Importing Structural VHDL Descriptions into Composer Schematics.

This tutorial will take you through the steps required to import structural VHDL descriptions into the cadence design framework. The schematics produced using this will be used to run Layout versus Schematic (LVS) tool to verify that your layouts match the VHDL descriptions. The main tool used for this is called vhdlin and it can be accessed through the CIW.

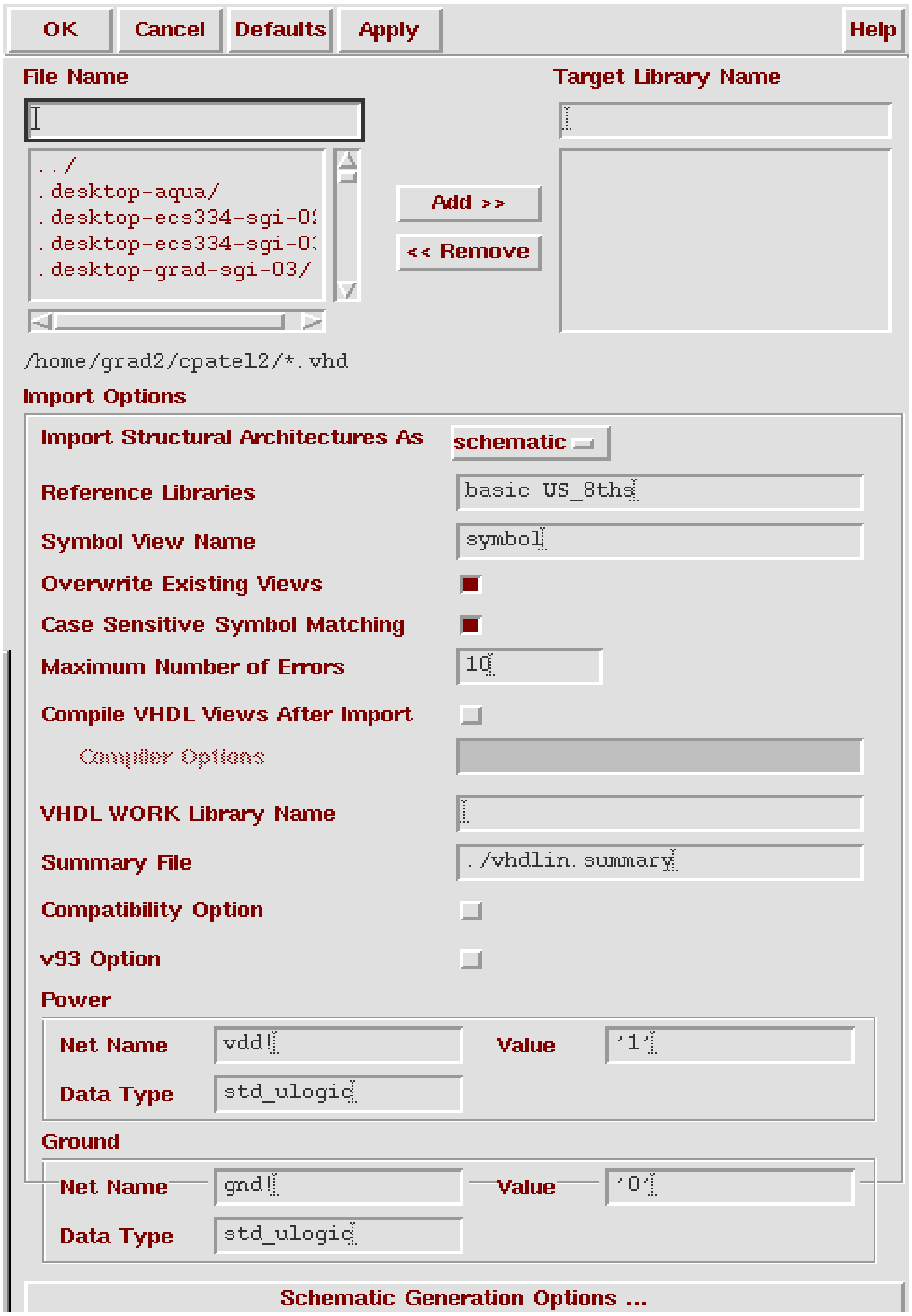
There are few points to be noted before going to the exact procedure. Also you will have to create the schematics of all the standard cells that you are using in you design manually i.e. any cells that have a behavioral statement. When the instances of these standard cells are called in a VHDL file the vhdlin tool will synthesis the schematic for you. The example used in this tutorial is that of an inverter where we will create a standard cell called inv and then call an instance of the inv in the inv1 entity.

First of all let us import a primitive cell called inv into the vhdl library. The example inv VHDL file is given here.

Example INV VHDL file:

```
--
-- Entity: inv
-- Architecture: structural
-- Author: cpatel2
-- Created On: 10/20/00 at 13:32
--
library IEEE;
use IEEE.std_logic_1164.all;

entity inv is

  port (
    A    : in  std_logic;
    Y   : out std_logic);

end inv;

architecture structural of inv is
begin

  Y <= not A;

end structural;
```

Open icfb from the same location as for previous tutorials. Go to file and choose import and then VHDL. The import vhdl form will show up. The figure shows the import vhdl form. Under the file name browser in the vhdlin form go and select the inv VHDL file and hit the Add button on the right hand side of the browser window. The vhdl file should show up in the list under the Target Library Name field. You can add multiple vhdl files if you want. In the Target Library Name field enter the library to import to as your design library (you might want to make a new library with ami 0.6 technology file for your project). Select schematic in the Import Structural Architecture as selection menu. In the reference library field you should have the following libraries (space separated names): basic, US_8ths ieee std. Remove any extra ones you have. Set the symbol view name to symbol. Turn off the overwrite option. Note that if you want to import an already imported design than you will have to turn this option on. Set the vhdl work library option to vhdl. You will not have to change any other fields in the form. Hit the OK button after rechecking all the fields in the form. You will get a message saying Started vhdlin in background in your CIW window. On completing the import process the system will come back with a successful or error run message (doesn't pop up on RHEL 5, check the vhdlin.summary file as well as vhdlin.log and vhdlin.err in your run directory). You can do View-> Refresh in the Library Manager to verify that the cells have been imported. View the log file to find errors. The vhdlin may not work successfully if you have any errors in your vhdl code. Thus to be absolutely sure compile the vhdl code before starting the import process using the commands shown in the VHDL tutorial.



File Name

../
.desktop-aqua/
.desktop-ecs334-sgi-0
.desktop-ecs334-sgi-0
.desktop-grad-sgi-03/

Target Library Name

/home/grad2/cpatel2/*.vhd

Import Options

Import Structural Architectures As	schematic
Reference Libraries	basic US_8ths
Symbol View Name	symbol
Overwrite Existing Views	■
Case Sensitive Symbol Matching	■
Maximum Number of Errors	10
Compile VHDL Views After Import	□
Compiler Options	
VHDL WORK Library Name	
Summary File	./vhdlin.summary
Compatibility Option	□
v93 Option	□

Power

Net Name	vdd!	Value	'1'
Data Type	std_ulogic		

Ground

Net Name	gnd!	Value	'0'
Data Type	std_ulogic		

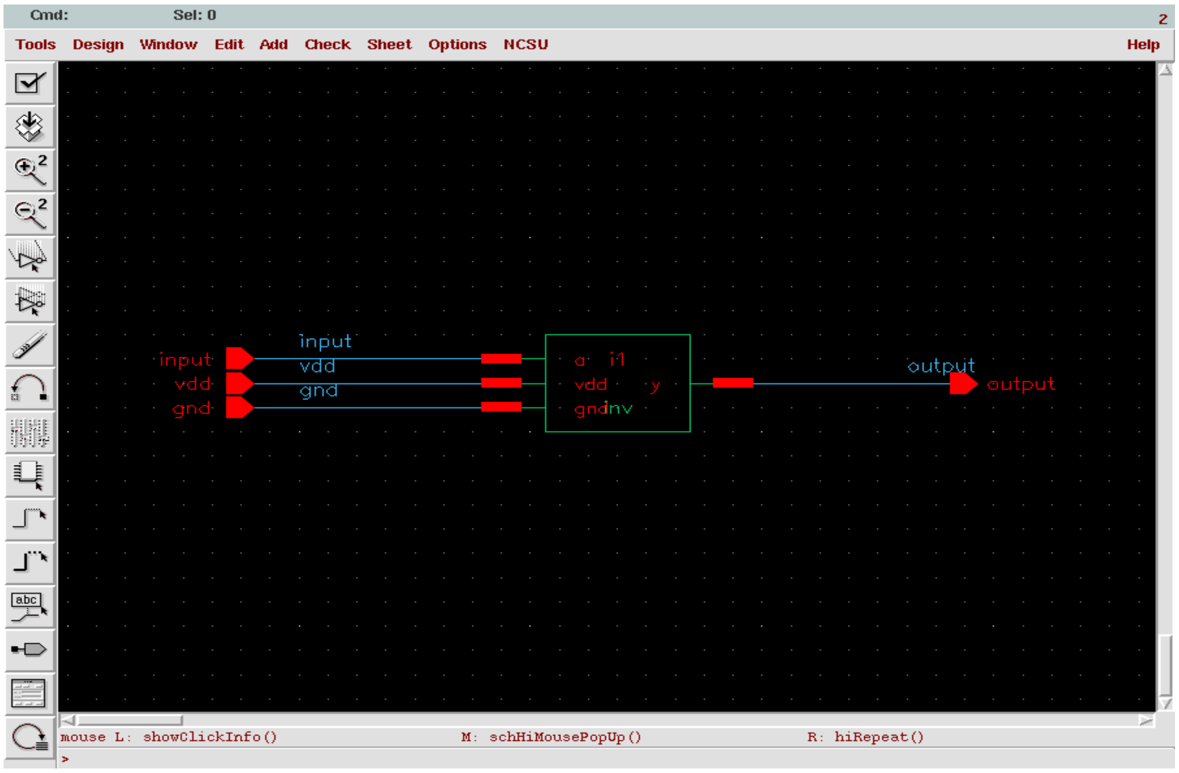
On successful completion of the import process the library will have a cell called inv with entity, structural and symbol views. Check the symbol view to see that all the inputs and output ports are present. Now the next step is to manually generate a schematic for the inv primitive cell. Create a new cellview called schematic for the inv and make a CMOS schematic for the inverter using composer schematic tool. Ensure that you can do a check and save on your design with no warnings. All the pins that you place in your schematic have to match with the ports in you symbol view. Thus now you have a primitive cell called inv that you can use anywhere you want to place an inverter in your design. In a similar fashion create the same cell views for all the primitive cells that you will require in your design.

The next step is to use the inv primitive cell and place an instance of it in the inv1 entity and then verify the inv1 layout with the schematic. The example inv1 vhdl file is shown below.

Example inv1 VHDL file:

```
--
-- Entity: inverter
-- Architecture : structural
-- Author: cpatel2
-- Created On: 10/20/00 at 13:32
--
library IEEE;
use IEEE.std_logic_1164.all;

entity inv1 is

  port (
    Input    : in  std_logic;
    Output   : out std_logic);

end inv1;

architecture structural of inv1 is

component inv
    port ( A   : in  std_logic;
           Y   : out std_logic);
end component inv;

for I1: inv use entity work.inv(structural);
begin

I1: inv port map (Input,Output);
end structural;
```

Import the inv1 module into a schematic using the same procedure as explained for the inv. Click on the structural view and you should have a schematic as shown in the figure below.



mouse L: showClickInfo() M: schHiMousePopUp() R: hiRepeat()

The inv1 schematic will have an instance of the inv primitive cell and will have the inputs and outputs as defined in the inv1 file. Click on the inv instance and choose design hierarchy descend read to look at the schematic of the inv standard cell. Click design hierarchy return to return to the inv1 schematic. Now create a layout for the inv1 cell and get the extracted view. Run LVS between the extracted view and the structural view and verify that the netlists match.