



**Fall Semester 2021-2022**

**ECE5030 - Scripting Languages for VLSI Design Automation**

**M.Tech VLSI Design**

**School of Electronics Engineering**

**Vellore Institute of Technology**

**Name: Shreyas S Bagi**

**Register Number: 21MVD0086**

**Slot: L3+L4**

## Lab Task 04

### TCL – File Handling and Automation

#### Section 1 Decimal to other Base Conversion

**Aim:** Write a TCL script that prompts the user to enter a decimal number (number without fractions). The script should convert the given decimal number to its corresponding binary, hexadecimal and octal and display the results to the user. The program should continue doing the task until the user enters “exit”.

#### Source Code or TCL Script code:

```
#!/usr/bin/tclsh
# Read Decimal Number
##### Command Description acting like Functions #####
proc Decimal_Conversion {Decimal} {
    set i 0
    # Converting to Decimal to Binary only Integer part
    set binary_num [list ]
    set binary_num1 [list ]
    set a $Decimal
    while {$a != 0} {
        #puts "Entered loop"
        set x [expr $a%2]
        #puts "$x"
        lappend binary_num $x
        set a [expr $a/2]
    }
    set binary_num [lreverse $binary_num]
    set bin_num [string map {" " ""} $binary_num]
    puts "The Binary Number of $Decimal is : $bin_num"
    # Converting to Decimal to Octal only integer part
    set octal_num [list ]
    set a $Decimal
    while {$a != 0} {
        #puts "Entered loop"
        set x [expr $a%8]
        #puts "$x"
        lappend octal_num $x
        set a [expr $a/8]
    }
    set octal_num [lreverse $octal_num]
    set oct_num [string map {" " ""} $octal_num]
    puts "The Octal Number of $Decimal is : $oct_num"
    # Converting to Decimal to Hexadecimal only integer part
    set hexdec_num [list ]
    set a $Decimal
    while {$a != 0} {
```

```
#puts "Entered loop"
set x [expr $a%16]
#puts "$x"
if {$x == 10} {
set x A
} elseif {$x == 11} {
set x B
} elseif {$x == 12} {
set x C
} elseif {$x == 13} {
set x D
} elseif {$x == 14} {
set x E
} elseif {$x == 15} {
set x F
}
lappend hexdec_num $x
set a [expr $a/16]
}
set hexdec_num [lreverse $hexdec_num]
set hex_num [string map {" " ""} $hexdec_num]
puts "The Hexadecimal Number of $Decimal is : $hex_num"
}

puts "Enter The Decimal Number: "
gets stdin Decimal
puts [Decimal_Conversion $Decimal]
set monitor 1
puts "#####"
while {$monitor != 0} {
puts "Do you convert Some more "
puts "Just Enter the Decimal Number"
puts "To quit please type quit"
gets stdin Onceagain
set Decimal $Onceagain
if {$Onceagain == "quit"} {
puts "Ending the Conversion"
set monitor 0
puts "BYE BYE BYE BYE BYE BYE BYE BYE"
exit
} else {
set i 1
incr monitor
}
while {"quit" != $Decimal && $i} {
puts [Decimal_Conversion $Decimal]
set i 0;
}
}
```

## Output Screenshots :

**Example 1 :** User has given decimal input 255, 123 and quit

```

Applications Places System 21MVD0086@TT237A SENSE010:~/Scripting/TCL/TCL_Practice_Lab
21MVD0086@TT237A SENSE010:~/Scripting/TCL/TCL_Practice_Lab
File Edit View Search Terminal Help

[21MVD0086@TT237A SENSE010 TCL_Practice_Lab]$ tclsh decmial_conversion_d.tcl
Enter The Decimal Number:
255
The Binary Number of 255 is : 11111111
The Octal Number of 255 is : 377
The Hexadecimal Number of 255 is : FF

#####
Do you convert Some more
Just Enter the Decimal Number
To quit please type quit
123
The Binary Number of 123 is : 1111011
The Octal Number of 123 is : 173
The Hexadecimal Number of 123 is : 7B

Do you convert Some more
Just Enter the Decimal Number
To quit please type quit
255
The Binary Number of 255 is : 11111111
The Octal Number of 255 is : 377
The Hexadecimal Number of 255 is : FF

Do you convert Some more
Just Enter the Decimal Number
To quit please type quit
quit
Ending the Conversion
BYE BYE BYE BYE BYE BYE BYE BYE
[21MVD0086@TT237A SENSE010 TCL_Practice_Lab]$
  
```

**Figure 1.1** In this screenshot shows the decimal conversion of 255, 123 and typing quit.

**Inference:**

1. Writing TCL Script and how to execute it in Terminal window.
2. Get familiarised with Variable declaration, Associative Arrays and List Data, Control Structure and Procedures using in the script.
3. Automation of VLSI Testvector Generation is achieved.

## Section 2 : Hexadecimal Generation using TCL

**Aim:** Write a TCL script to generate all the possible inputs in hexadecimal that can be applied as an input vector for any design module. Ask the user to provide the input vector size for generating the inputs.

### Source Code or TCL Script code:

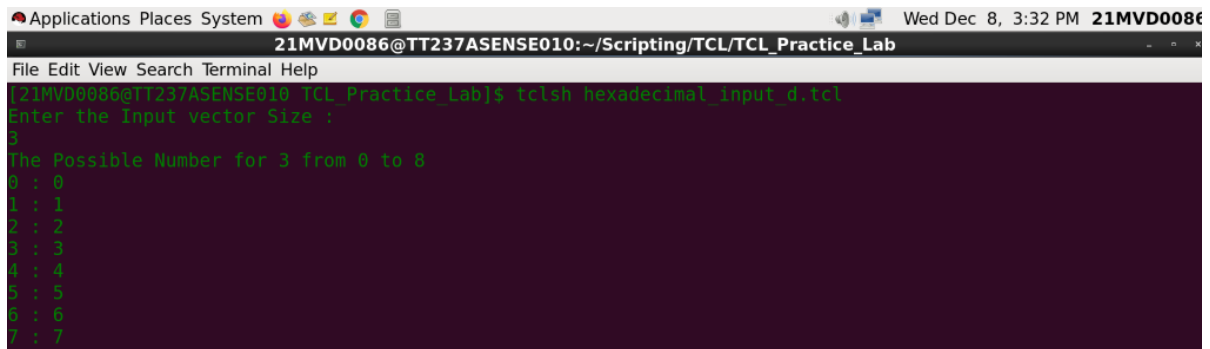
```
#!/usr/bin/tclsh

puts "Enter the Input vector Size : "

gets stdin Size_t
set Size [expr 2**$Size_t]
puts "The Possible Number for $Size_t from 0 to $Size"
for {set i 0} {$i<$Size} {incr i} {
# Converting to Decimal to Hexadecimal only Integer part
#puts "$i"
set hexdec_num [list ]
set a $i
if {$a == 0} {
set hexdec_num 0
puts "$i : $hexdec_num"
} elseif {$a == 1} {
set hexdec_num 1
puts "$i : $hexdec_num"
} else {
while {$a != 0} {
#puts "Entered loop"
set x [expr $a%16]
#puts "$x"
if {$x == 10} {
set x A
} elseif {$x == 11} {
set x B
} elseif {$x == 12} {
set x C
} elseif {$x == 13} {
set x D
} elseif {$x == 14} {
set x E
} elseif {$x == 15} {
set x F
}
lappend hexdec_num $x
set a [expr $a/16]
}
set hexdec_num [lreverse $hexdec_num]
set hex_num [string map {" " ""} $hexdec_num]
puts "$i : $hex_num"
}
```

}

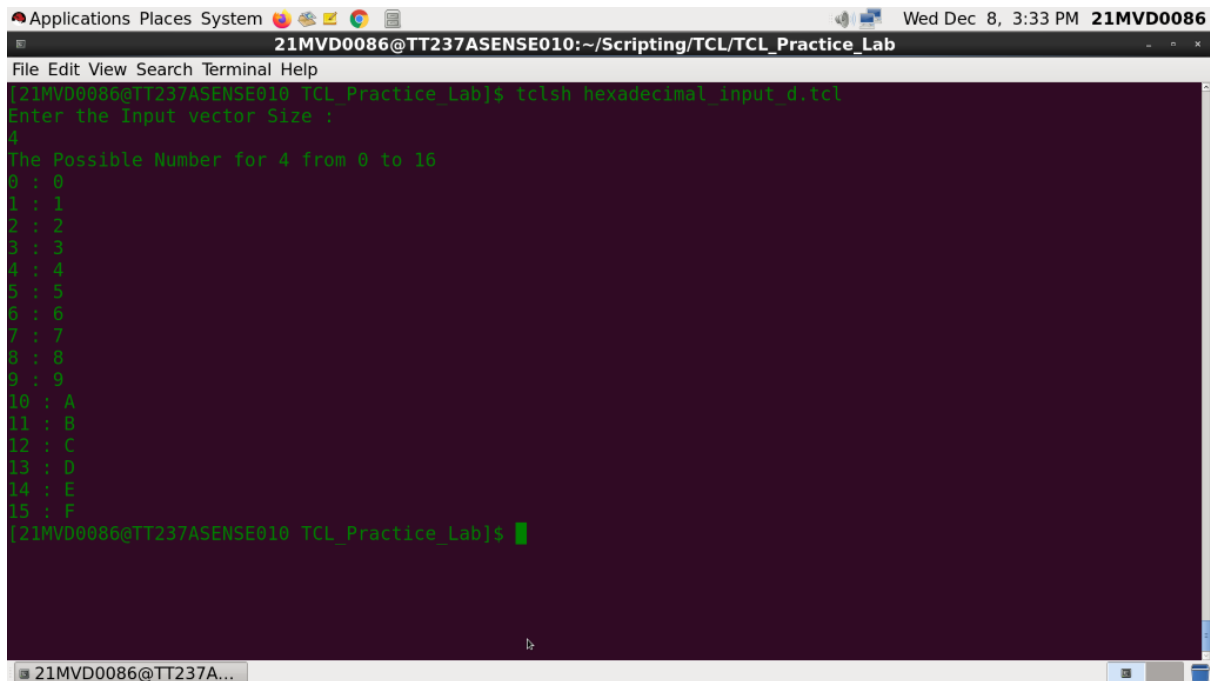
## Output Screenshots :



A terminal window titled '21MVD0086@TT237A5ENSE010:~/Scripting/TCL/TCL\_Practice\_Lab'. The prompt is '[21MVD0086@TT237A5ENSE010 TCL\_Practice\_Lab]\$'. The user has entered 'tclsh hexadecimal\_input\_d.tcl'. The program prompts 'Enter the Input vector Size :'. The user has entered '3'. The program outputs 'The Possible Number for 3 from 0 to 8'. The output is a list of numbers from 0 to 7, each preceded by its index from 0 to 7.

```
[21MVD0086@TT237A5ENSE010 TCL_Practice_Lab]$ tclsh hexadecimal_input_d.tcl
Enter the Input vector Size :
3
The Possible Number for 3 from 0 to 8
0 : 0
1 : 1
2 : 2
3 : 3
4 : 4
5 : 5
6 : 6
7 : 7
```

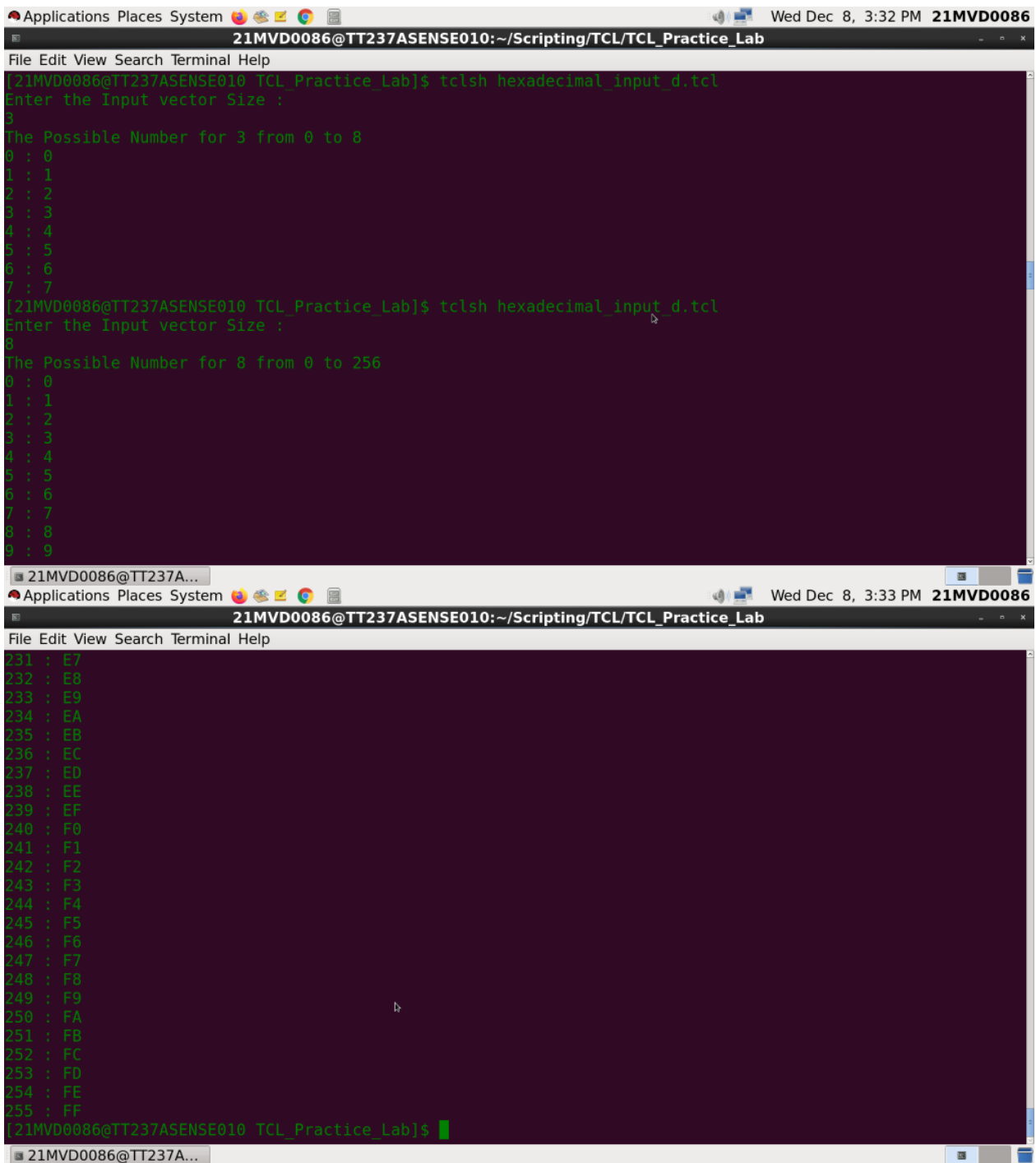
**Figure 2.1** The Possible Number generation for 3-bit Number.



A terminal window titled '21MVD0086@TT237A5ENSE010:~/Scripting/TCL/TCL\_Practice\_Lab'. The prompt is '[21MVD0086@TT237A5ENSE010 TCL\_Practice\_Lab]\$'. The user has entered 'tclsh hexadecimal\_input\_d.tcl'. The program prompts 'Enter the Input vector Size :'. The user has entered '4'. The program outputs 'The Possible Number for 4 from 0 to 16'. The output is a list of numbers from 0 to 15, each preceded by its index from 0 to 15. Numbers 10 through 15 are represented in hexadecimal (A through F).

```
[21MVD0086@TT237A5ENSE010 TCL_Practice_Lab]$ tclsh hexadecimal_input_d.tcl
Enter the Input vector Size :
4
The Possible Number for 4 from 0 to 16
0 : 0
1 : 1
2 : 2
3 : 3
4 : 4
5 : 5
6 : 6
7 : 7
8 : 8
9 : 9
10 : A
11 : B
12 : C
13 : D
14 : E
15 : F
[21MVD0086@TT237A5ENSE010 TCL_Practice_Lab]$
```

**Figure 2.2** The Possible Number generation for 4-bit Number.



The image consists of two screenshots of a terminal window. The top screenshot shows the execution of a TCL script named 'hexadecimal\_input\_d.tcl'. The user enters '3' for the input vector size, and the script outputs a list of possible numbers from 0 to 7. The bottom screenshot shows the same script being executed with an input vector size of '8', resulting in a list of possible numbers from 0 to 255, displayed in hexadecimal format (e.g., E7, E8, ..., FF).

```
21MVD0086@TT237AENSE010:~/Scripting/TCL/TCL_Practice_Lab
File Edit View Search Terminal Help
[21MVD0086@TT237AENSE010 TCL_Practice_Lab]$ tclsh hexadecimal_input_d.tcl
Enter the Input vector Size :
3
The Possible Number for 3 from 0 to 8
0 : 0
1 : 1
2 : 2
3 : 3
4 : 4
5 : 5
6 : 6
7 : 7
[21MVD0086@TT237AENSE010 TCL_Practice_Lab]$ tclsh hexadecimal_input_d.tcl
Enter the Input vector Size :
8
The Possible Number for 8 from 0 to 256
0 : 0
1 : 1
2 : 2
3 : 3
4 : 4
5 : 5
6 : 6
7 : 7
8 : 8
9 : 9
21MVD0086@TT237AENSE010:~/Scripting/TCL/TCL_Practice_Lab
File Edit View Search Terminal Help
231 : E7
232 : E8
233 : E9
234 : EA
235 : EB
236 : EC
237 : ED
238 : EE
239 : EF
240 : F0
241 : F1
242 : F2
243 : F3
244 : F4
245 : F5
246 : F6
247 : F7
248 : F8
249 : F9
250 : FA
251 : FB
252 : FC
253 : FD
254 : FE
255 : FF
[21MVD0086@TT237AENSE010 TCL_Practice_Lab]$
```

**Figure 2.3** The Possible Number generation for 8-bit Number.



**Inference:**

1. Writing TCL Script and how to execute it in Terminal window.
2. Get familiarised with Variable declaration, Associative Arrays and List Data, Control Structure and Procedures using in the script.
3. Automation of VLSI Testvector Generation is achieved.
4. Generation of Possible Testcases for n-bit Variable.

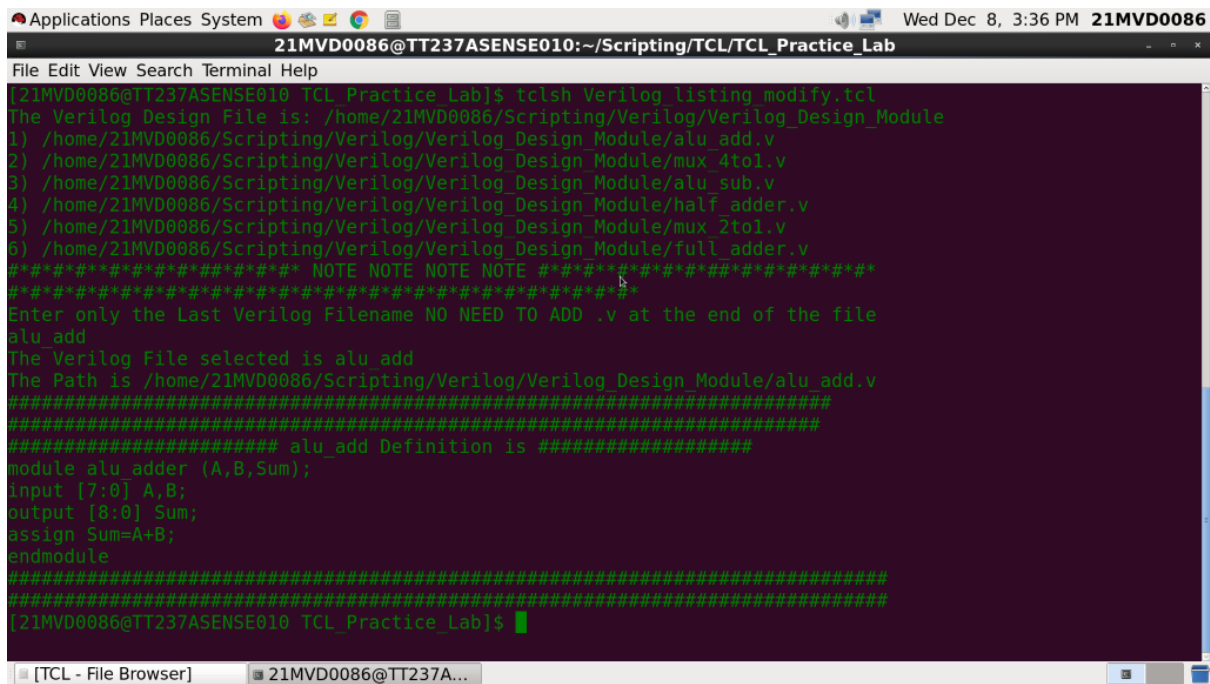
### Section 3 : Verilog File Listing

**Aim:** Write a TCL script that prompts the user with the list of .v files in the directory. Ask the user to enter any of the file to be read. Pick the file from the directory and display the contents of that file, line by line.

#### Source Code or TCL Script code:

```
#!/usr/bin/tclsh
set Dir "/home/21MVD0086/Scripting/Verilog/Verilog_Design_Module"
puts -nonewline "The Verilog Design File is: $Dir\n"
#####
set folderinput [glob -d $Dir *.v]
set j 0
foreach i $folderinput {
incr j
puts "$j) $i"
}
#####
##### User Selecting Verilog File #####
puts "#####"
NOTE NOTE NOTE NOTE #####
puts "#####"
puts "Enter only the Last Verilog Filename NO NEED TO ADD .v at the end of the file"
gets stdin Verilog_file_select
puts "The Verilog File selected is $Verilog_file_select"
set vext {.v}
set E {/}
set Verilogselect [concat $Dir$E$Verilog_file_select$vext]
puts "The Path is $Verilogselect"
puts
"#####"
puts
"#####"
puts "##### $Verilog_file_select Definition is
#####"
set FH1 [open "$Verilogselect" r]
while {[gets $FH1 data] >= 0} {
puts "$data"
}
close $FH1
puts
"#####"
#####
puts
"#####"
#####
```

## Output Screenshots :

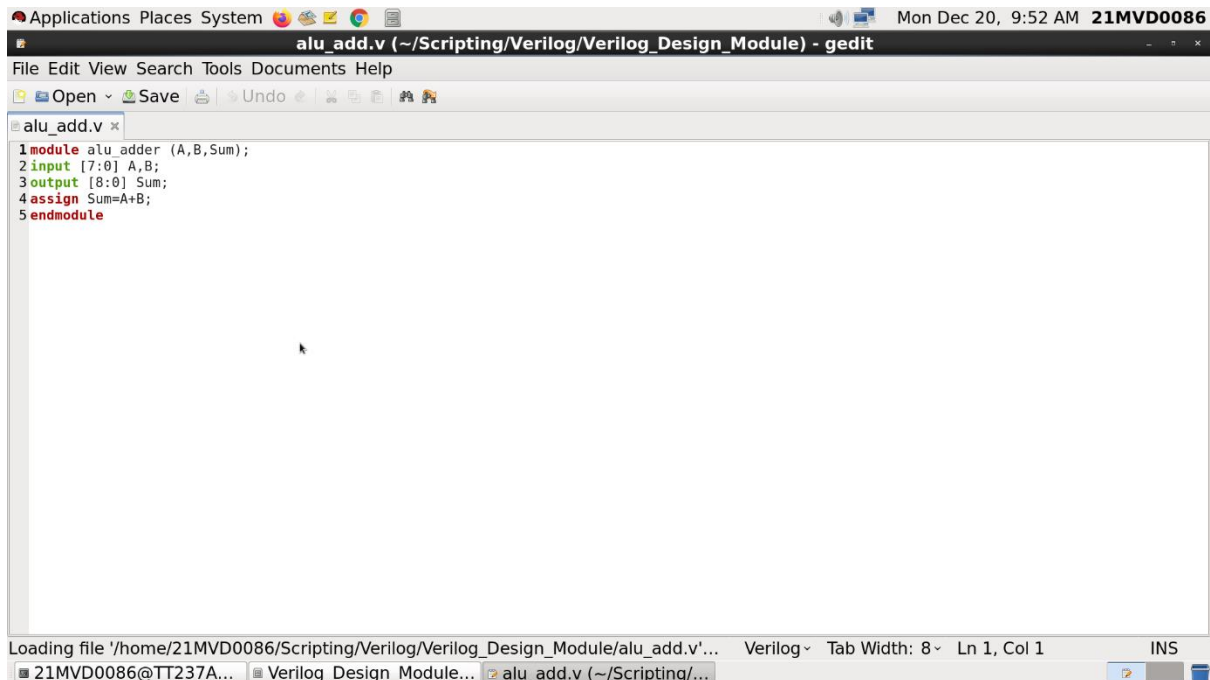


```

21MVD0086@TT237A5ENSE010:~/Scripting/TCL/TCL_Practice_Lab
File Edit View Search Terminal Help
[21MVD0086@TT237A5ENSE010 TCL Practice Lab]$ tclsh Verilog_listing_modify.tcl
The Verilog Design File is: /home/21MVD0086/Scripting/Verilog/Verilog_Design_Module
1) /home/21MVD0086/Scripting/Verilog/Verilog_Design_Module/alu_add.v
2) /home/21MVD0086/Scripting/Verilog/Verilog_Design_Module/mux_4to1.v
3) /home/21MVD0086/Scripting/Verilog/Verilog_Design_Module/alu_sub.v
4) /home/21MVD0086/Scripting/Verilog/Verilog_Design_Module/half_adder.v
5) /home/21MVD0086/Scripting/Verilog/Verilog_Design_Module/mux_2to1.v
6) /home/21MVD0086/Scripting/Verilog/Verilog_Design_Module/full_adder.v
***** NOTE NOTE NOTE NOTE *****
Enter only the Last Verilog Filename NO NEED TO ADD .v at the end of the file
alu_add
The Verilog File selected is alu add
The Path is /home/21MVD0086/Scripting/Verilog/Verilog_Design_Module/alu_add.v
*****
***** alu_add Definition is *****
module alu adder (A,B,Sum);
input [7:0] A,B;
output [8:0] Sum;
assign Sum=A+B;
endmodule
*****
[21MVD0086@TT237A5ENSE010 TCL Practice Lab]$

```

**Figure 3.1** The Screenshot shows Verilog files present in the current directory and user selected Alu adder file.



```

Applications Places System Mon Dec 20, 9:52 AM 21MVD0086
alu_add.v (~/.Scripting/Verilog/Verilog_Design_Module) - gedit
File Edit View Search Tools Documents Help
Open Save Undo
alu_add.v x
1 module alu adder (A,B,Sum);
2 input [7:0] A,B;
3 output [8:0] Sum;
4 assign Sum=A+B;
5 endmodule
Loading file '/home/21MVD0086/Scripting/Verilog/Verilog_Design_Module/alu_add.v'... Verilog Tab Width: 8 Ln 1, Col 1 INS
21MVD0086@TT237A... Verilog_Design_Module... alu_add.v (~/.Scripting/...

```

**Figure 3.2** The Screenshot shows Alu adder Verilog file.

```

21MVD0086@TT237A SENSE010:~/Scripting/TCL/TCL_Practice_Lab
File Edit View Search Terminal Help
4) /home/21MVD0086/Scripting/Verilog/Verilog_Design_Module/half_adder.v
5) /home/21MVD0086/Scripting/Verilog/Verilog_Design_Module/mux_2to1.v
6) /home/21MVD0086/Scripting/Verilog/Verilog_Design_Module/full_adder.v
##### NOTE NOTE NOTE #####
#####
Enter only the Last Verilog Filename NO NEED TO ADD .v at the end of the file
mux_4to1
The Verilog File selected is mux_4to1
The Path is /home/21MVD0086/Scripting/Verilog/Verilog_Design_Module/mux_4to1.v
#####
##### mux_4to1 Definition is #####
module mux_4to1 (Y,I,Sel);
input [3:0] I;
input [1:0] Sel;
output Y;
wire Y1,Y2;
wire [1:0]YX;
mux_2to1 G1 (Y1,I[1:0],Sel[0]);
mux_2to1 G2 (Y2,I[3:2],Sel[0]);
assign YX={Y2,Y1};
mux_2to1G2 (Y,YX[1:0],Sel[1]);
endmodule
#####
[21MVD0086@TT237A SENSE010 TCL_Practice_Lab]$

```

**Figure 3.3** The Screenshot shows Verilog files present in the current directory and user selected Mux 4 to 1.

```

mux_4to1.v (~ /Scripting/Verilog/Verilog_Design_Module) - gedit
File Edit View Search Tools Documents Help
Open Save Undo Redo
mux_4to1.v x
1 module mux_4to1 (Y,I,Sel);
2 input [3:0] I;
3 input [1:0] Sel;
4 output Y;
5 wire Y1,Y2;
6 wire [1:0]YX;
7 mux_2to1 G1 (Y1,I[1:0],Sel[0]);
8 mux_2to1 G2 (Y2,I[3:2],Sel[0]);
9 assign YX={Y2,Y1};
10 mux_2to1G2 (Y,YX[1:0],Sel[1]);
11 endmodule

```

**Figure 3.4** The Screenshot shows Mux 4 to 1 Verilog file.

**Inference:**

1. Writing TCL Script and how to execute it in Terminal window.
2. Get familiarised with Variable declaration, Associative Arrays and List Data, Control Structure and Procedures using in the script.
3. Verilog File Listing and Selection of file.
4. Reading the contents of file and printing the file.

## Section 4 : Verilog Simulation TCL Automation

**Aim:** Write a TCL script such that the script should prompt the user with the list of .v files in the directory. Ask the user to enter the list of files to be compiled and simulated. The script should then automatically compile the corresponding design and the testbench modules, perform the simulation. Create a separate directory and save the contents of the transcript in separate log files named as design\_name.log.

### Source Code or TCL Script code:

```
#!/usr/bin/tclsh
#../././Scripting/Verilog/
set Dir_name $argv
puts "The Directory Path: $Dir_name"
puts "$Dir_name"
set listfiles [glob -dir $Dir_name *]
set i 1
foreach filevar $listfiles {
puts "$i) $filevar"
incr i
}
puts "-----NOTE-----"
puts "Enter The last Directory of the Path only"
puts "-_*_*_*_*_*NOTE-*_*_*_* NOTE -*_*NOTE-*_*NOTE-*_*_*_*_*_*_*_*"
puts "*****Enter the Only Design File*****"
gets stdin folderselect
puts "$folderselect"
set Verilog_name [concat $Dir_name$folderselect]
set veriloglist [glob -dir $Verilog_name *.v]
set j 1
foreach filevar1 $veriloglist {
puts "$j) $filevar1"
incr j
}
puts "Enter the Verilog file with ----- NO extension ----provide from above list"
gets stdin Verilog_file
set VERIDREAD $Verilog_file
set verilogmodule $Verilog_file
set ext {.v}
set fileform {/}
set Verilog_file [concat $Dir_name$folderselect$fileform$Verilog_file$ext]
puts "$Verilog_file"
#####
puts "-_*_*_*_*_*NOTE-*_*_*_* NOTE -*_*NOTE-*_*NOTE-*_*_*_*_*_*_*_*"
puts "-----NOTE-----"
puts "***Enter the Only Testbench Module File***"
puts "***Enter the Only Testbench Module File***"
puts "The Directory Path: $Dir_name"
puts "Type in Keyboard as Testbench"
```

```
gets stdin folderselect
set Verilog_name [concat $Dir_name$folderselect]
set veriloglist [glob -dir $Verilog_name *.v]
set j 1
foreach filevar1 $veriloglist {
puts "$j) $filevar1"
incr j
}
puts "Enter the Verilog file with ----- NO extension ----provide from above list"
gets stdin Verilogtb_file
set VERITBREAD $Verilogtb_file
set verilogtbmodule $Verilogtb_file
set ext {.v}
set fileform {/}
set Verilogtb_file [concat $Dir_name$folderselect$fileform$Verilogtb_file$ext]
puts "$Verilogtb_file"
#####

puts "The files to be compiled"
set verilog_files [list $Verilog_file $Verilogtb_file]
foreach i $verilog_files {
puts "The File location is $i"
}

puts "[lindex $verilog_files 0]"
puts "[lindex $verilog_files 1]"

exec vlib work
exec vmap work work
exec vlog [lindex $verilog_files 0] > $Dir_name/LOG_Files/compile.log
exec vlog [lindex $verilog_files 1] >> $Dir_name/LOG_Files/compile.log
exec vsim -c -do run.do work.$VERITBREAD >$Dir_name/LOG_Files/sim.log
```

## Output Screenshots :

The Verilog Directory has Verilog\_Design\_Module, Testbench.

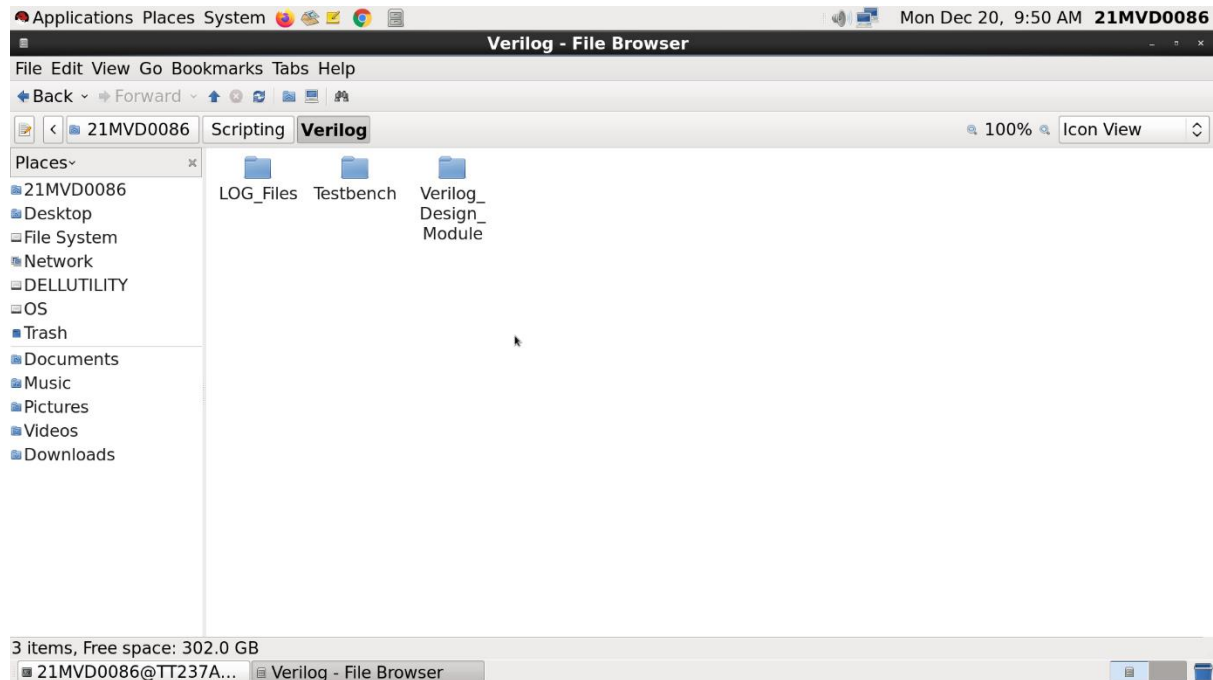


Figure 4.1 Verilog Directory having Testbench and Verilog\_Design\_Module.

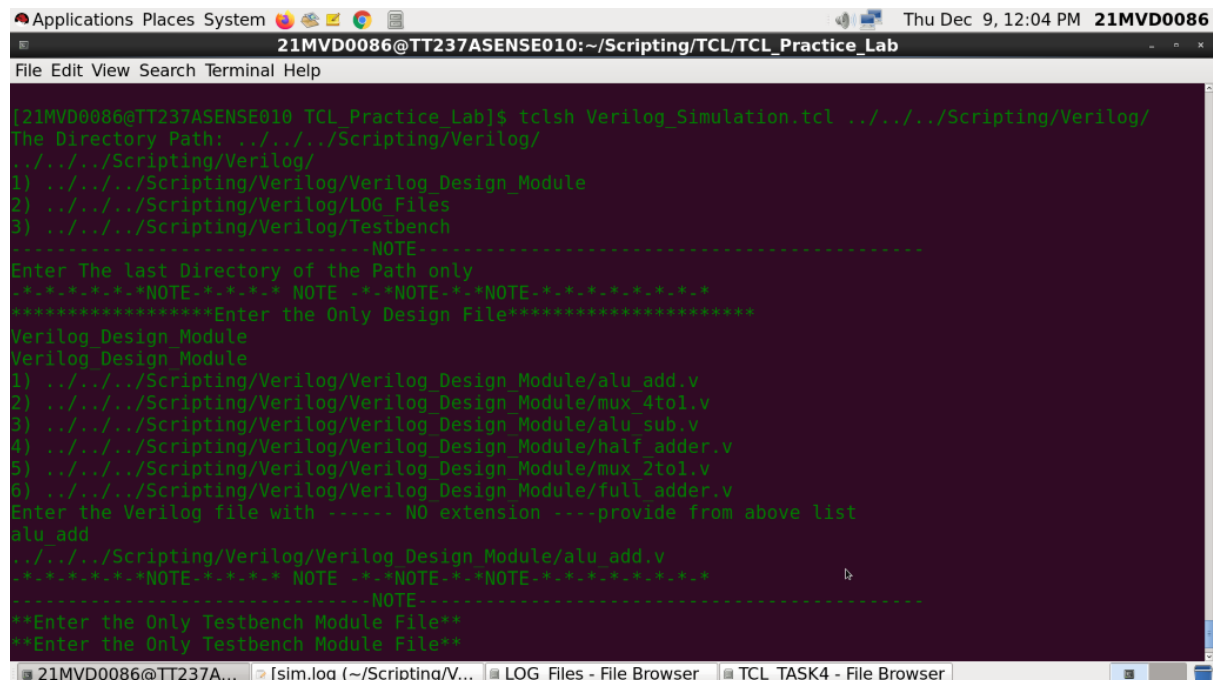
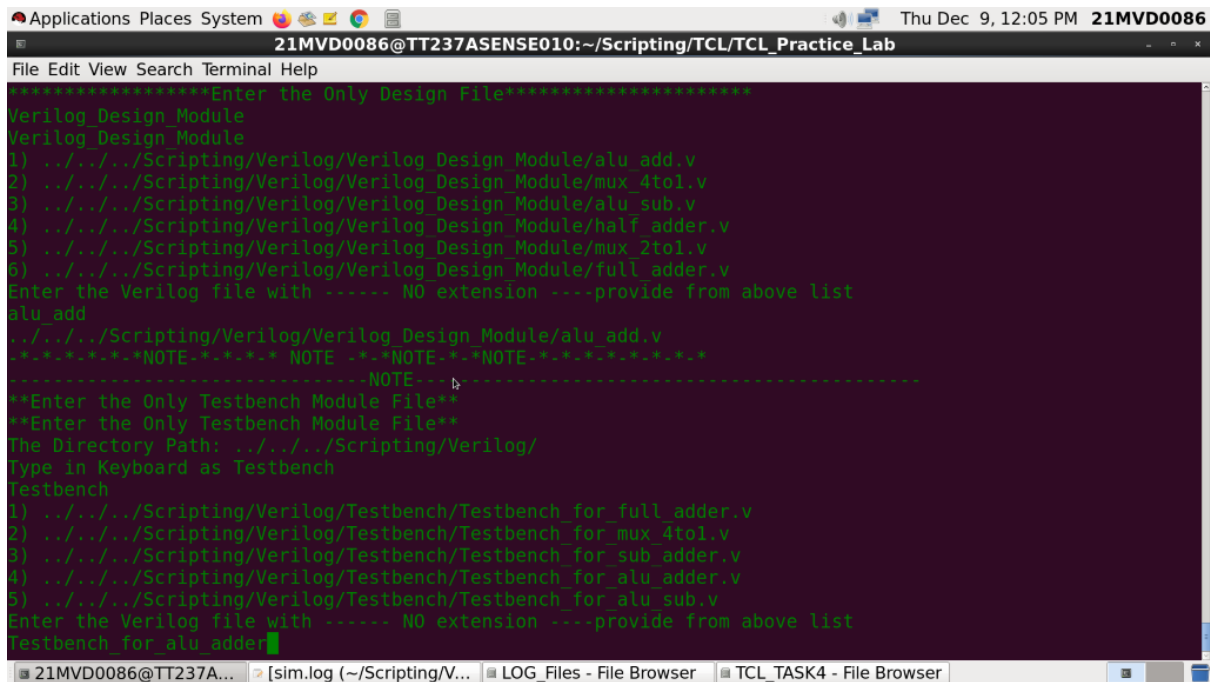


Figure 4.2 Prompting User to Verilog\_Design\_Module folder and listing the Verilog files present Verilog\_Design\_Module.



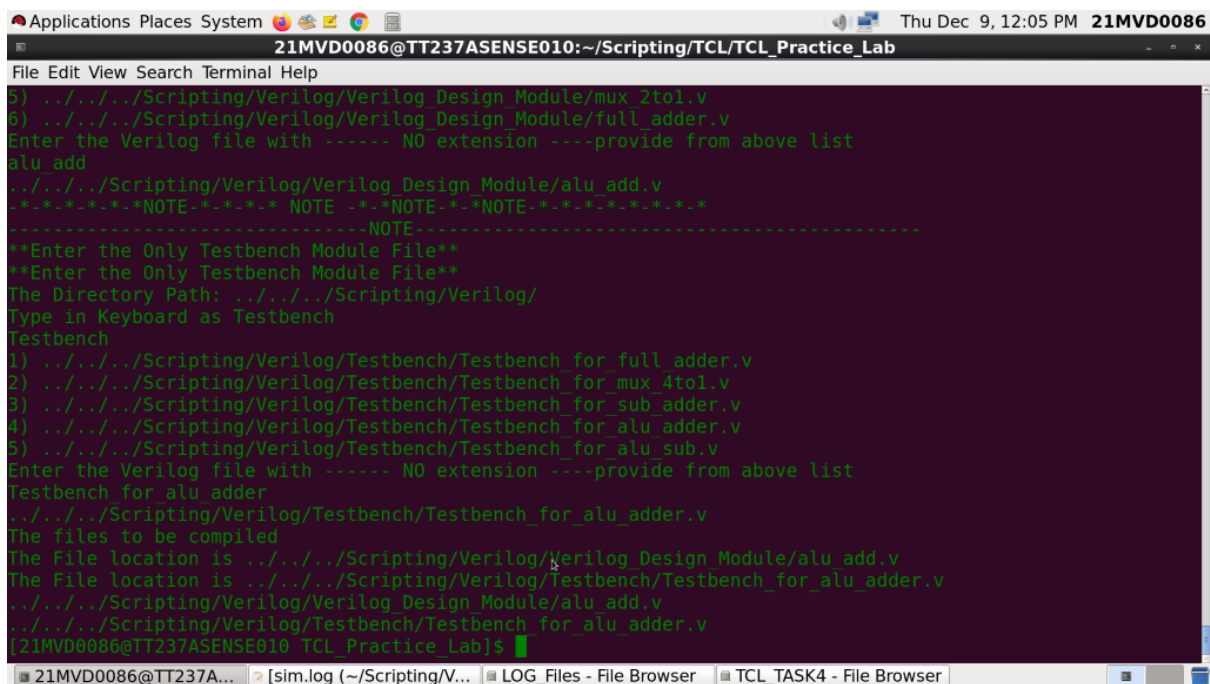


```

Applications Places System Thu Dec 9, 12:05 PM 21MVD0086
21MVD0086@TT237ASENSE010:~/Scripting/TCL/TCL_Practice_Lab
File Edit View Search Terminal Help
*****Enter the Only Design File*****
Verilog_Design_Module
Verilog_Design_Module
1) ../../../../Scripting/Verilog/Verilog_Design_Module/alu_add.v
2) ../../../../Scripting/Verilog/Verilog_Design_Module/mux_4to1.v
3) ../../../../Scripting/Verilog/Verilog_Design_Module/alu_sub.v
4) ../../../../Scripting/Verilog/Verilog_Design_Module/half_adder.v
5) ../../../../Scripting/Verilog/Verilog_Design_Module/mux_2to1.v
6) ../../../../Scripting/Verilog/Verilog_Design_Module/full_adder.v
Enter the Verilog file with ----- NO extension ----provide from above list
alu_add
../../../../Scripting/Verilog/Verilog_Design_Module/alu_add.v
*****NOTE***** NOTE *****NOTE*****NOTE*****
*****NOTE*****
**Enter the Only Testbench Module File**
**Enter the Only Testbench Module File**
The Directory Path: ../../../../Scripting/Verilog/
Type in Keyboard as Testbench
Testbench
1) ../../../../Scripting/Verilog/Testbench/Testbench_for_full_adder.v
2) ../../../../Scripting/Verilog/Testbench/Testbench_for_mux_4to1.v
3) ../../../../Scripting/Verilog/Testbench/Testbench_for_sub_adder.v
4) ../../../../Scripting/Verilog/Testbench/Testbench_for_alu_adder.v
5) ../../../../Scripting/Verilog/Testbench/Testbench_for_alu_sub.v
Enter the Verilog file with ----- NO extension ----provide from above list
Testbench_for_alu_adder
21MVD0086@TT237A... [sim.log (~/.Scripting/V... LOG_Files - File Browser TCL_TASK4 - File Browser

```

Figure 4.3 Prompting User to Testbench folder and listing the Verilog files present. Asking user to Enter proper testbench file corresponding to the design file.



```

Applications Places System Thu Dec 9, 12:05 PM 21MVD0086
21MVD0086@TT237ASENSE010:~/Scripting/TCL/TCL_Practice_Lab
File Edit View Search Terminal Help
3) ../../../../Scripting/Verilog/Verilog_Design_Module/mux_2to1.v
6) ../../../../Scripting/Verilog/Verilog_Design_Module/full_adder.v
Enter the Verilog file with ----- NO extension ----provide from above list
alu_add
../../../../Scripting/Verilog/Verilog_Design_Module/alu_add.v
*****NOTE***** NOTE *****NOTE*****NOTE*****
*****NOTE*****
**Enter the Only Testbench Module File**
**Enter the Only Testbench Module File**
The Directory Path: ../../../../Scripting/Verilog/
Type in Keyboard as Testbench
Testbench
1) ../../../../Scripting/Verilog/Testbench/Testbench_for_full_adder.v
2) ../../../../Scripting/Verilog/Testbench/Testbench_for_mux_4to1.v
3) ../../../../Scripting/Verilog/Testbench/Testbench_for_sub_adder.v
4) ../../../../Scripting/Verilog/Testbench/Testbench_for_alu_adder.v
5) ../../../../Scripting/Verilog/Testbench/Testbench_for_alu_sub.v
Enter the Verilog file with ----- NO extension ----provide from above list
Testbench_for_alu_adder
../../../../Scripting/Verilog/Testbench/Testbench_for_alu_adder.v
The files to be compiled
The File location is ../../../../Scripting/Verilog/Verilog_Design_Module/alu_add.v
The File location is ../../../../Scripting/Verilog/Testbench/Testbench_for_alu_adder.v
../../../../Scripting/Verilog/Verilog_Design_Module/alu_add.v
../../../../Scripting/Verilog/Testbench/Testbench_for_alu_adder.v
21MVD0086@TT237ASENSE010 TCL_Practice_Lab]$
21MVD0086@TT237A... [sim.log (~/.Scripting/V... LOG_Files - File Browser TCL_TASK4 - File Browser

```

Figure 4.4 The screenshot shows the Verilog file compiled in Modelsim and creates log files saved in LOG\_Files folder.

```

1Model Technology ModelSim ALTERA vlog 10.1d Compiler 2012.11 Nov  2
2  2012
3-- Compiling module alu_adder
4
5Top level modules:
6    alu_adder
7Model Technology ModelSim ALTERA vlog 10.1d Compiler 2012.11 Nov  2
8  2012
9-- Compiling module Testbench_for_alu_adder
10
11Top level modules:
12    Testbench_for_alu_adder

```

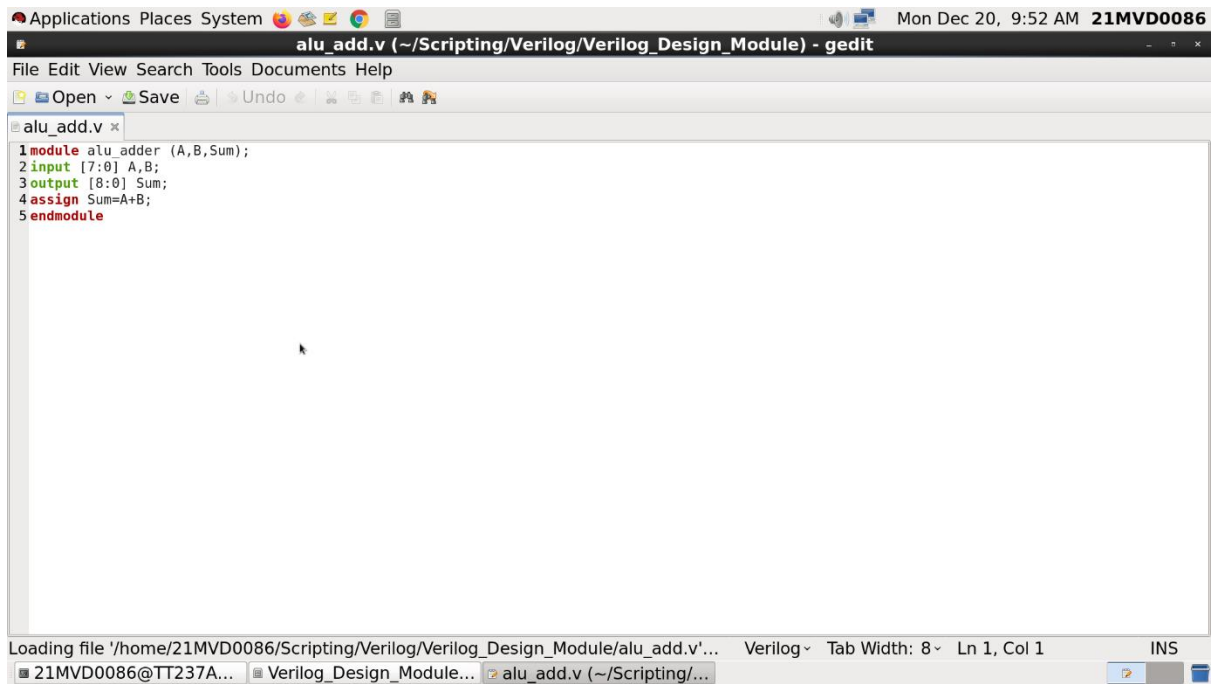
Figure 4.5 The compile log shows the design module name and testbench module name.

```

1Reading /home/altera/13.0/modelsim_ase/tcl/vsim/pref.tcl
2
3# 10.1d
4
5# vsim -do run.do -c work.Testbench_for_alu_adder
6# Loading work.Testbench_for_alu_adder
7# Loading work.alu_adder
8# do run.do
9#
10#      0 Sum=xxxxxxxx, A=xxxxxxxx,B=xxxxxxxx
11#      2 Sum=011011000, A=01010001,B=10000111
12#      4 Sum=011000101, A=11000101,B=00000000
13#      6 Sum=110000001, A=11100010,B=10011111
14#      8 Sum=10111110, A=11100111,B=01110111
15#     10 Sum=010011001, A=01000110,B=01010011
16#     12 Sum=000011001, A=00001011,B=00001110
17#     14 Sum=101001101, A=11000011,B=10001010
18#     16 Sum=011100111, A=01010000,B=10010111
19#     18 Sum=001111100, A=00010101,B=01100111
20#     20 Sum=101101011, A=01110100,B=11110111
21#     22 Sum=001111101, A=01010000,B=00101101
22#     24 Sum=001111111, A=00111101,B=01000010
23#     26 Sum=010100100, A=00100010,B=10000010
24#     28 Sum=100001011, A=01111101,B=10001110
25#
26# ** Note: $finish : ../../Scripting/Verilog/Testbench/Testbench_for_alu_adder.v(28)
27# Time: 30 ps Iteration: 0 Instance: /Testbench_for_alu_adder

```

Figure 4.6 The Sim log shows the simulation output.



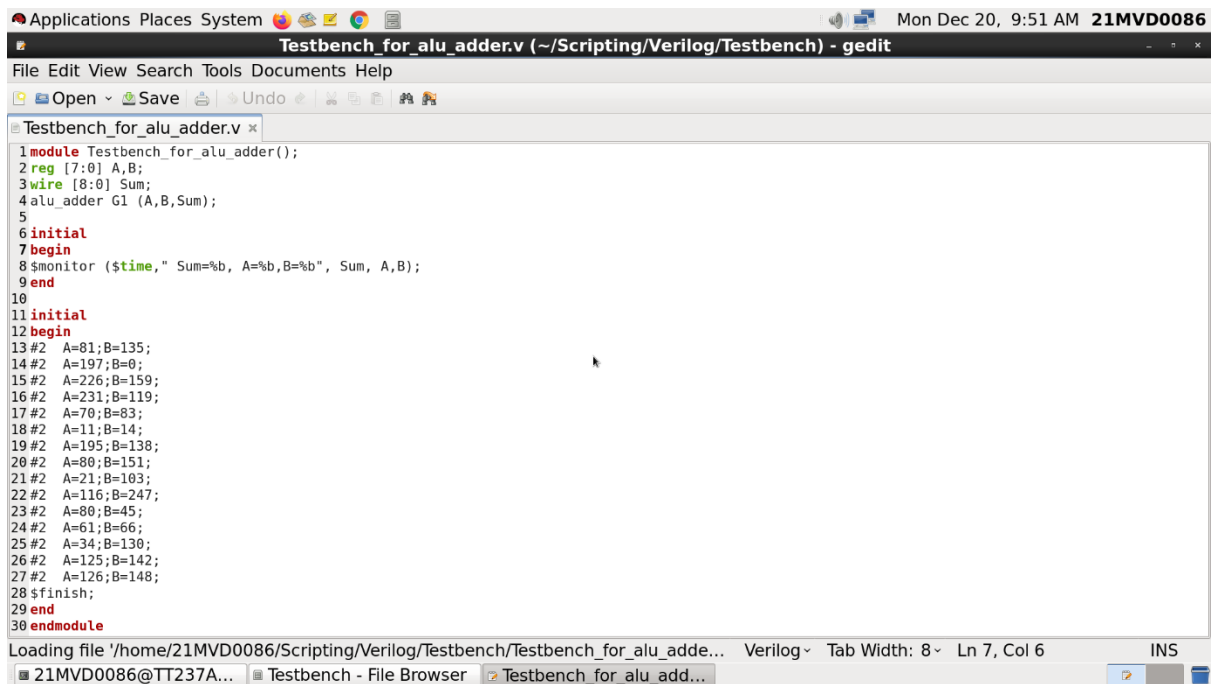
```

1 module alu_adder (A,B,Sum);
2 input [7:0] A,B;
3 output [8:0] Sum;
4 assign Sum=A+B;
5 endmodule

```

Loading file '/home/21MVD0086/Scripting/Verilog/Verilog\_Design\_Module/alu\_add.v'... Verilog Tab Width: 8 Ln 1, Col 1 INS

Figure 4.7 The Verilog Design Module File.



```

1 module Testbench_for_alu_adder();
2 reg [7:0] A,B;
3 wire [8:0] Sum;
4 alu_adder G1 (A,B,Sum);
5
6 initial
7 begin
8 $monitor ($time," Sum=%b, A=%b,B=%b", Sum, A,B);
9 end
10
11 initial
12 begin
13 #2 A=81;B=135;
14 #2 A=197;B=0;
15 #2 A=226;B=159;
16 #2 A=231;B=119;
17 #2 A=70;B=83;
18 #2 A=11;B=14;
19 #2 A=195;B=138;
20 #2 A=80;B=151;
21 #2 A=21;B=103;
22 #2 A=116;B=247;
23 #2 A=80;B=45;
24 #2 A=61;B=66;
25 #2 A=34;B=130;
26 #2 A=125;B=142;
27 #2 A=126;B=148;
28 $finish;
29 end
30 endmodule

```

Loading file '/home/21MVD0086/Scripting/Verilog/Testbench/Testbench\_for\_alu\_adder.v'... Verilog Tab Width: 8 Ln 7, Col 6 INS

Figure 4.8 The Verilog Testbench Module File.

```

21MVD0086@TT237A... Thu Dec 9, 12:07 PM 21MVD0086
21MVD0086@TT237A...:~/Scripting/TCL/TCL_Practice_Lab
File Edit View Search Terminal Help
3) ../../../../Scripting/Verilog/Verilog_Design_Module/mux_2to1.v
6) ../../../../Scripting/Verilog/Verilog_Design_Module/full_adder.v
Enter the Verilog file with ----- NO extension ----provide from above list
alu_sub
../../../../Scripting/Verilog/Verilog_Design_Module/alu_sub.v
*****NOTE***** NOTE *****NOTE*****NOTE*****
-----NOTE-----
**Enter the Only Testbench Module File**
**Enter the Only Testbench Module File**
The Directory Path: ../../../../Scripting/Verilog/
Type in Keyboard as Testbench
Testbench
1) ../../../../Scripting/Verilog/Testbench/Testbench_for_full_adder.v
2) ../../../../Scripting/Verilog/Testbench/Testbench_for_mux_4to1.v
3) ../../../../Scripting/Verilog/Testbench/Testbench_for_sub_adder.v
4) ../../../../Scripting/Verilog/Testbench/Testbench_for_alu_adder.v
5) ../../../../Scripting/Verilog/Testbench/Testbench_for_alu_sub.v
Enter the Verilog file with ----- NO extension ----provide from above list
Testbench_for_alu_sub
../../../../Scripting/Verilog/Testbench/Testbench_for_alu_sub.v
The files to be compiled
The File location is ../../../../Scripting/Verilog/Verilog_Design_Module/alu_sub.v
The File location is ../../../../Scripting/Verilog/Testbench/Testbench_for_alu_sub.v
../../../../Scripting/Verilog/Verilog_Design_Module/alu_sub.v
../../../../Scripting/Verilog/Testbench/Testbench_for_alu_sub.v
[21MVD0086@TT237A... TCL_Practice_Lab]$

```

Figure 4.9 The User has selected Alu Subtractor as design module and testbench file of it.

```

compile.log (~/.Scripting/Verilog/LOG_Files) - gedit
File Edit View Search Tools Documents Help
Open Save Undo Redo
Verilog_Simulation.tcl x sim.log x compile.log x
1 Model Technology ModelSim ALTERA vlog 10.1d Compiler 2012.11 Nov 2 2012
2 -- Compiling module alu_sub
3
4 Top level modules:
5     alu_sub
6 Model Technology ModelSim ALTERA vlog 10.1d Compiler 2012.11 Nov 2 2012
7 -- Compiling module Testbench_for_alu_sub
8
9 Top level modules:
10     Testbench_for_alu_sub

```

Figure 4.10 The User has selected Alu Subtractor as design module and testbench file of it.

```

1 Reading /home/altera/13.0/modelsim_ase/tcl/vsim/pref.tcl
2
3 # 10.1d
4
5 # vsim -do run.do -c work.Testbench_for_alu_sub
6 # Loading work.Testbench_for_alu_sub
7 # Loading work.alu_sub
8 # do run.do
9 #
10 #      0 Diff=xxxxxxxxxxxxxxxx, A=xxxxxxxxxxxxxxxx, B=xxxxxxxxxxxxxxxx
11 #      2 Diff=0001111100100100, A=1110101001100011, B=1010101100111111
12 #      4 Diff=1100000100111010, A=010111110100101, B=1101110100101111
13 #      6 Diff=01001000001100101, A=1100100110111101, B=0011100101011000
14 #      8 Diff=1101111011111111, A=0000000101111101, B=0100001111111110
15 #     10 Diff=1110011011101010, A=0010000110110010, B=0101001111011100
16 #     12 Diff=001010110110001001, A=1001001001010100, B=0010010011001011
17 #     14 Diff=1110110010101001, A=1000100111010000, B=1010110100100111
18 #     16 Diff=11101100111111001, A=0100100110111111, B=0110111111100110
19 #     18 Diff=11101001111001111, A=0101011001001111, B=1000001010000000
20 #     20 Diff=1111000101110001, A=0110100111011101, B=0111100001101100
21 #     22 Diff=0010101000110101, A=0111100001101010, B=0000111000110101
22 #     24 Diff=1110110000101011, A=1001111000000101, B=1100000111011010
23 #     26 Diff=11011101001000011, A=1001001001001000, B=1101100000000101
24 #     28 Diff=11100100000011100, A=0011000110101110, B=0110100110010010
25 #     30 Diff=01001100110001101, A=1010110110101110, B=0001010000100001
26 #
27 ** Note: $finish : ...../Scripting/Verilog/Testbench/Testbench_for_alu_sub.v(28)
28 Time: 40 ps Iteration: 0 Instance: /Testbench_for_alu_sub

```

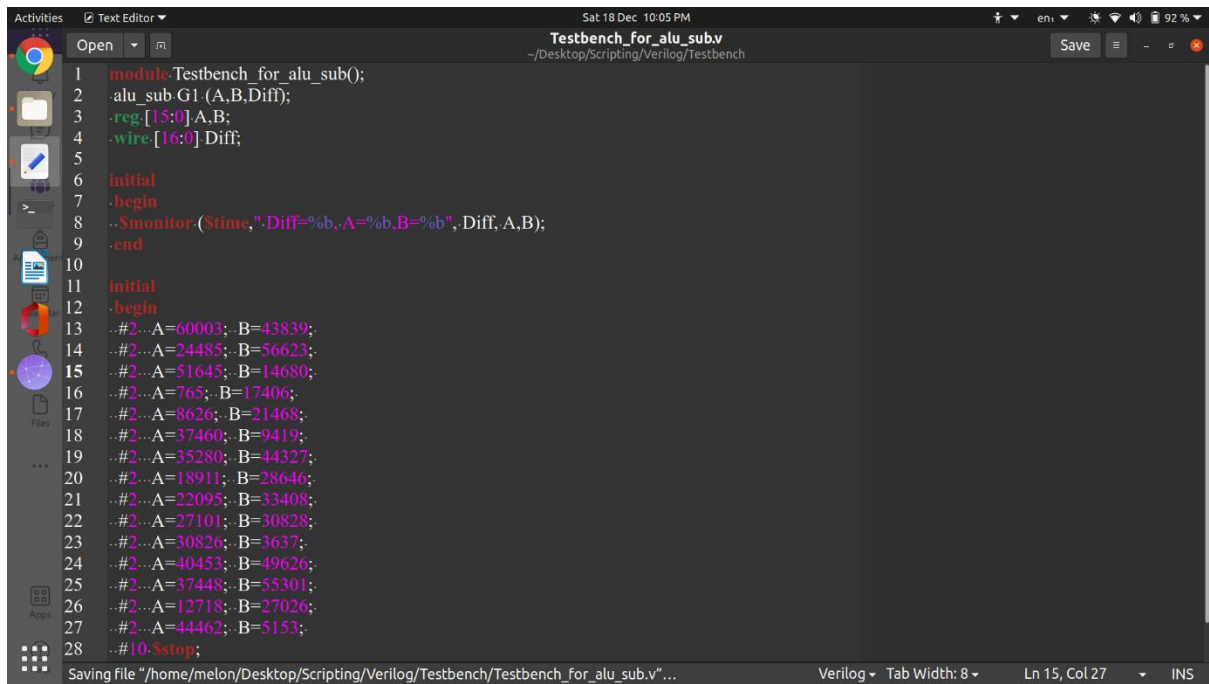
Figure 4.11 The Sim log shows the simulation output.

```

1 module alu_sub (A,B,Diff);
2 input [15:0] A,B;
3 output [16:0] Diff;
4 assign Diff=A-B;
5 endmodule

```

Figure 4.12 The Verilog Design Module File.



```

1 module Testbench_for_alu_sub();
2   alu_sub G1 (A,B,Diff);
3   reg [15:0] A,B;
4   wire [16:0] Diff;
5
6   initial
7   .begin
8     $monitor($time, "%Diff=%b, A=%b, B=%b", Diff, A, B);
9   .end
10
11   initial
12   .begin
13     #2...A=60003; B=43839;
14     #2...A=24485; B=56623;
15     #2...A=51645; B=14680;
16     #2...A=765; B=17406;
17     #2...A=8626; B=21468;
18     #2...A=37460; B=9419;
19     #2...A=35280; B=44327;
20     #2...A=18911; B=28646;
21     #2...A=22095; B=33408;
22     #2...A=27101; B=30828;
23     #2...A=30826; B=3637;
24     #2...A=40453; B=49626;
25     #2...A=37448; B=55301;
26     #2...A=12718; B=27026;
27     #2...A=44462; B=5153;
28     #10: $stop;
  
```

Figure 4.13 The Verilog Testbench Module File.

**Inference:**

1. Writing TCL Script and how to execute it in Terminal window.
2. Get familiarised with Variable declaration, Associative Arrays and List Data, Control Structure and Procedures using in the script.
3. Automating the Simulation of Verilog files without opening the Modelsim and using TCL commands.