

**Walkthrough 1
Standard Cell Design
Entry and Simulation of Standard
Library Components**

Author Neil Cole

Date Thu 24-Mar-2011 10:38 am

1	Introduction	1
2	Conventions used in this document	1
2.1	Entering Parameters for commands	2
3	Accessing and Configuring the Design Environment	2
3.1	Creating a Design Environment	2
3.2	Using the Transport Form	3
4	Configuring and Starting Cadence NC VHDL Simulation Environment	4
4.1	Critical Issue	4
4.2	Starting NCVHDL for the first time	5
4.3	Selecting the Run Mode	6
4.4	Configuring the Open Design Directory Form.	6
4.5	Create cds.lib form	7
4.6	New cds.lib file form	8
4.7	NCLaunch Form Final Actions	9
4.7.1	Compiling the Example Files	14
4.7.2	Elaborate command	17
4.7.3	Elaborate form	17
4.7.4	Sending the signals to the waveform window	21
4.7.4.1	Using the Console window to limit the simulation time	23
4.7.5	Viewing the Output Signals	23
5	Synthesis Using SOC Encounter	25
5.0.1	Further information	25
5.1	Invoking the Synthesis tool	25
5.2	Reading in the HDL file	27
5.3	Elaboration	27
5.4	Generating the Output Netlist	29
5.5	Exiting the Synthesis tool	30
5.5.1	Checking the verilog file	30
5.6	That was easy	30
5.7	Why is everything text based? or Why is the graphics tool so dumb?	30
6	Importing and converting the synthesised design into the Schematic Environment	30
6.0.1	Loading the Configuration file	33
6.0.2	Target Library Name	33
6.0.3	Verilog Files to Import	33
6.1	Checking in Cadence	34
7	Simulating the design in Verilog	35

1 Introduction

This document is intended to replace the digital Design Kit to Verilog exercise in the current 4407. The changes in the licensing structure. Whilst sub optimal in terms of course development the intention is to replicate the functions covered in the original walkthroughs

Compared to the original walkthrough the change are:

Table 1:

Change	Original	Update
Design Kit	Mietec 2um	AMS 0.35
Simulator	leapfrog	NCVHDL
Environment	Unix Direct	UNIX/Linux Transport
Design Environment	4.1	4.5/4.6

2 Conventions used in this document

The following are conventions that will be used in this document.

- Where user input is required it will be delineated with "<>".
- Where a special function key such as escape, control or return/enter is indicated in a command it will also be enclosed in "<>" brackets. For example "<RETURN>".
- Please be aware of spaces in words and commands. Like Windows and common English UNIX and Linux require spaces between commands.
- Unix and Linux commands are lower case unless stated. Unix and Linux are case sensitive.
- Operations involving the mouse will use the left hand mouse button unless otherwise stated.
- Options selected from sub menus will be indicated in the following manner

Main Menu Item -> Sub menu Item -> Sub Menu Item

- e.g. File -> New -> Library

2.1 Entering Parameters for commands

Unlike windows most UNIX/Linux parameters are delineated by a "<SPACE>". For example

- `grep -i fred *.v`

3 Accessing and Configuring the Design Environment

3.1 Creating a Design Environment

Students using the AMI design system are provided with a Transport menu that facilitates easy access to appropriate design directories and associated applications software.

Accessing and Configuring the design environment is achieved using the “transport” tool. Students should start a UNIX terminal window and invoke the transport tool by typing the command:

- transport & <RETURN>

The *Transport* form should appear as shown below.

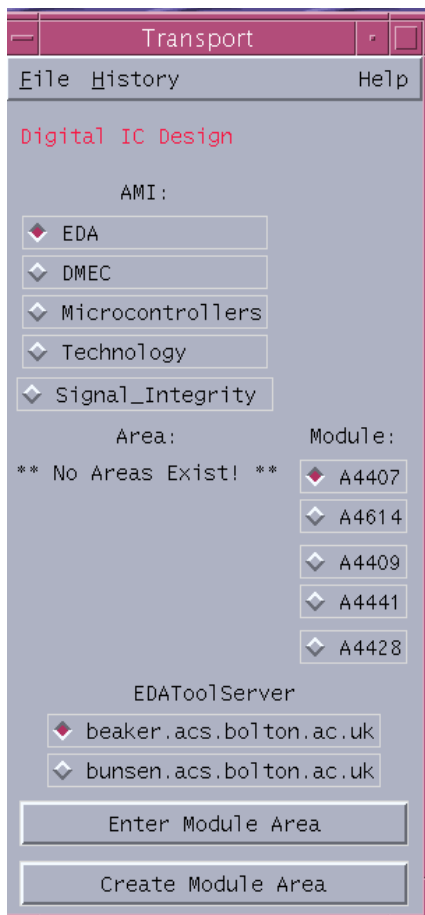


Fig: Transport Menu with no module areas created for A4407

The menu enables a module design area to be created and then entered by specifying the module being studied. Modules are classified by type and function.

3.2 Using the Transport Form

Click on the **EDA** button to display a list of modules within that classification.

- Select the module **A4407**

If you are using the transport tool for the first time for this module there will be no design areas present and the “Area” field will be blank. In this case you will need to create the default structure by using the “Create Module Area” button.

- Click on the **Create Module Area** button to the design areas.

This may take a few seconds. Once this has run successfully the “Area” field will be populated. As shown in the figure “Transport Menu with module areas specified”.

Now select the Walkthroughs area. Only the *Walkthroughs* area will be used for this exercise.

- Click on the **Walkthroughs** button

The *Transport* menu should now be as shown below.

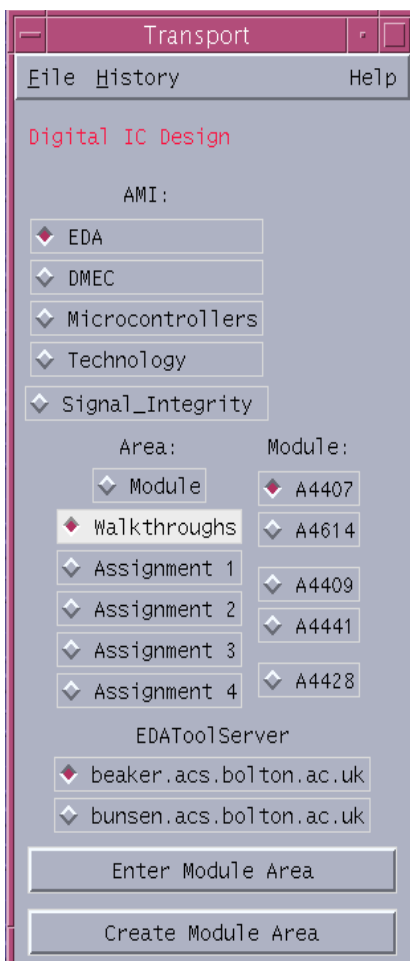


Fig: Transport Menu with module areas specified

Now click on the **Enter Module Area** button to open up a configured UNIX terminal window. All subsequent commands will be typed in this window. This window will be in the correct location, in this case the "Walkthrough" area. For the 4407 module this will be physically located at "~AMI/A4407/Walkthroughs". This area is also configured to allow you to run all the relevant tools and

4 Configuring and Starting Cadence NC VHDL Simulation Environment

4.1 Critical Issue

Both the Cadence schematic/layout/analogue/mixed signal and the Cadence NCSIM environments use a single common file the cds.lib. However these are not compatible and the existence of a file in one format in a directory will destroy the function of the other tool in that environment.

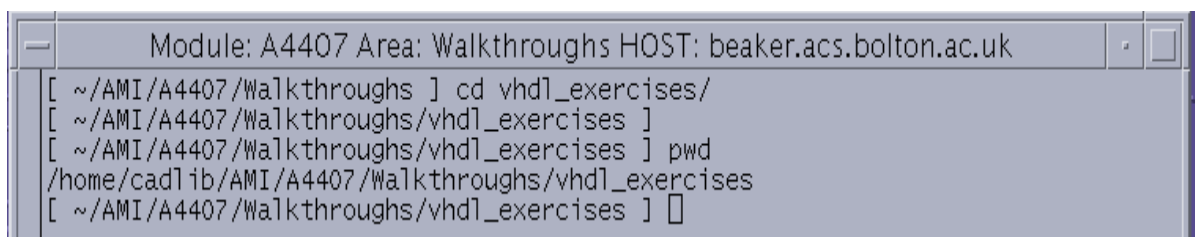
We are already using the Walkthroughs area for the Cadence design tools. Hence we need to use a separate directory for the VHDL work.

Your Walkthroughs area has a directory in it called vhd1_exercises. All VHDL work should be carried out in this area.

To move from a terminal window opened in the Walkthroughs area to the vhd1_exercises area type the following command:

- `cd vhd1_exercises <RETURN>`

The location prompt on the left hand side of the terminal window will change to reflect the new location.



```
Module: A4407 Area: Walkthroughs HOST: beaker.acs.bolton.ac.uk
[ ~/AMI/A4407/Walkthroughs ] cd vhd1_exercises/
[ ~/AMI/A4407/Walkthroughs/vhd1_exercises ]
[ ~/AMI/A4407/Walkthroughs/vhd1_exercises ] pwd
/home/cad1ib/AMI/A4407/Walkthroughs/vhd1_exercises
[ ~/AMI/A4407/Walkthroughs/vhd1_exercises ] □
```

Walkthroughs window showing commands and responses for moving into vhd1_exercises area.

You can check the location using the following command:

- `pwd`

You should make sure you are in the correct area before invoking any tools.

4.2 Starting NCVHDL for the first time

Make sure you are in the correct directory, (Walkthroughs/vhdl_exercises).

The command used to invoke the simulation environment is:

- `nclaunch & <RETURN>`

Hint: The `&` allows the tool to run in background so you can still run commands in the terminal window. Otherwise you will be unable to use the terminal window until you exit the tool.

The first time you do so, or when you start a new design area, using the command “`nclaunch -new`” you will be presented with the selection windows shown below. The second time you use the command in this area you will be taken directly to the selected simulation environment. You need to make the following selections for this course module:

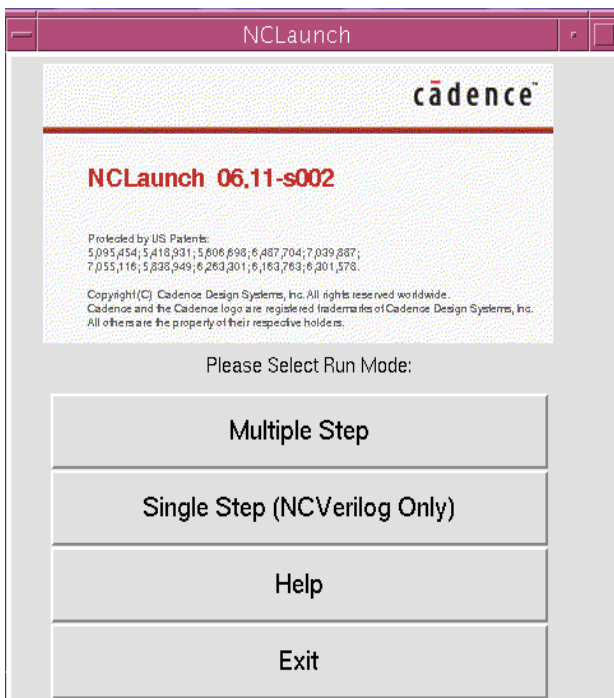


Fig: NCLaunch Startup Form

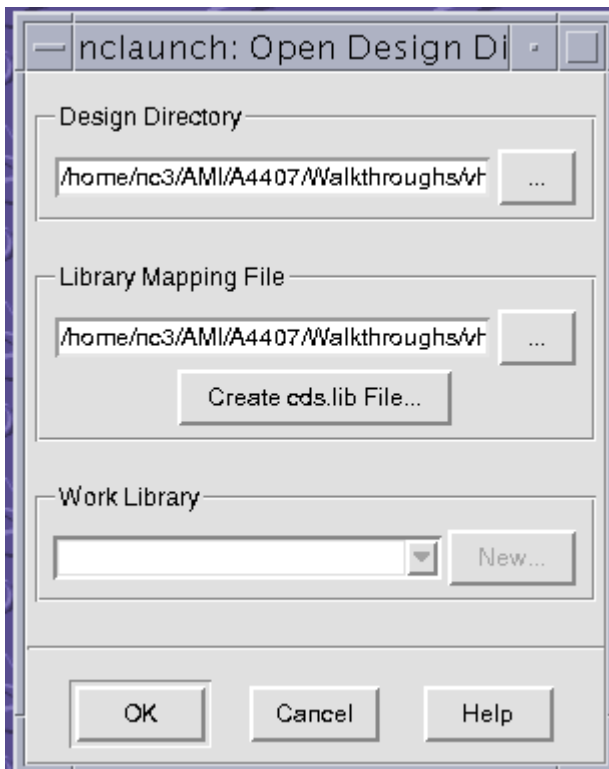
4.3 Selecting the Run Mode

You need to select the multiple step option for VHDL.

- **Multiple Step**

The next form will then be displayed. This will be the Open Design Directory Form.

4.4 Configuring the Open Design Directory Form.



You need to do the following actions:

- **Select the “Create cds.lib File” button to create the cds.lib for the tool.**

This will also open the “Create a cds.lib” form.

4.5 Create cds.lib form

You need to do the following actions on the “Create a cds.lib file form:

- **Select the “Save” on this form.**

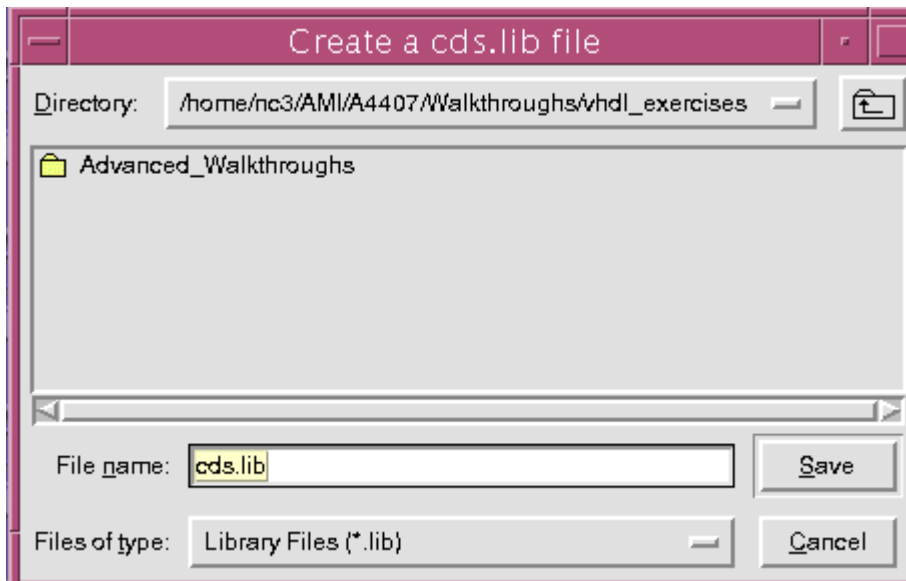


Fig: Create cds.lib file form

When “Save” has been selected the library selection form will be displayed.

4.6 New cds.lib file form

We are going to use the standard IEEE libraries for this exercise.

You need to do the following actions on the “New cds.lib file” form:

- **Select the “IEEE” libraries on the form**
- **“OK” the form.**

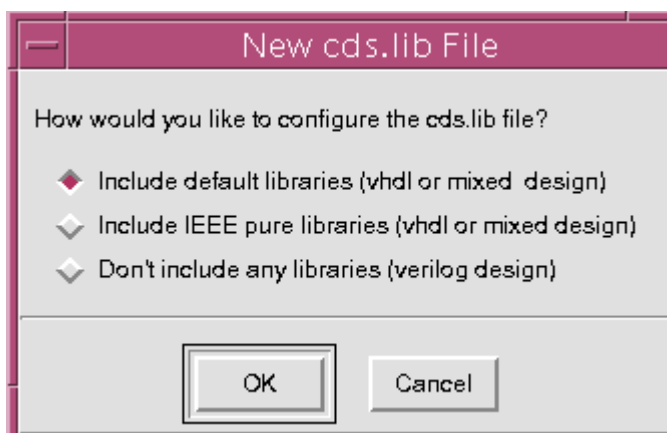


Fig: New cds.lib file form

You will then be returned to the nlaunch form. The worklib and the other entries will now be filled in.

4.7 NCLaunch Form Final Actions

The NCLaunch form should now resemble the one shown below.

You need to do the following actions on the NCLaunch form:

- “OK” the form.

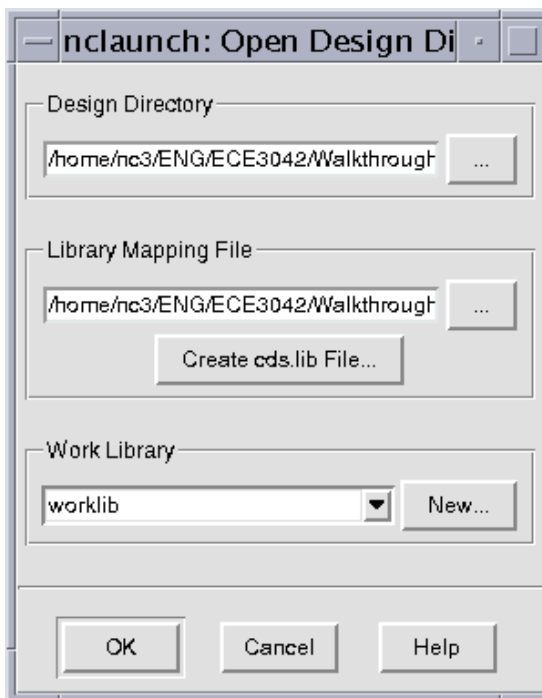


Fig: NCLaunch Open Design Form Completed

This will then close and you can now use the VHDL simulation environment will be displayed for you. You can see that the standard libraries i.e. iee and the worklib library are displayed for use.

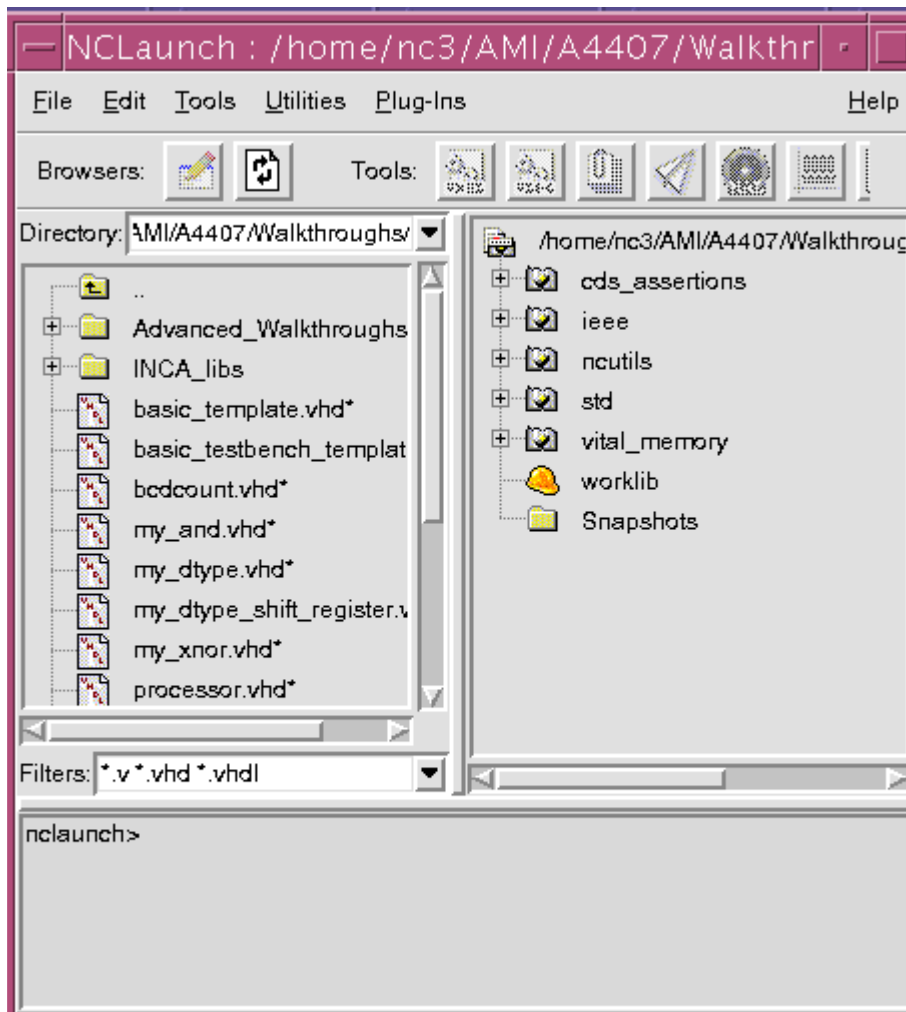


Fig: NCLaunch VHDL Simulation Tool

0.7. Making Life Easier

To make life easier we are going to use “nedit” rather than the default text editor to edit VHDL files. “nedit” is a context sensitive editor with similar commands to windows editors. It can be invoked directly at a terminal by type “nedit”. We are going to set the VHDL tool to use nedit. Select the Edit->Preferences option from the NCLaunch tool bar to display the preferences section and make sure the editor command includes nedit as shown below. Initially the form will display the lines shown below:

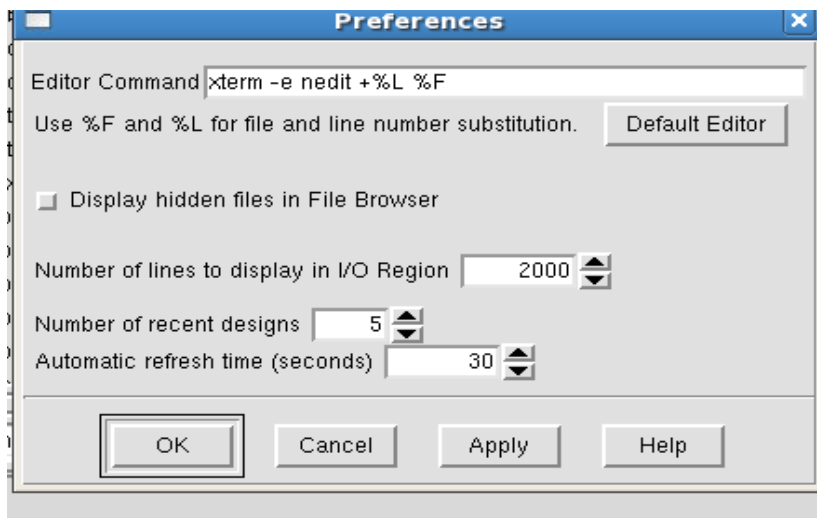


Fig: NCLaunch preferences form

We are going to change this so the “Editor Command” is:

- `nedit +%L %F`

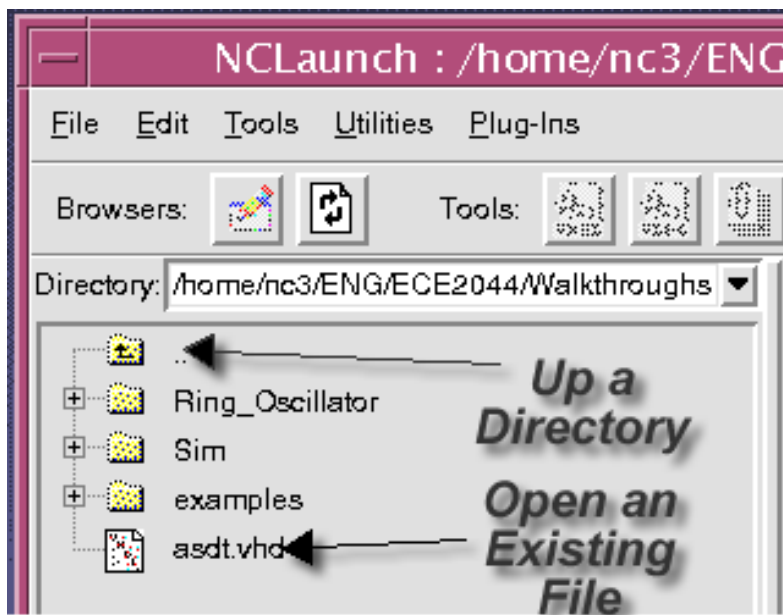
2. Using the Environment

2.1. Creating a new vhdl file

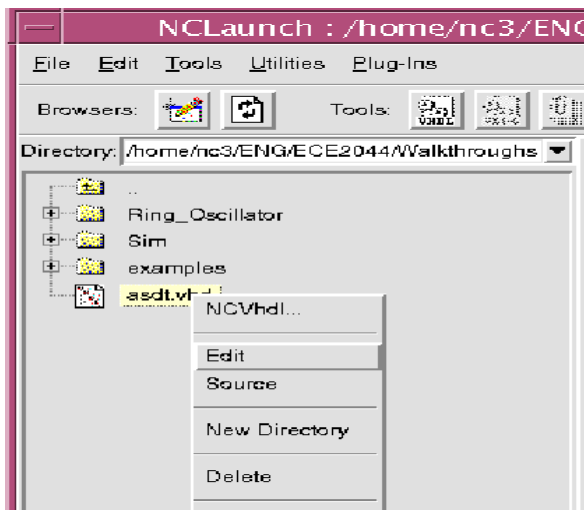
Use the File->Edit New File command on the nclaunch menu to open the new file.

2.2. Opening an existing VHDL File for edit

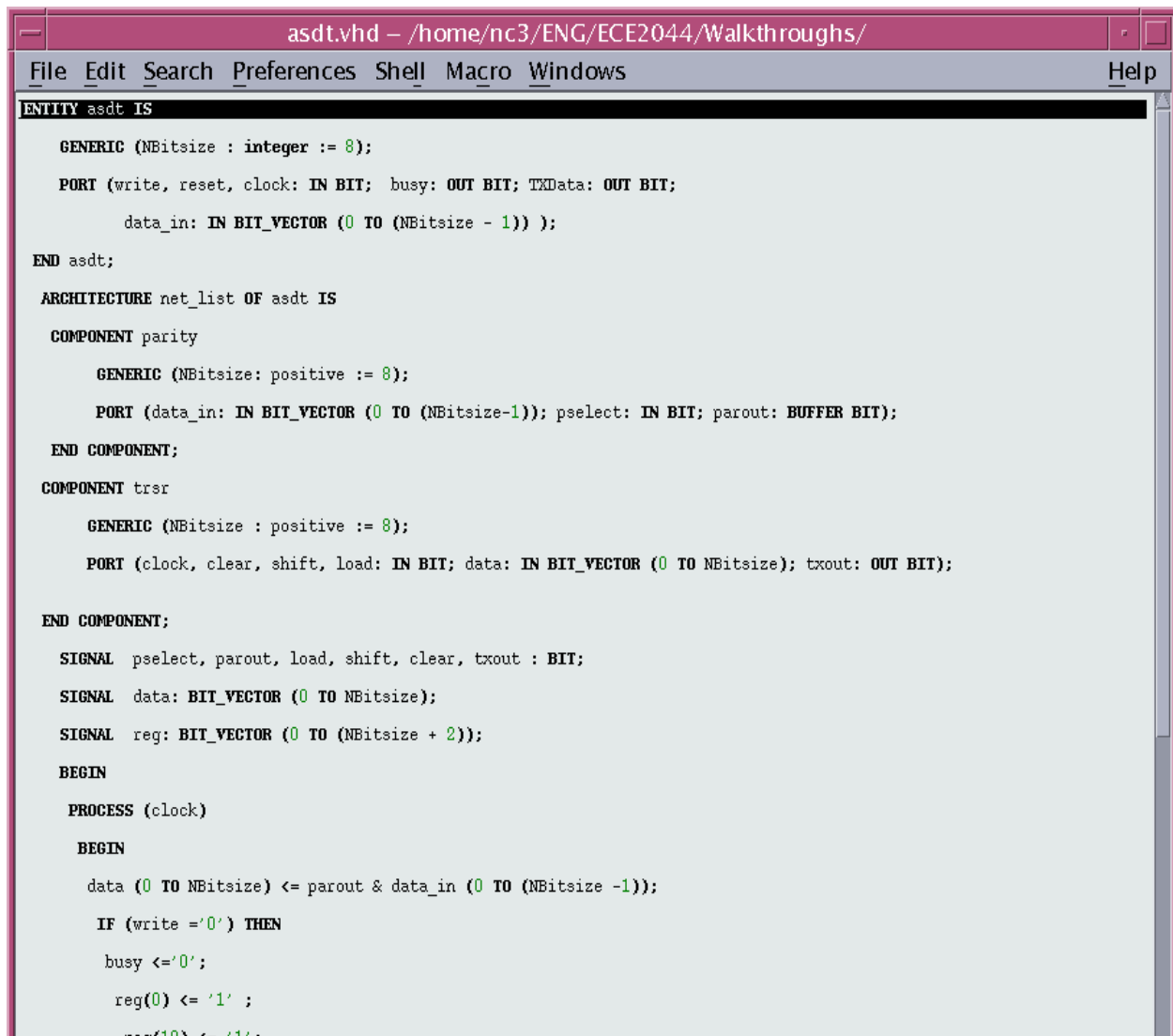
You can either open up an existing VHDL file or create one from scratch. To open an existing file navigate to the correct directory using the mouse and file browser in the left hand pane of the NCLaunch tool.



- Left click on the file to select
- Right click to display the context sensitive menu and select the Edit option from the menu



The file will be opened for editing in the nedit tool. It will also open up a terminal window that can be iconised. “nedit” is a context sensitive editor and will colour the structures in that it recognises. It uses the file extension to decide on the language to use.



```
asdt.vhd - /home/nc3/ENG/ECE2044/Walkthroughs/
File Edit Search Preferences Shell Macro Windows Help
ENTITY asdt IS
    GENERIC (NBitsize : integer := 8);
    PORT (write, reset, clock: IN BIT; busy: OUT BIT; TXData: OUT BIT;
          data_in: IN BIT_VECTOR (0 TO (NBitsize - 1)) );
END asdt;

ARCHITECTURE net_list OF asdt IS
    COMPONENT parity
        GENERIC (NBitsize: positive := 8);
        PORT (data_in: IN BIT_VECTOR (0 TO (NBitsize-1))); pselect: IN BIT; parout: BUFFER BIT);
    END COMPONENT;

    COMPONENT trsr
        GENERIC (NBitsize : positive := 8);
        PORT (clock, clear, shift, load: IN BIT; data: IN BIT_VECTOR (0 TO NBitsize); txout: OUT BIT);
    END COMPONENT;

    SIGNAL pselect, parout, load, shift, clear, txout : BIT;
    SIGNAL data: BIT_VECTOR (0 TO NBitsize);
    SIGNAL reg: BIT_VECTOR (0 TO (NBitsize + 2));

    BEGIN
        PROCESS (clock)
            BEGIN
                data (0 TO NBitsize) <= parout & data_in (0 TO (NBitsize -1));

                IF (write = '0') THEN
                    busy <= '0';
                    reg(0) <= '1';
                    reg(10) <= '1';
```

2.2.1. Saving the File

Use the File -> Save option to save the file when you have made changes.

2.2.2. Closing the File

Use the File -> Exit option to save/close the file.

3. Using the VHDL Files

3.1. Compiling the VHDL file

For this walkthrough we will use the `bcdcount.vhd` and `testt_bcdcount.vhd` files used in the original walkthrough.

To compile the VHDL file select the file and either:

- Right click and select “NCVHDL”
- Select the Tools -> Compile to open up the compile form
- Select the Tools VHDL icon

Read the bottom section of the form to ensure that the VHDL has compiled correctly. Errors will be indicated by entries in the information box at the bottom of the form:

4.7.1 Compiling the Example Files

Compile the following files, in this order, by selecting them in the file browser

- `bcdcount.vhd`
- `testt_bcdcount.vhd`

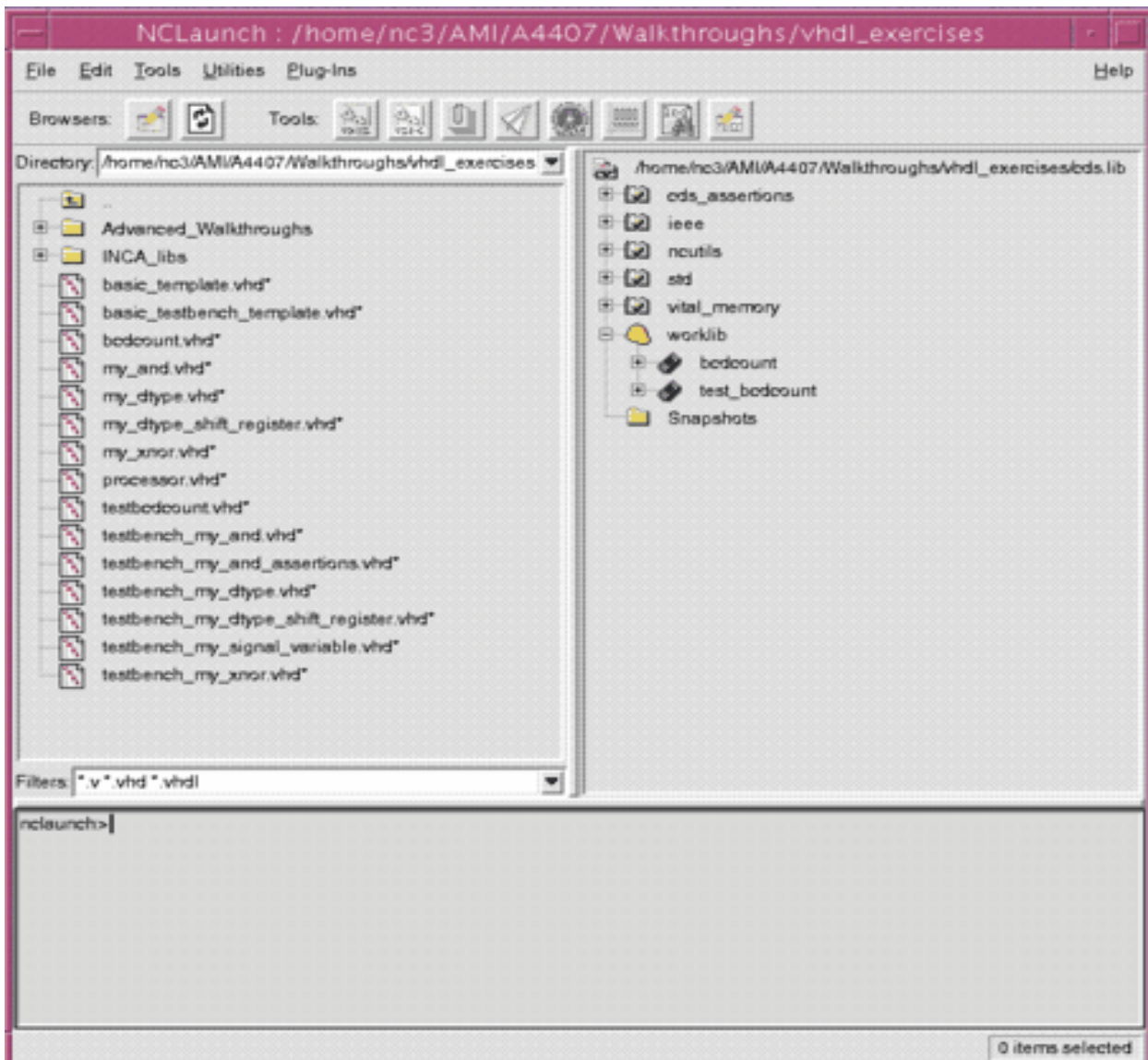


Fig: NCLaunch window format after successful compilation

In order to interpret any error information you need to examine the messages in detail. Where multiple errors occur scroll up the form until you reach the first error and resolve that before trying to resolve later errors. Later errors are quite often caused by the first error. The system will provide as much information as it can on the subject

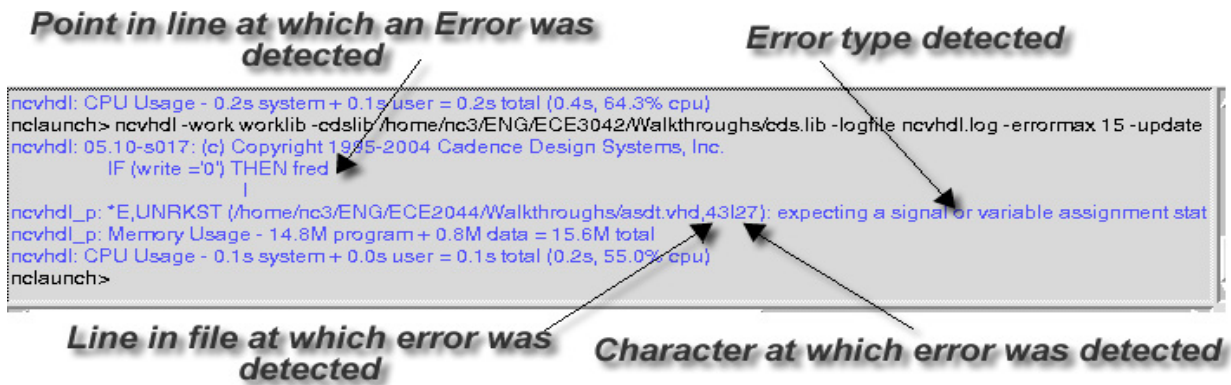


Fig: Showing error messages and information from compilation

You can use this information to help you locate and identify the errors.

3.2. Compiling a Test Bench

Compiling a test bench is exactly the same as compiling a normal VHDL file.

3.3. Elaborating the Test Bench

Once you have successfully compiled both the VHDL file and the test bench VHDL file you can run a simulation. You only need to elaborate the test bench. All the other files which you have compiled will be included automatically. All the architectures will be compiled and installed into the worklib library. When you have successfully compiled the files you will see that the right hand side pane can be expanded to show the entities and architectures that have been created.

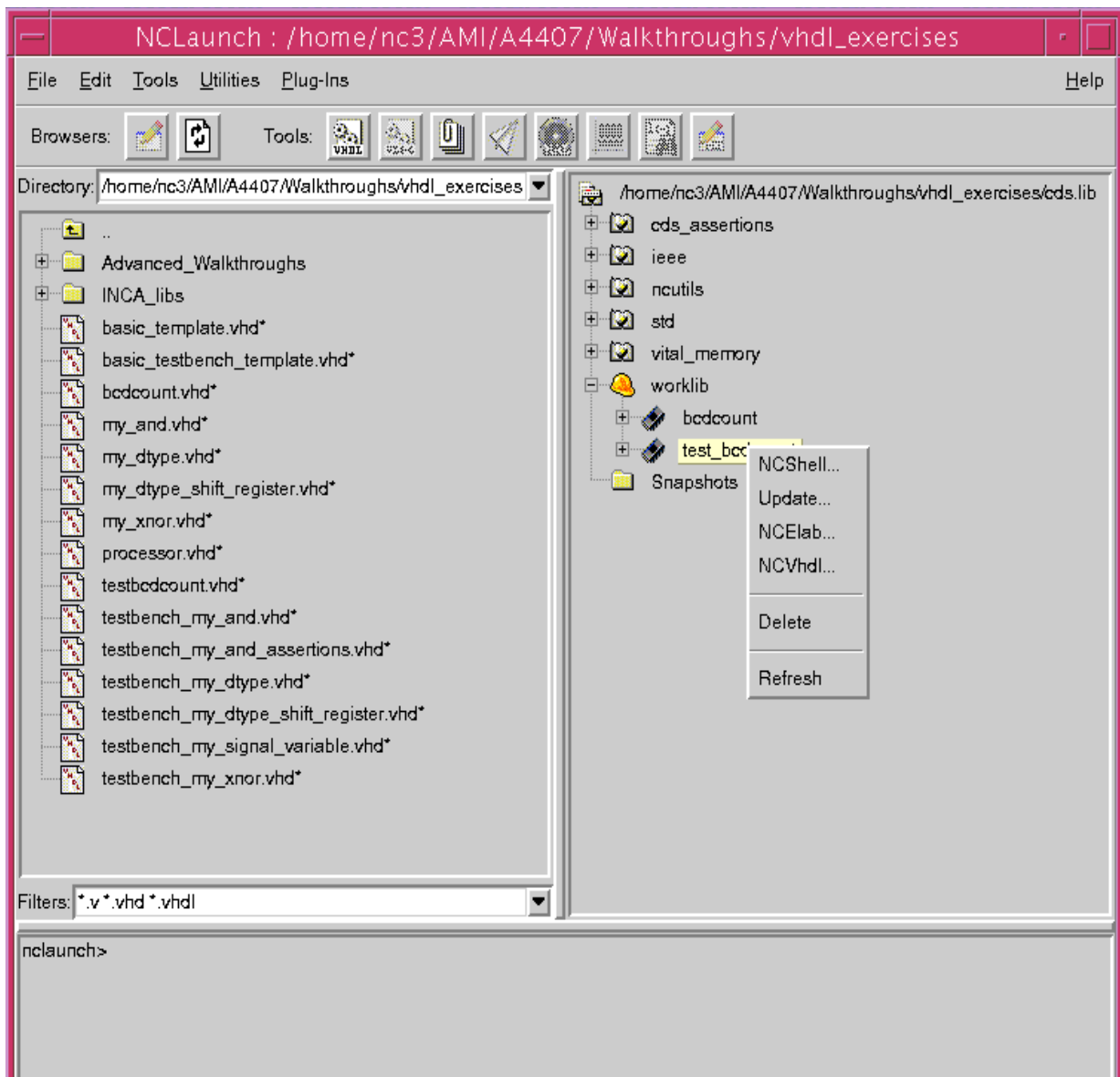


Fig: NCElab Command, File Worklib and Snapshot structure

4.7.2 Elaborate command

To compile a file for simulation select the entry in the worklib e.g. test_bcdcount and right click to display the context sensitive Menu. Select the Elaboration from this menu..

- Select the NCElab option

4.7.3 Elaborate form

This will display the Elaborate form. No changes are needed just "OK" the form. This will create the "Snapshot" view which can be simulate.

- OK this form

If the test bench can be successfully elaborated an entry will appear in the snapshot section of the right hand side viewer. If you are not successful you need to scroll back up the file entry or examine the ncelab.log file to see what the errors are and then correct them.

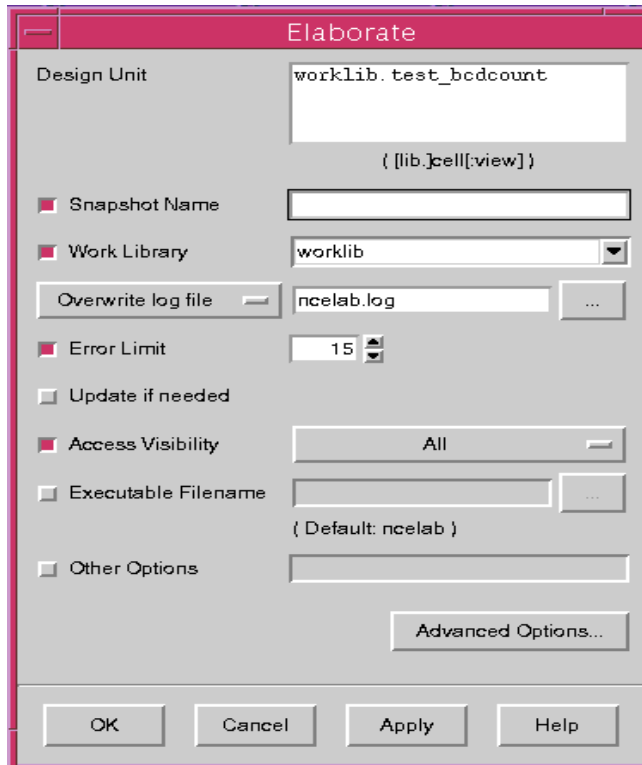


Fig: Elaborate form

3.4. Running and Viewing the Simulation

This will elaborate the test bench. The elaboration operation will include all the VHDL entities referenced by the test bench. If there are mismatches between the VHDL components and the test benches entries for the components there will be errors generated at this stage. If there are no errors then a entry under the snapshot directory will be created.

Entry and Simulation of Standard Library Components

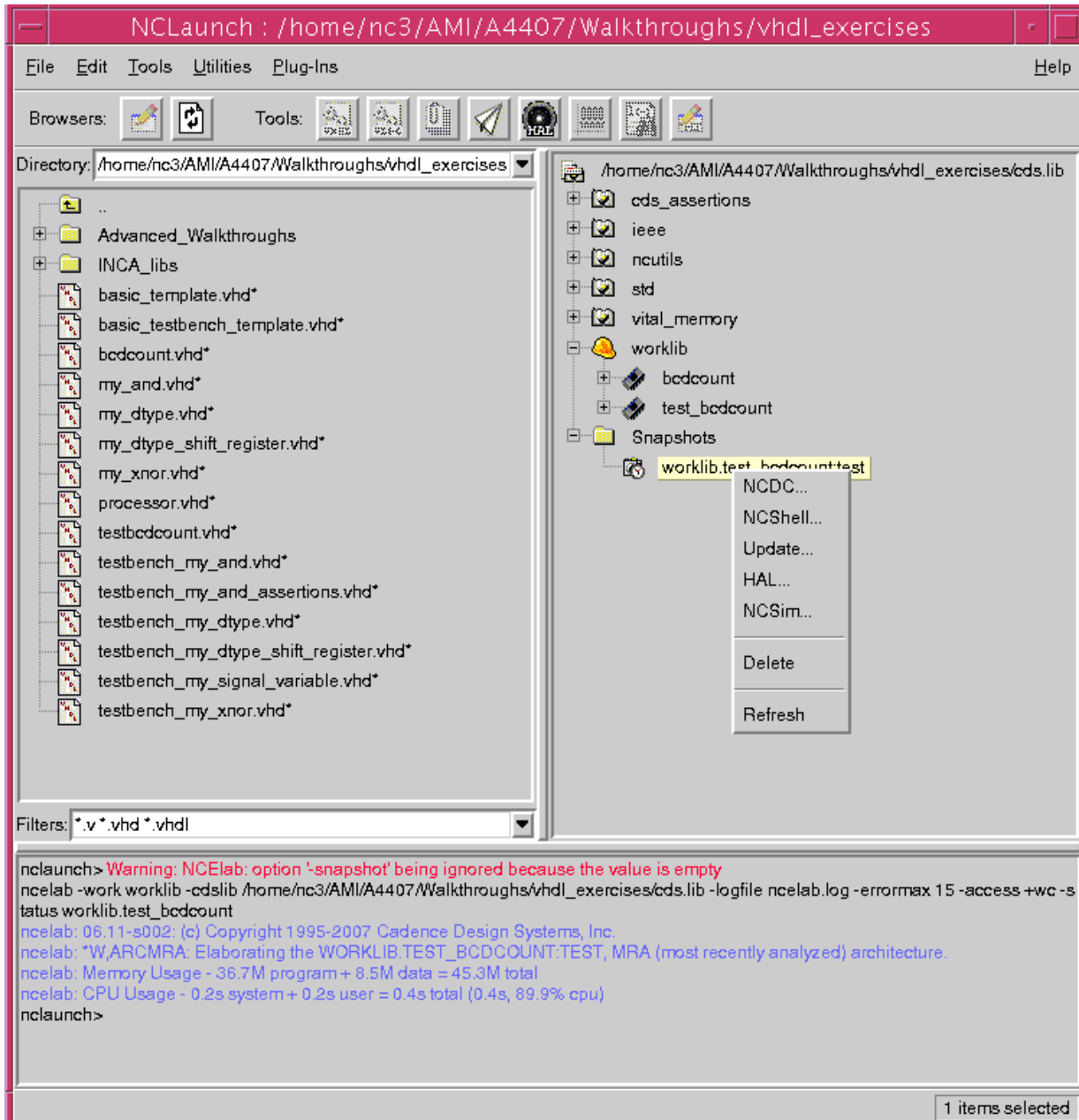


Fig: Snapshot selection and NCSIM Simulation

Select the desired snap shot by left clicking on it once, then right click to display the context

sensitive menu.

- Select the NCSim option that should now be available

This will open the simulation form. “OK” this form to open the simulation tool.

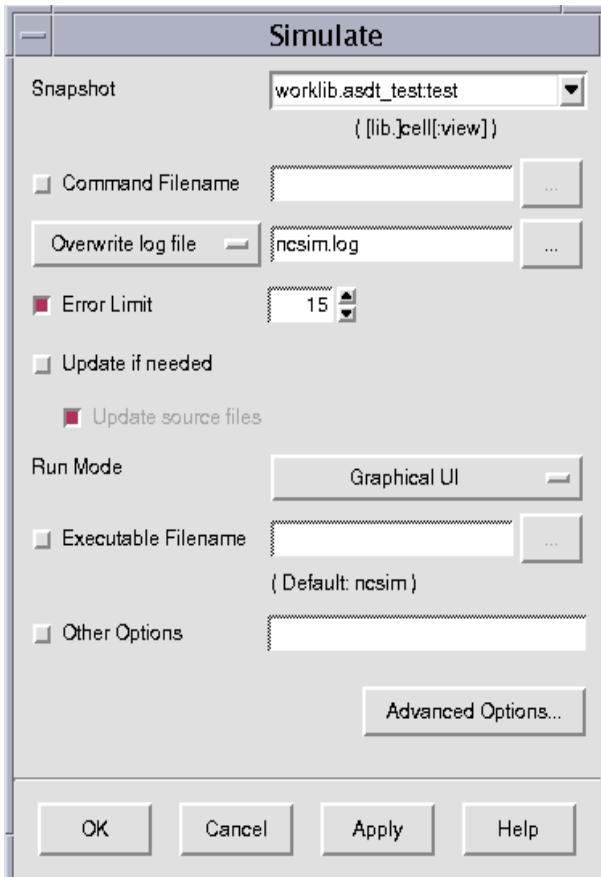


Fig: Simulation form

3.5. Running the Simulation

This will open the multiple windows used by the simulation environment. In particular it will open the “Console” and the “Design Browser” SimVision Windows. We have already covered their uses in the previous walkthrough. However sections of it will be repeated here. Students are recommended to use the additional documents detailing SimVision facilities.

3.5.1. Console

You can use the Console to control the simulation by either typing commands in the console section or using the options in the Simulation-> menu on the Console. The same options are available on the Design Browser but the system output from the NCSIM environment is shown in the Console so it is worth keeping an eye on.

3.5.2. Design Browser

You can use the Design Browser to select the signals that you want displayed during the simulation. You can also use this to display variables and other internal variables during simulation should you wish. If you click on the entries with “+” you can expand the design hierarchy and select the signals you wish to have plotted.

3.6. Selecting the Signals and Variables you want Displayed

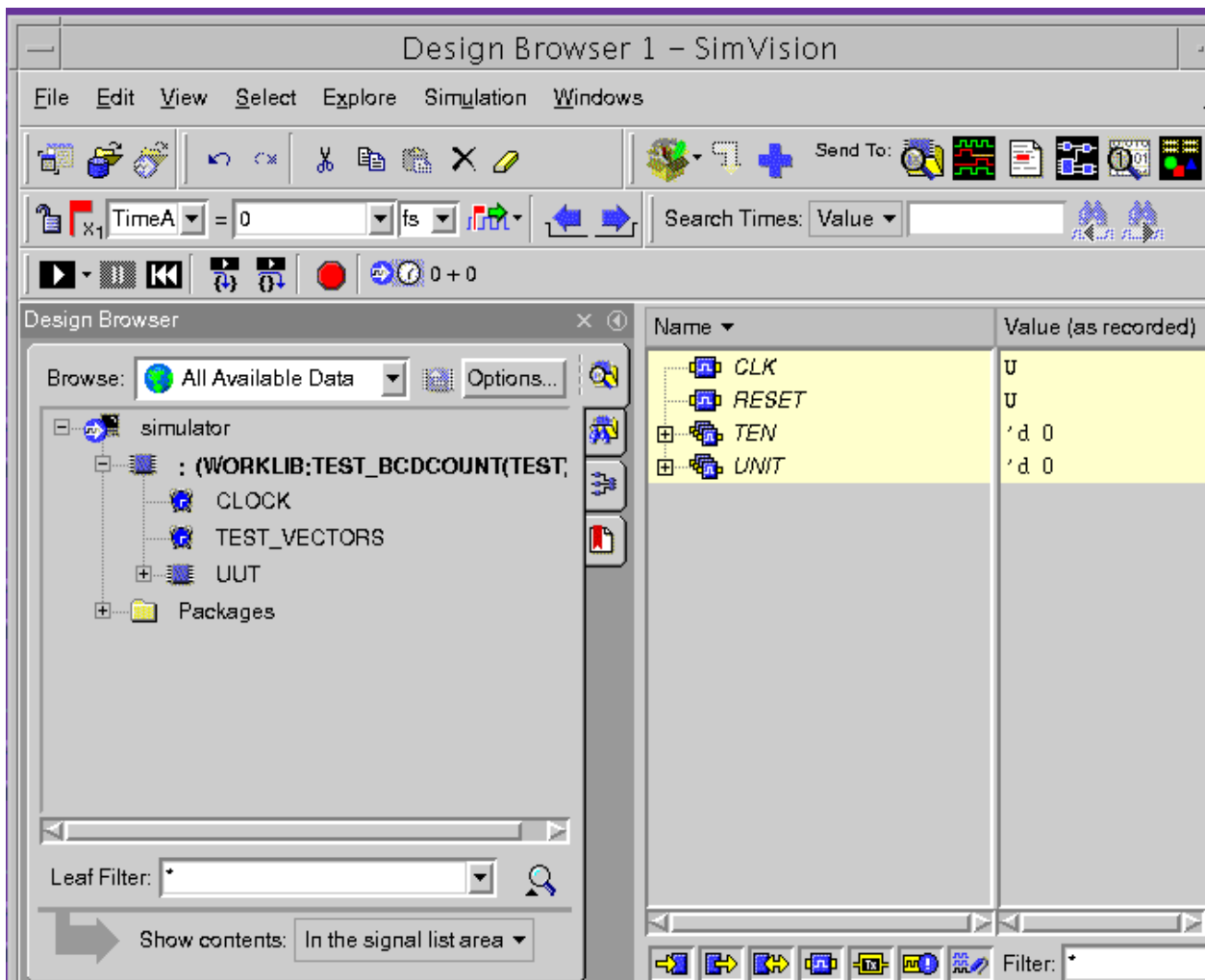


Fig: Design browser with top level signals selected.

4.7.4 Sending the signals to the waveform window

Select the Signal/Variables you wish to display, right click whilst in the right hand pane to display the context sensitive menu and then select “Send To Waveform Window”.

MAKE SURE YOU DO THIS BEFORE YOU START THE SIMULATION!

Entry and Simulation of Standard Library Components

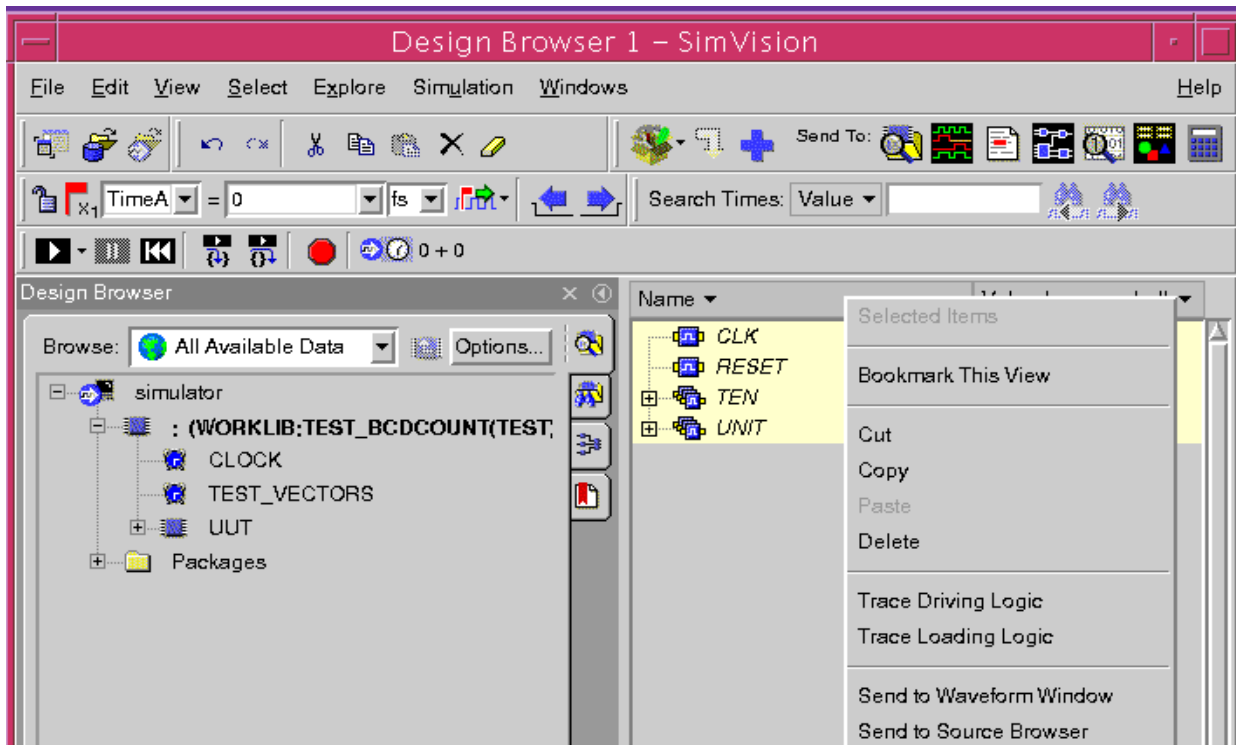


Fig: Waveform window showing context sensitive signals menu

This will open the Waveform window with the signals selected. You can add additional signals and variables should you so wish at any time by following this procedure.

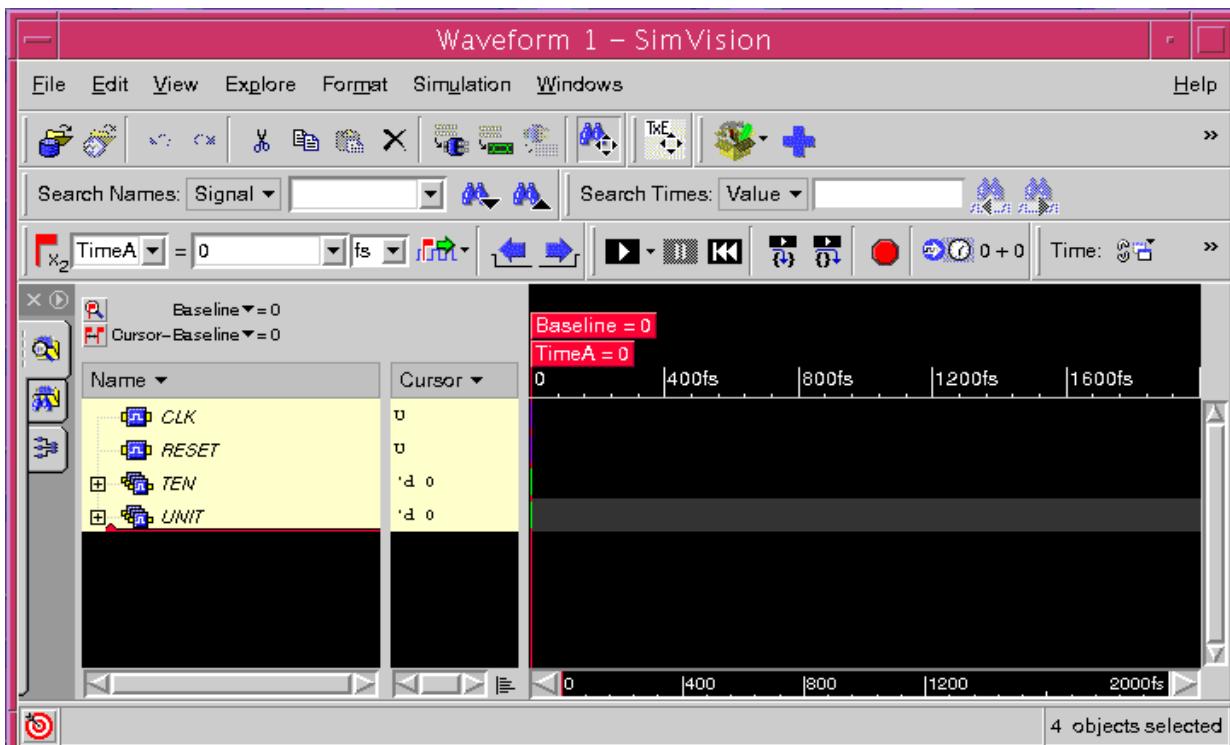


Fig: Initial Waveform Simvision window after signal selection

At this point you have not run the simulation so the values of the signals are unknown.

3.7. Running the simulation

You can run the simulation to the end of time i.e. infinity and interrupt it when you want, run it for a defined period or run it until a specific event occurs, using the Simulation menu options.

In this example I elected to run the simulation for 100us by entering the following command into the console window. Be aware of the spaces and the syntax used!

```
· run 2 us
```

4.7.4.1 Using the Console window to limit the simulation time

The simulation for the test_bcdcount will run continuously. So we need to set a finish time. Use the

run 2 us command in the Simvision Console window.

4.7.5 Viewing the Output Signals.

The signals in the Design Waveform window will now update.

Entry and Simulation of Standard Library Components

The first thing that needs to be done is for us to zoom out. Using the View -> Zoom -> Full X option. This will display the entire simulation run.

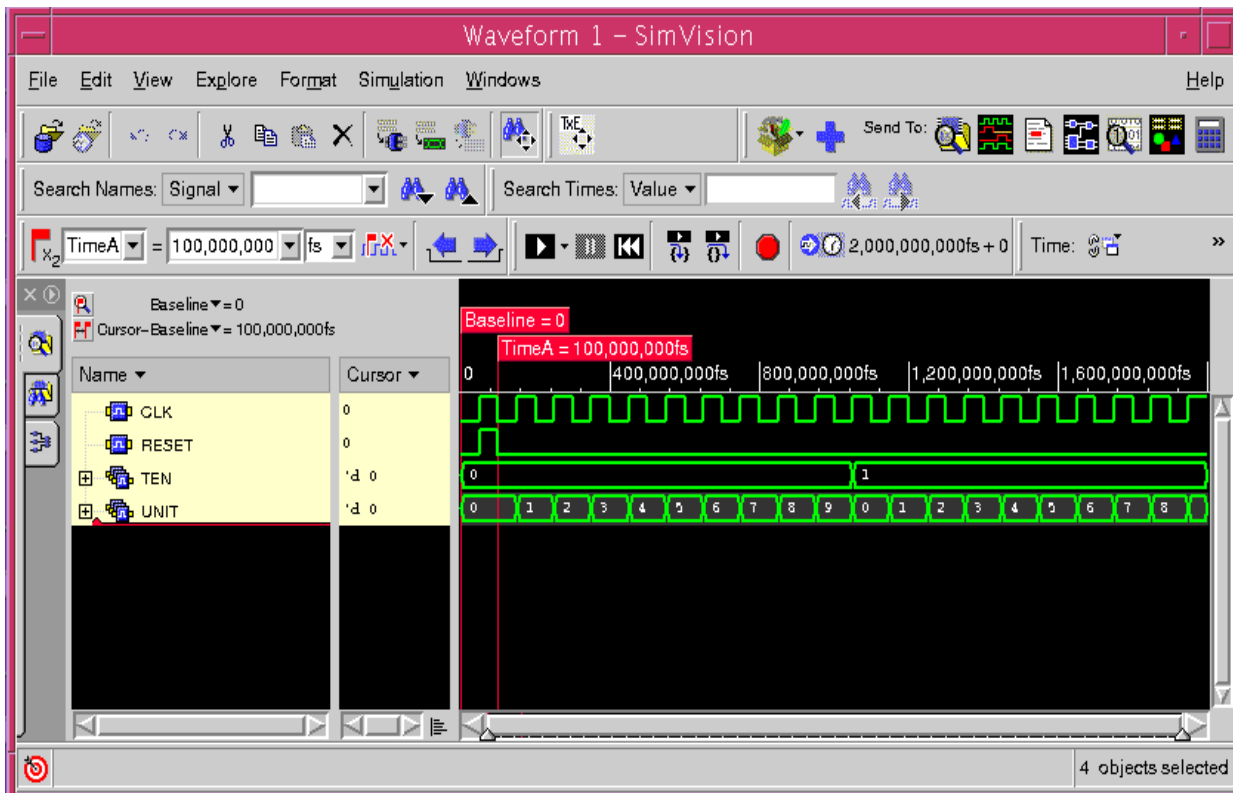


Fig: test_bcdcount 2us simulation waveform results

You can now use the View tools and the scroll bars to zoom in, zoom out and manipulate the views.

3.8. Congratulations

You have now compiled and simulated a VHDL module! All you need to do now is check the simulation and make sure that:

- The test bench does all the simulation tests you need
- The VHDL has the correct responses

The recommendation would be to keep this guide as a template to help jog your memory when doing further VHDL simulations.

5 Synthesis Using SOC Encounter

We are going to use the Cadence PKS to do a basic synthesis on the `bcdcount.vhd` file. As required this file conforms to the required RTL syntax and structure. Hence it can be synthesised.

5.0.1 Further information

The relevant libraries and configuration files have already been read into the synthesis tool allowing it to target the AMS 0.35um technology library. For those interested it is recommended to use the cadence only help invoked from the terminal window using the command:

- `cdnshelp <RETURN>`

Search for the

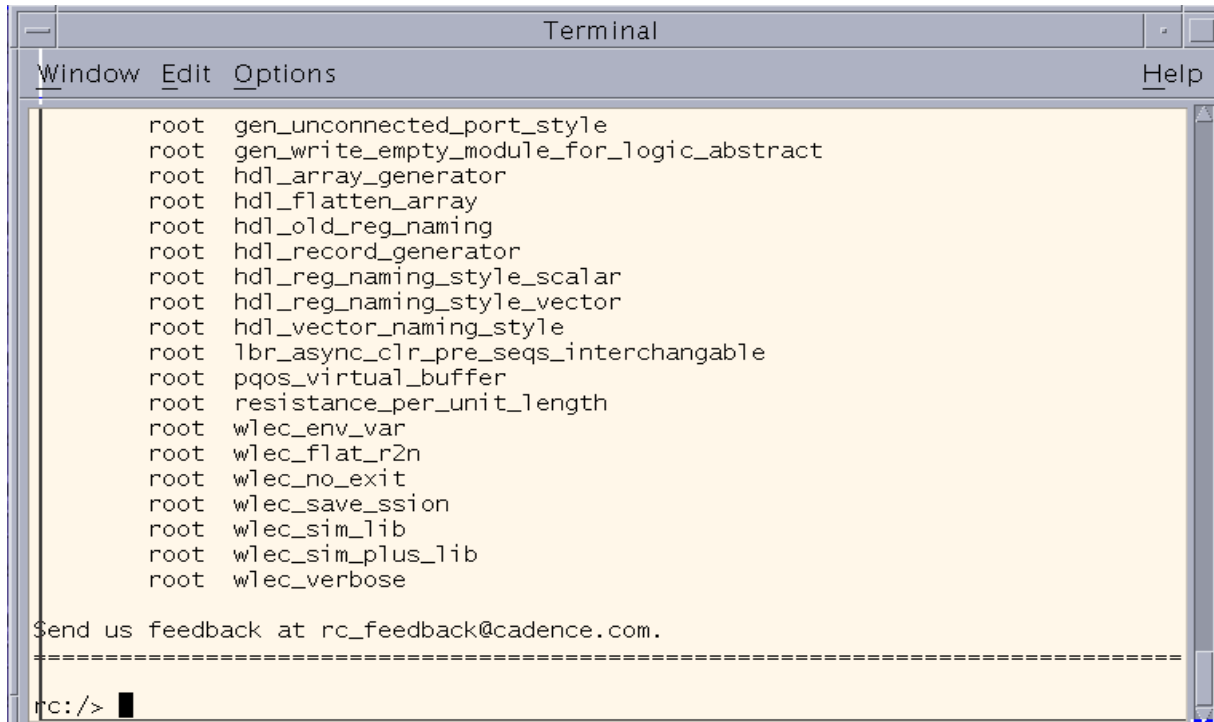
5.1 Invoking the Synthesis tool

Make sure you are in the `Walkthroughs/vhdl_exercises` directory. Then type the following command to invoke a configured synthesis tool:

- `amssynth <RETURN>`

This will invoke a configured SOC Encounter synthesis environment.

It will open up the graphics display and the terminal in which the command was typed will be converted into an interface to the synthesis tool. This will be signified by the prompt turning from the usual one, which indicates the current directory, to an `"rc:/>"` prompt.



The image shows a terminal window titled "Terminal" with a menu bar containing "Window", "Edit", "Options", and "Help". The terminal content is as follows:

```
root gen_unconnected_port_style
root gen_write_empty_module_for_logic_abstract
root hdl_array_generator
root hdl_flatten_array
root hdl_old_reg_naming
root hdl_record_generator
root hdl_reg_naming_style_scalar
root hdl_reg_naming_style_vector
root hdl_vector_naming_style
root lbr_async_clr_pre_seqs_interchangable
root pqos_virtual_buffer
root resistance_per_unit_length
root wlec_env_var
root wlec_flat_r2n
root wlec_no_exit
root wlec_save_ssion
root wlec_sim_lib
root wlec_sim_plus_lib
root wlec_verbose
```

Send us feedback at rc_feedback@cadence.com.

```
rc: /> █
```

Fig: Terminal configured for synthesis input after invocation of synthesis tools

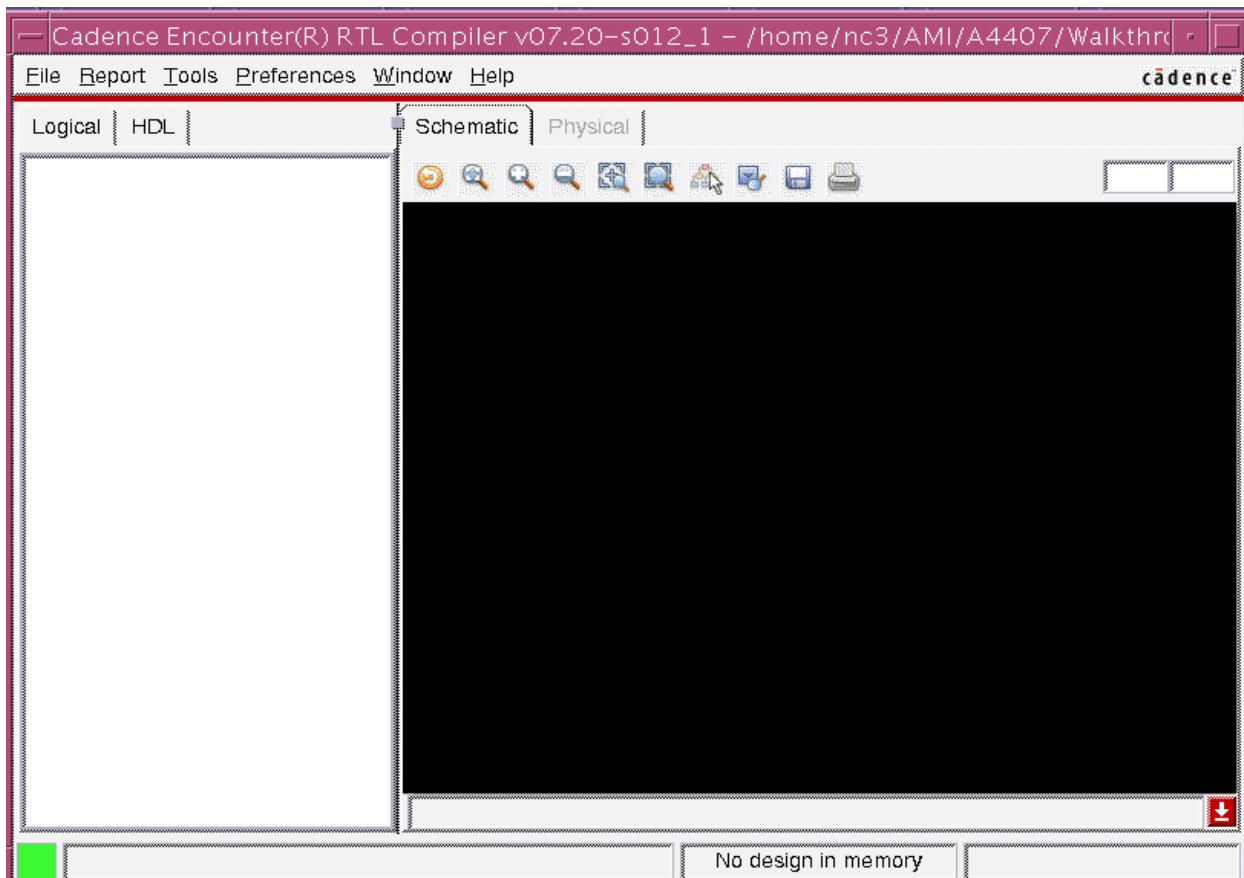


Fig: Graphics User interface to Synthesis tools

5.2 Reading in the HDL file

Synthesis can be carried out using either Verilog or VHDL files. The default is Verilog as we are using VHDL we need to specify this on the command. The following command assumes you were in the Walkthroughs/vhdl_exercises directory when you invoked the synthesis tools. You can read in the VHDL file bcdcount.vhd using the following command in the terminal window with the “rc:/>” prompt.

- `read_hdl -vhdl bcdcount.vhd`

Be careful of spaces and the differences between “_” and “-”. Yes it is larger on purpose!

This will read in the VHDL file into the synthesis tool

5.3 Elaboration

Elaboration is the initial conversion of RTL code into equivalent Boolean statements. In this phase the structure generated is based on “generic” blocks within the synthesis tool, rather than explicit gates on a design technology. In addition to other functions it allows you an

Entry and Simulation of Standard Library Components

overview of the structure generated by the RTL code as implemented. Allowing issues, discrepancies and misunderstandings to be identified. To elaborate the bcdcount.vhd we use the command:

- elaborate

If successful the graphic pane will change to reflect the generic structure.

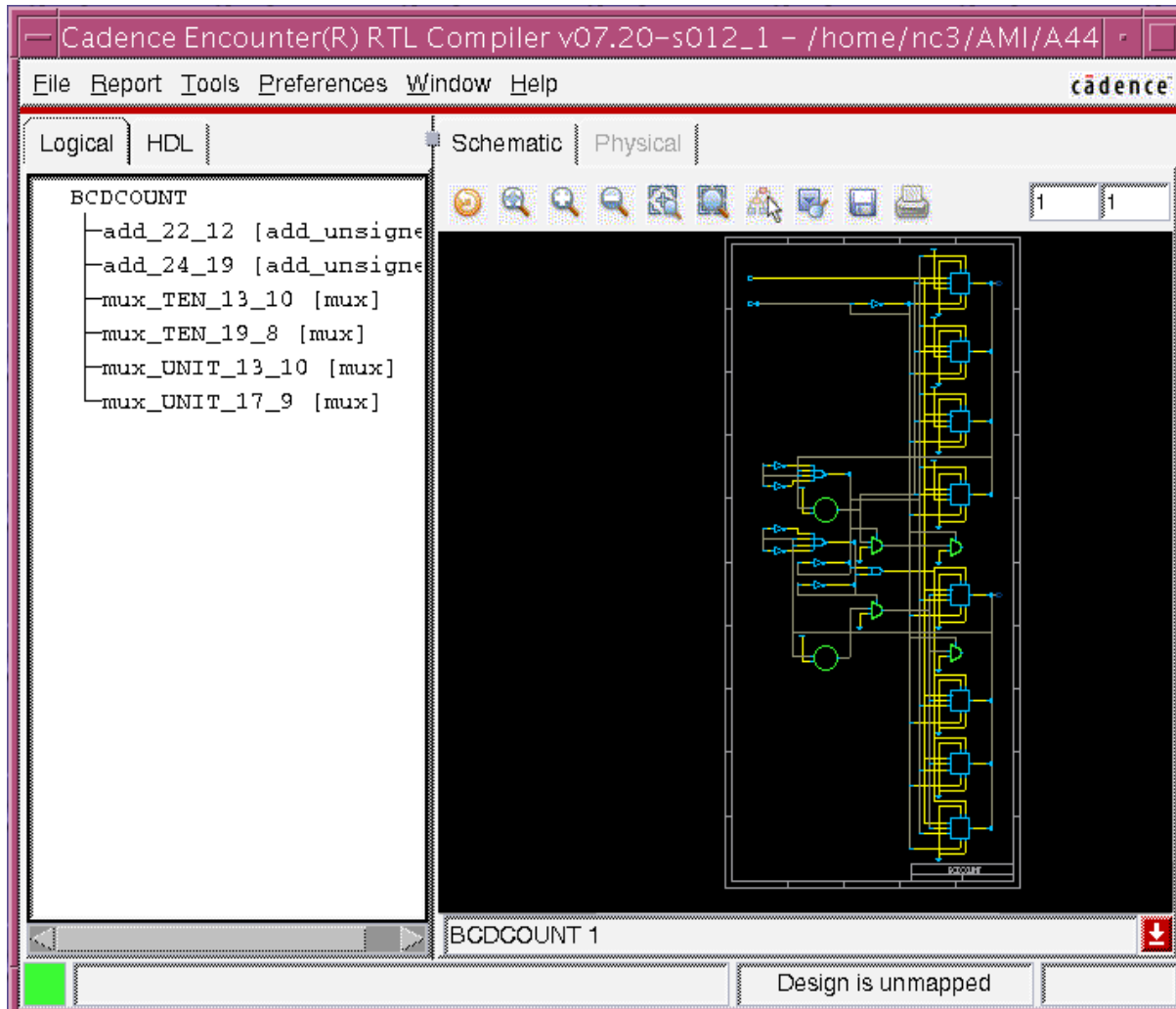


Fig: Synthesis Graphic pane showing generic structure for bcdcount

You can use the zoom icons and other options to examine the structure in detail.

Generating the gate mapped version

The next stage is to actually synthesise the gate level version of the netlist from the generic structures. This is the stage that explicitly targets the design library.

Use the command:

- **synthesize -to_mapped**

Watch the “z” in synthesise, they are American, and the “-” and “_” entries. If successful the graphics window should change again to reflect the gate level mapping.

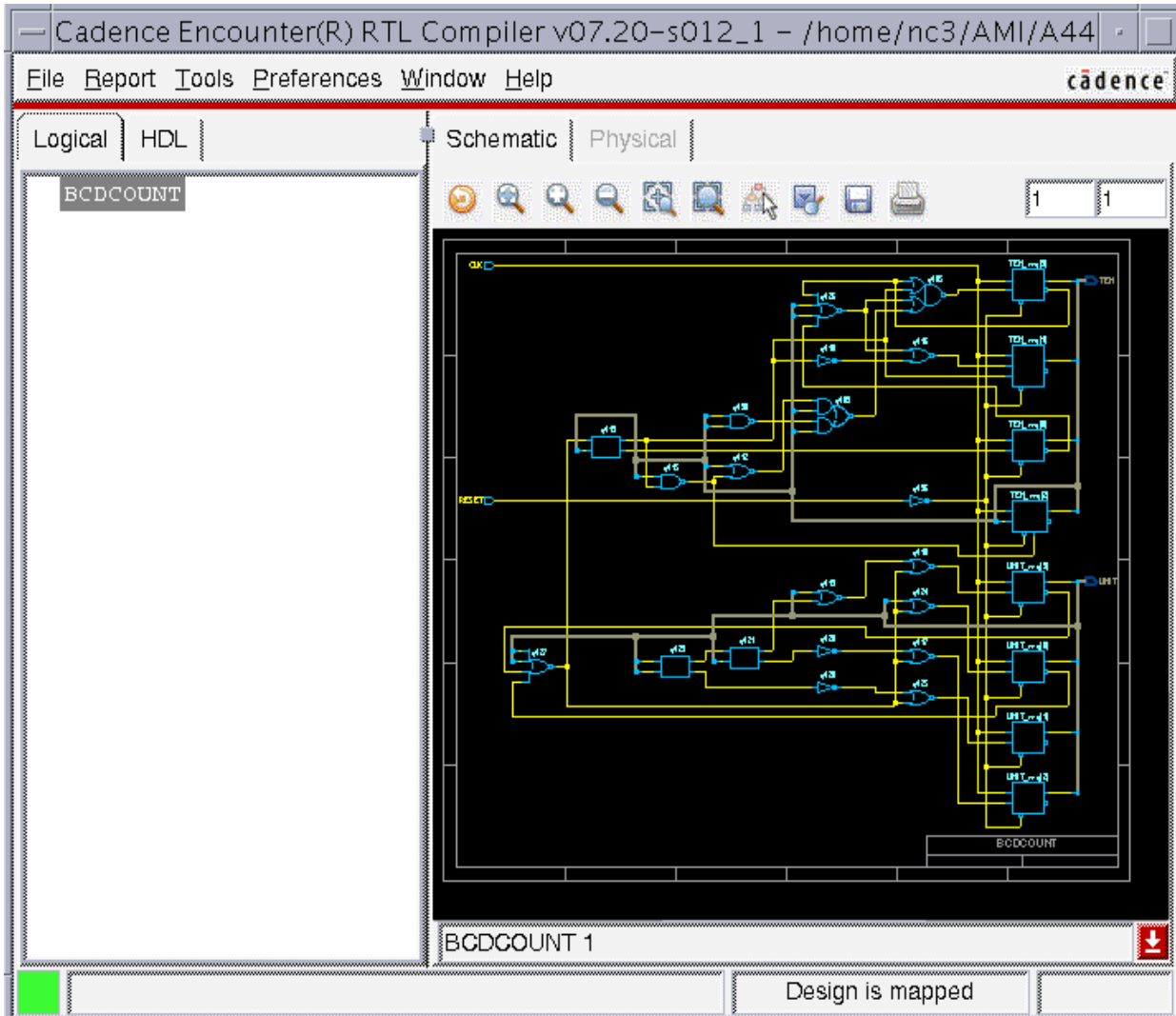


Fig: Synthesis Graphic pane showing Post "Synthesis" structure

5.4 Generating the Output Netlist

Whilst it can read both VHDL and Verilog RTL code. Cadences design flow from this point onwards is fixed on verilog. We need to write the output netlist. Then for this design flow we are going to import this as a schematic into the Cadence schematic environment.

To generate the output netlist we need to use the following command

- `write_hdl > bcdcount_gate.v`

Be careful of the “_” and “>” entries and try to remember the spaces correctly!

If successful this will generate a verilog version of the synthesised netlist.

5.5 Exiting the Synthesis tool

We can now exit the synthesis tool either use File->Exit on the synthesis gui or type exit at the “rc:/>” prompt

5.5.1 Checking the verilog file

We can use the now released terminal window to open the bcdcount_gate.v file to examine its contents. Use the command:

```
ncedit bcdcount_gate.v
```

- Close the file when you have finished.

5.6 That was easy

Well yes, we used defaults for every stage, added no timing constraints and were not doing any test path insertions. In addition we did no optimisation or any of a plethora of other options. It is also a somewhat simple design. Synthesis and in particular Physically Knowledagable Synthesis (PKS) , timing closure and analysis are significant parts of the actual work loads in producing modern DSM designs, cf, Magma, Cadence PKS, Synopsys. As is generating test benches for the code cf Open Vera.

5.7 Why is everything text based? or Why is the graphics tool so dumb?

You really only drive synthesis tools for a complex multi- level design using command files and structures. There are so many parameters, variables and files that need setting and configuring, even if we ignore issues associated with timing, test path insertion and synthesis directives that direct driving and a whole host of other activities graphical input is really on feasible for simple basic designs with little configuration.

In modern systems part of the normal design flow is to do scan/test path insertion as part of the synthesis operation.

6 Importing and converting the synthesised design into the Schematic Environment

If you do not still have the schematic environment open then return to, or open a new terminal

Entry and Simulation of Standard Library Components

window using transport, the Walkthroughs area, and reinvoke the tool.

Return to the Walkthroughs directory and start a Cadence design kit using the command

- amiselect fb

We now need to “import” the verilog design into the schematic environment in such a way that it creates a gate level schematic. To do this we need to do the following actions:

From the Cadence CIW windows select the command File -> Import -> Verilog this will invoke the Verilog In Form.

File -> Import -> Verilog

The form will resemble the somewhat the one shown below.

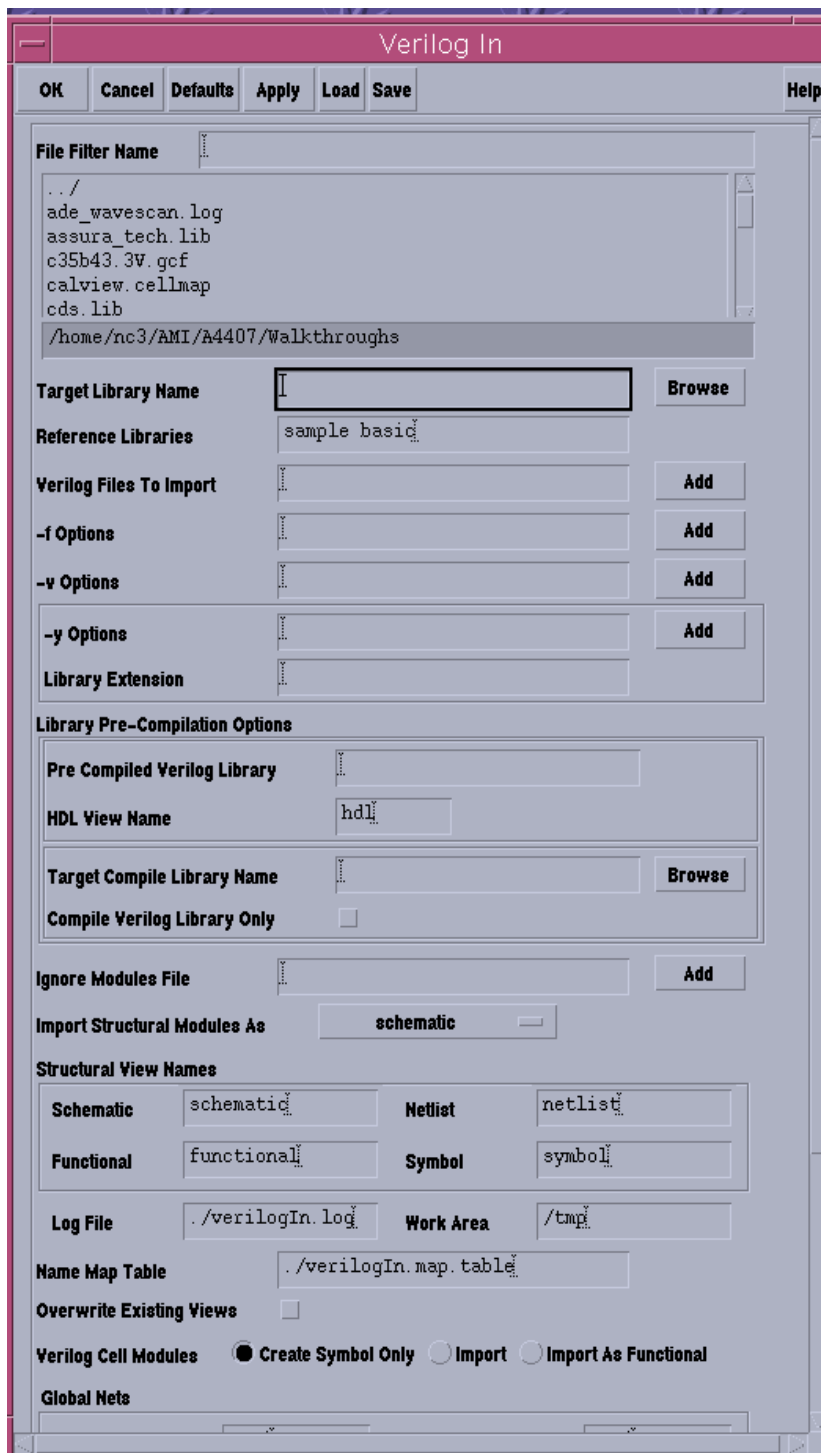


Fig: Unconfigured Verilog In Form

We need to make the following changes to the form

6.0.1 Loading the Configuration file

There is a configuration file automatically added to your design area that will configure the majority of the Verilog In form for you. Select the “Load” button on the top of the form to invoke the Load Parameters form. Type the following entry into the form

- **verilogin.conf**

Then “OK” the form. This will configure the form for the required function.

6.0.2 Target Library Name

This needs to be changed to the name of the library we used for our previous simulation exercise. In my case this was “examples”. Now we can type this in directly or we can use the “Browse” button on the right hand side of the Target Library Name field to invoke the library browser to allow us to select the library. It will be set to VERILOG_IN by the configuration file.

- Target Library Name

Hint: If you get this wrong it will create a new library of this name. There are issues with creating libraries with no techlib that mean the view may be correct but show as blank. Attaching a tech file or copying the schematic to a library with an attached techfile should resolve the problem. Alternatively correct the name and run the exercise again.

6.0.3 Verilog Files to Import

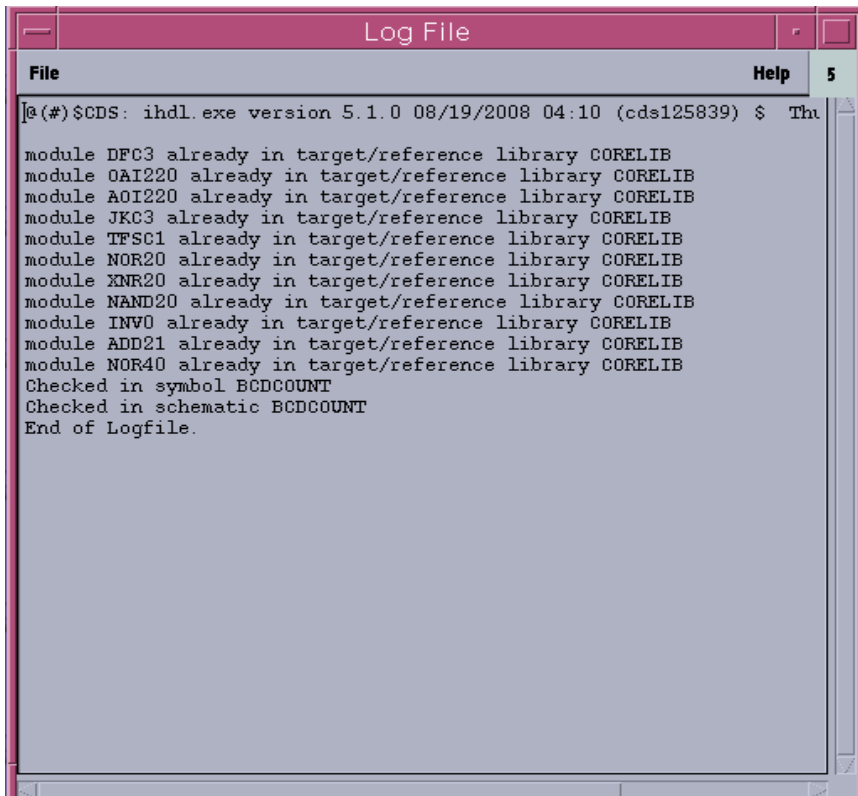
This needs to be set to the name and location of the bcdcount_gate.v file. We can use the browser field at the top to help us here. Use the browser to select the “vhdl_exercises” directory and then the “bcdcount_gate.v” file. Then use the “Add” button on the right hand side of the “Verilog Files To Import” field. This will add the file and path required. Alternatively the path you need to use, assuming you have followed my conventions would be:

- ./vhdl_exercises/bcdcount_gate.v

Importing the Design into the Cadence Library

Now all we need to do is select the “OK” function on the Verilog In form.

The log file for a successful import should resemble this:



```
File Help 5
[!(#)$CDS: ihdl.exe version 5.1.0 08/19/2008 04:10 (cds125839) $ Thu
module DFC3 already in target/reference library CORELIB
module OAI220 already in target/reference library CORELIB
module AOI220 already in target/reference library CORELIB
module JKC3 already in target/reference library CORELIB
module TFSC1 already in target/reference library CORELIB
module NOR20 already in target/reference library CORELIB
module XNR20 already in target/reference library CORELIB
module NAND20 already in target/reference library CORELIB
module INV0 already in target/reference library CORELIB
module ADD21 already in target/reference library CORELIB
module NOR40 already in target/reference library CORELIB
Checked in symbol BCDCOUNT
Checked in schematic BCDCOUNT
End of Logfile.
```

Fig: Successful import of the bcdcount into examples library

6.1 Checking in Cadence

We can now use the Cadence Library manager, CIW File -> Library Manager, if not already open. To look in our library, in my case “examples” and to see the imported design by opening the BCDCOUNT schematic. You will notice that both a schematic and a symbol have been created. Hence we can use this design on a schematic or within another structure if we should wish.

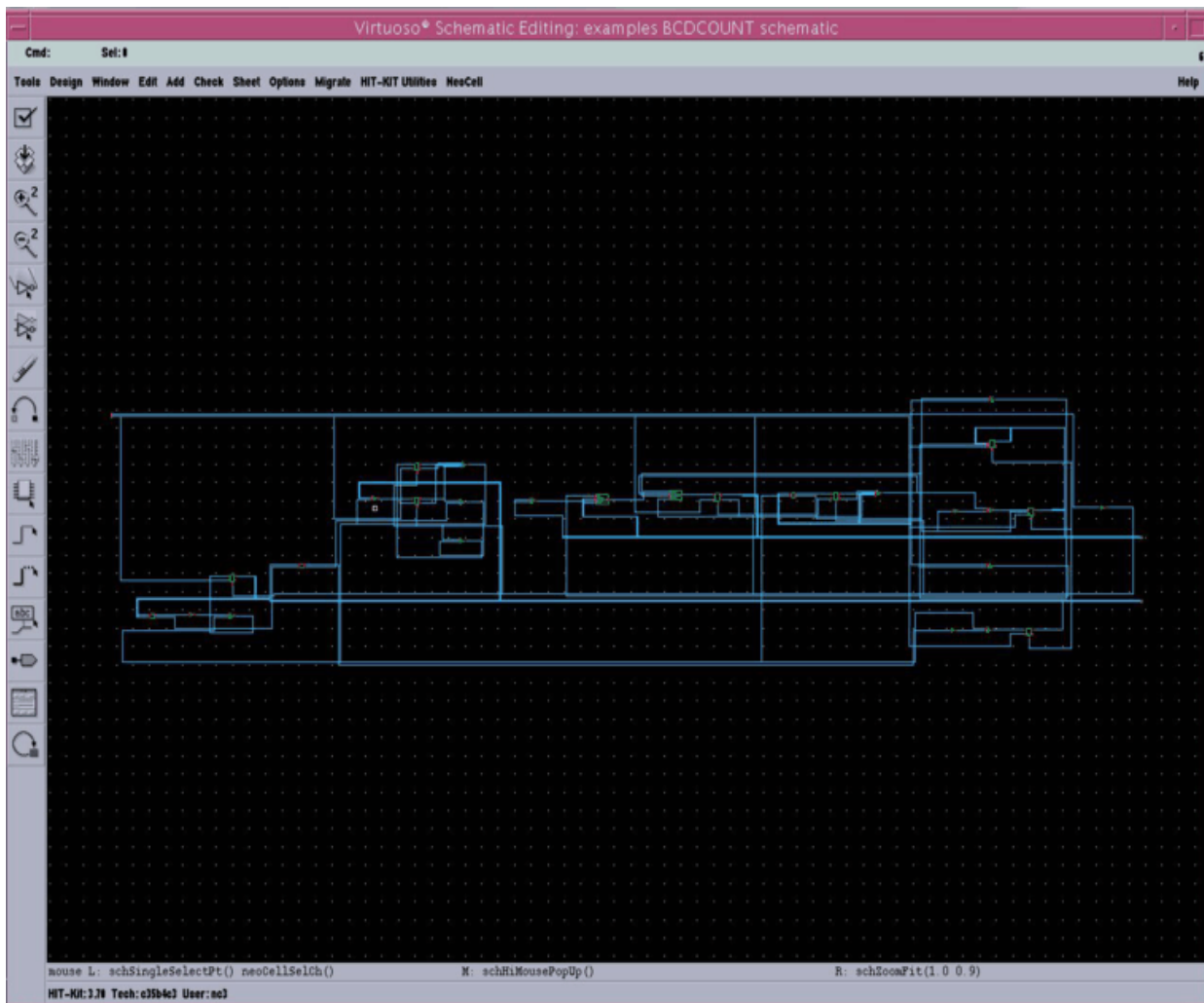


Fig: Imported BCDCOUNT from `bcdcount_gate.v` synthesised file

7 Simulating the design in Verilog

We have already covered the stages required to run a successful Verilog simulation of a design in Walkthrough 1. We now need to follow these stages to allow us to test the BCDCOUNT schematic.

Although you should refer back to the previous walkthrough. The following is the general flow for this part of the exercise.

Schematic Tools -> Simulation -> NCVerilog

To invoke the Virtuoso Verilog Environment for NC-Verilog

- Verilog Environment Commands -> Initialise Design
- Verilog Environment Commands -> Netlist

To Invoked the Editing form

Entry and Simulation of Standard Library Components

- Verilog Environment Commands -> Edit Test Fixture

Using the Edit Test Fixture Form

- Edit Test Fixture File Type -> Test Bench button
- Edit Test Fixture Select -> Edit button
- Edit Test Fixture Apply button

Edit the testfixture.template document to change the

- ``include "testfixture.verilog"`

to be

- ``include "my_testfixture.verilog"`

Save the file and close the nedit window.

Using the Edit Test Fixture Form

- Edit Test Fixture File Type -> Stimulus button
- Edit Test Fixture Select -> Edit button
- Edit Test Fixture Apply button

Edit the testfixture.template document to save the file as my_testfixture.verilog.

The current nedit window should update so the top header now reads my_testfixture.verilog. Edit this file to add in the requisite patterns for the CLK and RESET signals listed. Save the file, you may/may not want to close the nedit until you are certain of the syntax and function.

Hints:

RESET is high active, and any integer i.e. ns clock rate will be fine.

The result I got for my simulation are shown below

Entry and Simulation of Standard Library Components

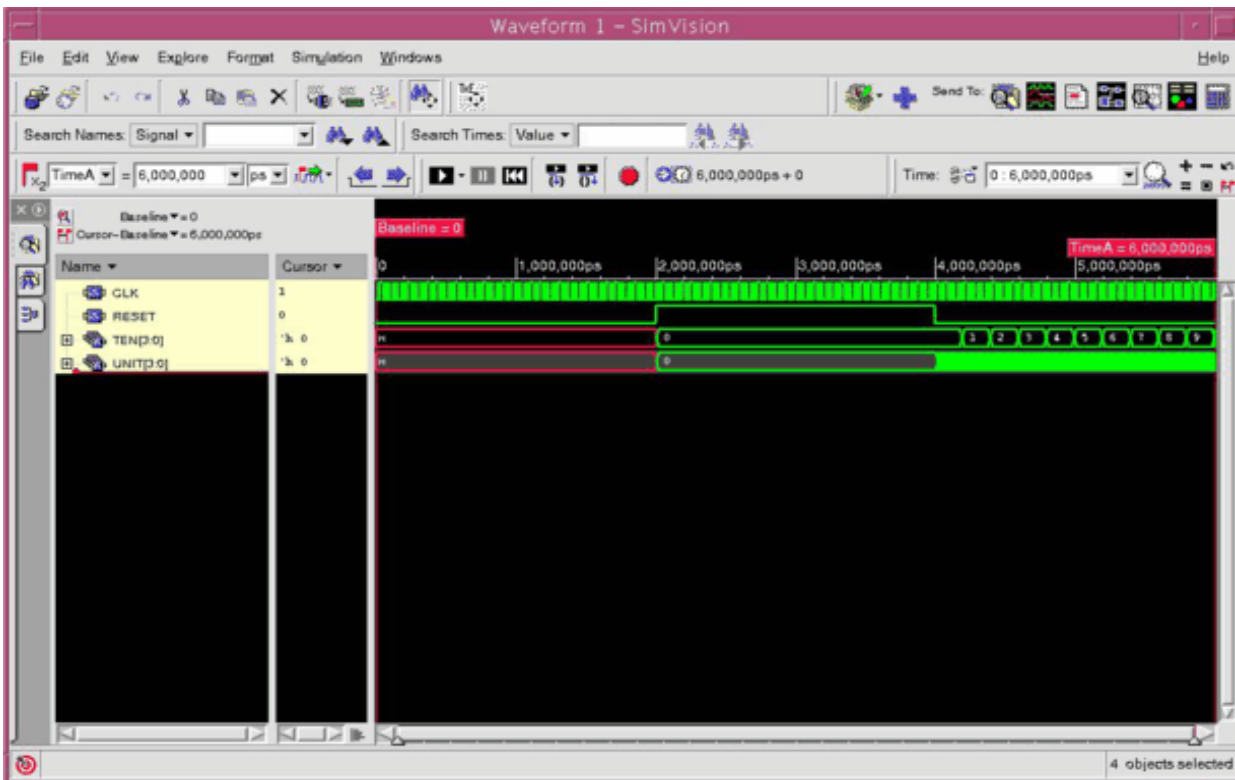


Fig: Simulation results for BCDCount