Tutorial for VCS

STEP 1: login to the Linux system on Linuxlab server. Start a terminal (the shell prompt).

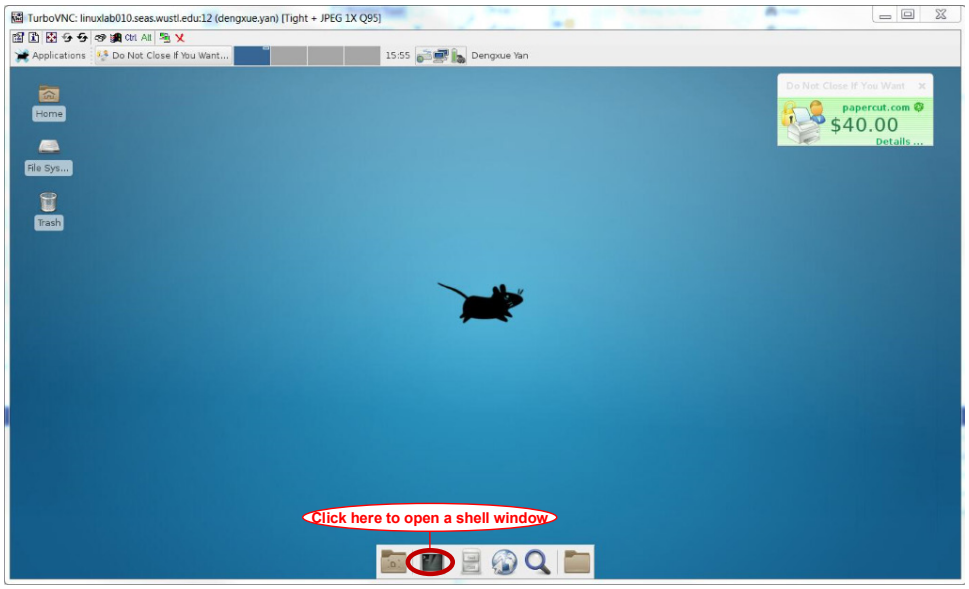

Fig. 1 The screen when you login to the Linuxlab through equeue

STEP 2: In the terminal, execute the following command:

module add ese461

You could perform "*module avail*" in the terminal to find the available modules on Linuxlab. Make sure *ese461* is presented when you execute this command.

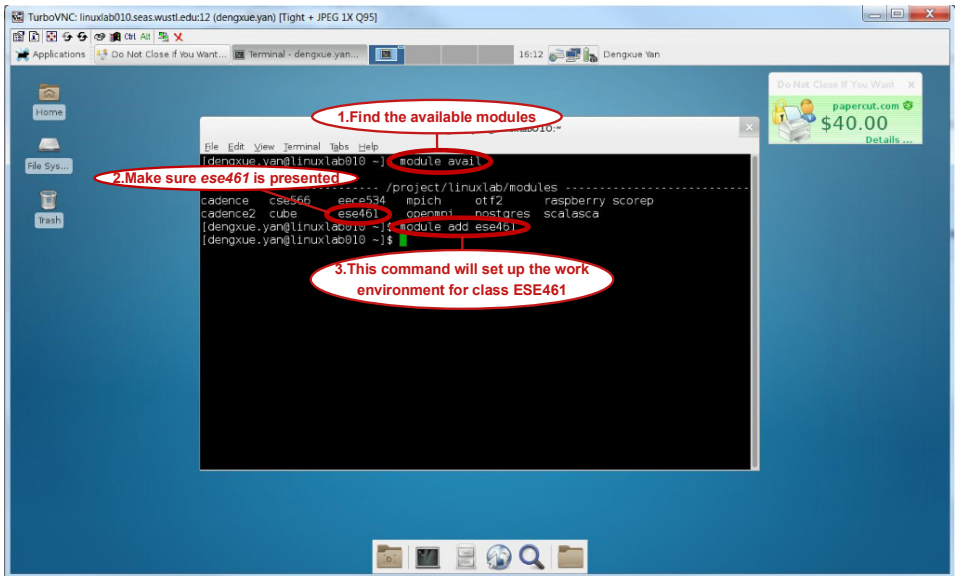

Fig. 2 Build work environment for class ESE461 using module

STEP 3: Getting started with Verilog

- **Creating a new folder (better if you have all the files for a project in a specific folder).**
- **Enter into this new folder and start writing your Verilog script in a new file (.v file). Example code for modeling an counter is here**
- **In addition to model code, Test Bench script has to be given in order to verify the functionality of your model (.v file). Example code of test bench for counter is here.**

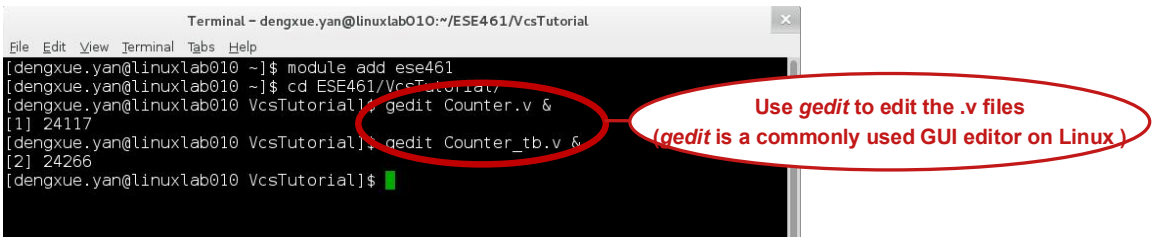

Fig. 3 Open gedit through teminal

STEP 4: Compiling and simulating your code

- **In the terminal, change the directory to where your model and test bench files (Counter.v and Counter_tb.v) are present by using this command:**
  cd <path>
  For example:
  cd ~/ESE461/VcsTutorial/
  (Remark: '~' means home directory on Linux)
- **Compile the files by typing in the terminal:**
  vcs <file>.v <file_tb>.v
  In the above example, it should be:
  vcs Counter.v Counter_tb.v
  There should be no error presented in the terminal. Otherwise you need to check your code and correct them according to the related message. The complier will print out detailed information about your mistakes in the code.

```
[dengxue.yan@linuxlab010 ~]$ cd ~/ESE461/VcsTutorial/
[dengxue.yan@linuxlab010 VcsTutorial]$ vcs Counter.v Counter_tb.v
                         Chronologic VCS (TM)
         Version J-2014.12-SP1-1 -- Fri Sep  9 23:35:54 2016
               Copyright (c) 1991-2014 by Synopsys Inc.
                         ALL RIGHTS RESERVED

This program is proprietary and confidential information of Synopsys Inc.
and may be used and disclosed only as authorized in a license agreement
controlling such use and disclosure.

Parsing design file 'Counter.v'
Parsing design file 'Counter_tb.v'
Top Level Modules:
       Counter_tb
TimeScale is 1 ns / 1 ps
Starting vcs inline pass...
1 module and 0 UDP read.
recompiling module Counter_tb
rm -f _csrc*.so linux_scvhdl_*.so pre_vcsobj_*.so share_vcsobj_*.so
ld -m elf_i386 -shared  -o .//../simv.daidir//_csrc0.so amcQwB.o
rm -f _csrc0.so
if [ -x ../simv ]; then chmod -x ../simv; fi
g++  -o ../simv  -m32 -m32   -Wl,-rpath-link=./ -Wl,-rpath='$ORIGIN'/simv.daidir
/ -Wl,-rpath='$ORIGIN'/simv.daidir//scsim.db.dir    _27906_archive_1.so _csrc0.s
o  SIM_l.o  _csrc0.so     rmapats_mop.o rmapats.o rmar.o  rmar_llvm_0_1.o rmar_l
lvm_0_0.o          /project/linuxlab/synopsys/vcs_mx/linux/lib/libzerosoft_rt_st
ubs.so /project/linuxlab/synopsys/vcs_mx/linux/lib/libvirsim.so /project/linuxla
b/synopsys/vcs_mx/linux/lib/liberrorinf.so /project/linuxlab/synopsys/vcs_mx/lin
ux/lib/libsnpsmalloc.so    /project/linuxlab/synopsys/vcs_mx/linux/lib/libvcsnew
.so /project/linuxlab/synopsys/vcs_mx/linux/lib/libuclinative.so   -Wl,-whole-ar
chive /project/linuxlab/synopsys/vcs_mx/linux/lib/libvcsucli.so -Wl,-no-whole-ar
chive          /project/linuxlab/synopsys/vcs_mx/linux/lib/vcs_save_restore_new.
o /project/linuxlab/synopsys/vcs_mx/linux/lib/ctype-stubs_32.a -ldl  -lc -lm -lp
thread -ldl
 /simv up to date
CPU time: .154 seconds to compile + .255 seconds to elab + .290 seconds to link
```

Successfully compiled

Fig. 4 The result of successfully executing vcs

```
[dengxue.yan@linuxlab010 VcsTutorial]$ vcs Counter.v Counter_tb.v
                         Chronologic VCS (TM)
         Version J-2014.12-SP1-1 -- Fri Sep  9 23:37:02 2016
               Copyright (c) 1991-2014 by Synopsys Inc.

This program is proprietary and confidential information of Synopsys Inc.
and may be used and disclosed only as authorized in a license agreement
controlling such use and disclosure.

The design hasn't changed and need not be recompiled.
If you really want to, delete file simv.daidir/.vcs.timestamp and
run VCS again.
```

Don't need to recompile because nothing changed

Fig. 5 The result of recompile the code when nothing changed

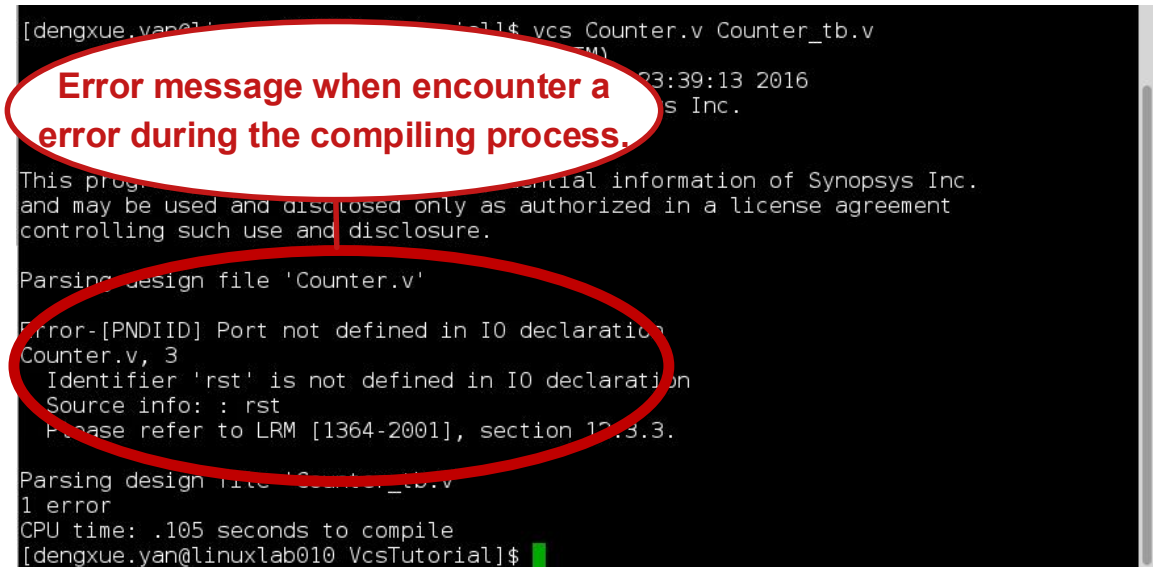

Fig.6 The result of vcs when it thinks there are mistakes in the code

- **A successfully compiling will print out on terminal "*../simv up to date*". And it should generate an executable file named "*simv*" in the same folder where your codes are present.**
- **Then in the terminal run:**
  ./simv
- **After the process finishes, "*VCS Simulation Report*" will be present on the terminal and a file named "*<file>.vcd*" will be generated in the same folder where your codes are present. This is the dump file we specified in the test bench code and we will use it to graphically display the simulation results.**

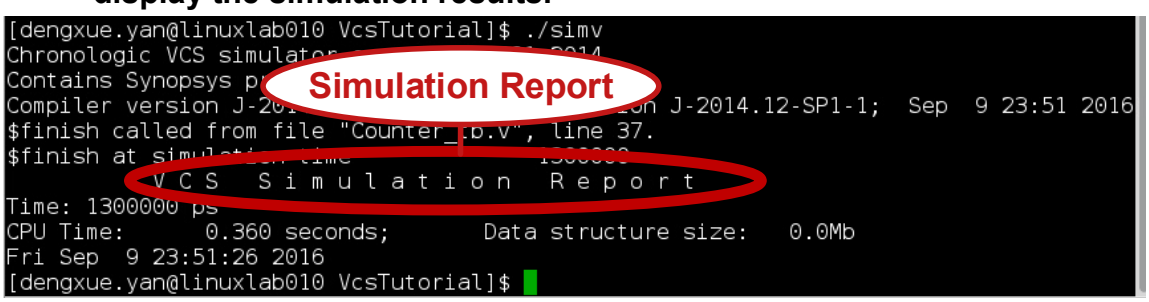

Fig. 7 Simulation Report

STEP 5: Displaying your Results graphically using *dve*

- **After simulation report and "<file>.vcd" is generated, now type the following command in the terminal:**
  dve
  This is a viewer to plot and verify your results.
  (Remark: an *"&"* can be placed behind the command, which means this command will run in background, so the terminal will be released)

```
[dengxue.yan@linuxlab010 VcsTutorial]$ dve &
[1] 1345
[dengxue.yan@linuxlab010 VcsTutorial]$
```

Fig. 8 Start dve on the terminal

- **Go to "*File/Open Database*" and select the "*.vcd*" file from the project folder.**

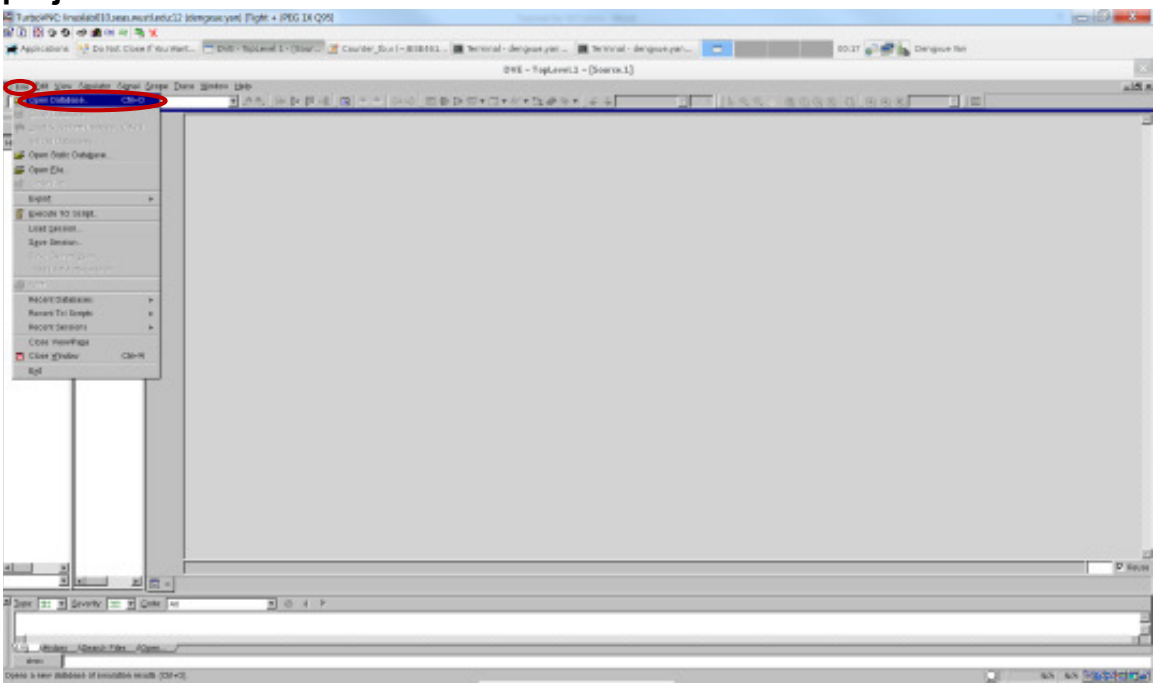

Fig. 9 dve open database (1)

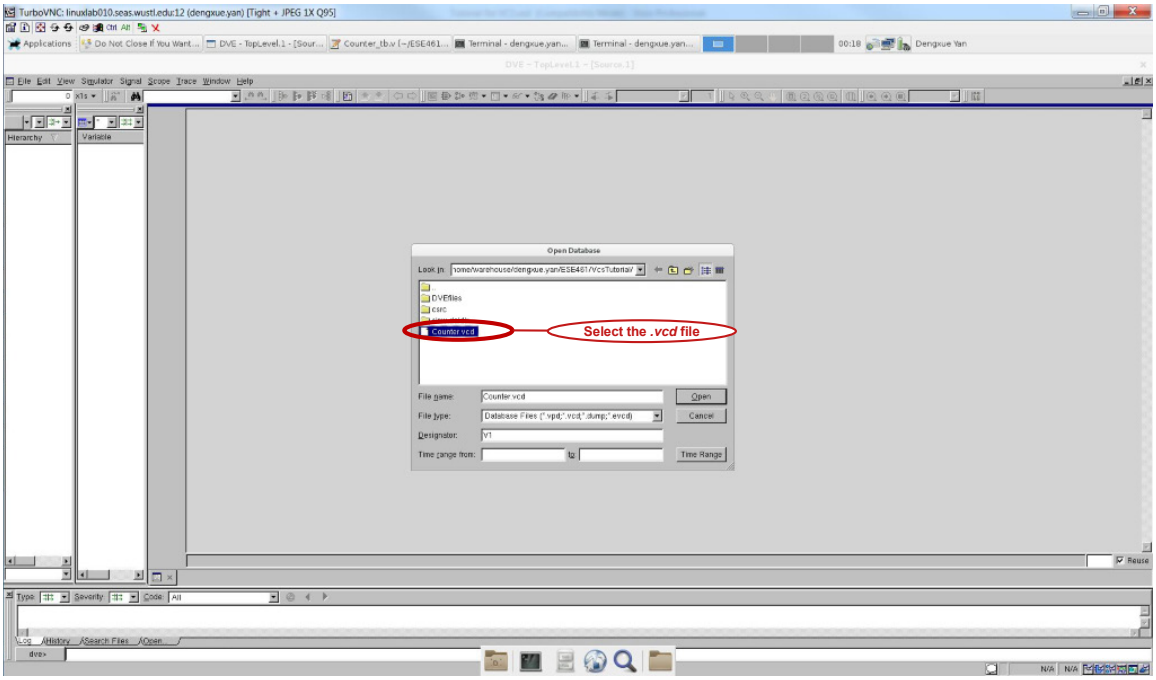

Fig. 10 dve open database (2)

- **Then you will find the name of your test bench model in the *Hierarchy* box (*Counter_tb* here). Expand it so that you can find *DUT* in the options.**

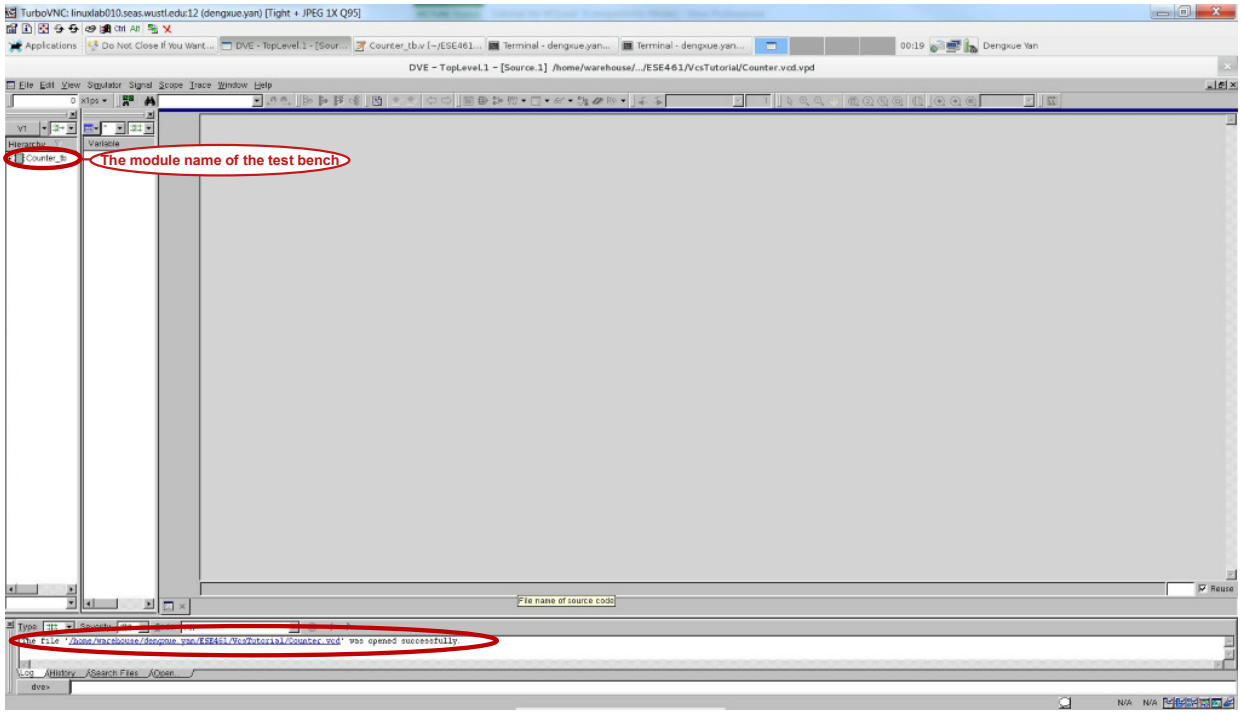

Fig. 11 dve open database (3)

- **If you click on *DUT* , select the signals listed(all or partial) and right click, you will find an option "*Add to Waves*".**

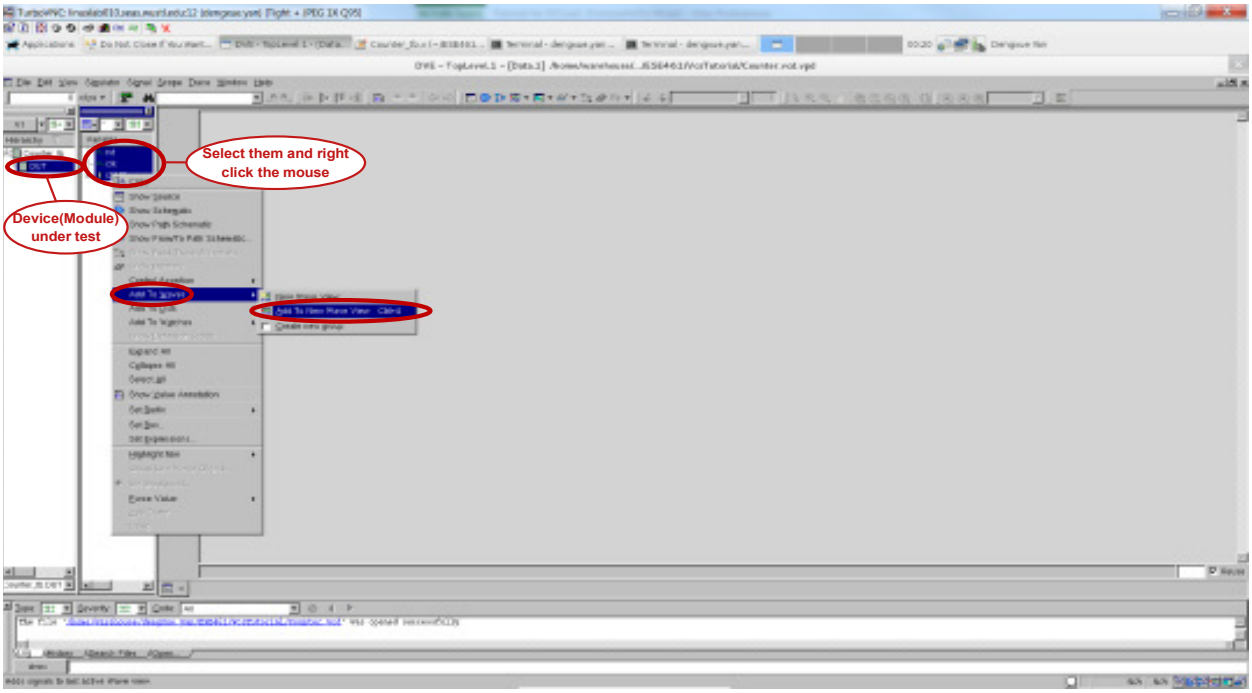

Fig. 12 dve open database (4)

- **Click on "*Add to New Wave View*" to see the waveforms of your Inputs and Outputs. You should see your results in a new window. Then adjust the size of the waveform and explore other options as well.**

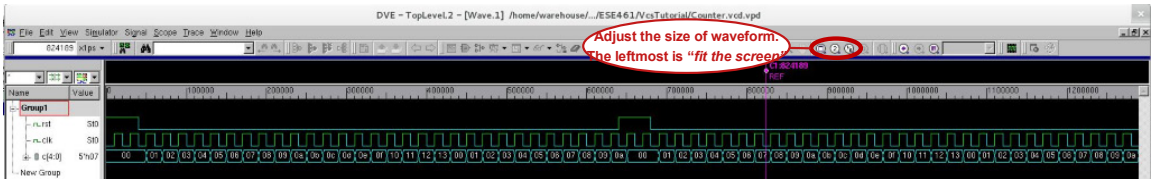

Fig. 13 The waveform display

Additional Option to run in Debug mode:

Instead of compiling the files directly as before, we can enable a debug flag during compilation by using following command

vcs -lca -debug_access+all Counter.v Counter_tb.v

Now run the code:

./simv -gui &

This should open the *dve* tool automatically and you can fully run your test bench or debug it step by step. To do this first select inputs and outputs from variable window and right click "*Add to the Waves*" as before. This should open the following window as shown in the Fig. 14 . Then click the tool button of blue arrow in brace or press *F11* to run the test bench step by step(Fig. 14 and Fig. 15). Or click the tool button of the blue arrow pointing downward or press *F5* to run the test bench fully(Fig. 16 ~ Fig. 18). Other tool options are also available and just explore them by yourself.

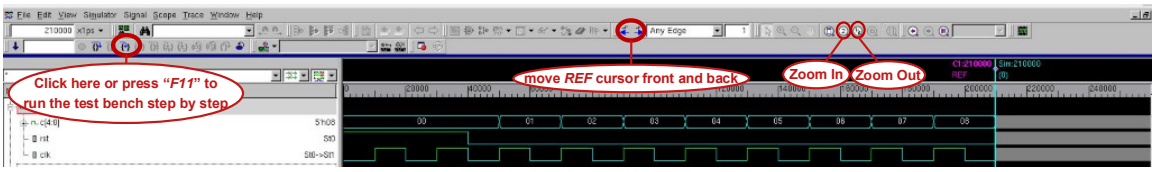

Fig. 14 The waveform display in debug mode

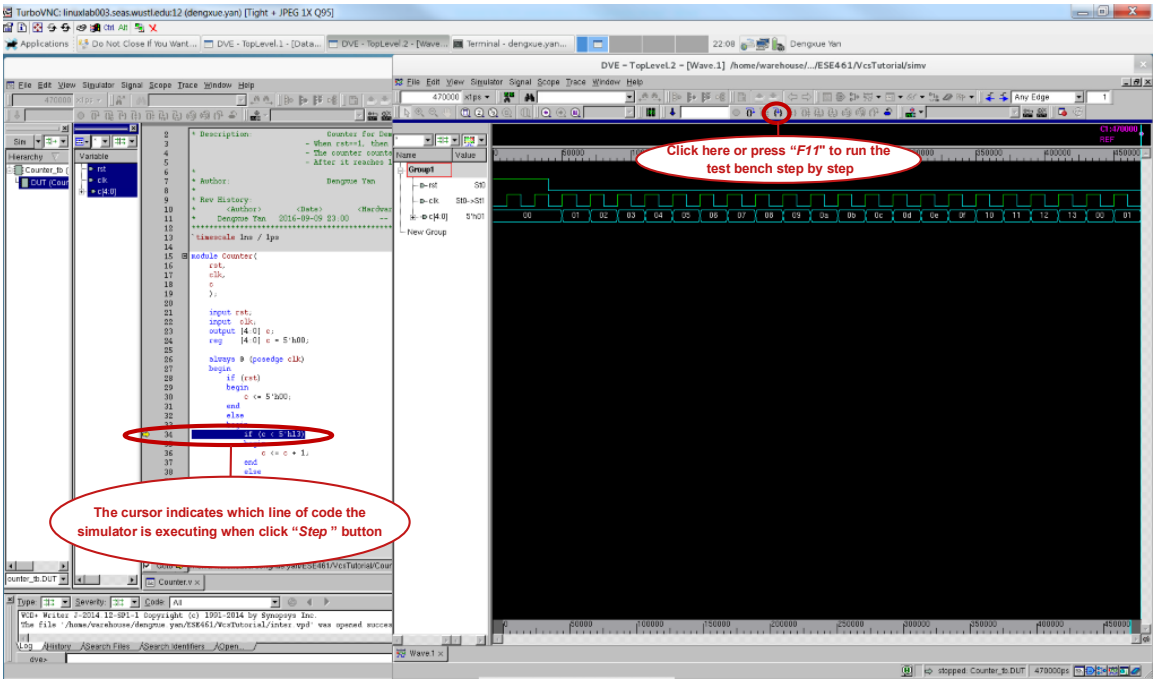

Fig.15 The code trace cursor in debug mode

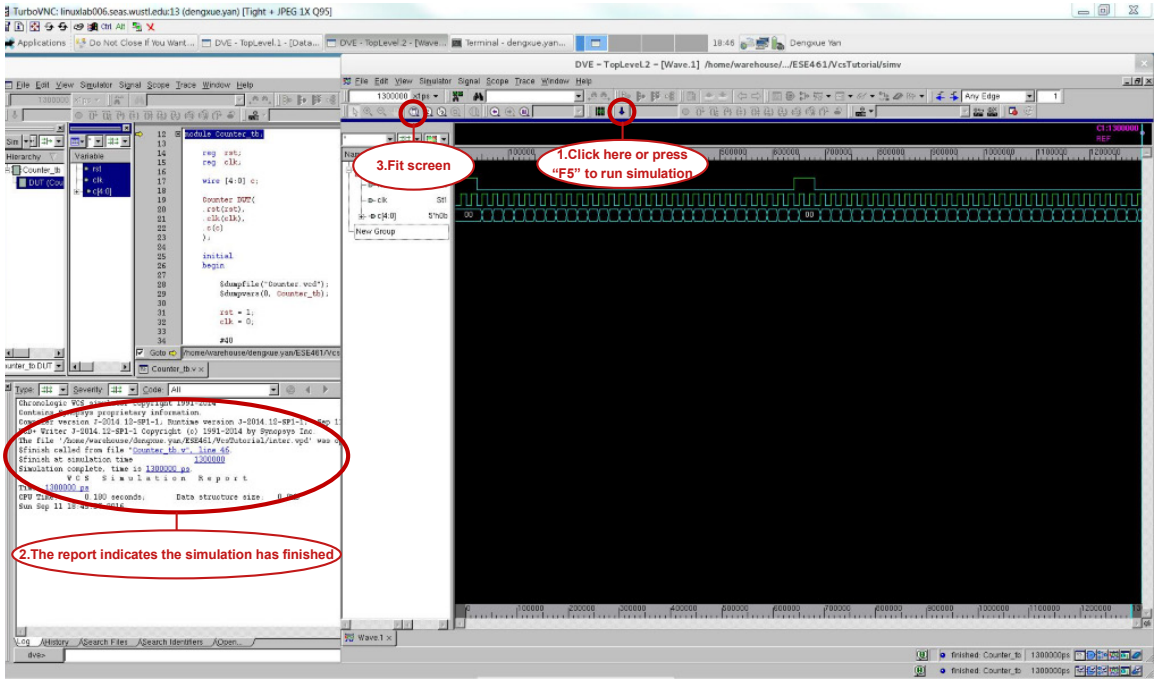

Fig.16 Fully "run" in debug mode(1)

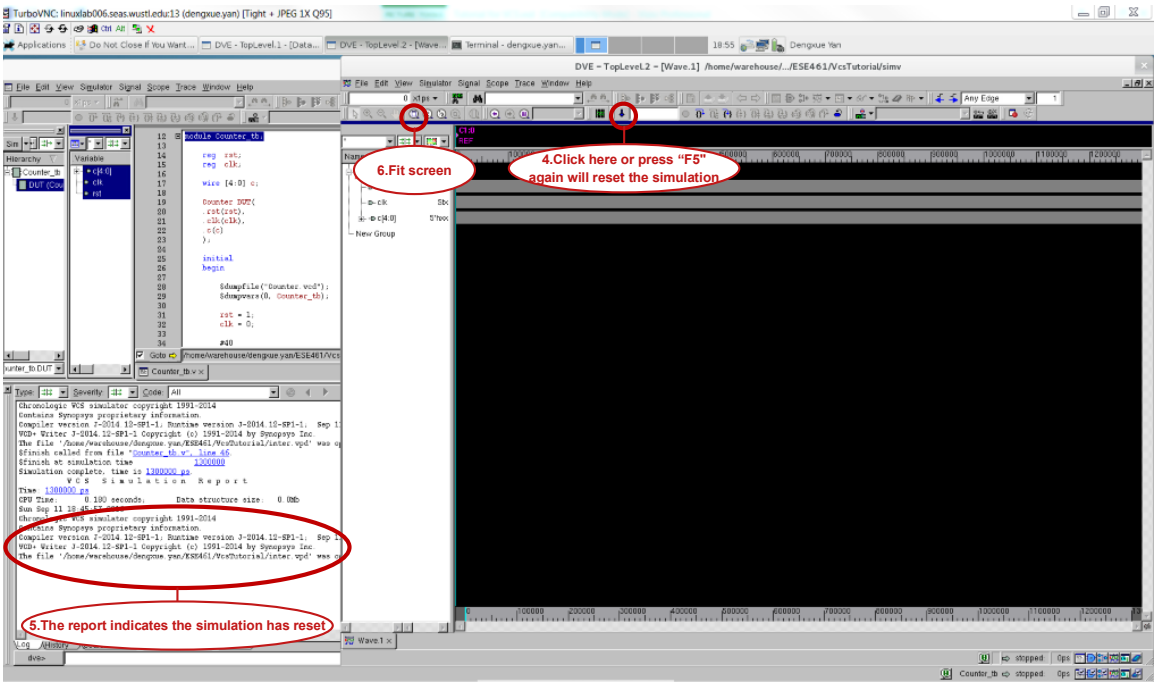

Fig.17 Fully "run" in debug mode(2) – simulation reset

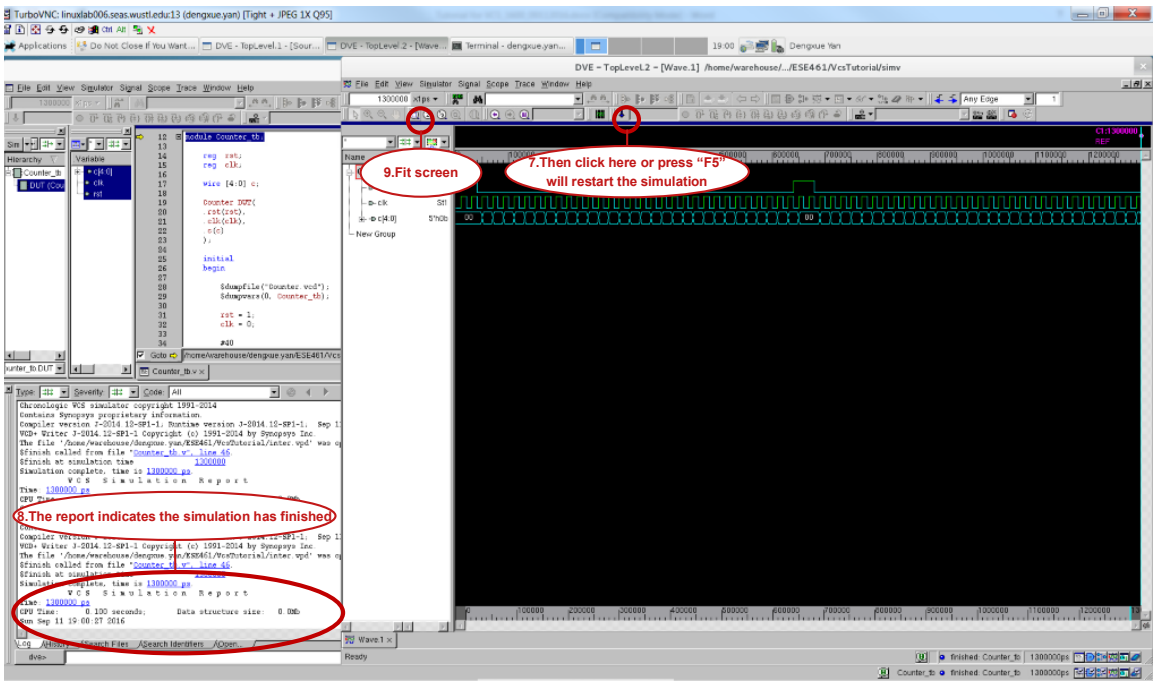

Fig.18 Fully run in debug mode(3) -- simulation restart

Reference:

If you are interested in exploring further, another example model and test bench codes are present in the following link

https://github.com/bangonkali/electronics/tree/master/verilog/adder

Reference for test bench syntax can be found here.