VHDL Tutorial

This tutorial will cover the steps involved in compiling, elaborating and simulating VHDL design files. The NC-simulator and the ncvhdl compiler under the cadence distribution will be used for this purpose. This tutorial will cover only the command line option of running all these tools. A later tutorial will cover the GUI option for the same.

The ncvhdl and nc-simulator binaries live under a different directory structure than the normal directory structure used by the icfb and other related tools in the cadence distribution. So you will require a separate .cshrc file to run these tools. The file is available at the class webpage and is called .cshrc.sim. Copy the file in your home directory and source it whenever you want to run VHDL simulations. The command for this is *source .cshrc.sim*. Source the normal .cshrc file using the command *source .cshrc* when you want to run any other tools except the nc-simulators and compilers. You will have to source one of the two files whenever you switch form icfb to ncvhdl and vice versa.

Next step is to copy some options files and setup a directory structure. It is important that you follow the steps involved in setting up the directory structure so that your designs are compatible with the GUI options. Change your directory to cadence using *cd cadence*. Then make a directory called vhdl in the cadence directory using *mkdir vhdl*. Change your working directory to vhdl using *cd vhdl*. Now you will have to copy two files into this directory. One is the cds.lib file that defines the path to some standard vhdl libraries and the hdl.var file that list the options required by the simulator for proper operation. They are available on the course webpage and a copy is provided below. Copy these files into your vhdl directory and make sure that they have the following lines.

cds.lib file :-

```
- -
- - This cds.lib file is made specifically to be used with the ldv toolset
- -
DEFINE ieee /cds/ldv/tools.sun4v/inca/files/IEEE
DEFINE std /cds/ldv/tools.sun4v/inca/files/STD
DEFINE vhdl /home/grad2/cpatel2/cadence/vhdl
```

Change the path in the last line to point to your vhdl directory. So you will have something like *your_home_directory/cadence/vhdl* in the last line.

hdl.var file :-

```
DEFINE WORK vhdl
```

After you have copied these files in your vhdl directory you are ready to enter vhdl code. This tutorial will cover the design and simulation of an inverter in VHDL. The directory structure that is explained in this tutorial should be implemented for each design. The only difference will be the names of the respective directories depending on your design names.

We will start by making a directory called inverter in the vhdl directory. Then create a directory called vhdl under the inverter directory. The commands required for the above steps are shown here.

```
mkdir inverter
cd inverter
mkdir vhdl
cd vhdl
```

Now make a file called vhdl.vhd under the vhdl directory and type in your VHDL code in this file. The code for the inverter is shown below and is also available on the course webpage. The emacs editor under sunserver1 has a special vhdl options menu when you edit any .vhd file. This will be a great help when learning VHDL and most of the default VHDL statements are present in the menu. Note that when you write you VHDL code for the inverter the entity is called inverter which is the same name that you used for the directory. Please name the top level entity in your design the same as the directory that it lives in. Please look at the comments in the code below for information about the specifics.

vhdl.vhd file for the inverter.

```
- -
- - Entity: inverter
- - Architecture : structural
- - Author: cpatel2
- - Created On: 10/20/00 at 13:32
- -
-- These are the standard libraries that you are defining.
library IEEE;
use IEEE.std_logic_1164.all;

-- The entity name should be same as the directory name that you are designing in.
-- Ports are the inputs and outputs from your circuit.
entity inverter is

  port (
    input    : in  std_logic;
    output   : out std_logic);
end inverter;

-- Define an architecture for the inverter entity which is either functional or behavioral.
architecture structural of inverter is

begin

-- Concurrent assignment statement.
  output <= not (input);

end structural;
```

Now before we go on to compiling and simulating we need to make a test bench to test the code that we just entered. A test bench is another VHDL file that uses your design file and gives it inputs and checks the outputs. The input and output is done using text files through the VHDL code using inbuilt I/O functions. For making a test bench make are directory called inverter_test in your ~/cadence/vhdl directory. Then make a vhdl directory under the inverter_test directory the same way as you did for the inverter. Inside the inverter_test/vhdl directory make a file called vhdl.vhd and enter the code for you test bench in it. The test bench file is available on the course webpage and is also shown below. Again the entity should have the same name as the directory you are designing in. The color formatting is not shown here as in the previous case but will be present if you use the emacs editor.

vhdl.vhd file for the test bench.

```
- -
- - Entity: inverter_test
- - Architecture : vhdl
- - Author: cpatel2
- - Created On: 10/20/00 at 01:55
- -

- - Define the libraries that you are going to use. The two textio libraries define your input and
- - output functions

library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_textio.all;
use STD.textio.all;

- - The entity name here is also the name as the directory that you are designing in.
- - The circuit has no ports because this is a test bench file that you are using to test the inverter.
entity inverter_test is

end inverter_test;

- - Define an architecture for the inverter_test entity.
architecture test of inverter_test is

- - Define the inverter as a component that you will use in this file. The port list is the same as
- - the original port list of the inverter entity.
component inverter
   port (
   input       : in  std_logic;
   output      : out std_logic);
end component;
```

```
- - Define an instance of the inverter that you are going to use in this file.
for i1 : inverter use entity work.inverter(structural);
   signal ip,op : std_logic;
   signal clock : std_logic;

begin

- - Assign inputs and outputs to the various inverter ports.
i1 : inverter port map (ip,op);

- - Control the inputs to the circuit.

clk : process
   begin  -- process clk

    clock<='0','1' after 5 ns;
    wait for 10 ns;

  end process clk;

- - Read inputs from a input file called inverter_in.txt and write outputs to inverter_out.txt
io_process: process

  file infile  : text is in "inverter_in.txt";
  file outfile : text is out "inverter_out.txt";
  variable ip1,op1 : std_logic;
  variable buf : line;

begin

  while not (endfile(infile)) loop


    readline(infile,buf);
    read (buf,ip1);
    ip<=ip1;

    wait until falling_edge(clock);

    op1:=op;

    write(buf,op1);
    writeline(outfile,buf);

  end loop;
  wait;
```

end process io_process;

end test;

After creating the circuit file for the inverter and the test bench to test the same we are now ready to compile and simulate the circuit. But first we need to create the inverter_in.txt file that specifies the input sequences that will be assigned to the inputs to the circuit. This file is created in the ~/cadence/vhdl directory. The file should have one sequence per line. In our case the sequences will only be a single bit sequence because we have a single bit input. The inverter_in.txt file is shown below.

inverter_in.txt file

```
0
0
0
1
1
0
1
```

The sequence are assigned arbitrarily. The inverter_out.txt file will be generated automatically in the ~/cadence/vhdl directory when you simulate the circuit. As the circuit we have implemented is a simple inverter the inverter_out.txt file would look like the one shown below.

inverter_out.txt file

```
1
1
1
0
0
1
0
```

There are three steps involved in running a simulation. First is compiling the VHDL code, then elaborating the design and finally simulating. Thus we first need to compile the inverter entity as well as the test bench entity. Next we will elaborate both these entities. Now we will simulate the test bench. It is not required to simulate the inverter as it is because it doesn't have any inputs applied to the circuit.

Compiling the ~/cadence/vhdl/inverter/vhdl/vhdl.vhd file.

The command for the compilation step is shown below and should be run from the ~/cadence/vhdl directory.

```
ncvhdl -cdslib ~/cadence/vhdl/cds.lib -hdlvar ~/cadence/vhdl/hdl.var -messages inverter/vhdl/vhdl.vhd.
```

If there are no errors in your code then the output will be similar to the one shown below.

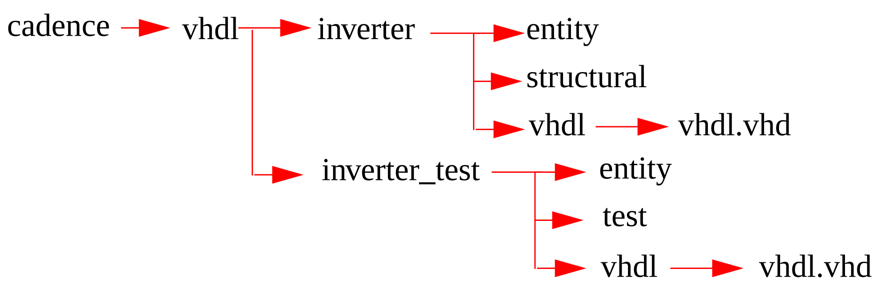
```
ncvhdl: v2.2.(s16): (c) Copyright 1995 - 2000 Cadence Design Systems, Inc.
inverter/vhdl/vhdl.vhd:
        errors: 0, warnings: 0
VHDL.INVERTER (entity):
        streams: 3, words: 9
VHDL.INVERTER:STRUCTURAL (architecture):
        STD_LOGIC_1164 function "NOT", line 21, is built-in
        1 function(s) built-in
        streams: 2, words: 92
```

Compiling the ~/cadence/vhdl/inverter_test/vhdl/vhdl.vhd file.

```
ncvhdl -cdslib ~/cadence/vhdl/cds.lib -hdlvar ~/cadence/vhdl/hdl.var -messages inverter_test/
vhdl/vhdl.vhd.
```

You should get a output similar to the one shown below with different values after a successful compilation.

After compiling the two designs you will have the following directory structure.



Elaborating vhdl.inverter:structural.

```
ncelab -cdslib ~/cadence/vhdl/cds.lib -hdlvar ~/cadence/vhdl/hdl.var -message
vhdl.inverter:structural
```

The output will look like the one shown below on successful completion.

```
ncelab: v2.2.(s16): (c) Copyright 1995 - 2000 Cadence Design Systems, Inc.
        Elaborating the design hierarchy:
        Building instance specific data structures.
        Design hierarchy summary:
                    Instances  Unique
             Components:  2       1
             Processes:   1       1
        Writing initial simulation snapshot: VHDL.INVERTER:STRUCTURAL
```

Elaborating vhdl.inverter_test:test.

```
ncelab -cdslib ~/cadence/vhdl/cds.lib -hdlvar ~/cadence/vhdl/hdl.var -messages
vhdl.inverter_test:test
```

Simulating vhdl.inverter_test:test.

```
ncsim -cdslib ~/cadence/vhdl/cds.lib -hdlvar ~/cadence/vhdl/hdl.var -messages
vhdl.inverter_test:test
```

If the simulation finished successfully there will be an inverter_out.txt file in your ~/cadence/vhdl directory. This file should compare to the one that was shown above. Verify that the ouputs from the circuit are the same as expected for each input sequence.