# ESE 566A Modern System-on-Chip Design, Spring 2017
# Tutorial for VCS

Electrical and System Engineering, Washington University

Revision: 01-12-2016

## 1. Introduction

In this tutorial you will gain experience compiling Verilog RTL into cycle-accurate executable simulators using Synopsys VCS. You will also learn how to use the "*dve*" Waveform Viewer to visualize the various signals in your simulated RTL designs. Figure 1.1 illustrates the basic VCS toolflow and how it fits into the larger ESE566A flow.

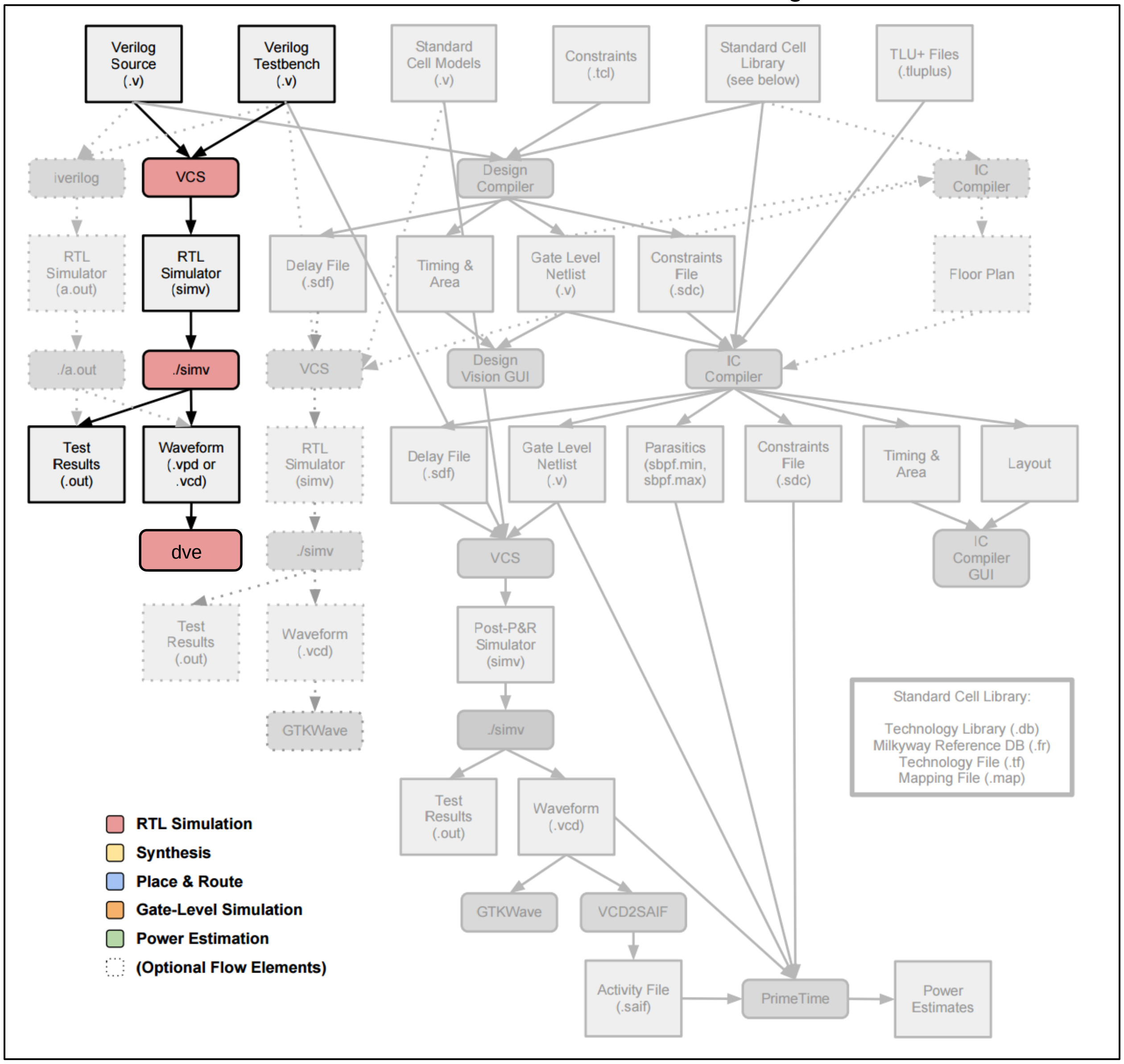


Figure 1.1 VCS Toolflow

VCS takes a set of Verilog files as input and produces an executable simulator as an output. VCS is capable of compiling both behavioral Verilog models and RTL Verilog models. Behavioral models are often not synthesizable, so any hardware we intend to synthesize will need to be written at the register transfer level. However, behavioral Verilog will still be useful when writing code we do not intend to synthesize, such as test harnesses. Please be conscious of this distinction, attempting to push behavioral code through the next step in the toolflow (Synthesis) will likely result in much pain and suffering.

## .2. Login to the Linux Lab server

Detailed explanation is in *ese566-linux-tutorial.pdf*

## 3: Getting started with Verilog

Creating a new folder (better if you have all the files for a project in a specific folder). Then, enter into this new folder and start writing your Verilog script in a new file (.v file). Example code for modeling a 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.

## 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 ~/ESE566A/VcsTutorial/
```

Compile the files by typing in the terminal:

```
% vcs -full64 -PP +lint=all,noVCDE +v2k -timescale=1ns/10ps <file>.v <file_tb>.v
```

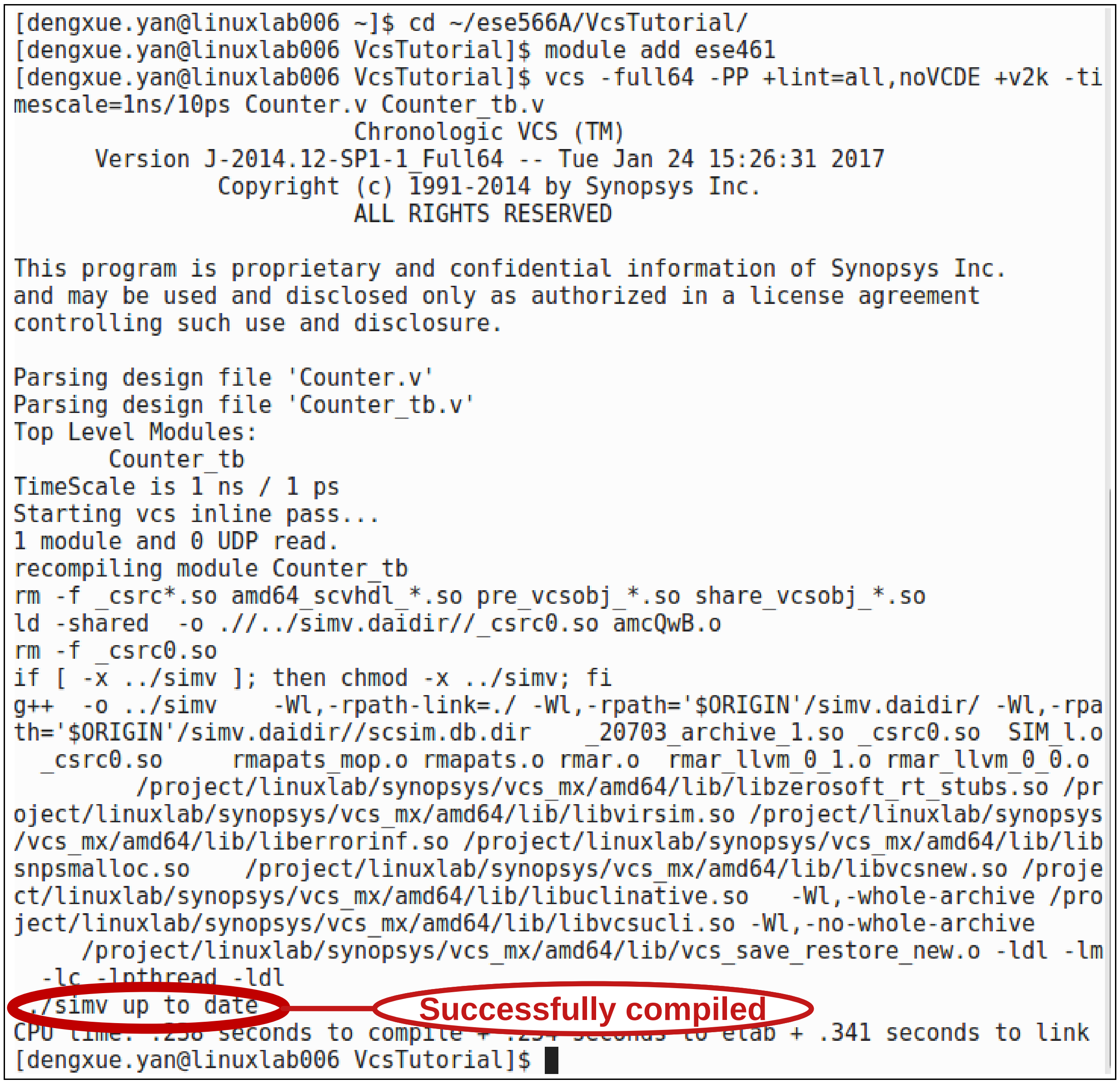
Important runtime flags:

- -full64 executes the 64-bit version of VCS.
- +lint=all,noVCDE turns on Verilog warnings except the VCDE warning. Since it is relatively easy to write legal Verilog code which is probably functionally incorrect, you will always want to use this argument. For example, VCS will warn you if you connect nets with different bitwidths or forget to wire up a port. Always try to eliminate all VCS compilation errors and warnings.
- +v2k enables support for various Verilog-2001 language features.
- +timescale specifies how the abstract delay units in a design map into real time units. This can also be provided in verilog source files as the ‘timescale compiler directive.
- -v indicates which Verilog files are part of a library (and thus should only be compiled if needed) and which files are part of the actual design (and thus should always be compiled).

In the above example, it should be:

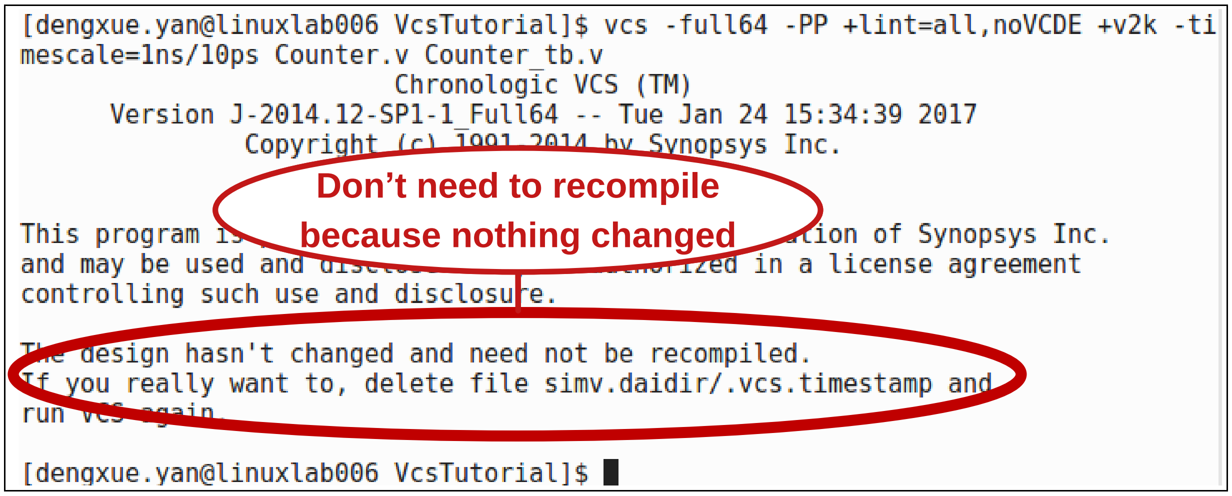
```
% vcs -full64 -PP +lint=all,noVCDE +v2k -timescale=1ns/10ps Counter.v
Counter_tb.v
```

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.



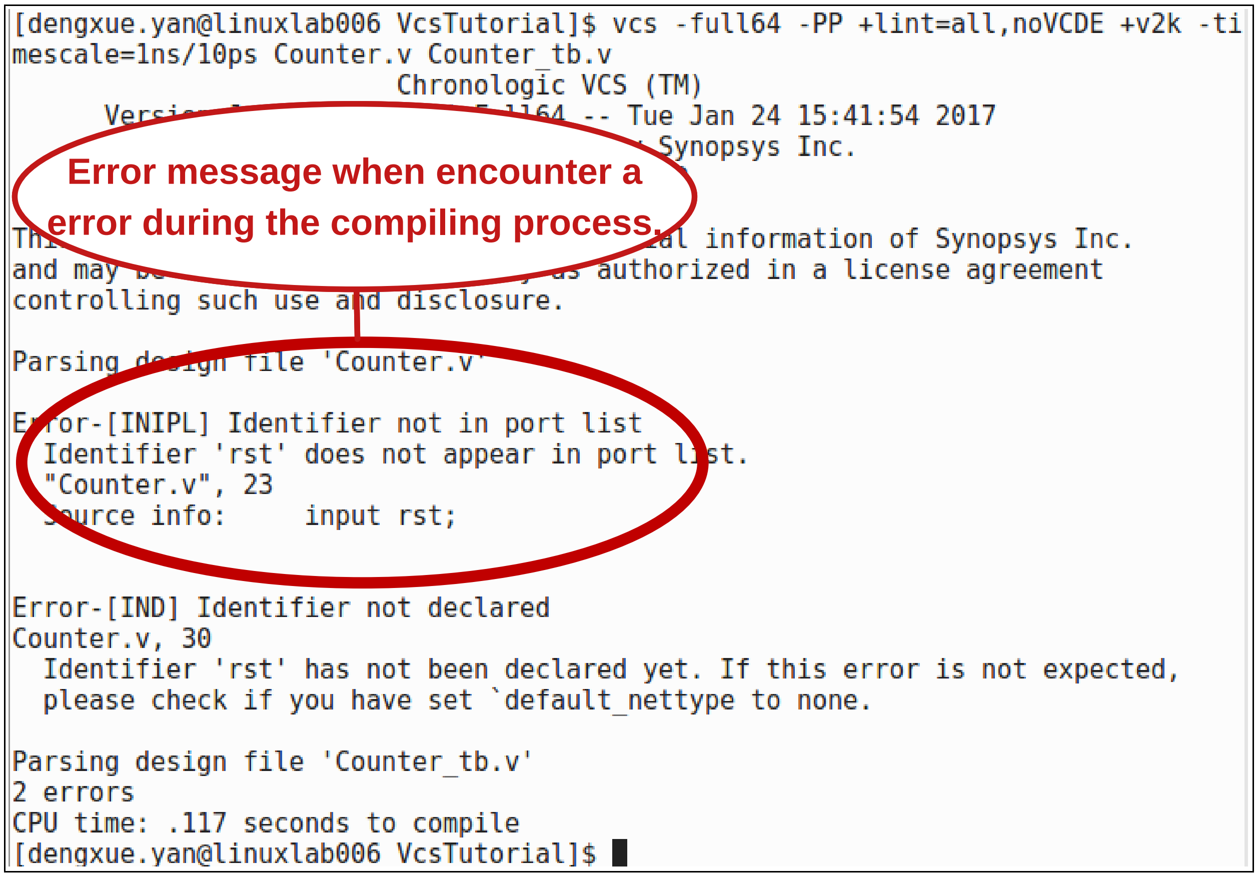
```
[dengxue.yan@linuxlab006 ~]$ cd ~/ese566A/VcsTutorial/
[dengxue.yan@linuxlab006 VcsTutorial]$ module add ese461
[dengxue.yan@linuxlab006 VcsTutorial]$ vcs -full64 -PP +lint=all,noVCDE +v2k -ti
mescale=1ns/10ps Counter.v Counter_tb.v
                         Chronologic VCS (TM)
      Version J-2014.12-SP1-1_Full64 -- Tue Jan 24 15:26:31 2017
               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 amd64_scvhdl_*.so pre_vcsobj_*.so share_vcsobj_*.so
ld -shared  -o .//../simv.daidir//_csrc0.so amcQwB.o
rm -f _csrc0.so
if [ -x ../simv ]; then chmod -x ../simv; fi
g++  -o ../simv    -Wl,-rpath-link=./ -Wl,-rpath='$ORIGIN'/simv.daidir/ -Wl,-rpa
th='$ORIGIN'/simv.daidir//scsim.db.dir    _20703_archive_1.so _csrc0.so  SIM_l.o
  _csrc0.so     rmapats_mop.o rmapats.o rmar.o  rmar_llvm_0_1.o rmar_llvm_0_0.o
         /project/linuxlab/synopsys/vcs_mx/amd64/lib/libzerosoft_rt_stubs.so /pr
oject/linuxlab/synopsys/vcs_mx/amd64/lib/libvirsim.so /project/linuxlab/synopsys
/vcs_mx/amd64/lib/liberrorinf.so /project/linuxlab/synopsys/vcs_mx/amd64/lib/lib
snpsmalloc.so    /project/linuxlab/synopsys/vcs_mx/amd64/lib/libvcsnew.so /proje
ct/linuxlab/synopsys/vcs_mx/amd64/lib/libuclinative.so   -Wl,-whole-archive /pro
ject/linuxlab/synopsys/vcs_mx/amd64/lib/libvcsucli.so -Wl,-no-whole-archive
     /project/linuxlab/synopsys/vcs_mx/amd64/lib/vcs_save_restore_new.o -ldl -lm
  -lc -lpthread -ldl
../simv up to date
CPU time: .238 seconds to compile + .254 seconds to elab + .341 seconds to link
[dengxue.yan@linuxlab006 VcsTutorial]$ █
```

Figure 4.1 The result of successfully executing vcs



```
[dengxue.yan@linuxlab006 VcsTutorial]$ vcs -full64 -PP +lint=all,noVCDE +v2k -ti
mescale=1ns/10ps Counter.v Counter_tb.v
                         Chronologic VCS (TM)
      Version J-2014.12-SP1-1_Full64 -- Tue Jan 24 15:34:39 2017
               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.

[dengxue.yan@linuxlab006 VcsTutorial]$ █
```

Figure 4.2 The result of recompile the code when nothing changed


Error message when encounter a error during the compiling process.

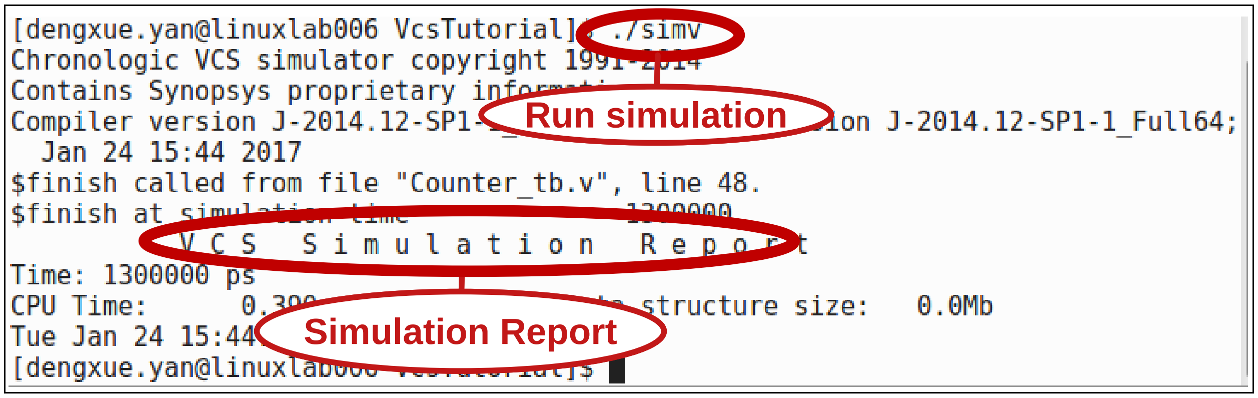
```
[dengxue.yan@linuxlab006 VcsTutorial]$ vcs -full64 -PP +lint=all,noVCDE +v2k -ti
mescale=1ns/10ps Counter.v Counter_tb.v
                         Chronologic VCS (TM)
      Versi                    164 -- Tue Jan 24 15:41:54 2017
                                             Synopsys Inc.

Thi                                          al information of Synopsys Inc.
and may b                                    authorized in a license agreement
controlling such use and disclosure.

Parsing design file 'Counter.v'

Error-[INIPL] Identifier not in port list
  Identifier 'rst' does not appear in port list.
  "Counter.v", 23
  Source info:     input rst;


Error-[IND] Identifier not declared
Counter.v, 30
  Identifier 'rst' has not been declared yet. If this error is not expected,
  please check if you have set `default_nettype to none.

Parsing design file 'Counter_tb.v'
2 errors
CPU time: .117 seconds to compile
[dengxue.yan@linuxlab006 VcsTutorial]$ █
```

Figure 4.3 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.



```
[dengxue.yan@linuxlab006 VcsTutorial]$ ./simv
Chronologic VCS simulator copyright 1991-2014
Contains Synopsys proprietary informati
Compiler version J-2014.12-SP1-                 ion J-2014.12-SP1-1_Full64;
  Jan 24 15:44 2017
$finish called from file "Counter_tb.v", line 48.
$finish at simulation time              1300000
           V C S   S i m u l a t i o n   R e p o r t
Time: 1300000 ps
CPU Time:      0.300                 structure size:    0.0Mb
Tue Jan 24 15:44
[dengxue.yan@linuxlab006 VcsTutorial]$ █
```

Figure 4.4 Simulation Report

## 5 Automated VCS Build Process

Typing each command via the command line is a tedious and error-prone process, and should typically be avoided. Instead, we make use of scripts to automate the process of building our tools for us. The following commands will first delete the simulator you previously built, and then regenerate it using the *makefile*(Example *makefile* is here, download it and put it into the folder *~/ese566A/VcsTutorial*).

```
% cd ~/ese566A/VcsTutorial
% rm simv
% rm -r simv.daidir
% make
% ./simv +verbose=1
```

The *make* program uses the *Makefile* located in the current working directory to generate the file given on the command line. Take a look at the *Makefile* located in *~/ese566A/VcsTutorial*. Makefiles are made up of variable assignments and a list of rules in the following form.

```
target : dependency1 dependency2 ... dependencyN
        command1
        command2
        ...
        commandN
```

Each rule has three parts: a target, a list of dependencies, and a list of commands. When a desired target file is “out of date” or does not exist, then the make program will run the list of commands to generate the target file. To determine if a file is “out of date”, the make program compares the modification times of the target file to the modification times of the files in the dependency list. If any dependency is newer than the target file, make will regenerate the target file. Locate in the *makefile* where the Verilog source files are defined. Find the rule which builds *simv*. More information about *makefiles* is online at http://www.gnu.org/software/make/manual. Not all *make* targets need to be actual files. For example, the clean target will remove all generated content from the current working directory. So, the following commands will first delete the generated simulator and then rebuild it.

```
% ~/ese566A/VcsTutorial
% make clean
% make simv
```

And run the following command you should run the *simv* file:

```
% make run
```

# 6. Viewing Trace Output With *dve*

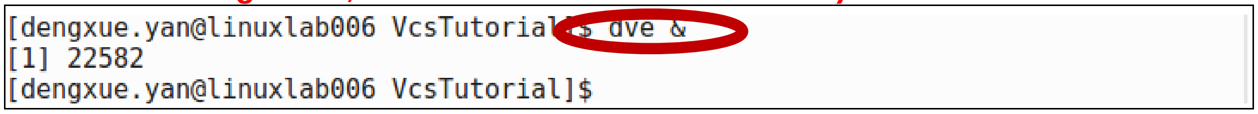
## 6.1 Basic operation

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@linuxlab006 VcsTutorial]$ dve &
[1] 22582
[dengxue.yan@linuxlab006 VcsTutorial]$
```

Figure 6.1 Start dve on the terminal

Go to “File->Open Database” and select the “*.vcd*” file from the project folder.

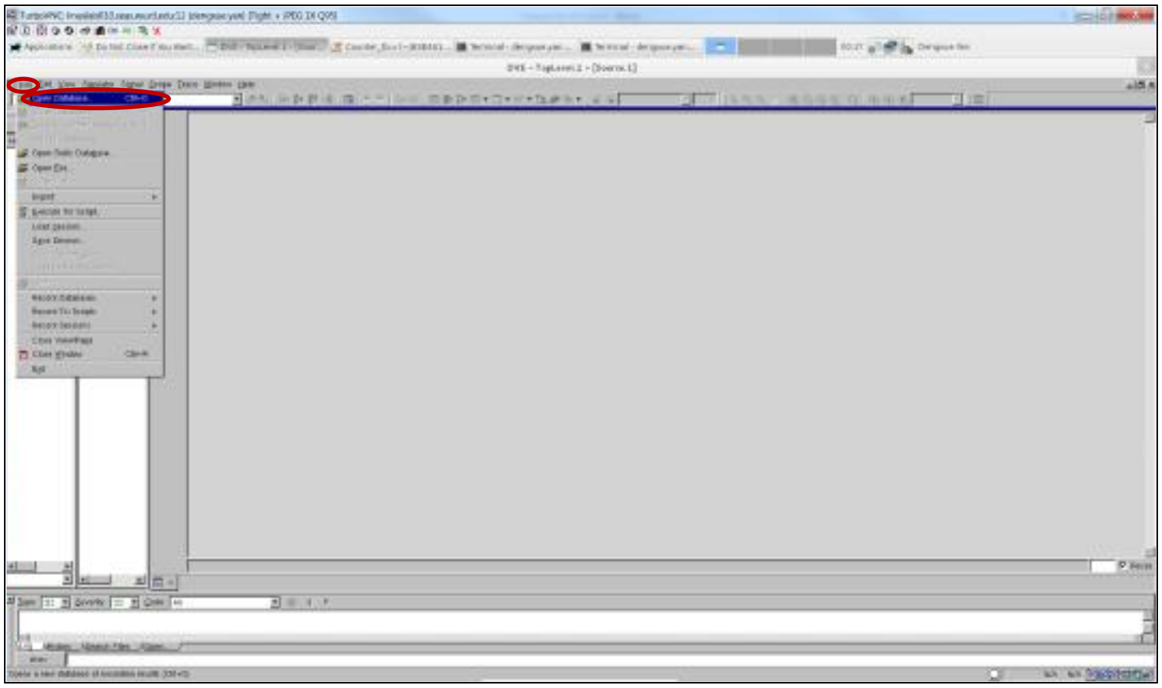


Figure 6.2 *dve* open database (1)

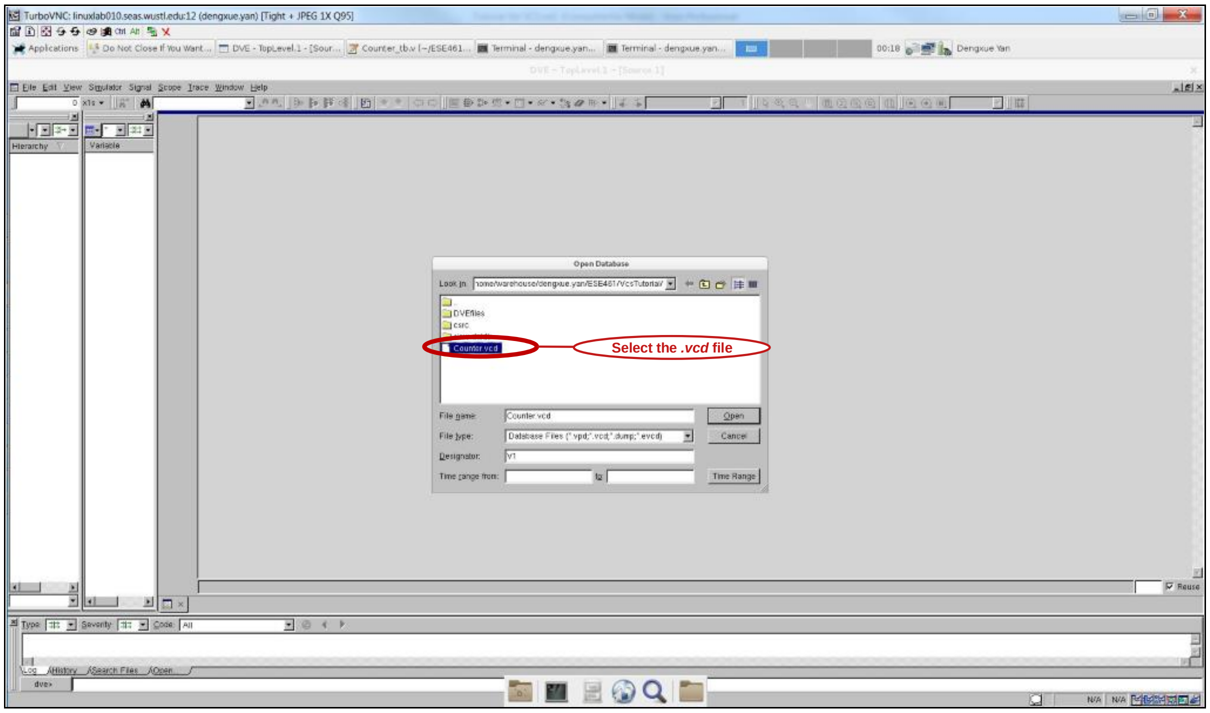
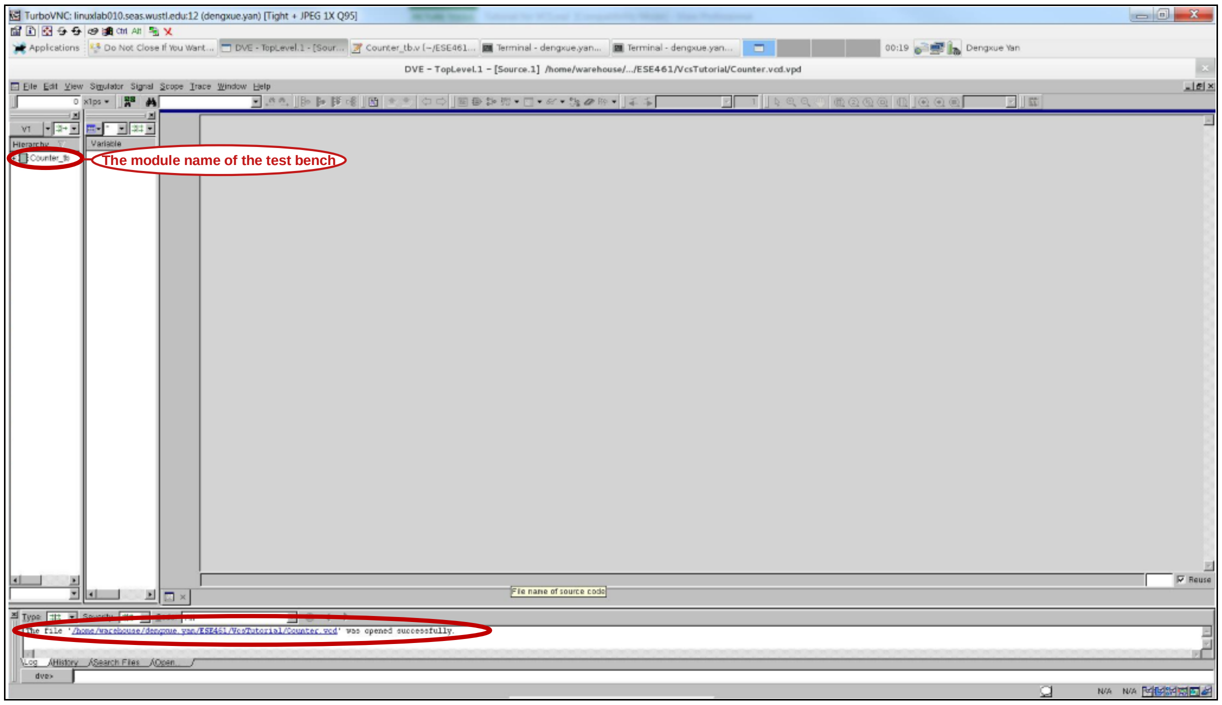


Figure. 6.3 *dve* open database (2)

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.


DVE - TopLevel.1 - [Source.1] /home/warehouse/.../ESE461/VcsTutorial/Counter.vcd.vpd
Hierarchy
Variable
Counter_tb
The module name of the test bench
File name of source code

```
The file '/home/warehouse/dengxue.yan/ESE461/VcsTutorial/Counter.vcd' was opened successfully.
```

Figure. 6.4 *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”.

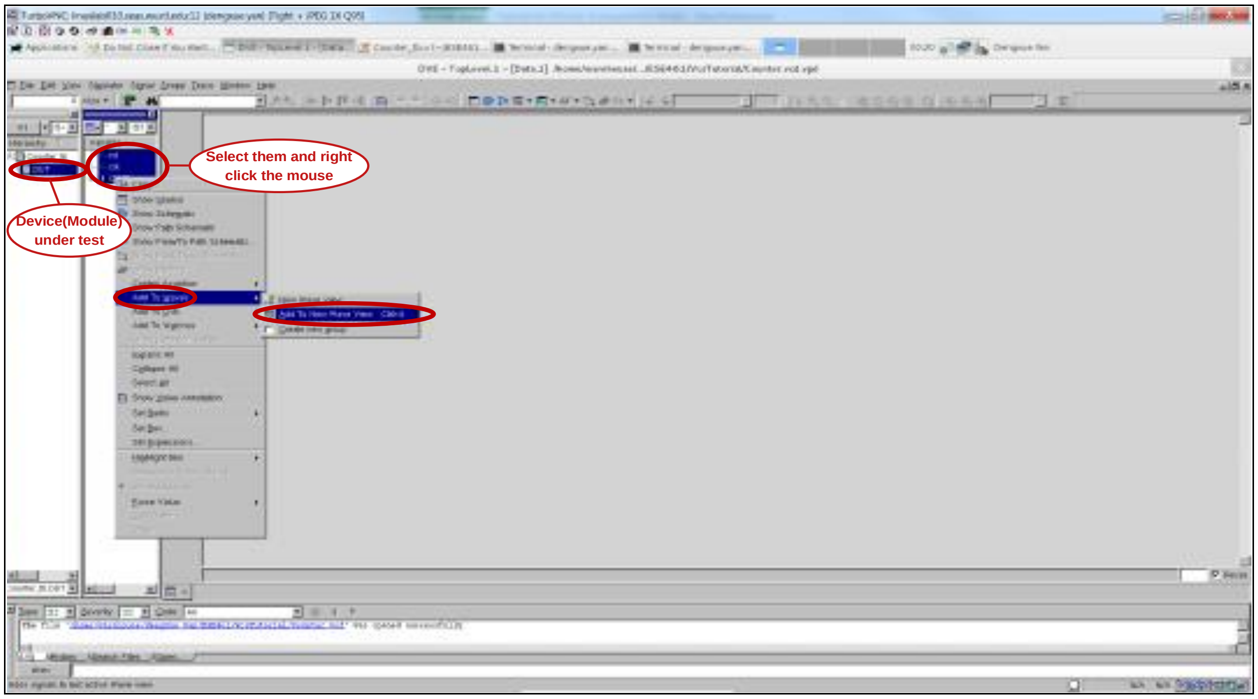

Select them and right click the mouse
Device(Module) under test

Figure. 6.5 *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.

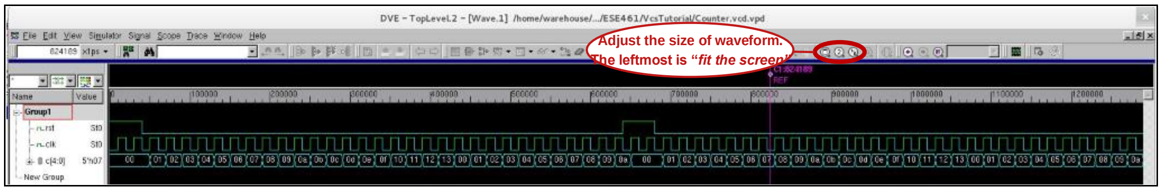

DVE - TopLevel.2 - [Wave.1] /home/warehouse/.../ESE461/VcsTutorial/Counter.vcd.vpd
Adjust the size of waveform. The leftmost is "fit the screen"
Name
Value
Group1
rst
clk
c[4:0]
New Group

Figure. 6.6 The waveform display

## 6.2 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 Figure 6.7 . Then click the tool button of blue arrow in brace or press F11 to run the test bench step by step(Figure 6.7 and Figure 6.8). Or click the tool button of the blue arrow pointing downward or press F5 to run the test bench fully(Figure 6.9 ~ Figure 6.11). Other tool options are also available and just explore them by yourself.

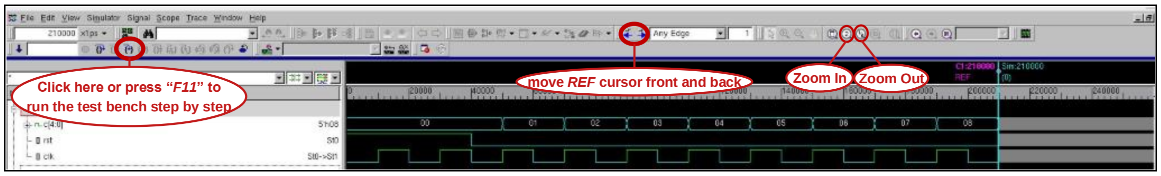
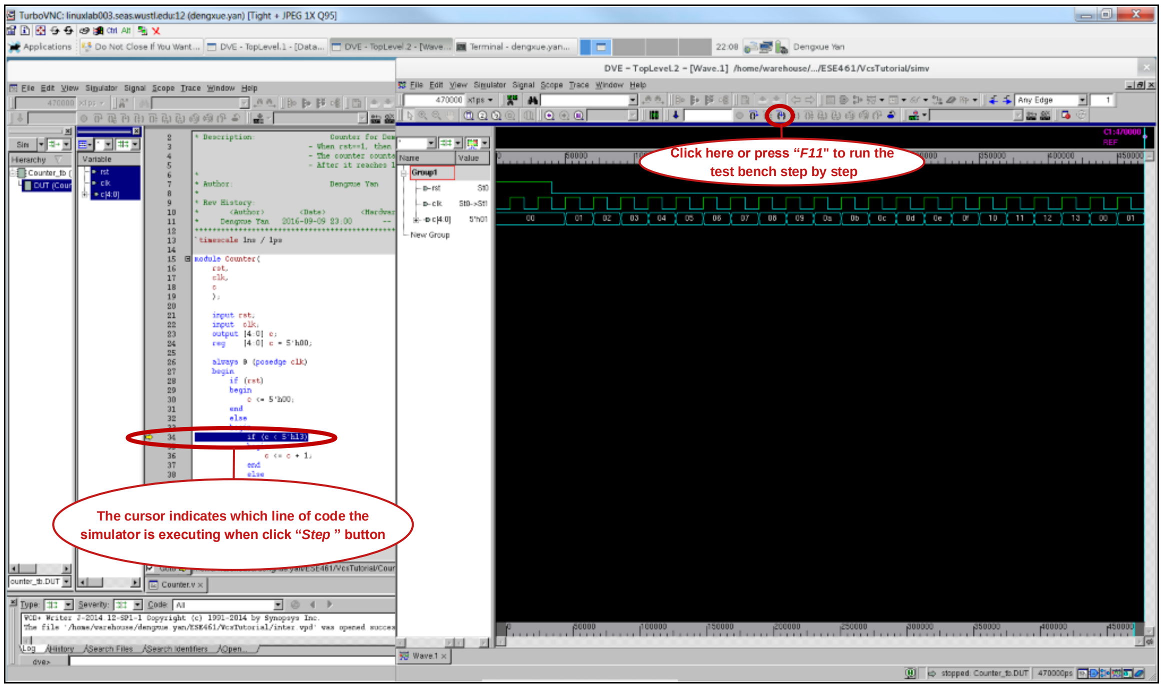
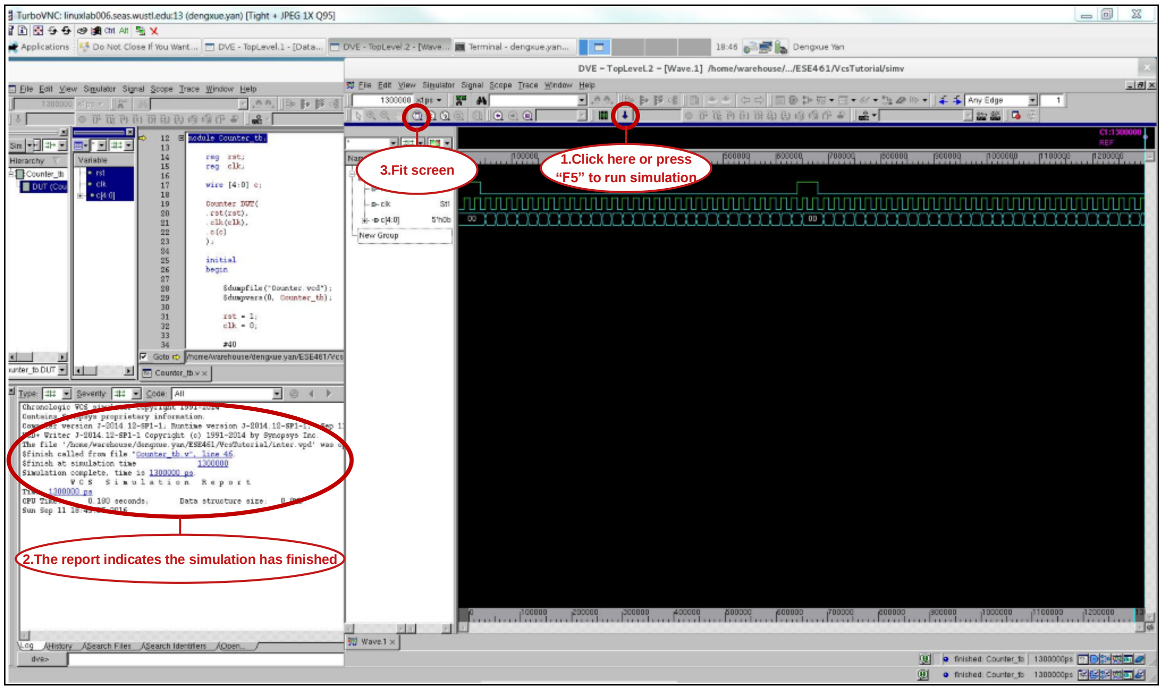

Click here or press "F11" to run the test bench step by step
move REF cursor front and back
Zoom In
Zoom Out

Figure 6.7 The waveform display in debug mode


Click here or press "F11" to run the test bench step by step
The cursor indicates which line of code the simulator is executing when click "Step " button

```
* Description:                    Counter for Dem
                             - When rst==1, then
                             - The counter counte
                             - After it reaches 1
*
* Author:                    Dengxue Yan
*
* Rev History:
*       <Author>        <Date>          <Hardwar
*     Dengxue Yan   2016-09-09 23:00       --
**************************************************
`timescale 1ns / 1ps

module Counter(
    rst,
    clk,
    c
    );

    input rst;
    input  clk;
    output [4:0] c;
    reg    [4:0] c = 5'h00;

    always @ (posedge clk)
    begin
        if (rst)
        begin
            c <= 5'h00;
        end
        else

            if (c < 5'h13)

                c <= c + 1;
            end
            else
```

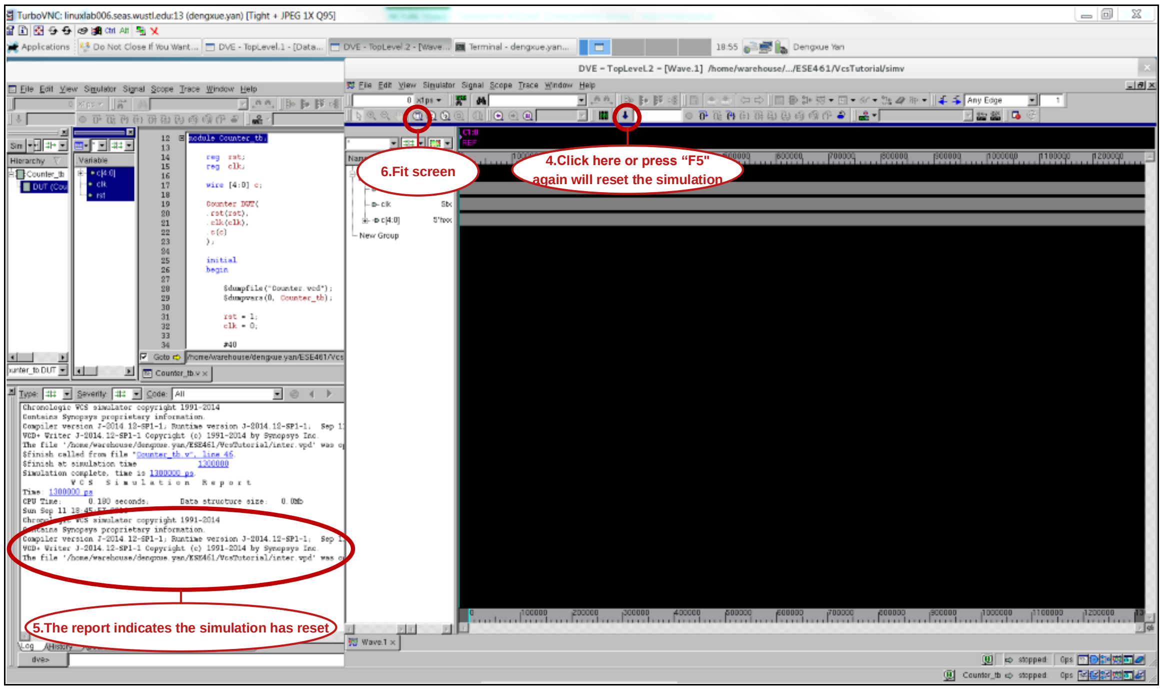
Figure 6.8 The code trace cursor in debug mode


3.Fit screen
1.Click here or press "F5" to run simulation
2.The report indicates the simulation has finished

```
module Counter_tb;

    reg  rst;
    reg  clk;

    wire [4:0] c;

    Counter DUT(
    .rst(rst),
    .clk(clk),
    .c(c)
    );

    initial
    begin

        $dumpfile("Counter.vcd");
        $dumpvars(0, Counter_tb);

        rst = 1;
        clk = 0;

        #40
```

```
Chronologic VCS simulator copyright 1991-2014
Contains Synopsys proprietary information.
Compiler version J-2014.12-SP1-1; Runtime version J-2014.12-SP1-1;  Sep 11
VCD+ Writer J-2014.12-SP1-1 Copyright (c) 1991-2014 by Synopsys Inc.
The file '/home/warehouse/dengxue.yan/ESE461/VcsTutorial/inter.vpd' was o
$finish called from file "Counter_tb.v", line 46.
$finish at simulation time              1300000
Simulation complete, time is 1300000 ps.
          V C S   S i m u l a t i o n   R e p o r t
Time: 1300000 ps
CPU Time:      0.180 seconds;       Data structure size:   0.0Mb
Sun Sep 11 18:45:   2016
```

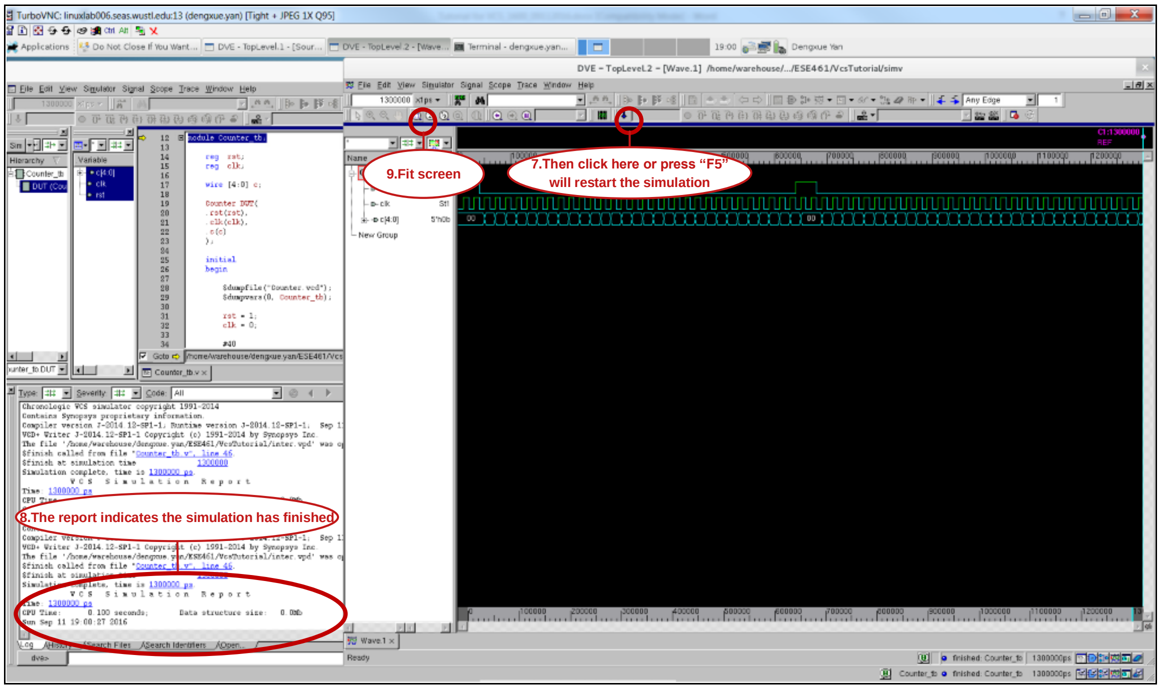
Figure 6.9 Fully "run" in debug mode(1)


4.Click here or press "F5" again will reset the simulation
6.Fit screen
5.The report indicates the simulation has reset
DVE - TopLevel.2 - [Wave.1] /home/warehouse/.../ESE461/VcsTutorial/simv
Counter_tb.v
Wave.1

```
module Counter_tb;

    reg rst;
    reg clk;

    wire [4:0] c;

    Counter DUT(
    .rst(rst),
    .clk(clk),
    .c(c)
    );

    initial
    begin

        $dumpfile("Counter.vcd");
        $dumpvars(0, Counter_tb);

        rst = 1;
        clk = 0;

        #40
```

```
Chronologic VCS simulator copyright 1991-2014
Contains Synopsys proprietary information.
Compiler version J-2014.12-SP1-1; Runtime version J-2014.12-SP1-1;  Sep 1
VCD+ Writer J-2014.12-SP1-1 Copyright (c) 1991-2014 by Synopsys Inc.
The file '/home/warehouse/dengxue.yan/ESE461/VcsTutorial/inter.vpd' was o
$finish called from file "Counter_tb.v", line 46.
$finish at simulation time                 1300000
Simulation complete, time is 1300000 ps.
           V C S   S i m u l a t i o n   R e p o r t
Time: 1300000 ps
CPU Time:      0.180 seconds;       Data structure size:   0.0Mb
Sun Sep 11 18:45:57 2016
Chronologic VCS simulator copyright 1991-2014
Contains Synopsys proprietary information.
Compiler version J-2014.12-SP1-1; Runtime version J-2014.12-SP1-1;  Sep 1
VCD+ Writer J-2014.12-SP1-1 Copyright (c) 1991-2014 by Synopsys Inc.
The file '/home/warehouse/dengxue.yan/ESE461/VcsTutorial/inter.vpd' was o
```

Figure 6.10 Fully "run" in debug mode(2) – simulation reset


7.Then click here or press "F5" will restart the simulation
9.Fit screen
8.The report indicates the simulation has finished
DVE - TopLevel.2 - [Wave.1] /home/warehouse/.../ESE461/VcsTutorial/simv
Counter_tb.v
Wave.1
Ready

```
Chronologic VCS simulator copyright 1991-2014
Contains Synopsys proprietary information.
Compiler version J-2014.12-SP1-1; Runtime version J-2014.12-SP1-1;  Sep 1
VCD+ Writer J-2014.12-SP1-1 Copyright (c) 1991-2014 by Synopsys Inc.
The file '/home/warehouse/dengxue.yan/ESE461/VcsTutorial/inter.vpd' was o
$finish called from file "Counter_tb.v", line 46.
$finish at simulation time                 1300000
Simulation complete, time is 1300000 ps.
           V C S   S i m u l a t i o n   R e p o r t
Time: 1300000 ps
CPU Time:      0.100 seconds;       Data structure size:   0.0Mb
Sun Sep 11 19:00:27 2016
```

Figure 6.11 Fully run in debug mode(3) -- simulation restart

## 7. Reference

[1] http://www.csl.cornell.edu/courses/ece5745/handouts/ece5745-tut1-vcs.pdf

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.