CPE/EE 427, CPE 527, VLSI Design I: Tutorial #5, Standard cell design flow (from vhdl to layout, mu0 processor)

Joel Wilder and Aleksandar Milenkovic, ECE Dept., The University of Alabama in Huntsville

1. INTRODUCTION

This tutorial steps you through the process of taking a vhdl design, simulating it using NCLaunch, synthesizing it using Build Gates, re-simulating the gate-level netlist, and then performing auto place-and-route to achieve a finished ASIC. Pads can also be added to your design as shown in previous tutorials. Thus, the vhdl-to-ASIC work flow is illustrated.

You will perform this work based on the 0.5um AMI nwell process (lambda = 0.30um).

2. PREPARE THE CADENCE TOOLS

From your home directory, change directories into your cadence working directory:

% cd cadence

Make a directory for lab5 and change into that directory:

% mkdir lab5

% cd lab5

3. VHDL SIMULATION USING NCLAUNCH

First, download the vhdl files you will use for this tutorial:

- mu0.vhd -- contains a vhdl description of the mu0 processor

  (to learn more about mu0 processor visit http://www.ece.uah.edu/%7Elacasa/tutorials/mu0/mu0desc_files/frame.htm)
- tb_mu0.vhd -- testbench for mu0 component

Next, start NCLaunch in a terminal window at the unix prompt:

% nclaunch -new

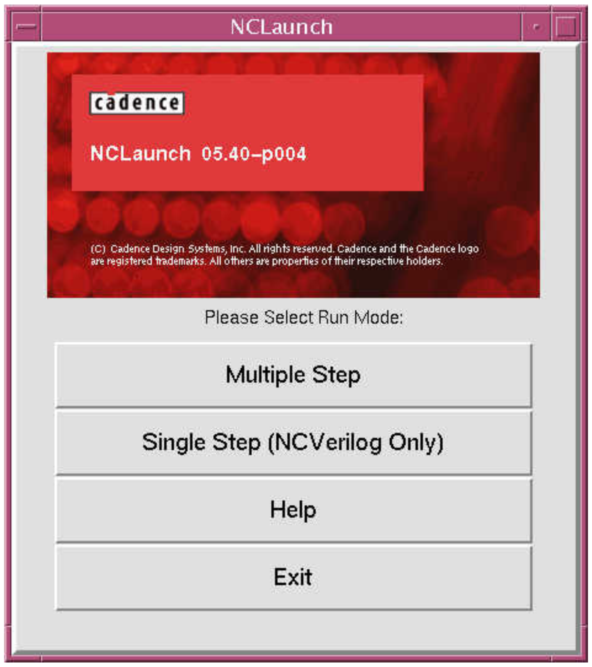
What is NCLaunch?

NCLaunch is a graphical user interface that helps you manage large design projects and lets you configure and launch your Cadence simulation tools.

NCLaunch is integrated into the Cadence Interleaved Native Compiled Architecture (INCA) and is a component of the SimVision analysis environment.

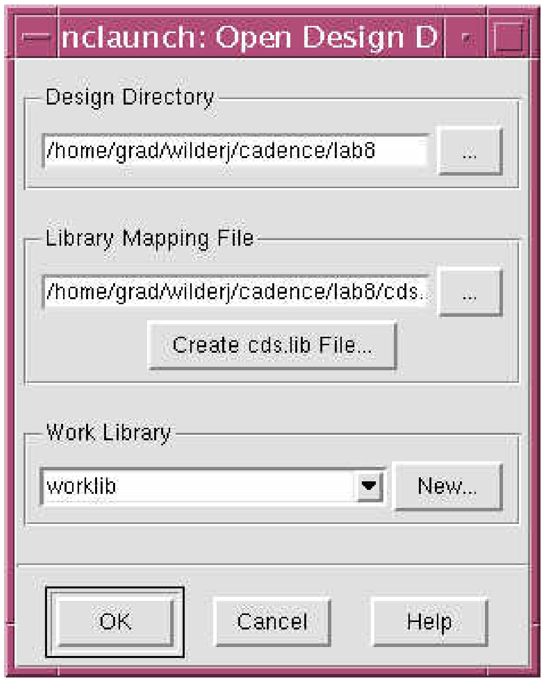
Want to learn more? Read the *NCLaunch User Guide*; it is intended for customers who want to simulate Verilog, VHDL, or mixed-language designs using the NCLaunch tool. This manual explains the complete functionality of the tool and gives examples of simulating with NCLaunch. In addition, it serves as a reference guide for finding specific details on using NCLaunch.

Select **Multiple Step** in the NCLaunch pop-up window:


NCLaunch
cadence
NCLaunch 05.40-p004
(C) Cadence Design Systems, Inc. All rights reserved. Cadence and the Cadence logo are registered trademarks. All others are properties of their respective holders.
Please Select Run Mode:
Multiple Step
Single Step (NCVerilog Only)
Help
Exit

The first step in the process of compiling the design units is to associate them with libraries.

Select **Create cds.lib File…** in the Open Design Window and then select **Save**:

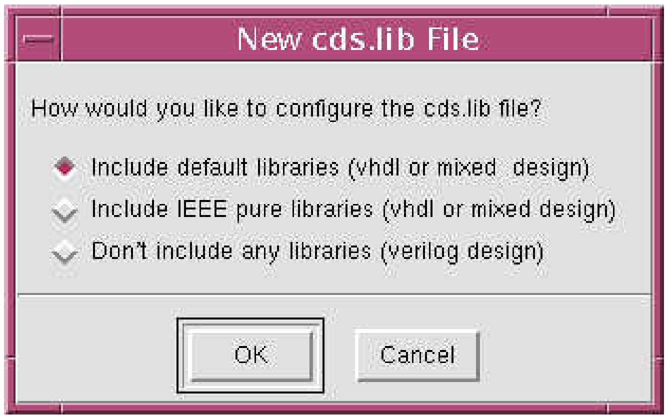


Design Directory: /home/grad/wilderj/cadence/lab8

Library Mapping File: /home/grad/wilderj/cadence/lab8/cds.

Work Library: worklib

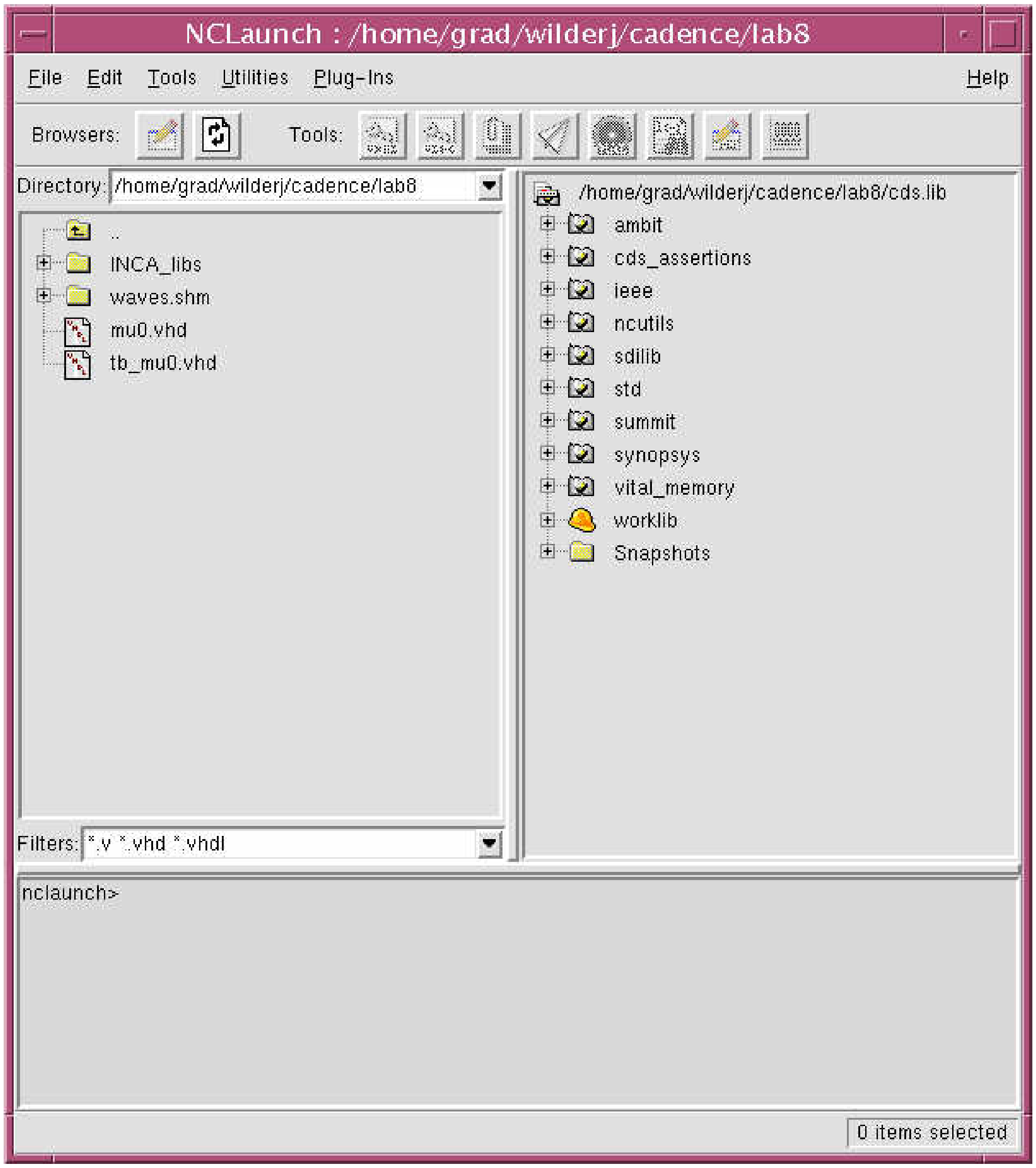
Select **Include default libraries** and press **OK:**



How would you like to configure the cds.lib file?

[✓ Include default libraries (vhdl or mixed design)]
Include IEEE pure libraries (vhdl or mixed design)
Don't include any libraries (verilog design)

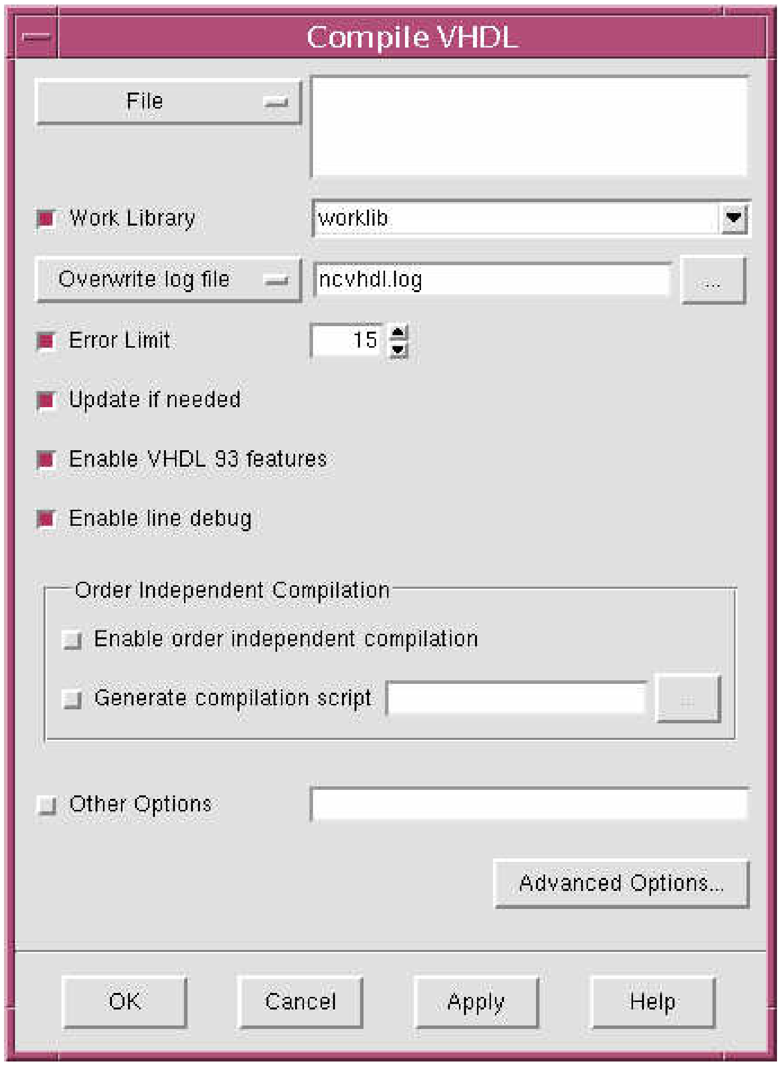
Press **OK** in the Open Design window and you should see:


NCLaunch : /home/grad/wilderj/cadence/lab8
File Edit Tools Utilities Plug-Ins Help
Browsers:
Tools:
Directory: /home/grad/wilderj/cadence/lab8
..
INCA_libs
waves.shm
mu0.vhd
tb_mu0.vhd
Filters: *.v *.vhd *.vhdl
/home/grad/wilderj/cadence/lab8/cds.lib
ambit
cds_assertions
ieee
ncutils
sdilib
std
summit
synopsys
vital_memory
worklib
Snapshots
0 items selected

```
nclaunch>
```

To perform compilation, you should first configure your compiler.

In Tools->VHDL Compiler, enable **VHDL 93 features**:



Compile VHDL

File

[✓] Work Library: worklib

Overwrite log file: ncvhdl.log

[✓] Error Limit: 15

[✓] Update if needed

[✓] Enable VHDL 93 features

[✓] Enable line debug

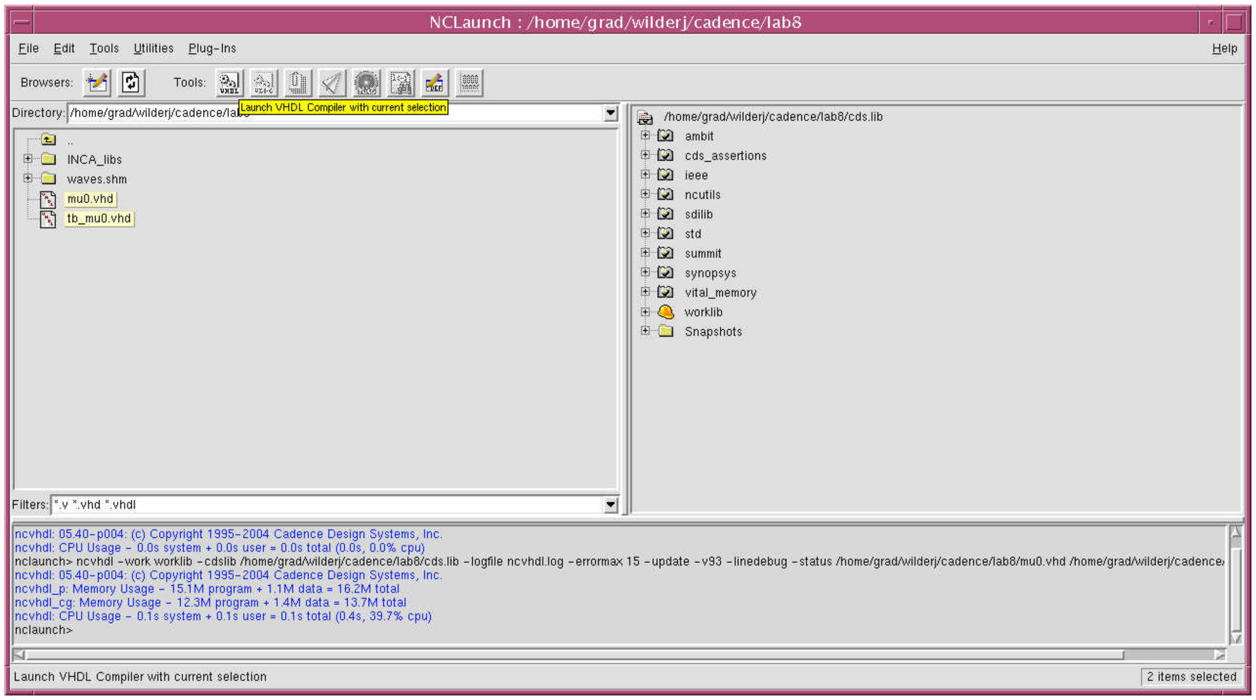
Order Independent Compilation

[ ] Enable order independent compilation

[ ] Generate compilation script

[ ] Other Options

Next, select the design units in the correct order (from the lowest design units to the top level units – i.e., select your testbench file last), and in the NCLaunch toolbar click on the VHDL compile icon (passing over the icon you will see the following text: 'Launch VHDL compiler with current settings'):

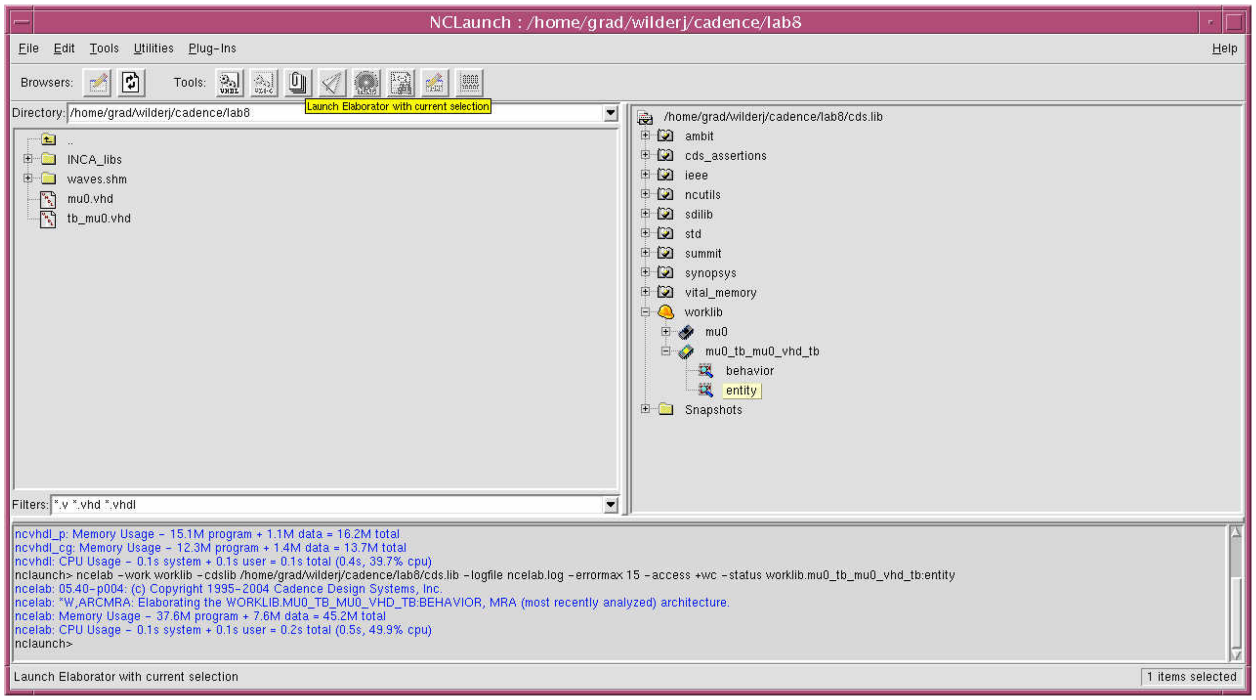

NCLaunch : /home/grad/wilderj/cadence/lab8
File Edit Tools Utilities Plug-Ins Help
Browsers: Tools:
Directory: /home/grad/wilderj/cadence/lab8
Launch VHDL Compiler with current selection
..
INCA_libs
waves.shm
mu0.vhd
tb_mu0.vhd
/home/grad/wilderj/cadence/lab8/cds.lib
ambit
cds_assertions
ieee
ncutils
sdilib
std
summit
synopsys
vital_memory
worklib
Snapshots
Filters: *.v *.vhd *.vhdl

```
ncvhdl: 05.40-p004: (c) Copyright 1995-2004 Cadence Design Systems, Inc.
ncvhdl: CPU Usage - 0.0s system + 0.0s user = 0.0s total (0.0s, 0.0% cpu)
nclaunch> ncvhdl -work worklib -cdslib /home/grad/wilderj/cadence/lab8/cds.lib -logfile ncvhdl.log -errormax 15 -update -v93 -linedebug -status /home/grad/wilderj/cadence/lab8/mu0.vhd /home/grad/wilderj/cadence/
ncvhdl: 05.40-p004: (c) Copyright 1995-2004 Cadence Design Systems, Inc.
ncvhdl_p: Memory Usage - 15.1M program + 1.1M data = 16.2M total
ncvhdl_cg: Memory Usage - 12.3M program + 1.4M data = 13.7M total
ncvhdl: CPU Usage - 0.1s system + 0.1s user = 0.1s total (0.4s, 39.7% cpu)
nclaunch>
```

Launch VHDL Compiler with current selection   2 items selected

The status should indicate the successful completion of the compilation process.

To perform Elaboration, expand worklib and the design units in it. Select the top level design unit (usually testbench, in our case tb_mu0) and select its entity:


NCLaunch : /home/grad/wilderj/cadence/lab8
File Edit Tools Utilities Plug-Ins Help
Browsers: Tools:
Directory: /home/grad/wilderj/cadence/lab8
Launch Elaborator with current selection
..
INCA_libs
waves.shm
mu0.vhd
tb_mu0.vhd
/home/grad/wilderj/cadence/lab8/cds.lib
ambit
cds_assertions
ieee
ncutils
sdilib
std
summit
synopsys
vital_memory
worklib
mu0
mu0_tb_mu0_vhd_tb
behavior
entity
Snapshots
Filters: *.v *.vhd *.vhdl

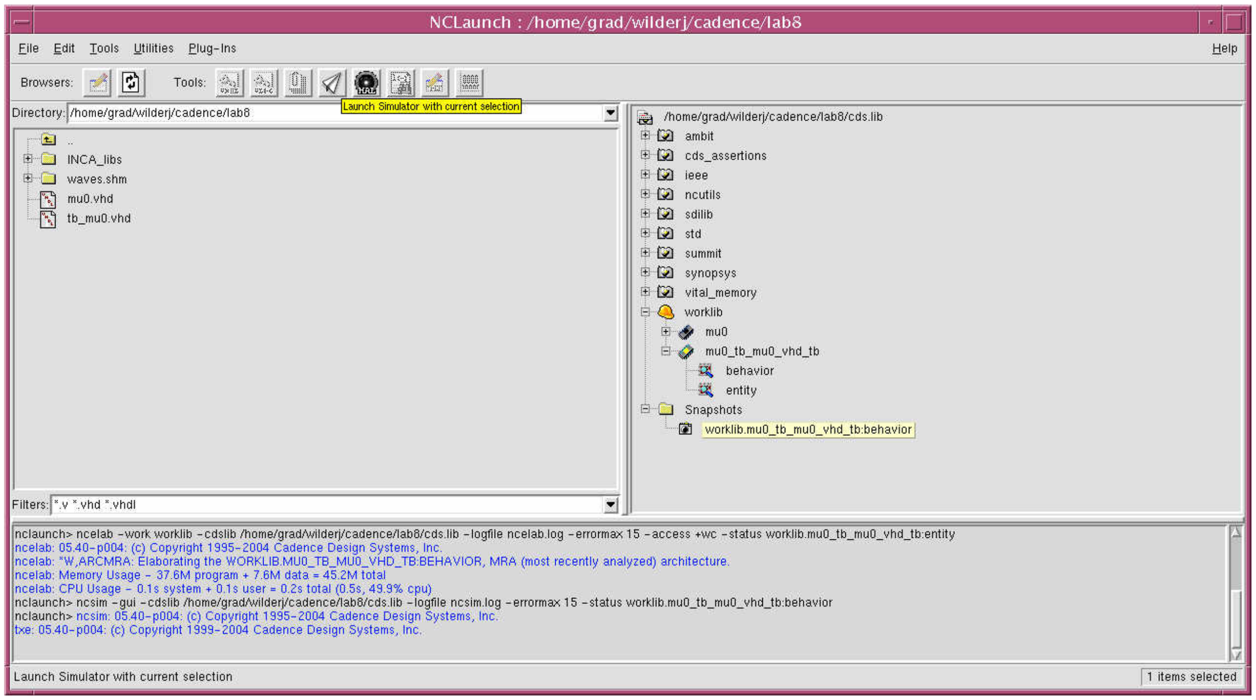
```
ncvhdl_p: Memory Usage - 15.1M program + 1.1M data = 16.2M total
ncvhdl_cg: Memory Usage - 12.3M program + 1.4M data = 13.7M total
ncvhdl: CPU Usage - 0.1s system + 0.1s user = 0.1s total (0.4s, 39.7% cpu)
nclaunch> ncelab -work worklib -cdslib /home/grad/wilderj/cadence/lab8/cds.lib -logfile ncelab.log -errormax 15 -access +wc -status worklib.mu0_tb_mu0_vhd_tb:entity
ncelab: 05.40-p004: (c) Copyright 1995-2004 Cadence Design Systems, Inc.
ncelab: *W,ARCMRA: Elaborating the WORKLIB.MU0_TB_MU0_VHD_TB:BEHAVIOR, MRA (most recently analyzed) architecture.
ncelab: Memory Usage - 37.6M program + 7.6M data = 45.2M total
ncelab: CPU Usage - 0.1s system + 0.1s user = 0.2s total (0.5s, 49.9% cpu)
nclaunch>
```

Launch Elaborator with current selection   1 items selected

Click on the Elaborator icon on the menu.

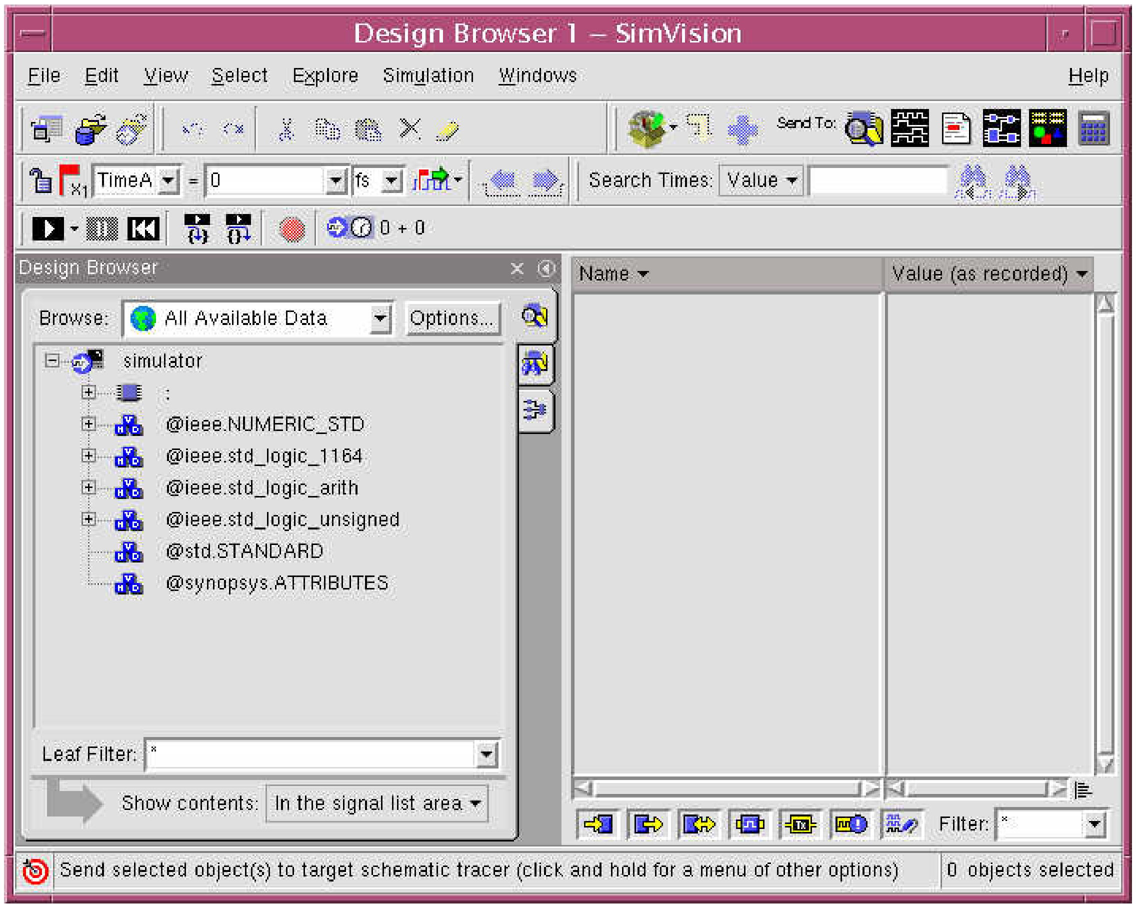
The status should indicate the successful completion of the elaboration phase.

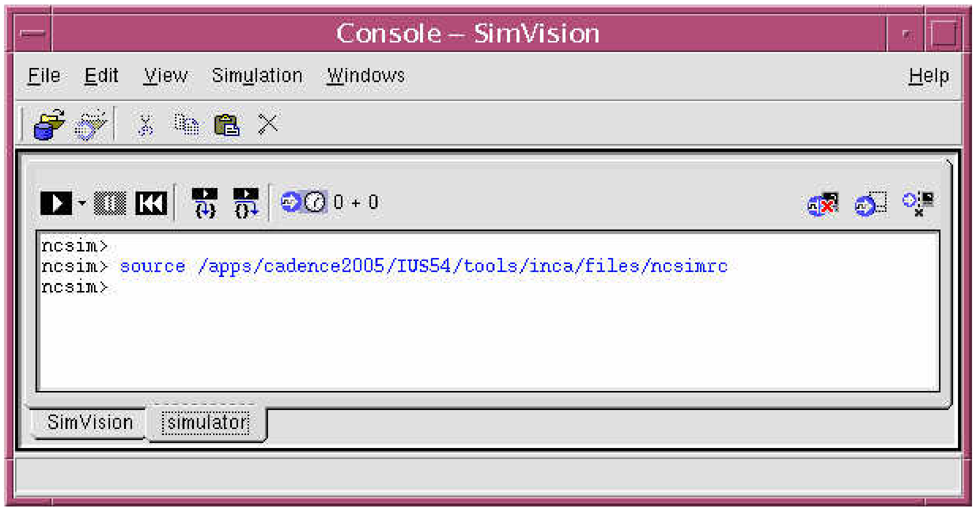
For simulation, expand the Snapshots directory and select the testbench. Then, click on the Simulation icon on the menu to get the simulation environment loaded:


NCLaunch : /home/grad/wilderj/cadence/lab8
File Edit Tools Utilities Plug-Ins Help
Browsers: Tools:
Launch Simulator with current selection
Directory: /home/grad/wilderj/cadence/lab8
..
INCA_libs
waves.shm
mu0.vhd
tb_mu0.vhd
Filters: *.v *.vhd *.vhdl
/home/grad/wilderj/cadence/lab8/cds.lib
ambit
cds_assertions
ieee
ncutils
sdilib
std
summit
synopsys
vital_memory
worklib
mu0
mu0_tb_mu0_vhd_tb
behavior
entity
Snapshots
worklib.mu0_tb_mu0_vhd_tb:behavior
Launch Simulator with current selection
1 items selected

```
nclaunch> ncelab -work worklib -cdslib /home/grad/wilderj/cadence/lab8/cds.lib -logfile ncelab.log -errormax 15 -access +wc -status worklib.mu0_tb_mu0_vhd_tb:entity
ncelab: 05.40-p004: (c) Copyright 1995-2004 Cadence Design Systems, Inc.
ncelab: *W,ARCMRA: Elaborating the WORKLIB.MU0_TB_MU0_VHD_TB:BEHAVIOR, MRA (most recently analyzed) architecture.
ncelab: Memory Usage - 37.6M program + 7.6M data = 45.2M total
ncelab: CPU Usage - 0.1s system + 0.1s user = 0.2s total (0.5s, 49.9% cpu)
nclaunch> ncsim -gui -cdslib /home/grad/wilderj/cadence/lab8/cds.lib -logfile ncsim.log -errormax 15 -status worklib.mu0_tb_mu0_vhd_tb:behavior
nclaunch> ncsim: 05.40-p004: (c) Copyright 1995-2004 Cadence Design Systems, Inc.
txe: 05.40-p004: (c) Copyright 1999-2004 Cadence Design Systems, Inc.
```

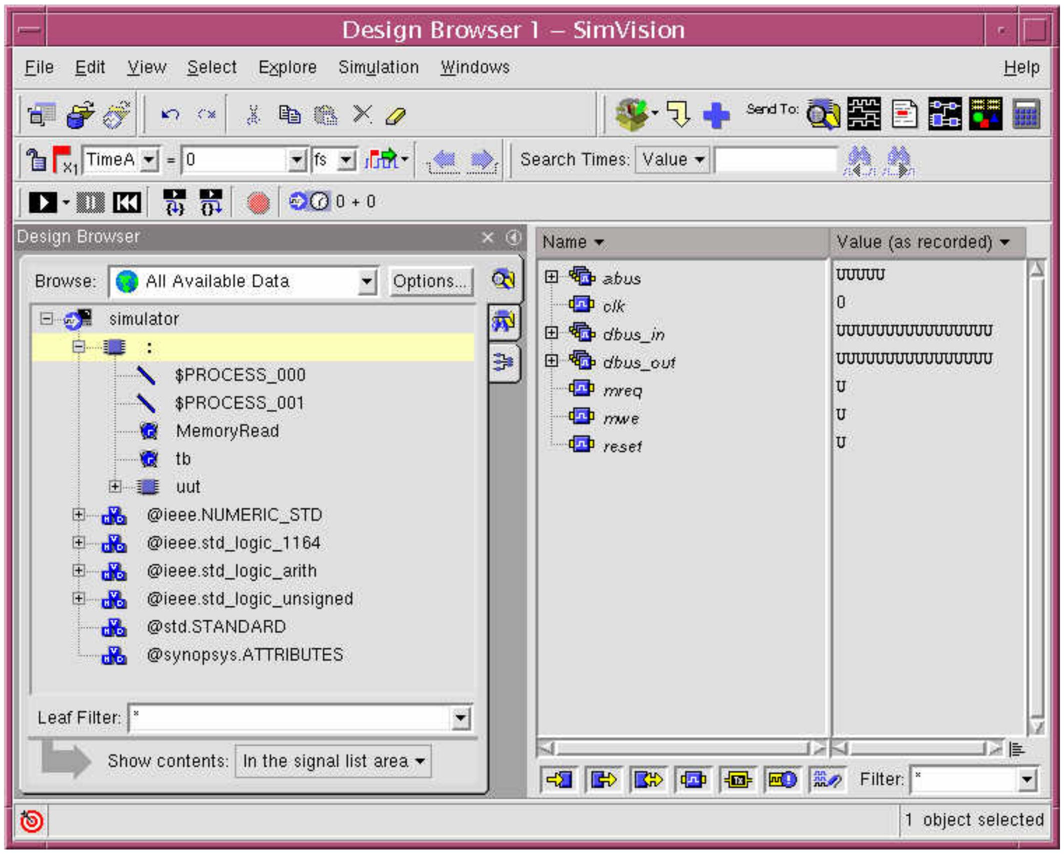
The SimVison console and Design Browser windows will appear:





```
ncsim>
ncsim> source /apps/cadence2005/IUS54/tools/inca/files/ncsimrc
ncsim>
```

In the Design Browser window, select the top level entity:



TimeA = 0 fs

Search Times: Value

0 + 0

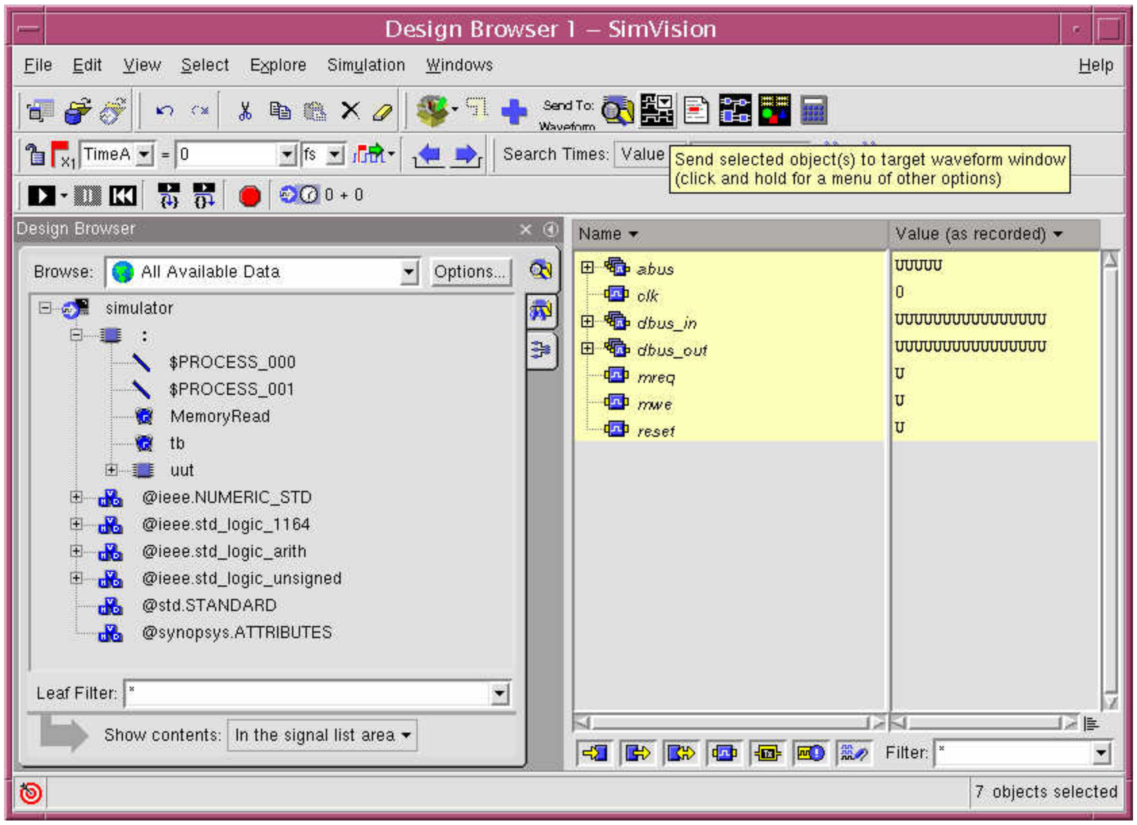
Browse: All Available Data

Leaf Filter: *

Show contents: In the signal list area

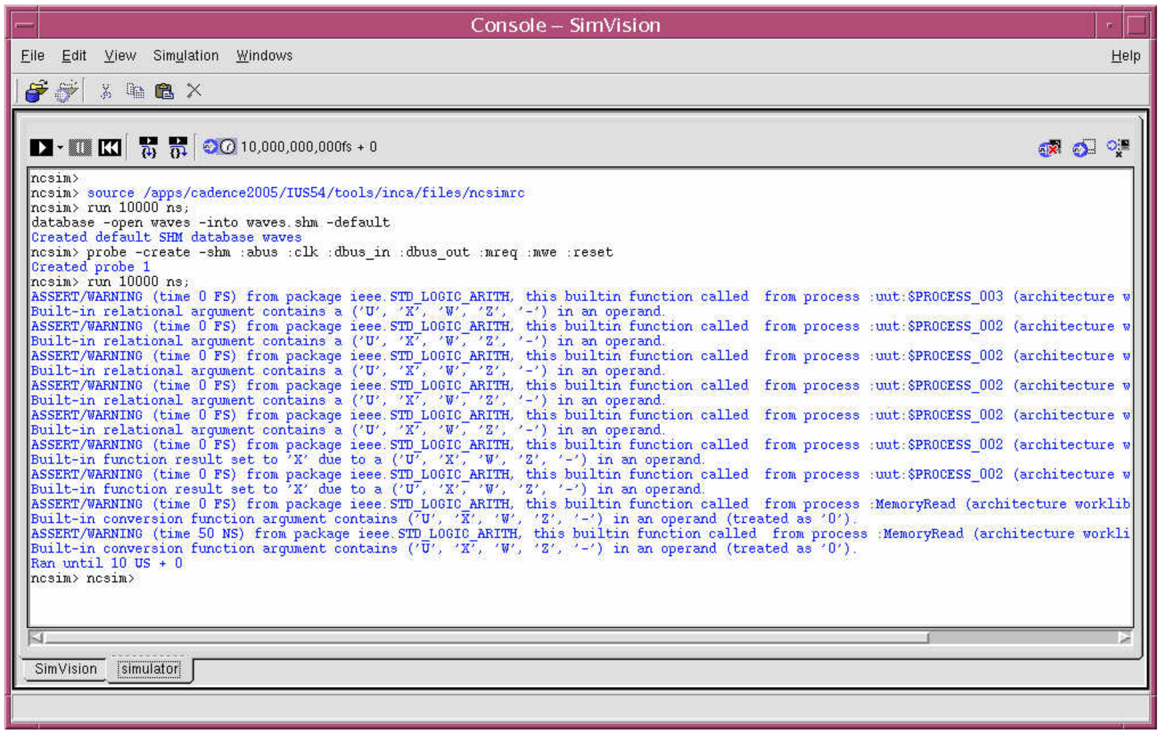
Name	Value (as recorded)
abus	UUUUU
clk	0
dbus_in	UUUUUUUUUUUUUUUU
dbus_out	UUUUUUUUUUUUUUUU
mreq	U
mwe	U
reset	U

Select the signals you want to inspect:

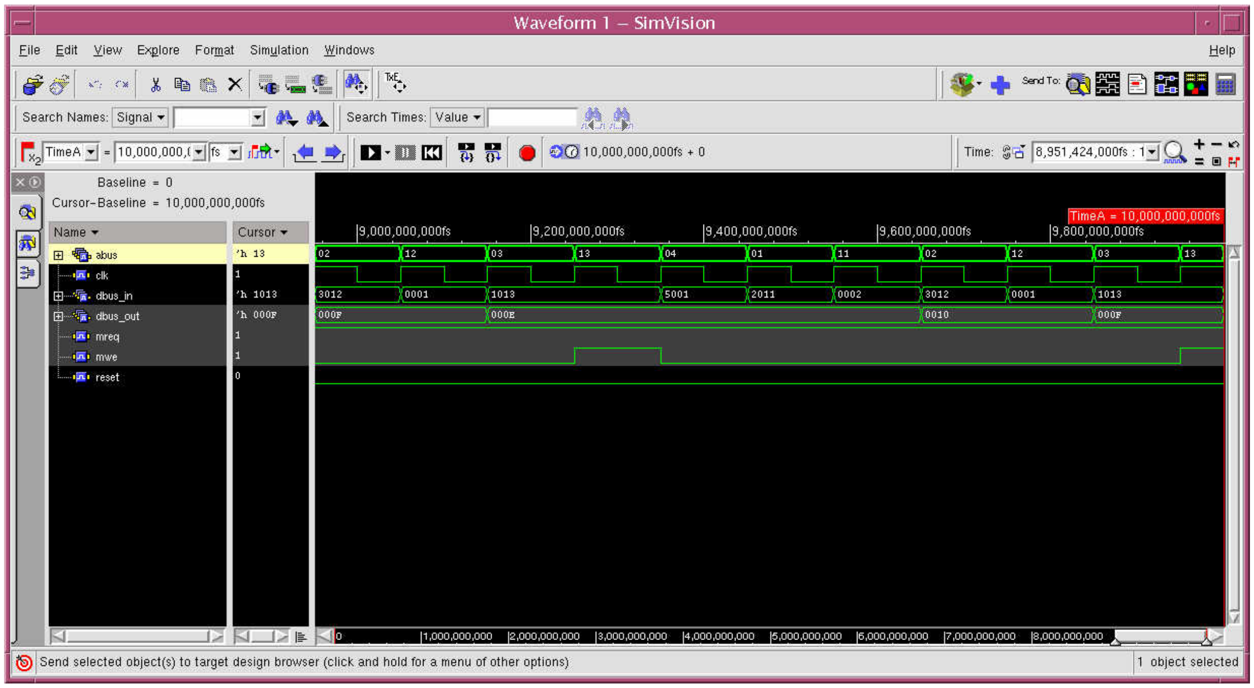


Name	Value (as recorded)
abus	UUUUU
clk	0
dbus_in	UUUUUUUUUUUUUUUU
dbus_out	UUUUUUUUUUUUUUUU
mreq	U
mwe	U
reset	U

Click on the 'waveform' icon on the menu to bring up the waveform window. In the SimVison console, type **run 10000 ns;** as shown:



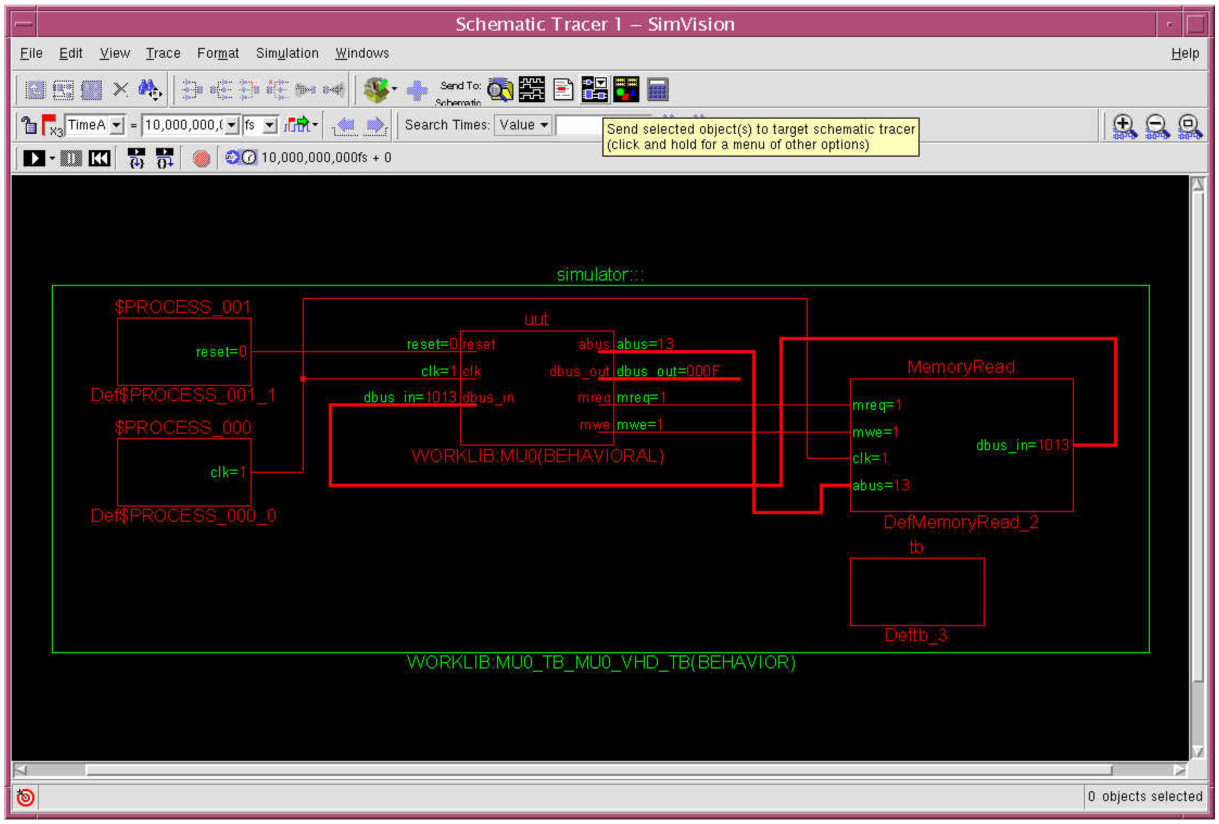
```
ncsim>
ncsim> source /apps/cadence2005/IUS54/tools/inca/files/ncsimrc
ncsim> run 10000 ns;
database -open waves -into waves.shm -default
Created default SHM database waves
ncsim> probe -create -shm :abus :clk :dbus_in :dbus_out :mreq :mwe :reset
Created probe 1
ncsim> run 10000 ns;
ASSERT/WARNING (time 0 FS) from package ieee.STD_LOGIC_ARITH, this builtin function called  from process :uut:$PROCESS_003 (architecture w
Built-in relational argument contains a ('U', 'X', 'W', 'Z', '-') in an operand.
ASSERT/WARNING (time 0 FS) from package ieee.STD_LOGIC_ARITH, this builtin function called  from process :uut:$PROCESS_002 (architecture w
Built-in relational argument contains a ('U', 'X', 'W', 'Z', '-') in an operand.
ASSERT/WARNING (time 0 FS) from package ieee.STD_LOGIC_ARITH, this builtin function called  from process :uut:$PROCESS_002 (architecture w
Built-in relational argument contains a ('U', 'X', 'W', 'Z', '-') in an operand.
ASSERT/WARNING (time 0 FS) from package ieee.STD_LOGIC_ARITH, this builtin function called  from process :uut:$PROCESS_002 (architecture w
Built-in relational argument contains a ('U', 'X', 'W', 'Z', '-') in an operand.
ASSERT/WARNING (time 0 FS) from package ieee.STD_LOGIC_ARITH, this builtin function called  from process :uut:$PROCESS_002 (architecture w
Built-in relational argument contains a ('U', 'X', 'W', 'Z', '-') in an operand.
ASSERT/WARNING (time 0 FS) from package ieee.STD_LOGIC_ARITH, this builtin function called  from process :uut:$PROCESS_002 (architecture w
Built-in function result set to 'X' due to a ('U', 'X', 'W', 'Z', '-') in an operand.
ASSERT/WARNING (time 0 FS) from package ieee.STD_LOGIC_ARITH, this builtin function called  from process :uut:$PROCESS_002 (architecture w
Built-in function result set to 'X' due to a ('U', 'X', 'W', 'Z', '-') in an operand.
ASSERT/WARNING (time 0 FS) from package ieee.STD_LOGIC_ARITH, this builtin function called  from process :MemoryRead (architecture worklib
Built-in conversion function argument contains ('U', 'X', 'W', 'Z', '-') in an operand (treated as '0').
ASSERT/WARNING (time 50 NS) from package ieee.STD_LOGIC_ARITH, this builtin function called  from process :MemoryRead (architecture workli
Built-in conversion function argument contains ('U', 'X', 'W', 'Z', '-') in an operand (treated as '0').
Ran until 10 US + 0
ncsim> ncsim>
```


Waveform 1 – SimVision
Baseline = 0
Cursor-Baseline = 10,000,000,000fs
TimeA = 10,000,000,000fs
abus
clk
dbus_in
dbus_out
mreq
mwe
reset
Send selected object(s) to target design browser (click and hold for a menu of other options)
1 object selected

Inspect the waveforms to ensure the design is working properly.

Other cool options:

Click on the 'schematic' icon on the menu to bring up the schematic tracer:



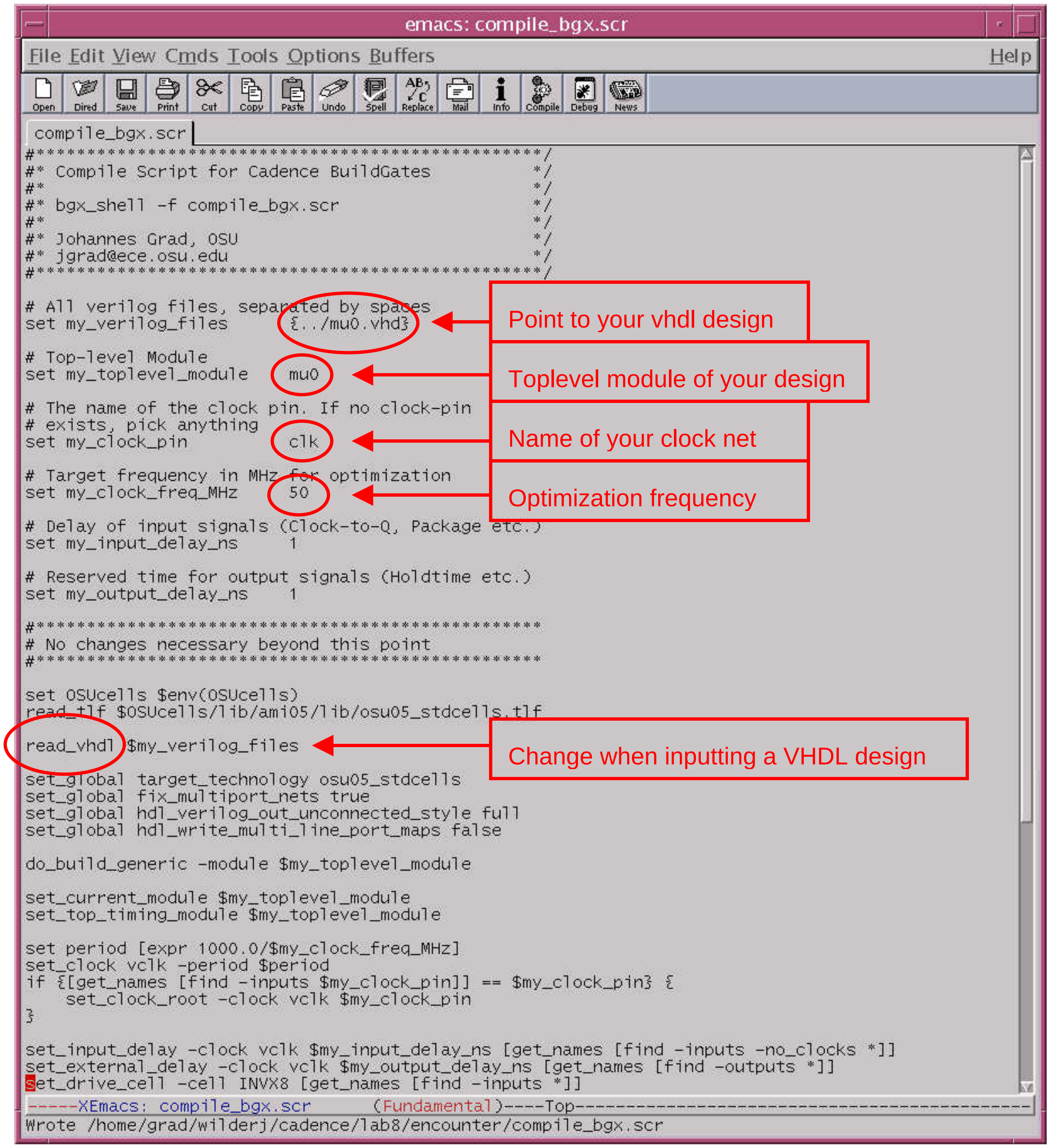
4. VHDL SYNTHESIS TO GATE-LEVEL NETLIST

Once you know your design is working properly through simulation, you can synthesize your vhdl design into a gate-level netlist in a similar fashion as was done for the verilog design work flow.

1. Create an encounter directory and copy in the technology files (for AMI 0.5um):

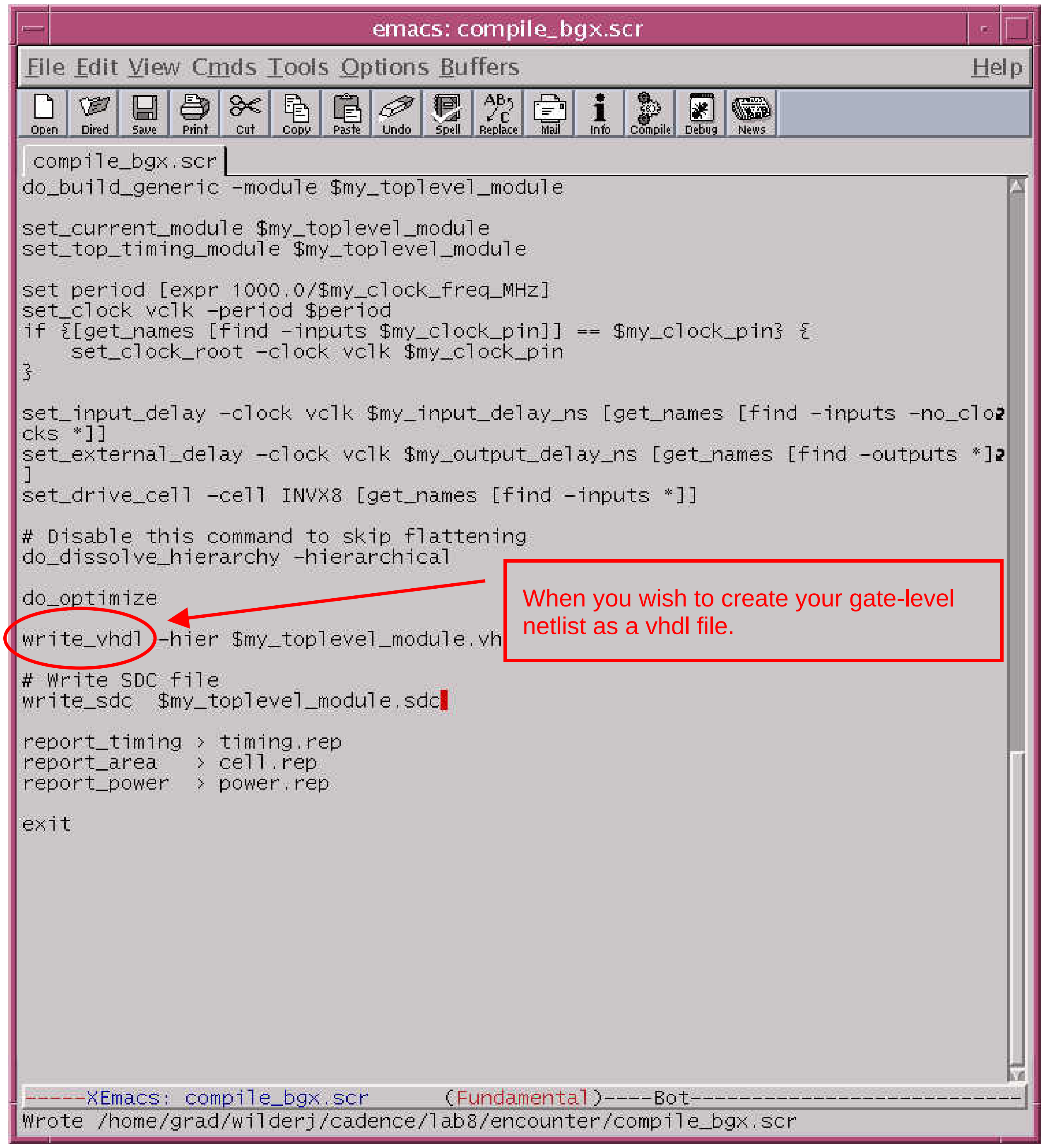
```
% mkdir encounter
% cd encounter
% cp /apps/iit_lib/osu/osu_stdcells/flow/ami05/* .
```

2. Modify the compile_bgx.scr as shown:


emacs: compile_bgx.scr
File Edit View Cmds Tools Options Buffers Help
Open Dired Save Print Cut Copy Paste Undo Spell Replace Mail Info Compile Debug News
compile_bgx.scr
Point to your vhdl design
Toplevel module of your design
Name of your clock net
Optimization frequency
Change when inputting a VHDL design

```
#*************************************************/
#* Compile Script for Cadence BuildGates         */
#*                                               */
#* bgx_shell -f compile_bgx.scr                  */
#*                                               */
#* Johannes Grad, OSU                            */
#* jgrad@ece.osu.edu                             */
#*************************************************/

# All verilog files, separated by spaces
set my_verilog_files    {../mu0.vhd}

# Top-level Module
set my_toplevel_module   mu0

# The name of the clock pin. If no clock-pin
# exists, pick anything
set my_clock_pin         clk

# Target frequency in MHz for optimization
set my_clock_freq_MHz    50

# Delay of input signals (Clock-to-Q, Package etc.)
set my_input_delay_ns     1

# Reserved time for output signals (Holdtime etc.)
set my_output_delay_ns    1

#************************************************
# No changes necessary beyond this point
#************************************************

set OSUcells $env(OSUcells)
read_tlf $OSUcells/lib/ami05/lib/osu05_stdcells.tlf

read_vhdl $my_verilog_files

set_global target_technology osu05_stdcells
set_global fix_multiport_nets true
set_global hdl_verilog_out_unconnected_style full
set_global hdl_write_multi_line_port_maps false

do_build_generic -module $my_toplevel_module

set_current_module $my_toplevel_module
set_top_timing_module $my_toplevel_module

set period [expr 1000.0/$my_clock_freq_MHz]
set_clock vclk -period $period
if {[get_names [find -inputs $my_clock_pin]] == $my_clock_pin} {
    set_clock_root -clock vclk $my_clock_pin
}

set_input_delay -clock vclk $my_input_delay_ns [get_names [find -inputs -no_clocks *]]
set_external_delay -clock vclk $my_output_delay_ns [get_names [find -outputs *]]
set_drive_cell -cell INVX8 [get_names [find -inputs *]]
-----XEmacs: compile_bgx.scr      (Fundamental)----Top------------------------------------------------
Wrote /home/grad/wilderj/cadence/lab8/encounter/compile_bgx.scr
```

2a. If you wish to *output* a vhdl gate-level netlist, modify compile_bgx.scr as shown:


emacs: compile_bgx.scr
File Edit View Cmds Tools Options Buffers Help
Open Dired Save Print Cut Copy Paste Undo Spell Replace Mail Info Compile Debug News
compile_bgx.scr

```
do_build_generic -module $my_toplevel_module

set_current_module $my_toplevel_module
set_top_timing_module $my_toplevel_module

set period [expr 1000.0/$my_clock_freq_MHz]
set_clock vclk -period $period
if {[get_names [find -inputs $my_clock_pin]] == $my_clock_pin} {
    set_clock_root -clock vclk $my_clock_pin
}

set_input_delay -clock vclk $my_input_delay_ns [get_names [find -inputs -no_clo
cks *]]
set_external_delay -clock vclk $my_output_delay_ns [get_names [find -outputs *]
]
set_drive_cell -cell INVX8 [get_names [find -inputs *]]

# Disable this command to skip flattening
do_dissolve_hierarchy -hierarchical

do_optimize

write_vhdl -hier $my_toplevel_module.vh

# Write SDC file
write_sdc  $my_toplevel_module.sdc

report_timing > timing.rep
report_area   > cell.rep
report_power  > power.rep

exit
```

When you wish to create your gate-level netlist as a vhdl file.

```
-----XEmacs: compile_bgx.scr      (Fundamental)----Bot--------------------------
Wrote /home/grad/wilderj/cadence/lab8/encounter/compile_bgx.scr
```

Two things to note: It's nice to create a gate-level netlist in a vhdl format so you can reuse your existing vhdl testbench. However, when you go to do the auto place-and-route, you will need a verilog gate-level netlist, so you will need to modify your compile_bgx.scr script to provide a verilog gate-level netlist also.

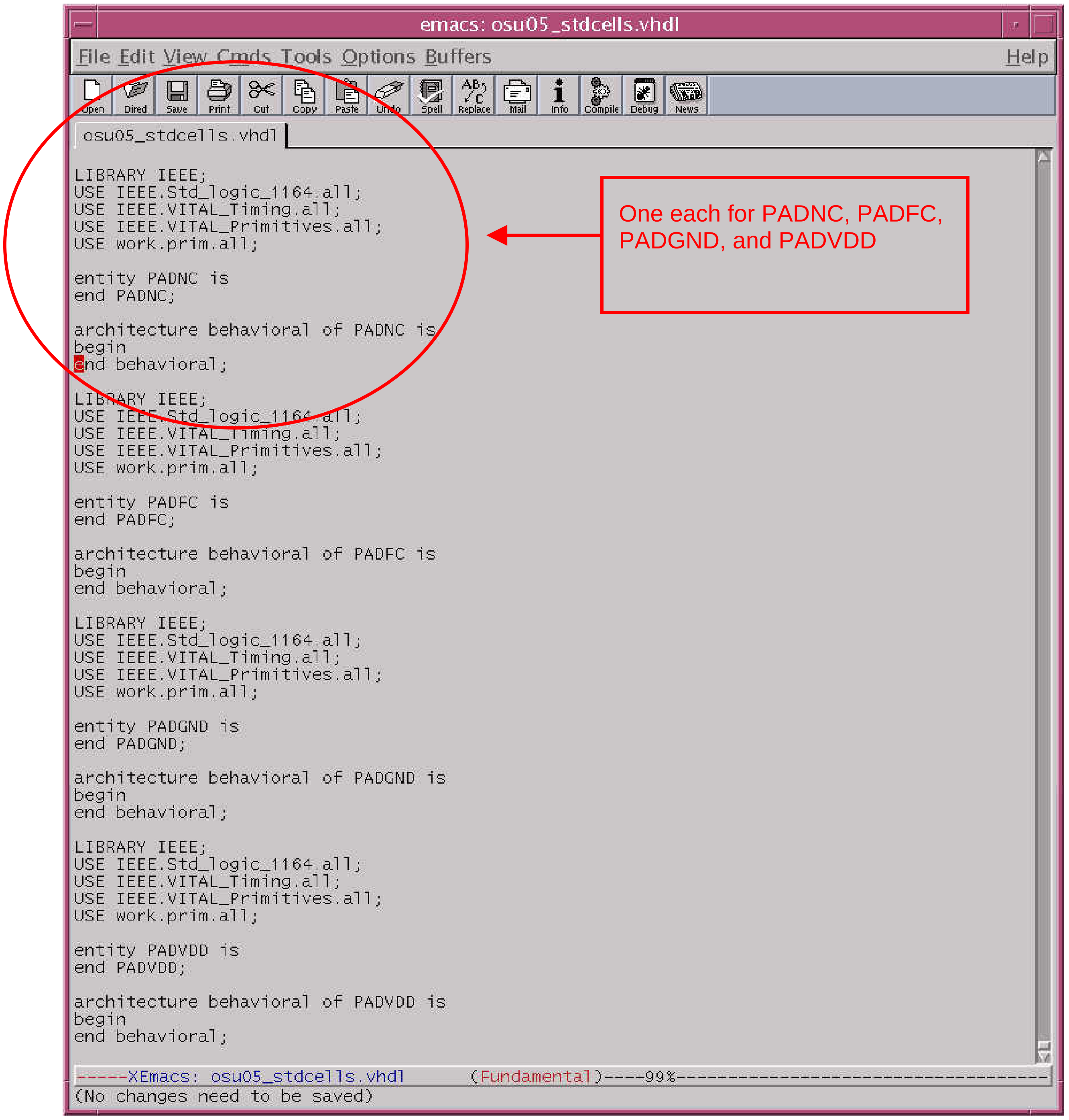
3. pks_shell -f compile_bgx.scr

-mu0.vh file created (gate-level netlist)

-mu0.sdc file created (timing constraints file for encounter)

Pads can be added to the mu0.vh file

3a.For simulation purposes in vhdl, once you have added pads to your design, you will need to modify the osu05_stdcells.vhdl file to fill in missing pads (PADNC, PADFC, PADGND, PADVDD) as shown below:


emacs: osu05_stdcells.vhdl
File Edit View Cmds Tools Options Buffers Help
Open Dired Save Print Cut Copy Paste Undo Spell Replace Mail Info Compile Debug News
osu05_stdcells.vhdl
One each for PADNC, PADFC, PADGND, and PADVDD

```
LIBRARY IEEE;
USE IEEE.Std_logic_1164.all;
USE IEEE.VITAL_Timing.all;
USE IEEE.VITAL_Primitives.all;
USE work.prim.all;

entity PADNC is
end PADNC;

architecture behavioral of PADNC is
begin
end behavioral;

LIBRARY IEEE;
USE IEEE.Std_logic_1164.all;
USE IEEE.VITAL_Timing.all;
USE IEEE.VITAL_Primitives.all;
USE work.prim.all;

entity PADFC is
end PADFC;

architecture behavioral of PADFC is
begin
end behavioral;

LIBRARY IEEE;
USE IEEE.Std_logic_1164.all;
USE IEEE.VITAL_Timing.all;
USE IEEE.VITAL_Primitives.all;
USE work.prim.all;

entity PADGND is
end PADGND;

architecture behavioral of PADGND is
begin
end behavioral;

LIBRARY IEEE;
USE IEEE.Std_logic_1164.all;
USE IEEE.VITAL_Timing.all;
USE IEEE.VITAL_Primitives.all;
USE work.prim.all;

entity PADVDD is
end PADVDD;

architecture behavioral of PADVDD is
begin
end behavioral;
-----XEmacs: osu05_stdcells.vhdl      (Fundamental)----99%-----------------------------------
(No changes need to be saved)
```

3b. Resimulate gate-level netlist (vhdl version, so you can use existing vhdl testbench) to ensure design still operating as you desire. (use above simulation procedure with NCLaunch)

4. **In order to use the existing encounter scripts, they require a verilog gate-level netlist.

This can be easily accomplished by changing the compile_bgx.scr file so that it writes a verilog netlist (as illustrated above in 2a)

Modify encounter.conf and encounter.tcl files as necessary (see previous labs)

5. Run encounter

```
encounter -init encounter.tcl
```

6. Check timing files to ensure slack time is met, perform checks (convert encounter post-route design into icfb schematics/layout), perform power analysis. Make any design changes as necessary. Once specifications have been met, you can send your design to the foundry for fabrication.