Amey Kulkarni
4th Nov.2012

NCVerilog Tutorial

To setup your cadence tools use your linuxserver.csumbc.edu account. We can connect to dedicated campus server.

Edit the file called .cshrc in your home directory. (This is basically for new students, those who used the cadence tools before can skip this)

I. `% Vi .cshrc` (this will open .cshrc file)

II. `Source/afs/umbc.edu/software/cadence/etc/setup_2008/cshrc.cadence`

III. Add above line by pressing "i" and then type out above line. Once typed it, you can press ESC : wq (write and save)

IV. Type "source cshrc.cadence" on terminal window.

The main objective is to understand the tool usage and its behavior. Here, I have taken very simple example of D Flip-flop.

File names and its descriptions are as follows:

`dff.v` = D flip flop top level module. (/data/eehpc0/amey/dff/dff.v)
`dff_tb.v` = Testbench for D flip flop. (/data/eehpc0/amey/dff/dff_tb.v)

To run NC-Verilog simulator two set-up files are required:

- A cds.lib file : This file contains statements that define your libraries and that map logical library names to physical directory paths.

  You can create your own txt file (Out of the scope of Tutorial). For time being, I am including the path for cds.lib

- An hdl.var file : This file defines which library is the work library. The hdl.var file also can contain definitions of other variables that determine how your design environment is configured, control the operation of NC-Verilog tools, and specify the locations of support files and invocation scripts.

To add these files to your cadence directory type following commands

```
cp  /afs/umbc.edu/software/cadence/etc/setup_2008/cds.lib   ~/cadence/
cp  /afs/umbc.edu/software/cadence/etc/setup_2008/hdl.var   ~/cadence/
```

Note: You can have more than one cds.lib or hdl.var

Running Verilog Simulations

There are two ways to run NC-Verilog simulator

- Single-step invocation: In this way of running the simulator, you issue one command, the ncverilog command. This command invokes a parser called ncvlog and an elaborator called ncelab to build the model, and then invokes the ncsim simulator to simulate the model.

- Multi-step invocation: In this way of running the simulator, you invoke ncvlog, ncelab, and ncsim separately

If you want to simulate directly, you can skip following theory part. But, it is always good to know this.

- **ncvlog** analyzes and compiles your Verilog source. This tool performs syntactic checking on the HDL design unit(s) (modules, macromodules, or UDPs) in the input source file(s) and generates an intermediate representation for each HDL design unit.

- **ncelab** elaborates the design hierarchy that defines the model. The elaborator takes as input the Library.Cell:View name of the top-level HDL design unit(s). It then constructs a design hierarchy based on the instantiation and configuration information in the design, establishes connectivity, and computes the initial values for all of the objects in the design.

- **ncsim** simulates Verilog using the native instruction streams to execute the dynamic behavior of the design.

Since, I will be running my example code by using Single step invocation with ncverilog.

You can always find more information by invoking cadence document window. Type `cdsdoc` command on terminal window

For getting help on command to run tools
`% tool-name –help (ex. Ncverilog –help)`

Compiling and Simulation cadence files

```
%ncverilog  -c dff.v
```

If your run is successful then the window will show you the outputs.

```
linux1[105]% ncverilog -c dff.v
ncverilog: 06.11-s008: (c) Copyright 1995-2007 Cadence Design Systems, Inc.
file: dff.v
                Caching library 'worklib' ....... Done
        Elaborating the design hierarchy:
        Building instance overlay tables: ..................... Done
        Loading native compiled code:     ..................... Done
        Building instance specific data structures.
        Design hierarchy summary:
                        Instances  Unique
                Modules:        1       1
                Registers:      1       1
                Always blocks:  1       1
        Writing initial simulation snapshot: worklib.dff:v
```

Now, here I have taken simple example. In case of huge designs which has top level and sub-modules.

Ex. `%ncverilog top.v sub1.b sub2.v`

Once you compiled your top level module. You can start compiling test bench module.

```
%ncverilog –c dff.v dff_tb.v
```

Remember if you run testbench without its top level file, it may give you a parsing error.

If everything goes well you can see following output at your terminal window.

```
linux1[113]% ncverilog dff.v dff_tb.v
ncverilog: 06.11-s008: (c) Copyright 1995-2007 Cadence Design Systems, Inc.
file: dff.v
file: dff_tb.v
                Caching library 'worklib' ....... Done
        Elaborating the design hierarchy:
        Building instance overlay tables: ..................... Done
        Loading native compiled code:     ..................... Done
        Building instance specific data structures.
        Design hierarchy summary:
                            Instances  Unique
                Modules:            2       2
                Registers:          4       4
                Scalar wires:       4       -
                Always blocks:      3       3
                Initial blocks:     1       1
                Pseudo assignments: 3       3
        Writing initial simulation snapshot: worklib.dff_tb:v
Loading snapshot worklib.dff_tb:v ..................... Done
ncsim> source /afs/umbc.edu/software/cadence/software_2008/IUS61/tools/inca/files/ncsimrc
ncsim> run
d=0, clk=1, rst=0,q=x

d=1, clk=1, rst=1,q=0

d=1, clk=1, rst=0,q=1

d=1, clk=1, rst=0,q=0

d=1, clk=1, rst=0,q=0

d=1, clk=1, rst=0,q=0

d=1, clk=1, rst=0,q=0

Simulation complete via $finish(1) at time 75 NS + 0
./dff_tb.v:16   #50 $finish;
ncsim> exit
```

(I know there should not be any “x” in the design, but remember we are not concentrating on design right now)

Now, we are done Compiling and Simulation.

After running compiler and simulator, you can observe that your current directory will contain “INCA_libs”. This folder will hold the snapshot of the simulations.

To invoke the snapshot:

```
%ncsim work.dff_tb
```

```
linux1[115]% ncsim work.dff_tb
ncsim: 06.11-s008: (c) Copyright 1995-2007 Cadence Design Systems, Inc.
ncsim> run
d=0, clk=1, rst=0,q=x

d=1, clk=1, rst=1,q=0

d=1, clk=1, rst=0,q=1

d=1, clk=1, rst=0,q=0

d=1, clk=1, rst=0,q=0

d=1, clk=1, rst=0,q=0

d=1, clk=1, rst=0,q=0

Simulation complete via $finish(1) at time 75 NS + 0
./dff_tb.v:16   #50 $finish;
ncsim> exit
```

Basically it will show you the same thing that had been generated after running simulator.

You can always refer to log files but delete them once you are done with the reports.

If you have any confusion with the tutorial contact *ameyk1@umbc.edu*