VHDL Tutorial

Cadence Setup

Instructions for setting up Cadence tools is provided on the webpage.

VHDL Setup

After following the above setup steps verify that you have a ***cds.lib*** and ***hdl.var*** file. Also make sure you have a directory called ***vhdl***

VHDL Example files

This tutorial will cover the design and simulation of an inverter in VHDL. The code for the inverter is shown below. The emacs editor under most machines has a special vhdl options menu when you edit any .vhd file. This will be a great help when learning VHDL as most of the default VHDL statements are present in the menu.

inverter.vhd file for the inverter.

```
-- These are the standard libraries that you are defining.
library IEEE;
use IEEE.std_logic_1164.all;

-- Keep the entity name same as the file name as it will be easier during compilation.
-- 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. The test bench file is also shown below.

inverter_test.vhd file for the test bench.

```
- - 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 file name.
- - 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 ***same directory as your code*** 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
0
1
```

The ***inverter_out.txt*** file will be generated automatically in the ***same directory as your code*** 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
1

The last file required is a run file that provides commands to the simulator. For this case the run file is very simple and tells the simulator to run for 90 ns and then exit.

ncsim.run

run 90 ns
exit

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 testbench entity. Next we will elaborate the testbench. Now we will simulate the testbench. It is not required to simulate the inverter because it doesn't have any inputs applied to the circuit.

Running VHDL Simulations

Compiling the inverter.vhd file.

The command for the compilation step is shown below and should be run from the directory where your setup files are located.

run_ncvhdl.bash -messages -linedebug -cdslib <path to cds.lib> -hdlvar <path to hdl.var> -smartorder inverter.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 - 2004 Cadence Design Systems, Inc.
inverter/vhdl/vhdl.vhd:
errors: 0, warnings: 0
VHDL.INVERTER (entity):
streams: 3, words: 9
VHDL.INVERTER:STRUCTURAL (architecture):
streams: 1, words: 99

Compiling the inverter_test.vhd file.

run_ncvhdl.bash -messages -linedebug -cdslib <path to cds.lib> -hdlvar <path to hdl.var> -smartorder inverter_test.vhd

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

Elaborating vhdl.inverter_test.

run_ncelab.bash -messages -access rwc -cdslib <path to cds.lib> -hdlvar <path to hdl.var> inverter_test

Simulating vhdl.inverter_test.

run_ncsim.bash -input ncsim.run –messages -cdslib <path to cds.lib> -hdlvar <path to hdl.var> inverter_test

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

If you run file is not properly written as compared to the number of inputs lines in the input file the simulation might hang. The simulation results should still be fine. You can exit the simulation by hitting control-C multiple times. Fix the errors in your run file to avoid this problem again.

Running VHDL using GUI mode (SimVision)

Simvision is a very powerful gui that can be used for debugging VDHL as well as verilog designs. The tools allows you to view waveforms for signals at any level of hierarchy, let you traverse through your design hierarchy, display schematic diagrams of your code, force signal values, set breakpoints, step through your code while simulating etc. The following ncsim command can be used to access all the features:

run_ncsim.bash -gui -cdslib <path to cds.lib> -hdlvar <path to hdl.var> inverter_test