

EE520 | EE620 EDA Tutorial 3 Assignment

Assigned: Friday, October 31

EDA Tutorial 3 Due: Friday, November 07, 1:00 PM

Revision History:

1	Original created for EE520 EE620 by: Mark A. Indovina
2	Updates for EE520 EE620 by: Mark A. Indovina

The grade for this project will be based on:

1. On the timely completion of your EDA Tutorial 3 assignment.
2. On the EDA Tutorial 3 project report, uploaded as a PDF copy to the mycourses drop box (**PDF only**) due: November 07, 1:00 PM.

1. INTRODUCTION	3
2. MODULE DESCRIPTION	4
3. DATABASE CHECKOUT	6
4. RTL SIMULATION / VERIFICATION.....	8
5. LOGIC SYNTHESIS & TEST INSERTION	14
6. DETAILED TIMING ANALYSIS	16
8. NOTES ON DATABASE ORGANIZATION.....	18
9. PROJECT REPORT	19
APPENDIX	20

1. INTRODUCTION

EDA Tutorial 3 tutorial will introduce you to an RTL verification, logic synthesis, test insertion, and detailed timing analysis design flow as shown in Figure 1; the database was created using Verilog HDL and various tools from Cadence Design Systems and Synopsys.

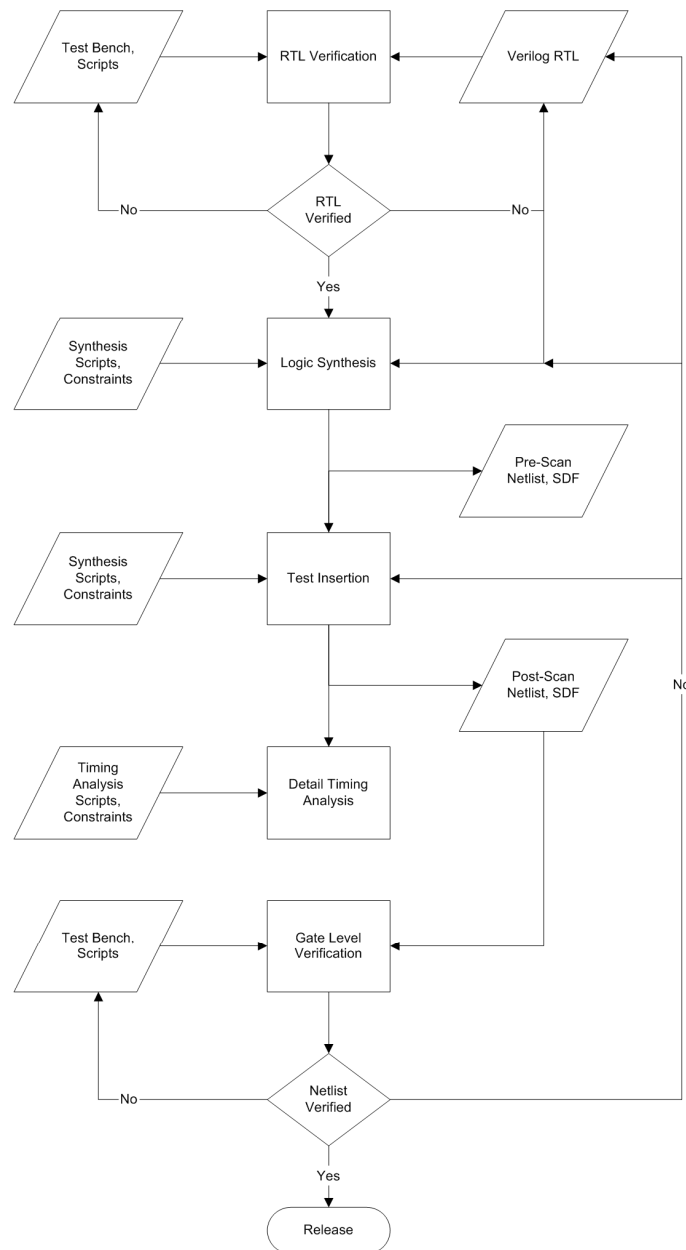


Figure 1 Simplified RTL Design Flow

2. MODULE DESCRIPTION

The module used during this exercise is the “Results Character Conversion” (RCC) block. During Project 3 you will work with the DTMF Receiver core RTL database which includes the RCC.

For reference, the RCC is used to process a signal spectrum calculated by a modified Discrete Fourier Transform (DFT), the resulting DFT signal spectrum is written in block format to the RCC. Once a block is written, the spectrum is analyzed for DTMF digit content (the in band tones you hear generated by wired and wireless telephones). If a digit is found, the resolved ASCII character representation is written to the Results circular buffer.

Note that the RCC is coded as an implicit state machine and performs the following checks to determine if a valid DTMF digit (and not speech or background noise) is present:

- Finds a pair of tones, one from the low frequency group, one from the high frequency group
- Determines if the signal amplitude difference between the low frequency tone and high frequency tone does not exceed 12 dB
- Determines if the signal amplitude difference between the high frequency tone and low frequency tone does not exceed 12 dB
- Determines if the DTMF digit is present for at least 45 mS
- Determines if an inter-digit “quiet period” exists between digits of at least 45 mS

Figure 2 presents a block diagram of the RCC. Review the source code and familiarize yourself with the behavior of this design unit. Note that both the GT and CHECK TWIST functions contain complex operations although the GT (gt_comp) function calls do NOT happen in parallel.

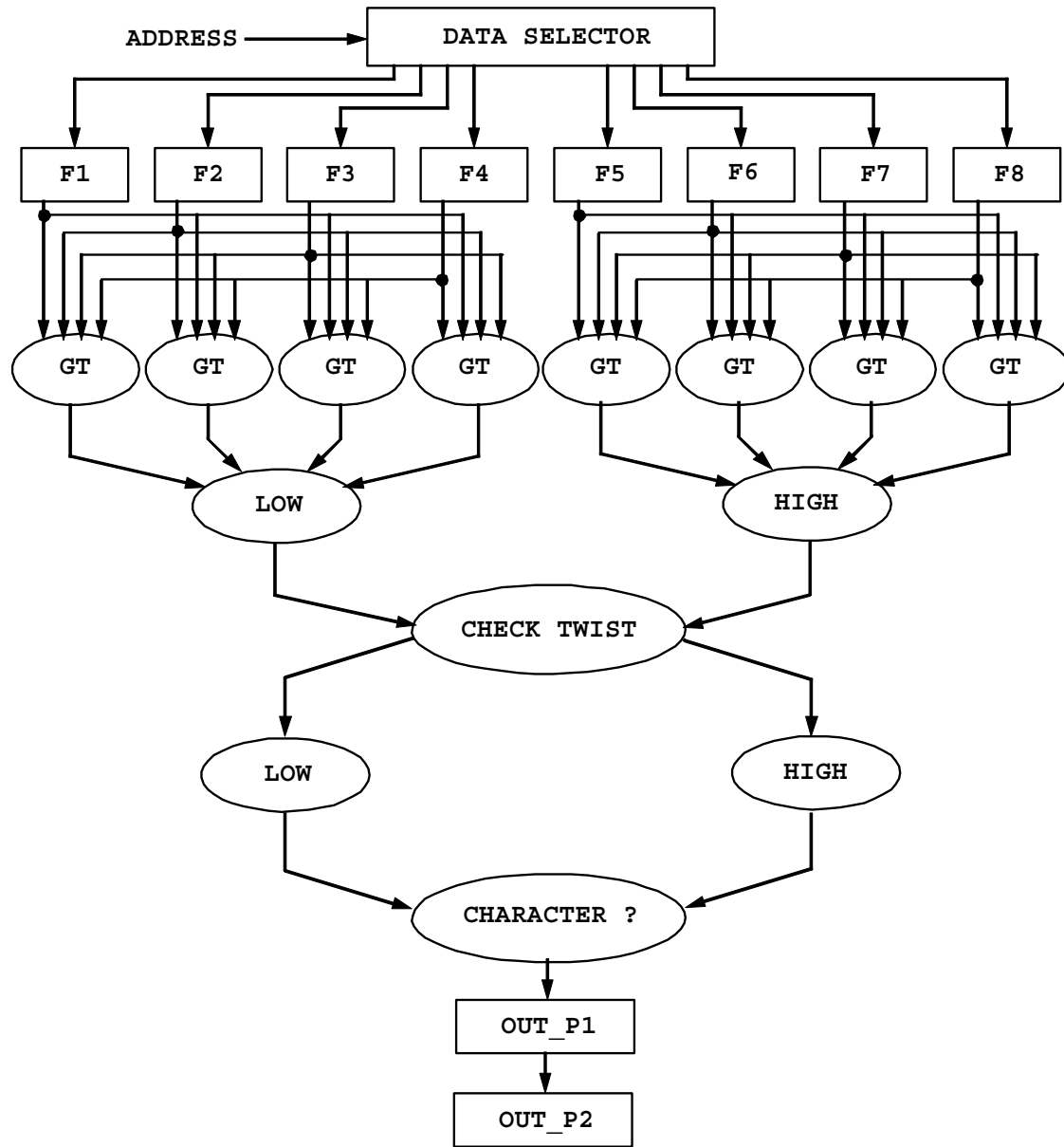


Figure 2 RCC Block Diagram

3. DATABASE CHECKOUT

To start, you will need to create a work area for the tutorial. Logon to one of the compute servers and create a work area for EDA Tutorial 3 as follows:

```
cd ee620  
mkdir eda_tut_3  
cd eda_tut_3
```

We will be using “*git*”, a distributed revision control and source code management tool, to manage data shared during HDL projects. *Git* is very comprehensive and contains lots of neat features. It is very easy to use, but requires practice and discipline to be utilized effectively.

We’ll briefly explore using *git* during EDA Tutorial 3, and you will utilize *git* to manage source files during Project 3. If you get stuck, or desire to know more about *git*, excellent resources can be found at <http://git-scm.com/documentation>. A brief online, interactive git tutorial can be found at <http://try.github.io/levels/1/challenges/1>.

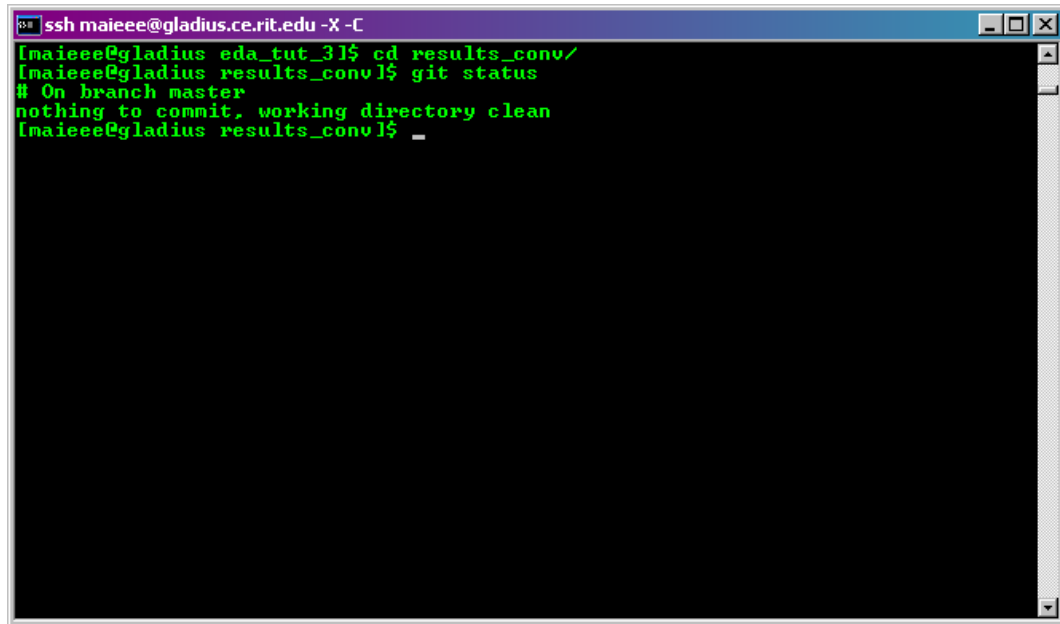
Let’s check out the tutorial database as follows:

```
git clone /classes/ee620/verilog/eda_tut_3/results_conv
```

Note that this version of the command does not allow you to use *git* across a network.

Once the database is cloned, change to the repository directory and check the repository status as follows:

```
cd results_conv  
git status
```

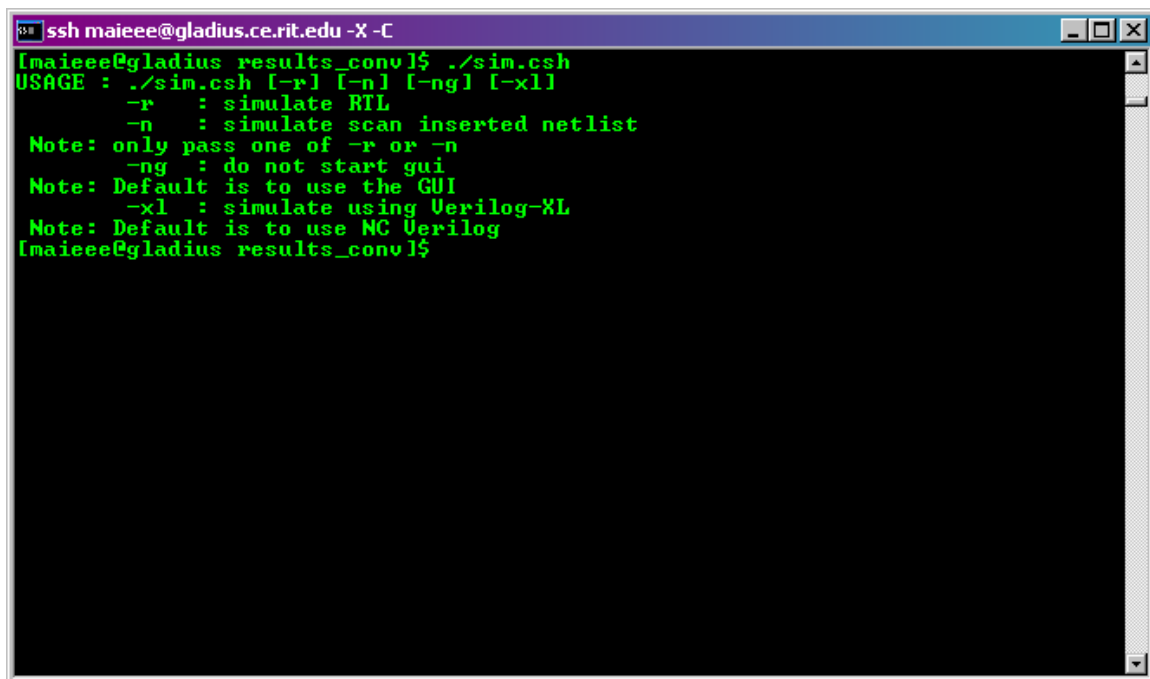
A terminal window titled 'ssh maieeee@gladius.ce.rit.edu -X -C' showing the execution of 'cd results_conv/' and 'git status'. The output of 'git status' is: '# On branch master', 'nothing to commit, working directory clean'. The prompt is '[maieeee@gladius results_conv]\$_'.

```
ssh maieeee@gladius.ce.rit.edu -X -C  
[maieeee@gladius eda_tut_3]$ cd results_conv/  
[maieeee@gladius results_conv]$ git status  
# On branch master  
nothing to commit, working directory clean  
[maieeee@gladius results_conv]$ _
```

If the *git status* command does not report “*nothing to commit, working directory clean*”, something went wrong – ask for assistance.

4. RTL SIMULATION / VERIFICATION

Your work directory contains a script, `sim.csh`, which you will use to simulate the design at the RTL and Netlist level. The simulation script has the ability to invoke both NC Verilog and Verilog XL. NC Verilog is a fast, native compiled Verilog simulator that supports Verilog 95, Verilog 2001 and SystemVerilog (versions of the executable also support mixed mode VHDL/Verilog/SystemC). Verilog XL is the “gold” standard for Verilog simulation, is based on the original Verilog simulator codebase, and is a fairly old (slow) simulator; unless otherwise specified, you will use NC Verilog for the lab exercises. The script requires command line options for proper execution; invoking the script without options will provide help as shown below.

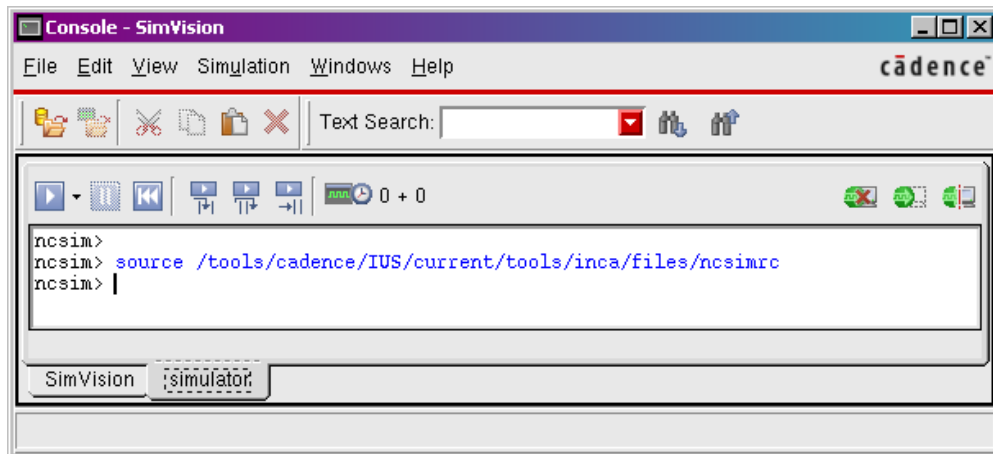
A terminal window titled 'ssh maieeee@gladius.ce.rit.edu -X -C' showing the output of the command './sim.csh'. The output displays the usage and options for the script.

```
maieeee@gladius results_conv1$ ./sim.csh
USAGE : ./sim.csh [-r] [-n] [-ng] [-xl]
        -r      : simulate RTL
        -n      : simulate scan inserted netlist
Note: only pass one of -r or -n
        -ng     : do not start gui
Note: Default is to use the GUI
        -xl     : simulate using Verilog-XL
Note: Default is to use NC Verilog
maieeee@gladius results_conv1$
```

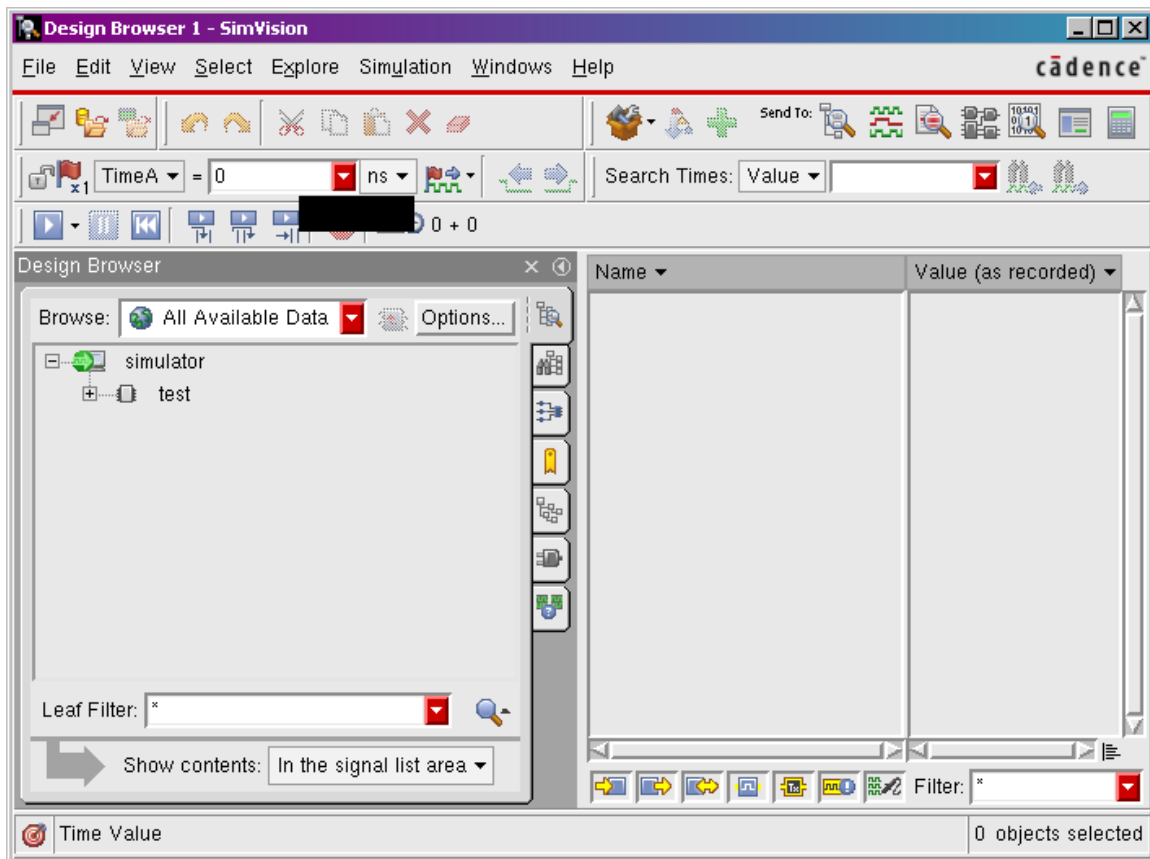
We will run RTL simulation using NC Verilog and the GUI using the following command:

```
./sim.csh -r
```


A simulator console window will open as shown below:

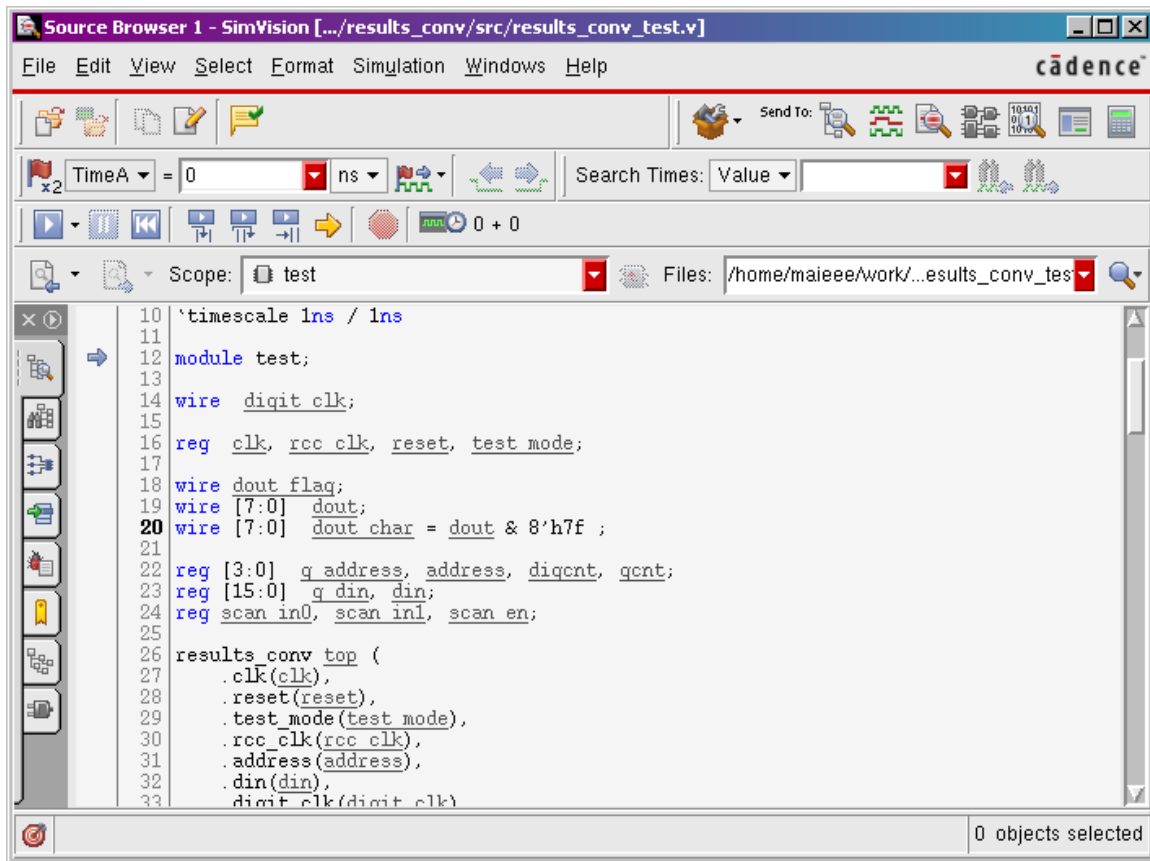


A design browser window will also open as shown below:



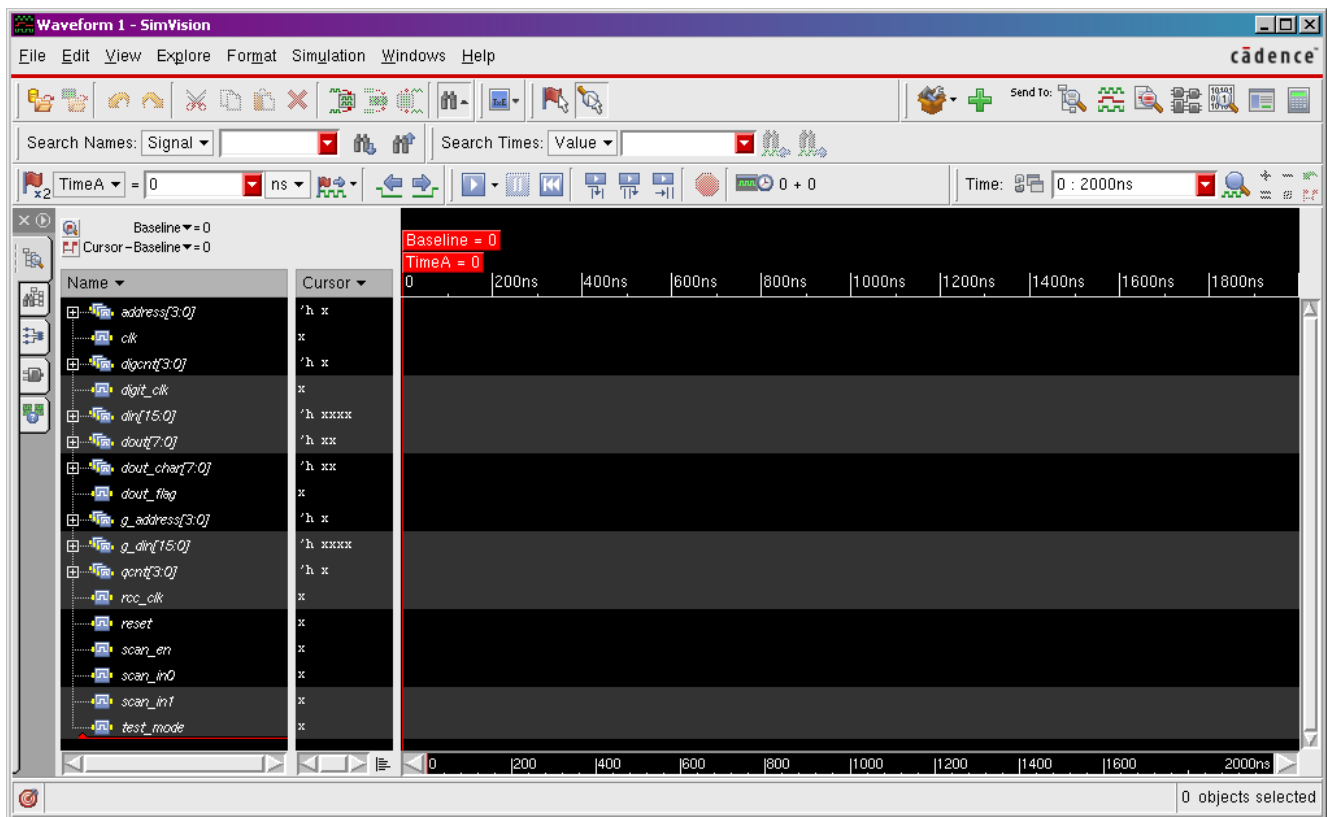
The design browser window is your main interface for debugging your design database. The design browser is a comprehensive analysis environment and includes various tools for hierarchy browsing, source display, waveform display, breakpoints, etc.

In the sub window Design Browser, click on **test** and then right click and select the menu item **Send To Source Browser**; a **Source Browser** window appears.



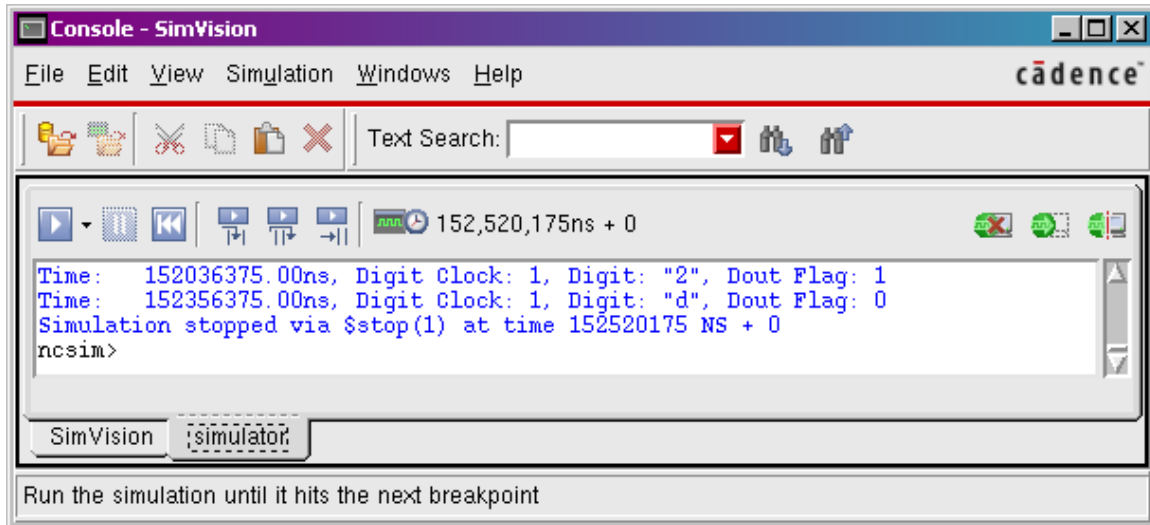
The **Source Browser** window will contain all design items (wires, registers, etc.) visible within the module **test** – which is the test bench. For lines that are highlighted, such as line 94, you could click on the line to set a breakpoint.

In the sub window Design Browser, click on **test** and then right click and select the menu item **Send To Waveform Window**; a **Waveform** window appears.

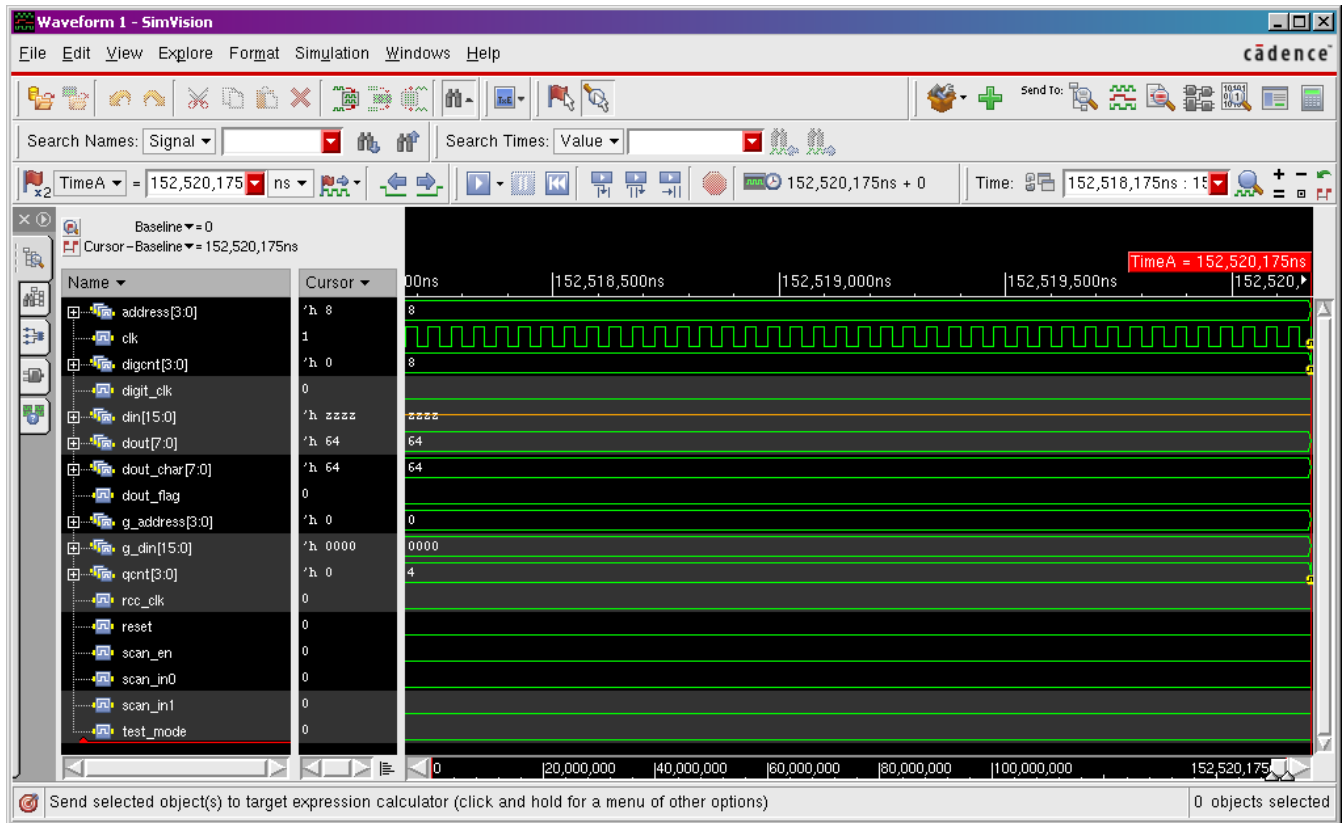


The **Waveform** window will contain all design items (wires, registers, etc.) visible within the module **test** – which is the test bench. In the **simulator Console window**, or **Source Browser** window, or **Waveform** window, start the simulation by pressing the **Run** button. Alternatively you could type a single **'r'** in the console window to start the simulation.

The simulation will run until either a breakpoint is reached (assuming you set a breakpoint), or the test bench stops the test. If you set a breakpoint, clear it and start the simulation. Once the test bench completes, the console will look like the following:



And the waveform window will update to show the signal traces.

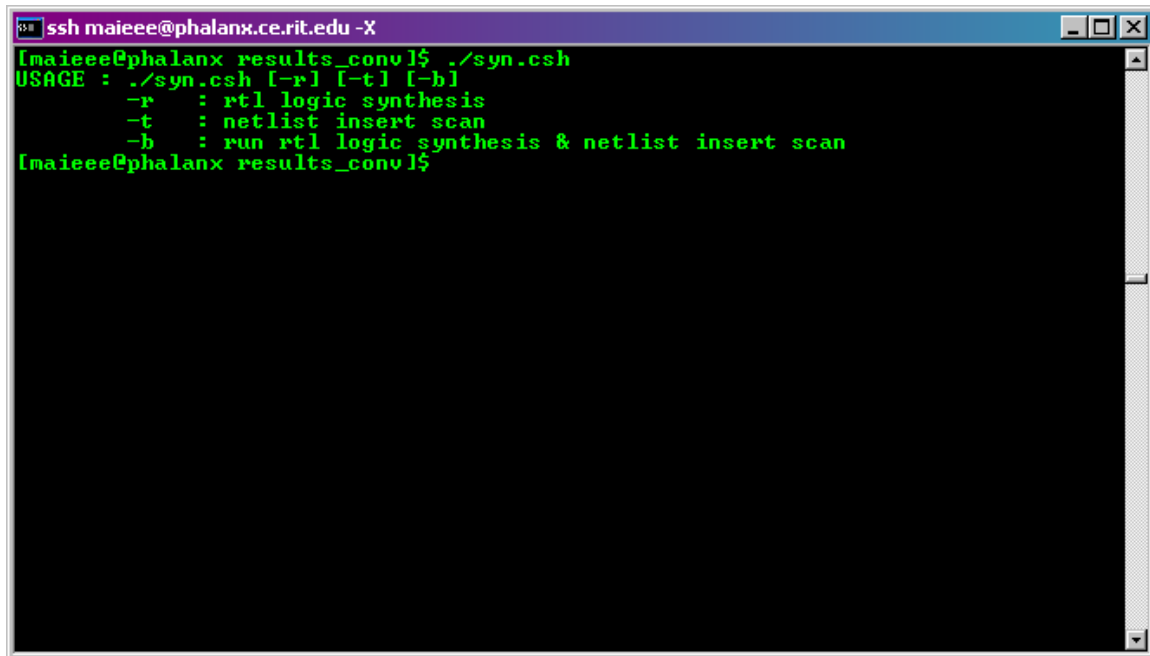


The waveform display includes various tools for zooming, panning, and modifying the waveform display to suite your needs; take the time to familiarize yourself with navigating the waveform display.

Issue the command `'quit'` or `'exit'` in the Console window to exit the software.

5. LOGIC SYNTHESIS & TEST INSERTION

Your work directory contains a script, `syn.csh`, which you will use to perform logic synthesis and test insertion with Synopsys Design Compiler. The script requires command line options for proper execution; invoking the script without options will provide help as shown below.



```
ssh maieeee@phalanx.ce.rit.edu -X
[maieeee@phalanx results_conv1]$ ./syn.csh
USAGE : ./syn.csh [-r] [-t] [-b]
        -r : rtl logic synthesis
        -t : netlist insert scan
        -b : run rtl logic synthesis & netlist insert scan
[maieeee@phalanx results_conv1]$
```

We will perform synthesis and test insertion in one step using the following command:

```
./syn.csh -b
```

Logic synthesis is a “noisy” process, and a large volume of status messages, warnings, and errors will scroll by; you should review the log file for issues that may need immediate attention.

Note that you will be targeting the TSMC 0.13 μ m (CL013G) Process; a copy of the data book for the library is on mycourses for your review. Assuming the process finishes successfully, you will find the resulting netlist, timing file, and reports in the directories:

netlist/	<- netlist directory
sdf/	<- timing annotation file directory
report/dc/	<- Design Compiler report directory

Please review these outputs, taking care **not to edit** any of the files. Note that if you wish more details on the reports, start the Synopsys Design Compiler using the following command:

dc_shell

Once the tool starts, you can '**man**' any of the report commands. Note that the report header contains the name of the report. For example the resources report header looks like the following:

```

ssh maieeee@phalanx.ce.rit.edu -X

*****
Report : resources
Design : results_conv
Version: D-2010.03-SP3
Date   : Thu Oct 17 13:11:58 2013
*****

Resource Sharing Report for design results_conv in file
/home/maieeee/work/eda_tut_3/results_conv/src/results_conv.v

=====
| Resource | Module      | Parameters | Contained | Contained Operations |
|-----|-----|-----|-----|-----|
| r353     | DW01_sub    | width=17   |           | sub_453 sub_522      |
| r356     | DW01_sub    | width=17   |           | sub_515 sub_528      |
| r365     | DW_cmp      | width=8     |           | eq_350               |
| r367     | DW01_sub    | width=17   |           | sub_457              |
| r369     | DW01_sub    | width=17   |           | sub_461              |
| r371     | DW01_sub    | width=17   |           | sub_521              |
| r373     | DW01_sub    | width=17   |           | sub_527              |
|-----|-----|-----|-----|-----|

No implementations to report
No multiplexors to report

```

At the **dc_shell** command prompt type the following:

dc_shell> man report_resources

This will return a manual page for this command. Issue the command '**quit**' or '**exit**' to exit the software.

It is highly commended that you review and familiarize yourself with the synthesis scripts and constraints too. These can be found in the directories:

dc/ <- Design Compiler Logic Synthesis Scripts
cons/ <- Design Constraints (for synthesis and timing analysis)

Similar to the help on the reports, you can also access manual pages for all the commands you find in the scripts and constraints using the '**man**' command inside of **dc_shell**.

6. DETAILED TIMING ANALYSIS

Your work directory contains a script, `pt.csh`, which you will use to perform detailed timing analysis on the post scan inserted netlist using Synopsys PrimeTime.

We will perform detailed timing analysis in one step using the following command:

```
./pt.csh
```

Assuming the analysis finishes successfully, you will find the resulting reports in the directories:

```
report/pt/          <- PrimeTime report directory
```

Please review these outputs, taking care **not to edit** any of the files. Note that if you wish more details on the reports, start the Synopsys PrimeTime using the following command:

```
pt_shell
```

Once the tool starts, you can '**man**' any of the report commands. Note that the report header contains the name of the report. For example the analysis coverage report header looks like the following:

```
ssh maieeee@phalanx.ce.rit.edu -X
*****
Report : analysis_coverage
        -status_details {untested violated}
        -sort_by slack
Design : results_conv
Version: D-2010.06-SP1
Date   : Sun Oct 27 17:13:32 2013
*****
Type of Check      Total      Met      Violated      Untested
-----
setup              855      570 < 67%>      0 < 0%>      285 < 33%>
hold               855      449 < 53%>      121 < 14%>      285 < 33%>
recovery           157        1 < 1%>        0 < 0%>      156 < 99%>
min_pulse_width    727      570 < 78%>      0 < 0%>      157 < 22%>
out_setup           12        12 <100%>      0 < 0%>        0 < 0%>
out_hold            12        12 <100%>      0 < 0%>        0 < 0%>
-----
All Checks         2618     1614 < 62%>     121 < 5%>     883 < 34%>
-----
Constrained      Related   Check      Slack   Reason
Pin              Pin        Type
-----
clear_flag_reg/RN(low)              min_pulse_width
                                   untested no_clock
clear_flag_reg/RN(rise)             CK(rise)  recovery untested false_paths
clear_flag_reg/SE                   CK(rise)  hold    untested false_paths
"report/pt/results_conv_tsmc18_scan_pt_rac.rpt" [readonly] 1186L, 90979C
```


At the `pt_shell` command prompt type the following:

```
pt_shell> man report_analysis_coverage
```

This will return a manual page for this command. Issue the command `'quit'` or `'exit'` to exit the software.

It is highly commended that you review and familiarize yourself with the timing analysis scripts and constraints too. These can be found in the directories:

```
pt/      <- PrimeTime Timing Analysis Scripts  
cons/    <- Design Constraints (for synthesis and timing analysis)
```

Similar to the help on the reports, you can also access manual pages for all the commands you find in the scripts and constraints using the `'man'` command inside of `pt_shell`.

7. GATE LEVEL SIMULATION / VERIFICATION

We will run gate level simulation of the scan inserted netlist using NC Verilog and the GUI using the following command:

```
./sim.csh -n
```

Observe a couple of things when the simulator launches: (i) due the fact that you are simulating the netlist, there are many more objects in the design hierarchy; (ii) timing data is being back annotated onto the netlist using an SDF file created during synthesis for a timing-accurate, gate-level simulation.

Follow the simulation procedure used during RTL simulation to verify the netlist. Note that netlist simulation is much slower than RTL simulation.

8. NOTES ON DATABASE ORGANIZATION

There is a README file and MANIFEST file at the top level of the database:

MANIFEST.txt Baseline listing of files checked into repository

README.txt Brief description of database organization

You might also notice there is a naming convention used for derived files such as reports and netlists. The convention is to add an extension to the name of the file as follows:

<technology><scan?>_<tool?>

Where:

<technology>	identifies the target process technology
<scan?>	optional identifier used to indicate that the design has been processed through test insertion; would typically identify what type of test protocol used, i.e. full-scan (scan), partial-scan, LSSD, etc.
<tool?>	optional identifier used to indicate what tool produced this file
<rpt?>	optional identifier used to indicate which report is contained in this file

For example, a listing of the report files generated during the tutorial would be as follows:

Design Compiler logic synthesis reports:

report/dc/results_conv_tsmc18_dc.rpt

Design Compiler test insertion reports:

report/dc/results_conv_tsmc18_scan_dc.rpt

PrimeTime detail timing reports for test inserted netlist:

report/pt/results_conv_tsmc18_scan_pt_ct.rpt
 report/pt/results_conv_tsmc18_scan_pt_rt.rpt
 report/pt/results_conv_tsmc18_scan_pt_rclk.rpt
 report/pt/results_conv_tsmc18_scan_pt_rac.rpt
 report/pt/results_conv_tsmc18_scan_pt_rcv.rpt
 report/pt/results_conv_tsmc18_scan_pt_rca.rpt
 report/pt/results_conv_tsmc18_scan_pt_rdt.rpt

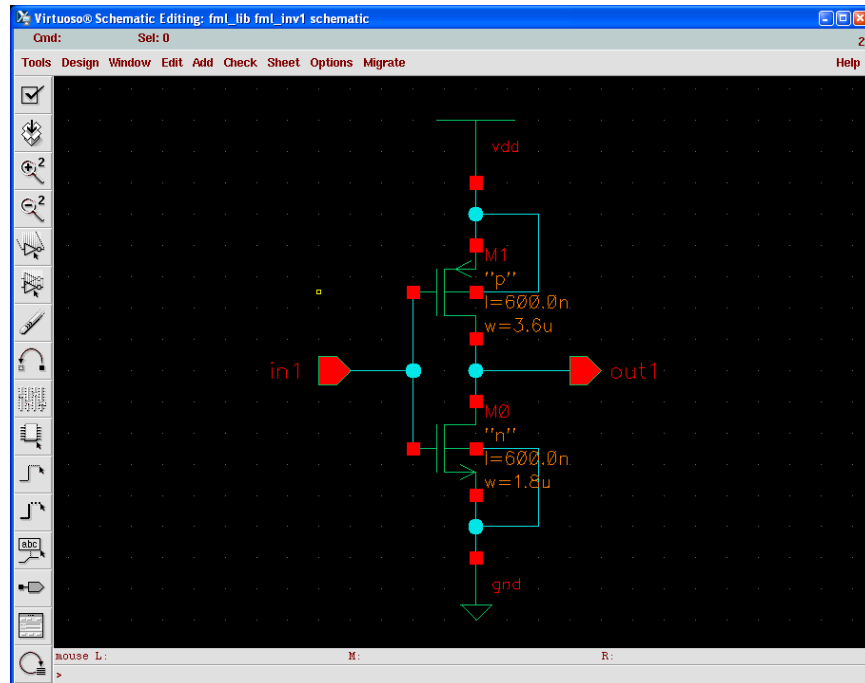
9. PROJECT REPORT

Your EDA Tutorial 3 PDF report will include the following:

1. Screen shot of your RTL simulation Console window expanded to show the log messages and that simulation is complete.
2. Screen shot of your Netlist simulation Console window expanded to show the log messages and that simulation is complete.
3. Screen shot of your RTL simulation Waveform window showing all signals visible within the module *test* and zoomed in to show simulation time between 99,990,000ns → 99,993,000ns.
4. Screen shot of your Netlist simulation Waveform window showing all signals visible within the module *test* and zoomed in to show simulation time between 99,990,000ns → 99,993,000ns.
5. Document the following from the logic synthesis report:
 - a. Total cell area for the pre-scan netlist
 - b. Total number of cells in the pre-scan netlist
 - c. Worst case timing path for the pre-scan netlist
 - d. Power consumption (dynamic and leakage) for pre-scan netlist
 - e. Total cell area for the post-scan netlist
 - f. Total number of cells in the post-scan netlist
 - g. Worst case timing path for the post-scan netlist
 - h. Power consumption (dynamic and leakage) for post-scan netlist
6. Document the following from the various timing analyzer reports:
 - a. Worst case timing path for the post-scan netlist
 - b. Total timing analysis coverage for the post-scan netlist

APPENDIX

How to get rid of the black background for printing and documentation purposes:



After you copy/paste the window containing the schematic in a word file, using ALT-PRT_SCR, right click on the image and select SHOW PICTURE TOOLBAR.

Once the toolbar opens, towards the right hand side there is a command SELECT TRANSPARENT COLOR. Click on it and then click anywhere on the black background in your window. See the result below.

