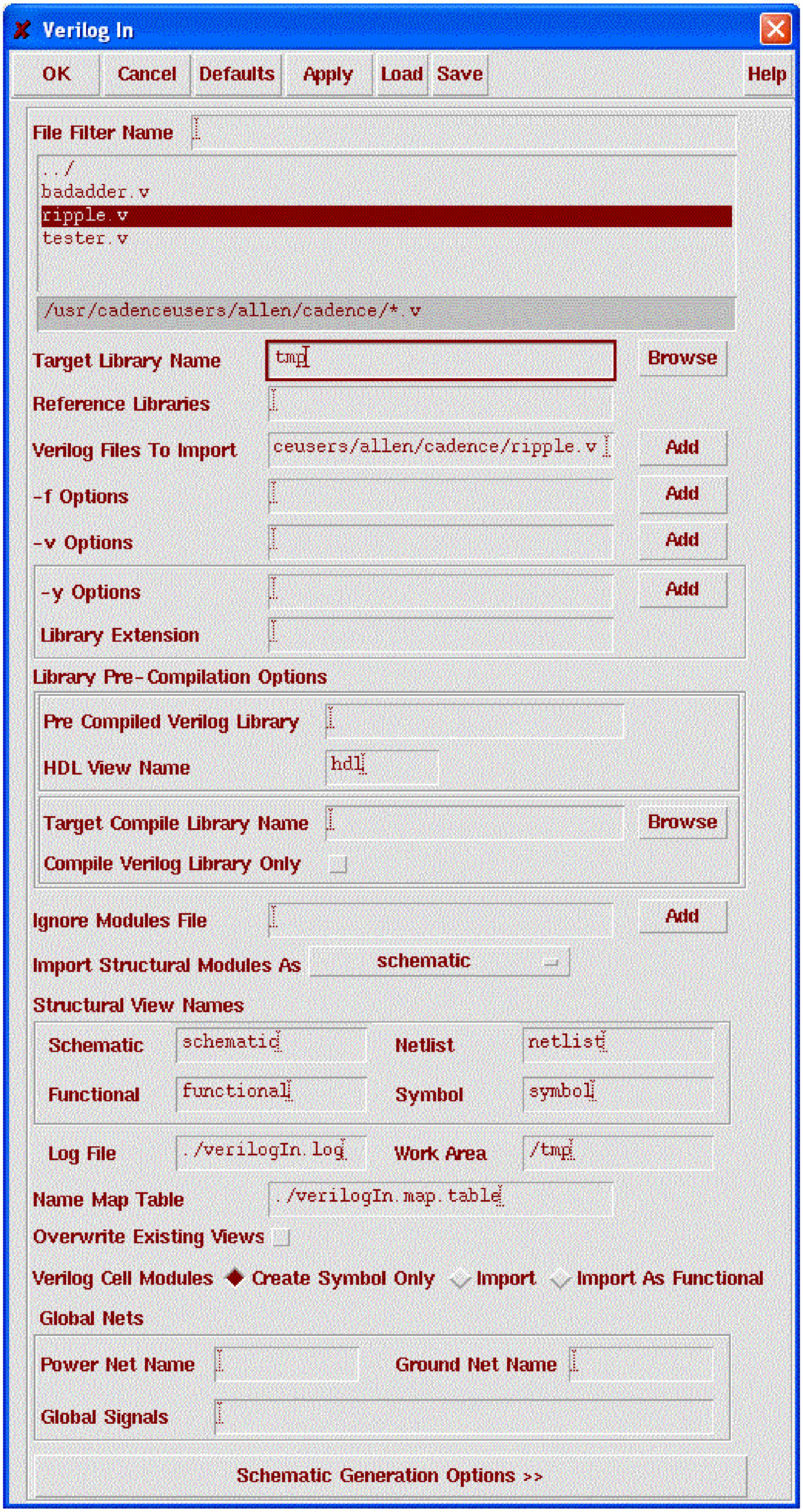
```
> layoutPlus&
```

In tha main window, choose File -> Import -> Verilog



```
ATION
ert string "-adobe-helvetica-medium-r-normal--0-100-75-75-p-0-iso8859-1" to type
```

This will open the Verilog In window.



File Filter Name

../
badadder.v
ripple.v
tester.v

/usr/cadenceusers/allen/cadence/*.v

Target Library Name: tmp

Reference Libraries:

Verilog Files To Import: ceusers/allen/cadence/ripple.v

-f Options:

-v Options:

-y Options:

Library Extension:

Library Pre-Compilation Options

Pre Compiled Verilog Library:

HDL View Name: hdl

Target Compile Library Name:

Compile Verilog Library Only □

Ignore Modules File:

Import Structural Modules As: schematic

Structural View Names

Schematic: schematic
Netlist: netlist
Functional: functional
Symbol: symbol

Log File: ./verilogIn.log
Work Area: /tmp

Name Map Table: ./verilogIn.map.table

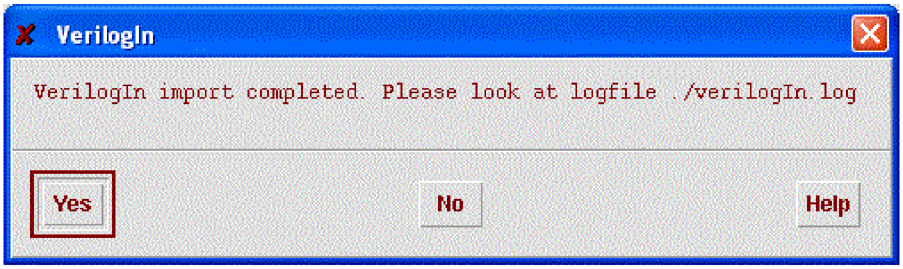
Overwrite Existing Views □

Verilog Cell Modules ◆ Create Symbol Only ◇ Import ◇ Import As Functional

Global Nets

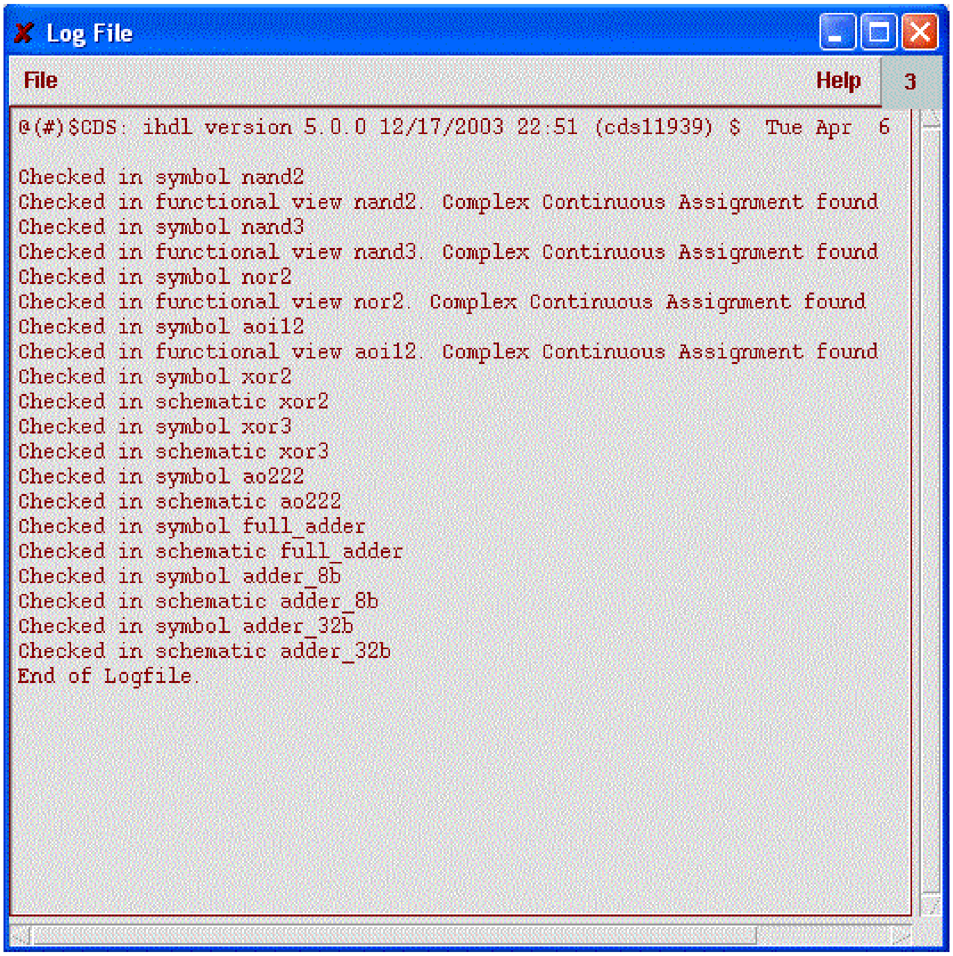
Power Net Name:
Ground Net Name:
Global Signals:

Input the correct Target Libray Name. Clear the Refrence Library Field. Browse the directory tree to find your verilog source file. You can put *.v into the File Filter Name to only show verilog source files. Select the desired source file (ripple.v in this example) and press the Add button for Verilog files to import. You may want to select the Overwrite Existing Views button if you want to change any of the schematics. Select Apply.



VerilogIn import completed. Please look at logfile ./verilogIn.log

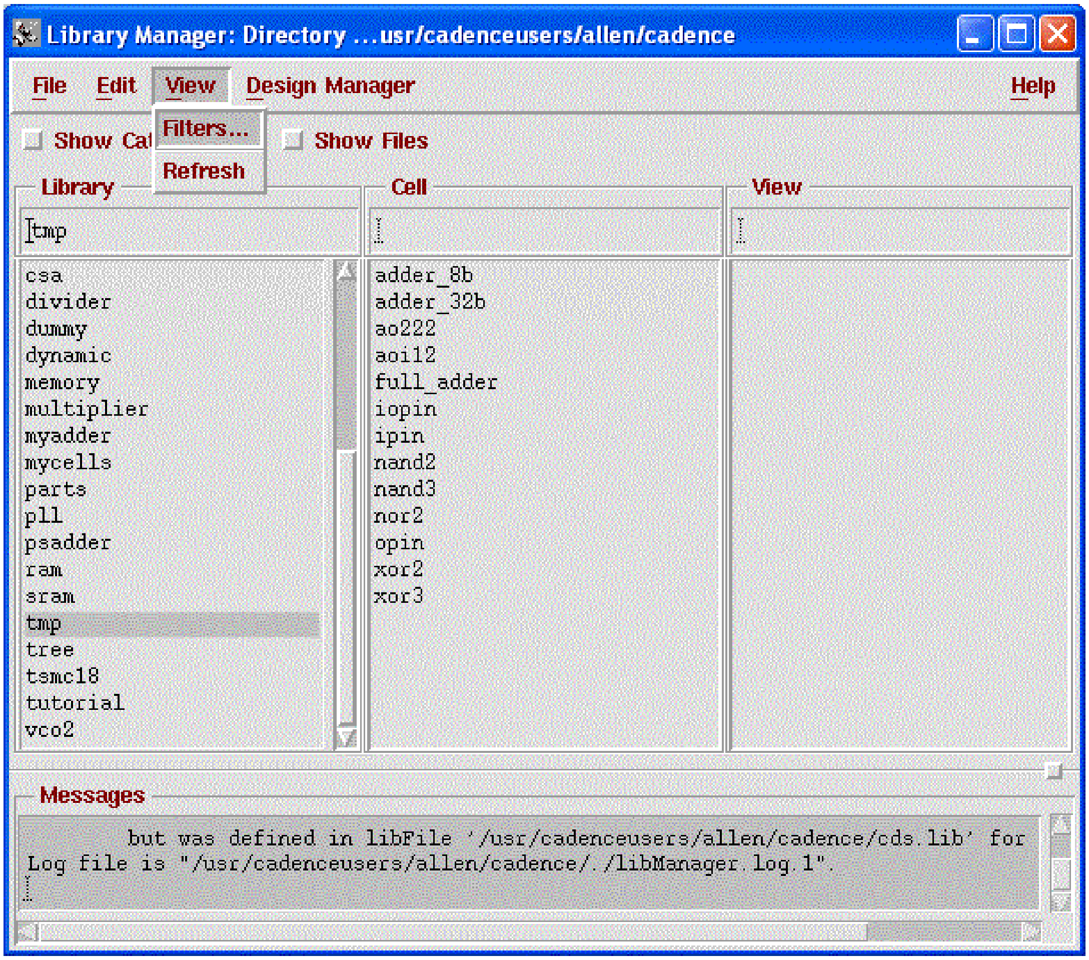
You will see a popup window asking you if you want to see the log file. Select Yes. If you did not see this popup, check the main window for errors.



```
@(#)$CDS: ihdl version 5.0.0 12/17/2003 22:51 (cds11939) $  Tue Apr  6

Checked in symbol nand2
Checked in functional view nand2. Complex Continuous Assignment found
Checked in symbol nand3
Checked in functional view nand3. Complex Continuous Assignment found
Checked in symbol nor2
Checked in functional view nor2. Complex Continuous Assignment found
Checked in symbol aoi12
Checked in functional view aoi12. Complex Continuous Assignment found
Checked in symbol xor2
Checked in schematic xor2
Checked in symbol xor3
Checked in schematic xor3
Checked in symbol ao222
Checked in schematic ao222
Checked in symbol full_adder
Checked in schematic full_adder
Checked in symbol adder_8b
Checked in schematic adder_8b
Checked in symbol adder_32b
Checked in schematic adder_32b
End of Logfile.
```

In the Log File, you can see that that the symbol views were created for all of the modules, Schematic vies were created for the structural modules, and functional views were created for the behavioral views (the functional view just contains the text source.) If you have already created schematic views for the behavioral modules and the pin names do not match, you will have to changes the net names.

To see the newly improrted cells, you may have to click refresh in the Library Manager.



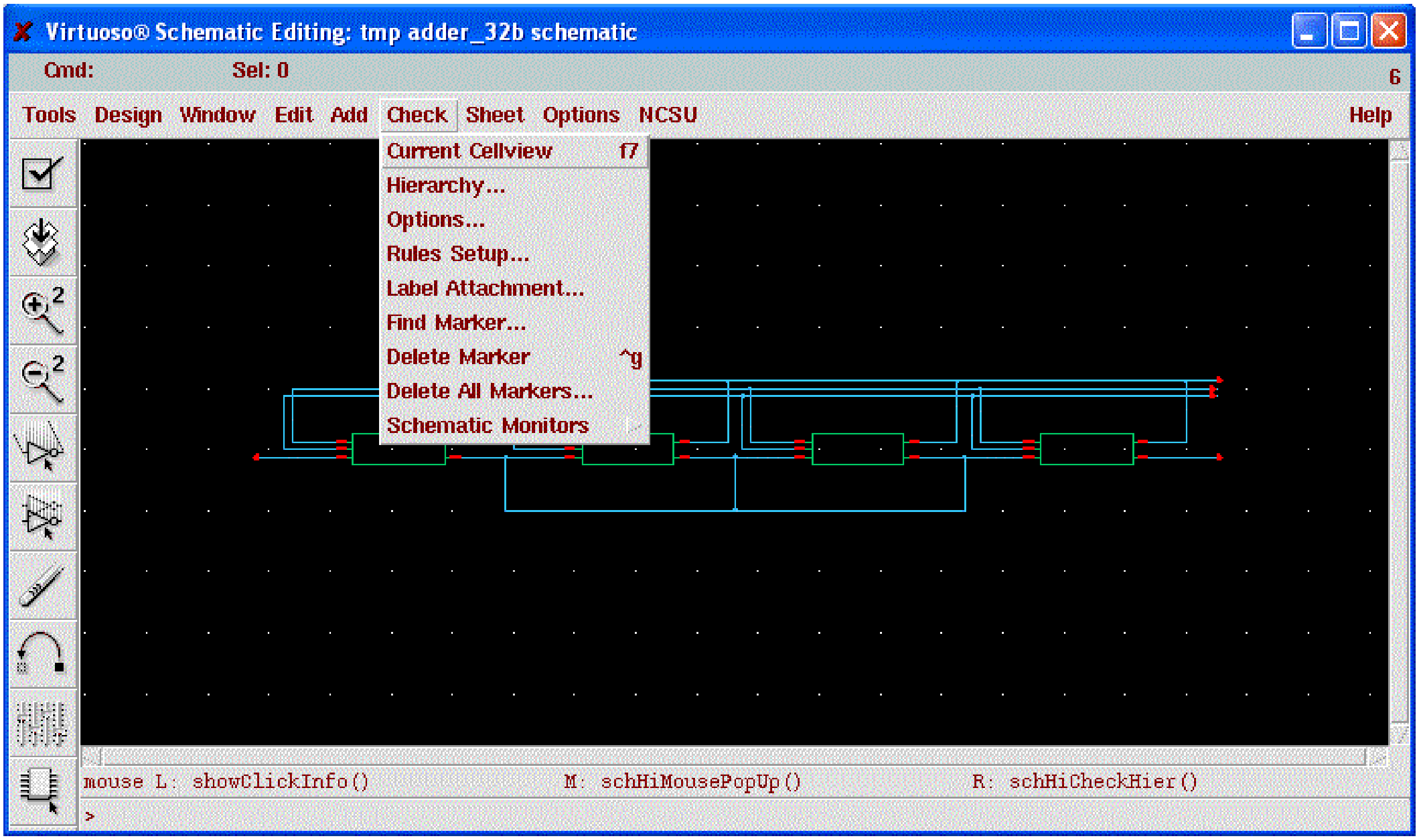
Library: tmp

```
csa
divider
dummy
dynamic
memory
multiplier
myadder
mycells
parts
pll
psadder
ram
sram
tmp
tree
tsmc18
tutorial
vco2
```

```
adder_8b
adder_32b
ao222
aoi12
full_adder
iopin
ipin
nand2
nand3
nor2
opin
xor2
xor3
```

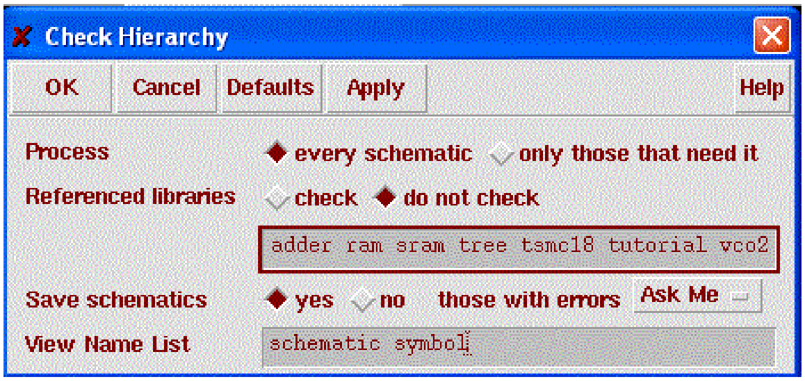
```
        but was defined in libFile '/usr/cadenceusers/allen/cadence/cds.lib' for
Log file is "/usr/cadenceusers/allen/cadence/./libManager.log.1".
```

Open the top level schematic to check the design.



```
mouse L: showClickInfo()          M: schHiMousePopUp()          R: schHiCheckHier()
>
```

Choose Check -> Hierarchy.



Process ◆ every schematic ◇ only those that need it

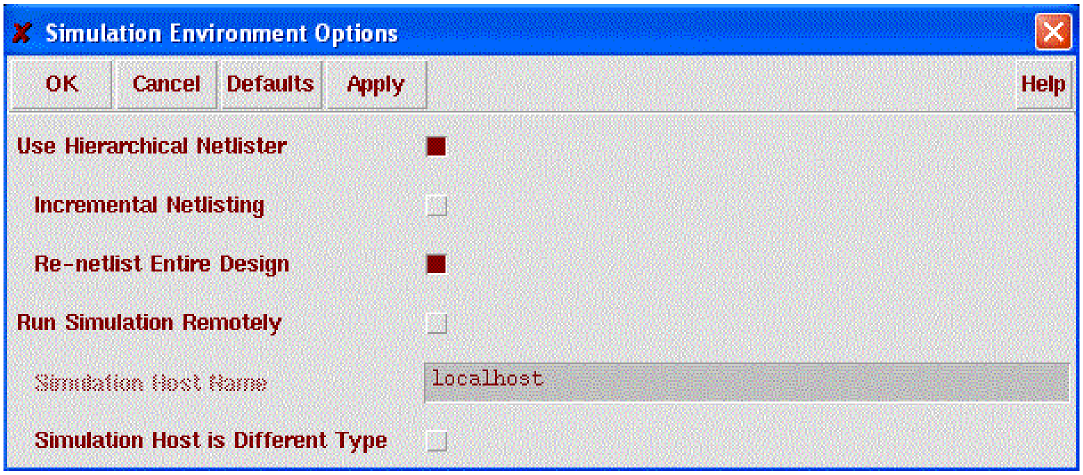
Referenced libraries ◇ check ◆ do not check

adder ram sram tree tsmc18 tutorial vco2

Save schematics ◆ yes ◇ no those with errors Ask Me

View Name List schematic symbol

Check every schematic and press OK. Look in the main window for errors. Netlist the design as usual except that it is a good idea to select Re-netlist Entire Design in the simulation options.



Use Hierarchical Netlister ■

Incremental Netlisting □

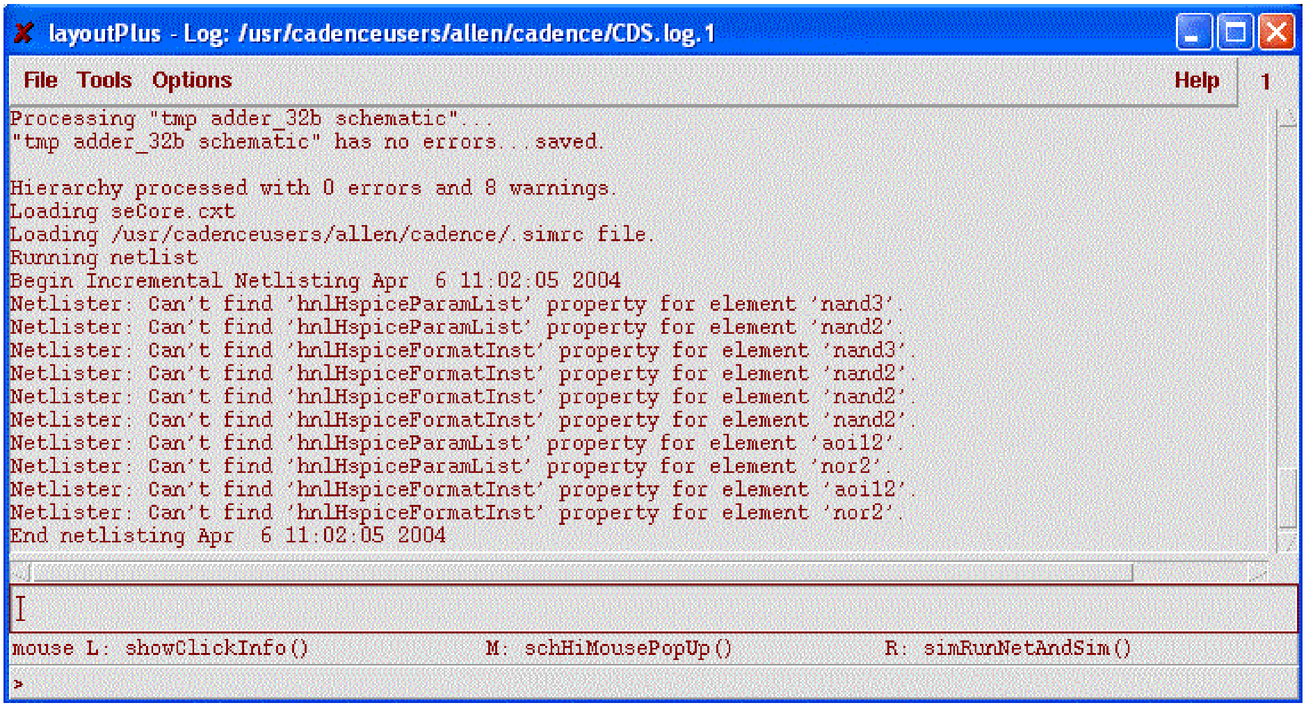
Re-netlist Entire Design ■

Run Simulation Remotely □

Simulation Host Name localhost

Simulation Host is Different Type □

When you netlist the design, check Re-netlist Entire Design just to be sure that you are using the current versions of all of your cells. In the example design, the modules nand2, nand3, aoi12, nor2 must all be implemented at the transistor level and will not netlist until these schematics are defined manually. If you have not yet defined the low-level cells, you will get the folling error while netlisting.



```
Processing "tmp adder_32b schematic"...
"tmp adder_32b schematic" has no errors...saved.

Hierarchy processed with 0 errors and 8 warnings.
Loading seCore.cxt
Loading /usr/cadenceusers/allen/cadence/.simrc file.
Running netlist
Begin Incremental Netlisting Apr  6 11:02:05 2004
Netlister: Can't find 'hnlHspiceParamList' property for element 'nand3'.
Netlister: Can't find 'hnlHspiceParamList' property for element 'nand2'.
Netlister: Can't find 'hnlHspiceFormatInst' property for element 'nand3'.
Netlister: Can't find 'hnlHspiceFormatInst' property for element 'nand2'.
Netlister: Can't find 'hnlHspiceFormatInst' property for element 'nand2'.
Netlister: Can't find 'hnlHspiceFormatInst' property for element 'nand2'.
Netlister: Can't find 'hnlHspiceParamList' property for element 'aoi12'.
Netlister: Can't find 'hnlHspiceParamList' property for element 'nor2'.
Netlister: Can't find 'hnlHspiceFormatInst' property for element 'aoi12'.
Netlister: Can't find 'hnlHspiceFormatInst' property for element 'nor2'.
End netlisting Apr  6 11:02:05 2004
```

```
mouse L: showClickInfo()              M: schHiMousePopUp()            R: simRunNetAndSim()
>
```

If you ever have to modify the verilog code and re-import the design, you should delete all of the cells that you want to update using the library manager. Then always check the entire hierarchy for errors.

Tips

Always remember that verilog is case sensitive but spice is not.

Dont use reserved keywords in verilog to name modules or wires -- and, xor, ...

Vectors in verilog -- a[31:0] -- will turn into busses in schematic -- a<31:0> -- and will netlist into indivitual nets in spice -- a31, a31, ... a0. If your verilog source has a net called a0, you will have problems.

Debug your verilog before trying to import the design to schematic.

If your verilog modules contains any behavioral code -- includeing assign statements -- it will not produce a schematic.

Sometimes the schematic will produduce routing errors if you module is very complex so keep modules small and simple.