Date:20/12/2021

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

Date:20/12/2021

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:

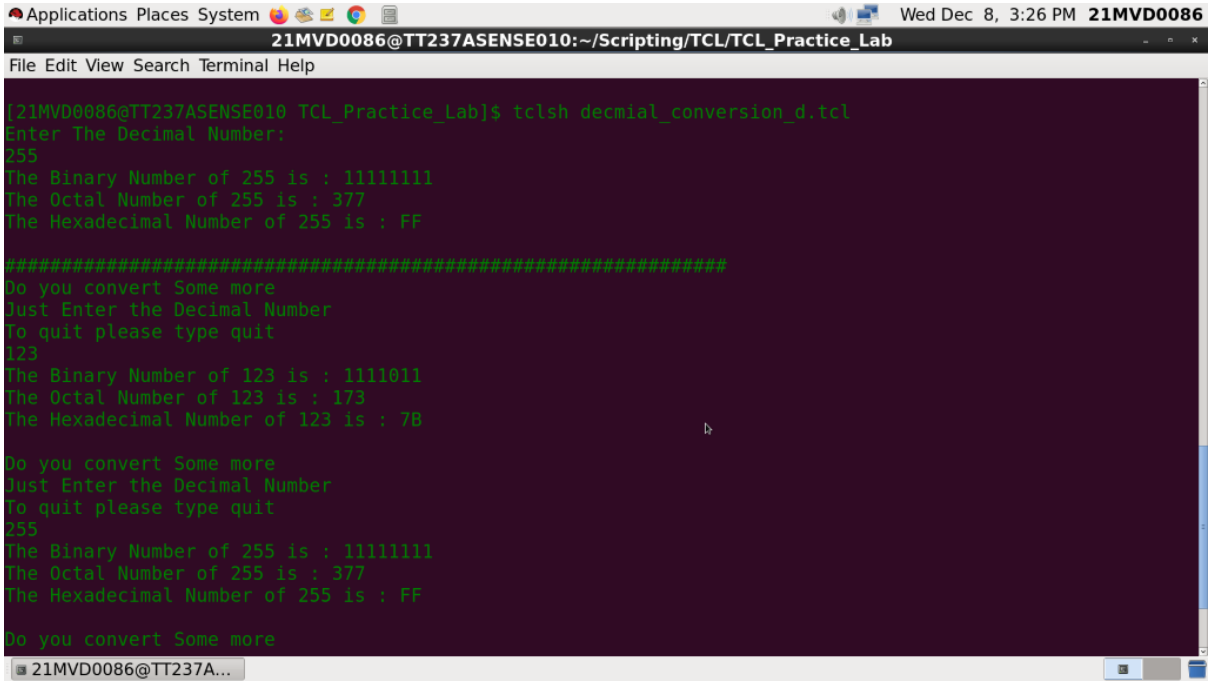
```
#! /usr/bin/tclsh
# Read Decimal Number
############# Command Description acting like Functions ###############
proc Decimal_Conversion {Decimal} {
set i 0
# Converting to Decimal to Binary only Integer part
set binary_num [list ]
set binary_num1 [list ]
set a $Decimal
while {$a != 0} {
#puts "Entered 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;
}
}
```

Date:20/12/2021

Output Screenshots :

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



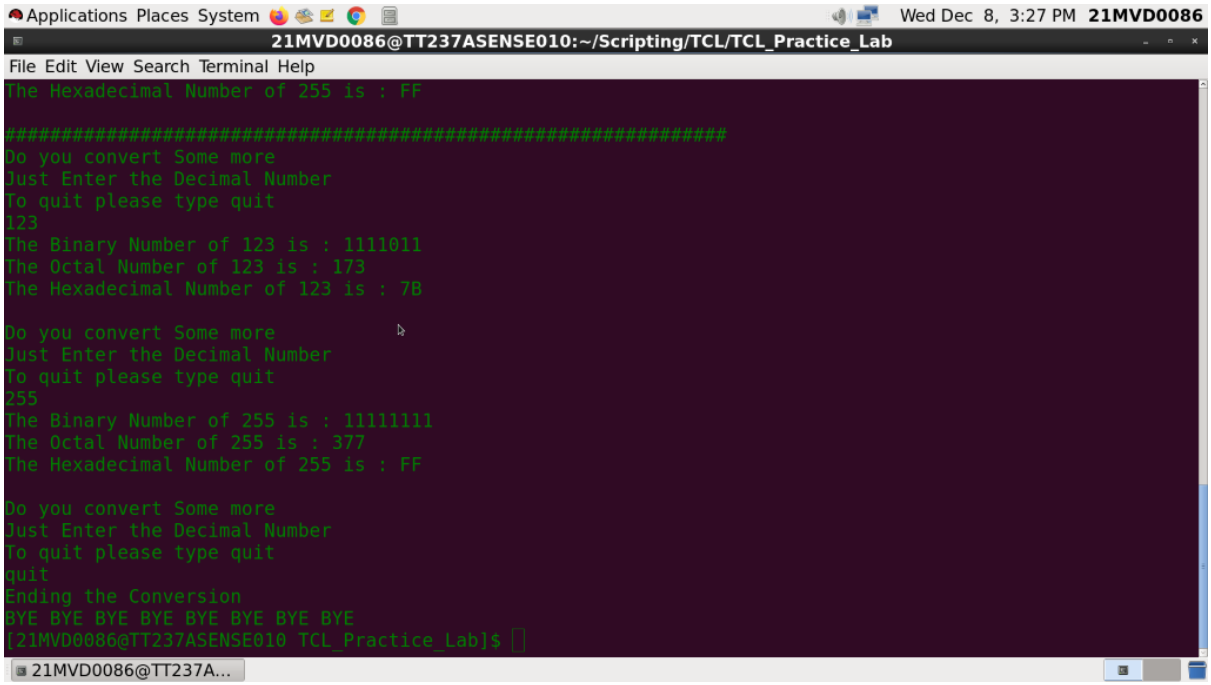


Figure 1.1 In this screenshot shows the decimal conversion of 255, 123 and typing quit.

Date:20/12/2021

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.

Date:20/12/2021

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 :

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

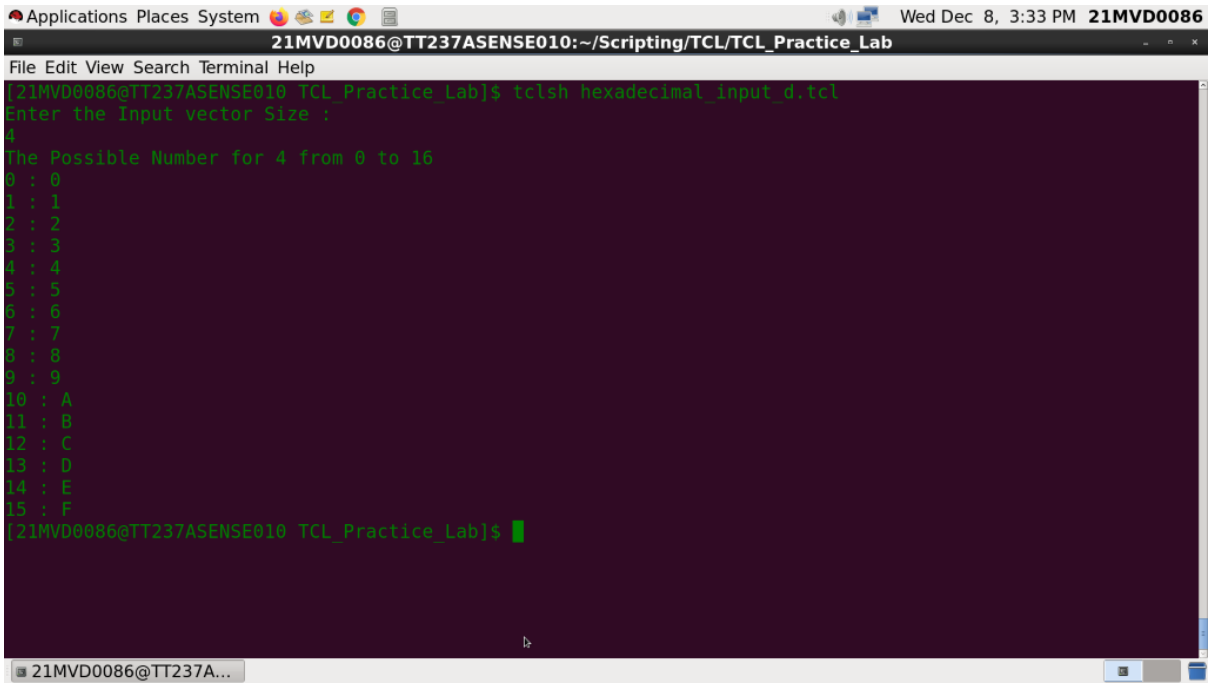


Figure 2.2 The Possible Number generation for 4-bit Number.

Date:20/12/2021

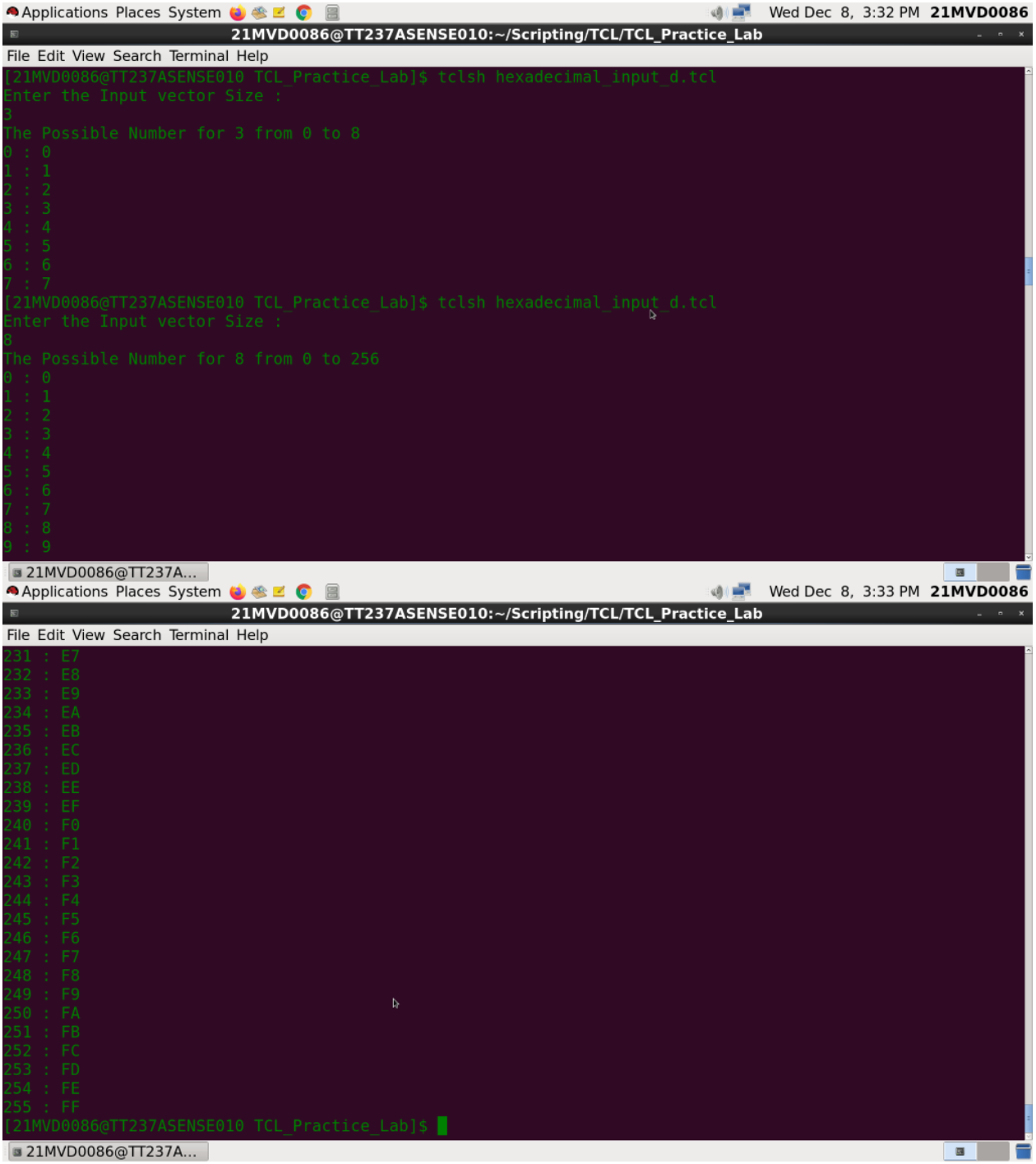


Figure 2.3 The Possible Number generation for 8-bit Number.

Date:20/12/2021

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.

Date:20/12/2021

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 :

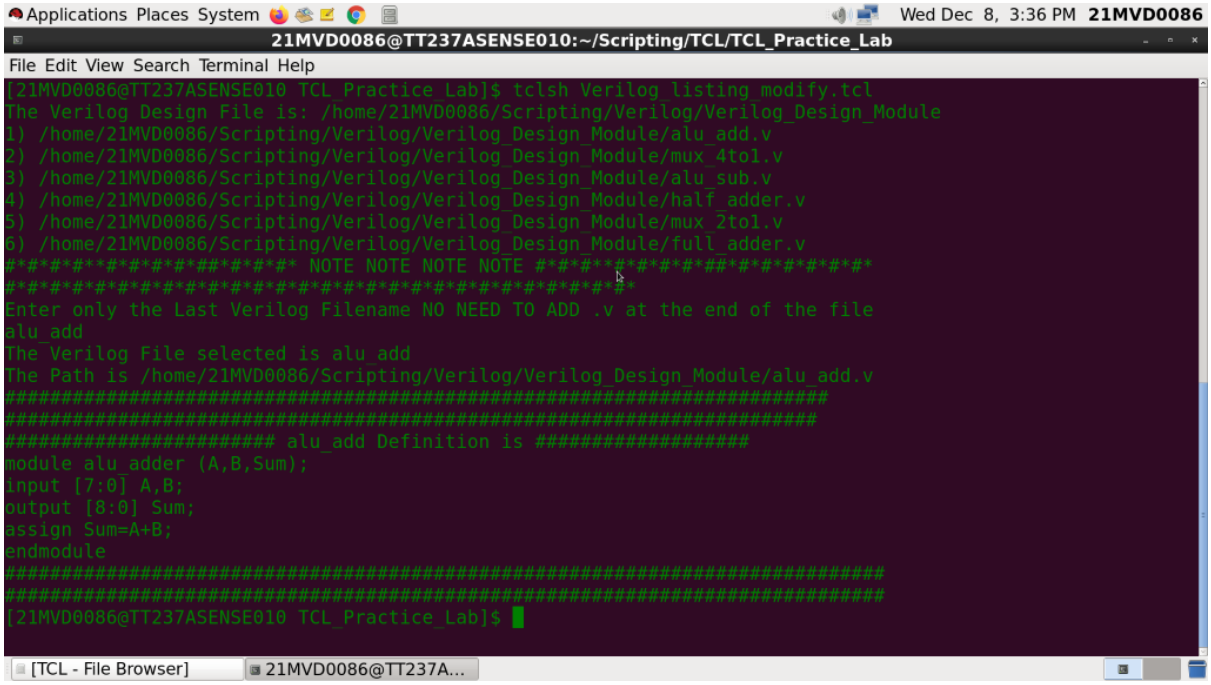


Figure 3.1 The Screenshot shows Verilog files present in the current directory and user selected Alu adder file.

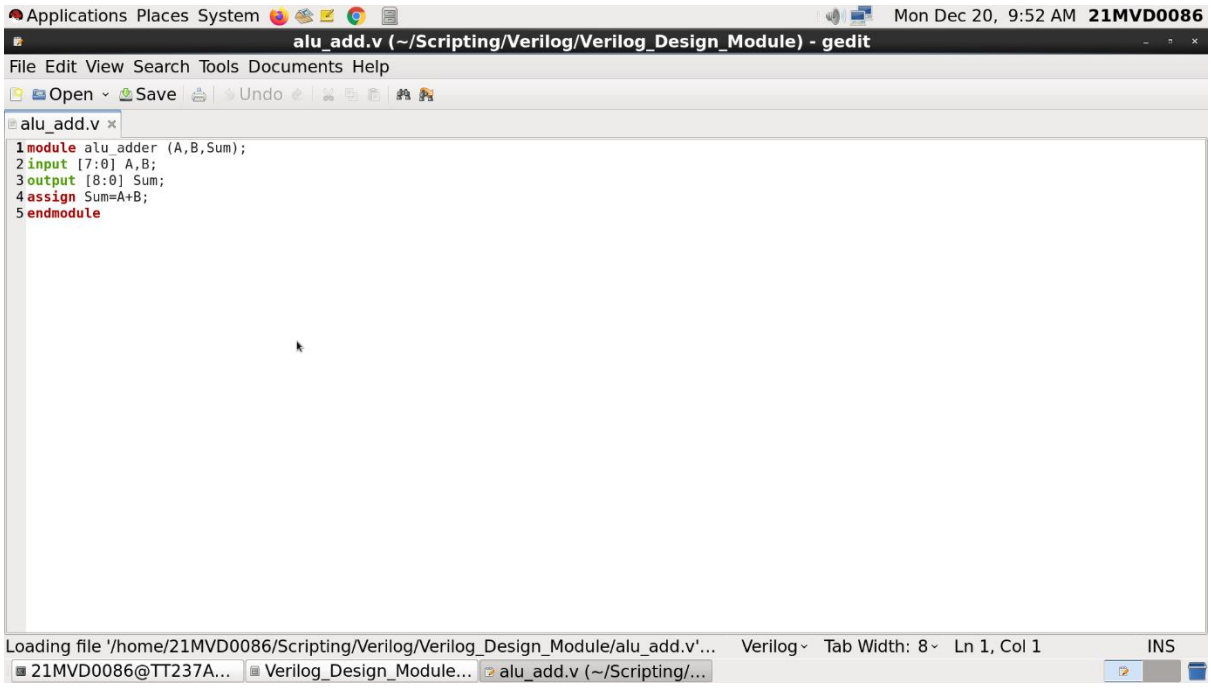


Figure 3.2 The Screenshot shows Alu adder Verilog file.

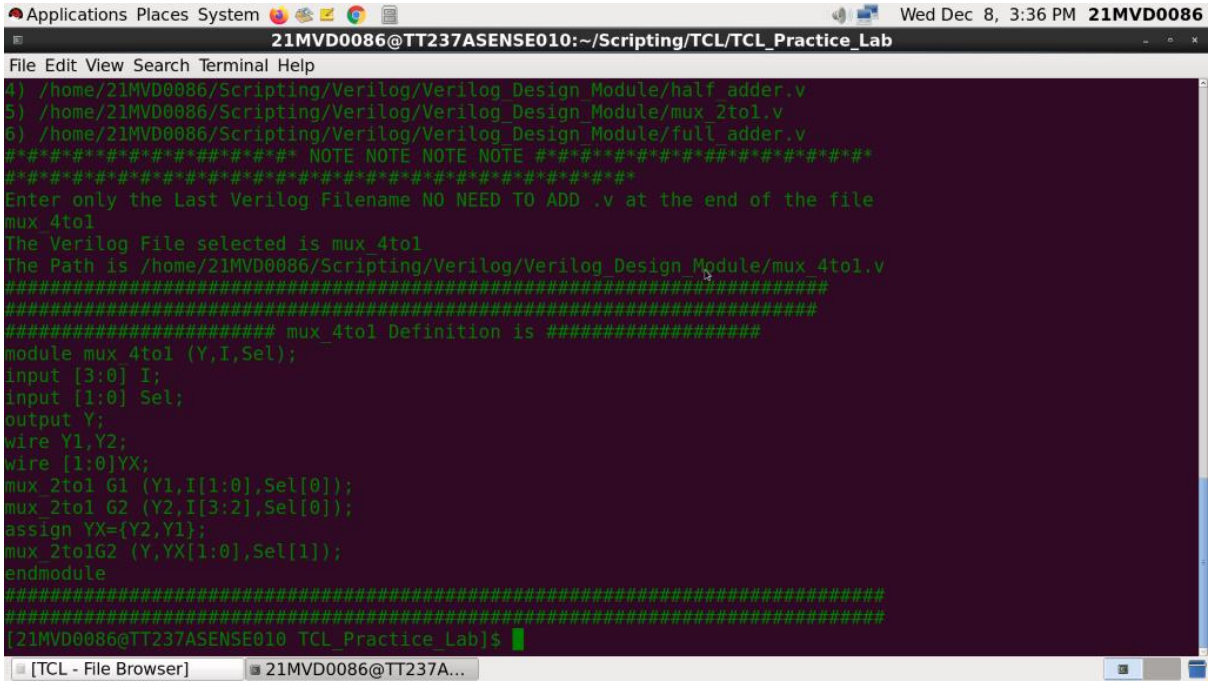


Figure 3.3 The Screenshot shows Verilog files present in the current directory and user selected Mux 4 to 1.

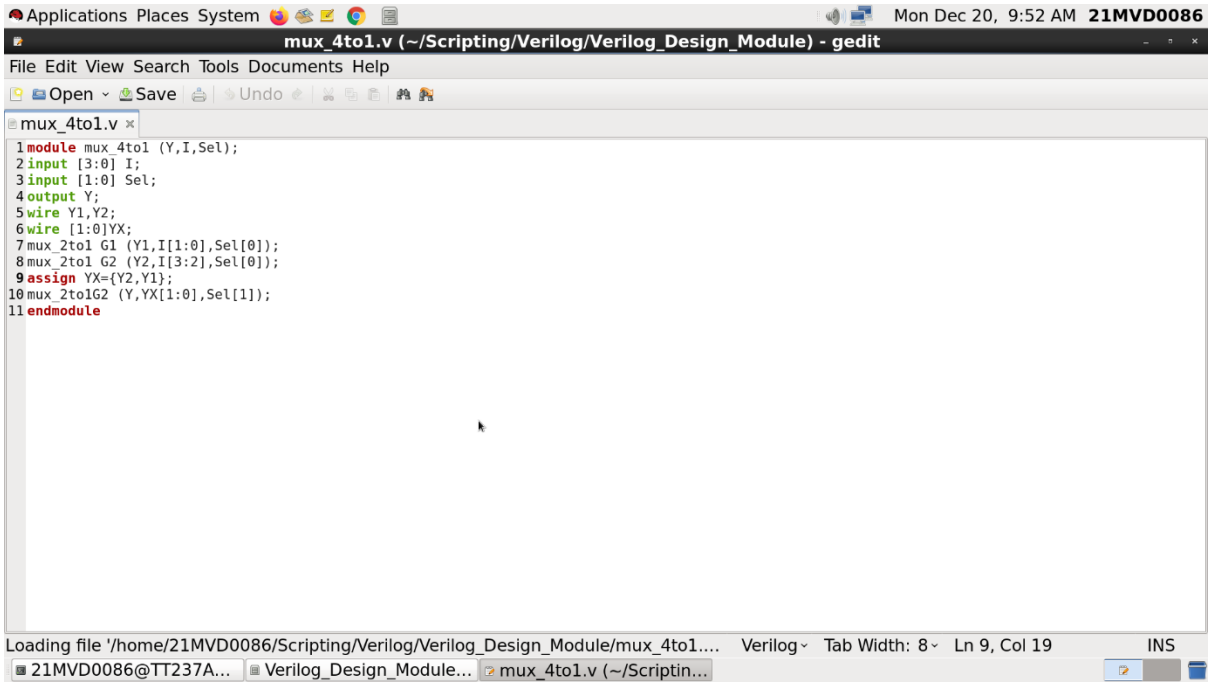


Figure 3.4 The Screenshot shows Mux 4 to 1 Verilog file.

Date:20/12/2021

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.

Date:20/12/2021

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.

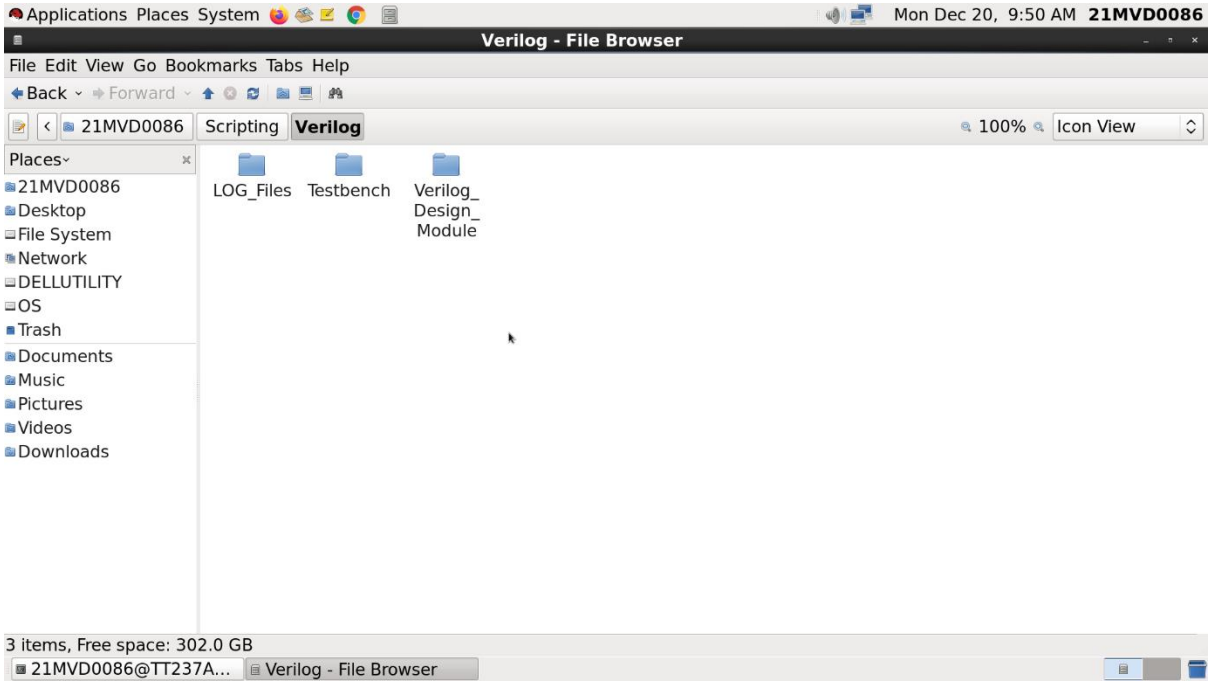


Figure 4.1 Verilog Directory having Testbench and Verilog_Design_Module.

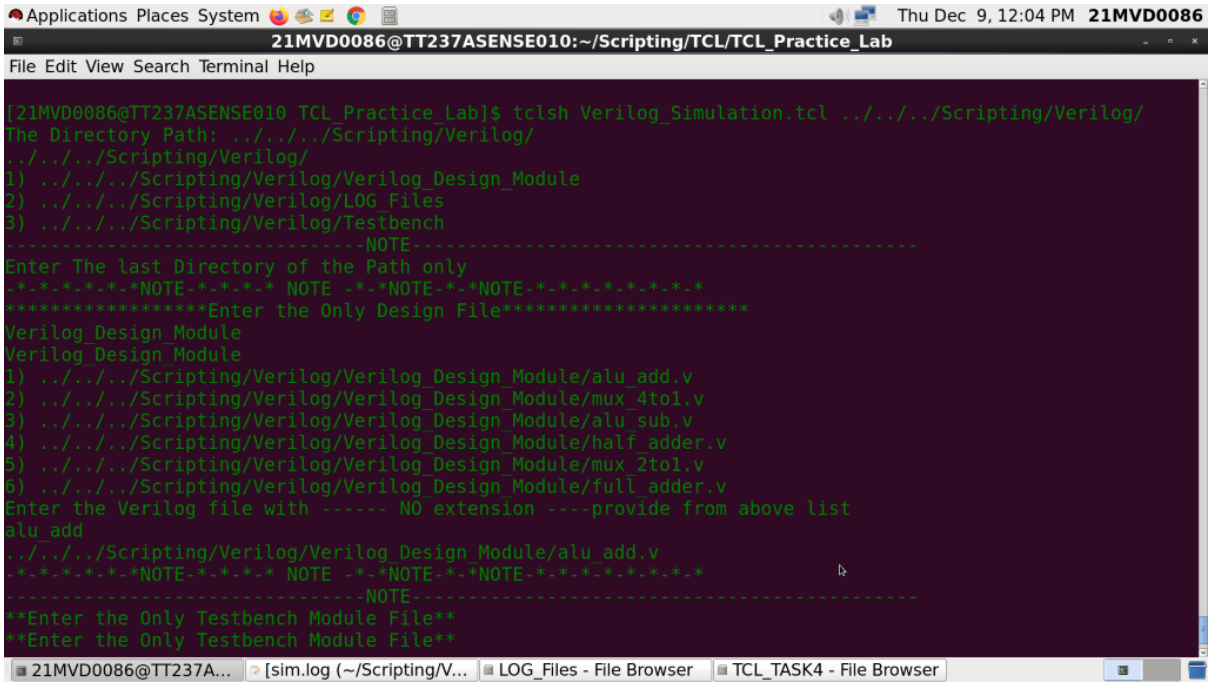


Figure 4.2 Prompting User to Verilog_Design_Module folder and listing the Verilog files present Verilog_Design_Module.

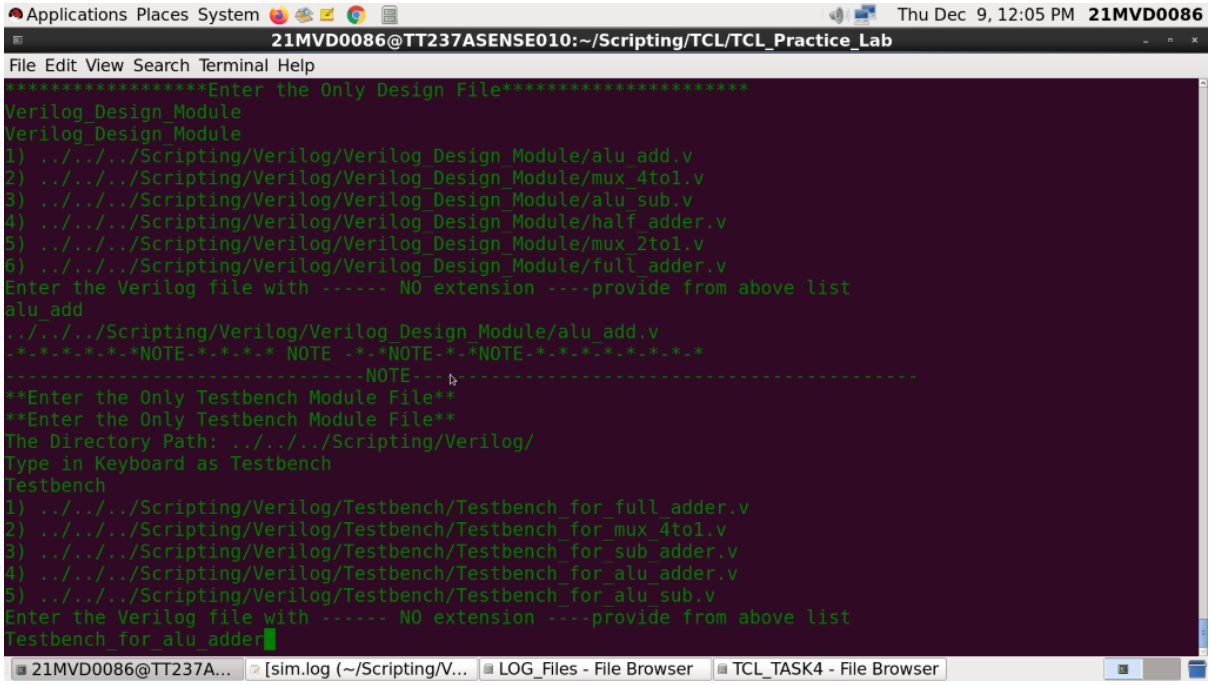


Figure 4.3 Prompting User to Testbench folder and listing the Verilog files present. Asking user to Enter proper testbench file corresponding to the design file.

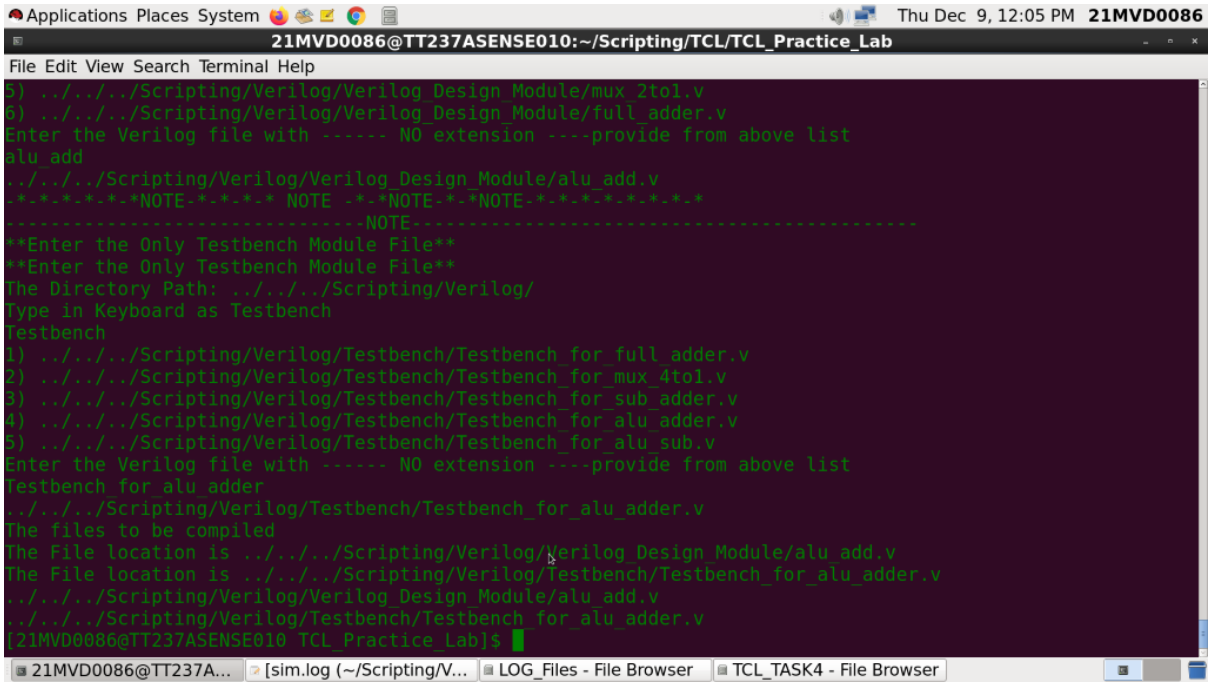


Figure 4.4 The screenshot shows the Verilog file complied in Modelsim and creates log files saved in LOG_Files folder.

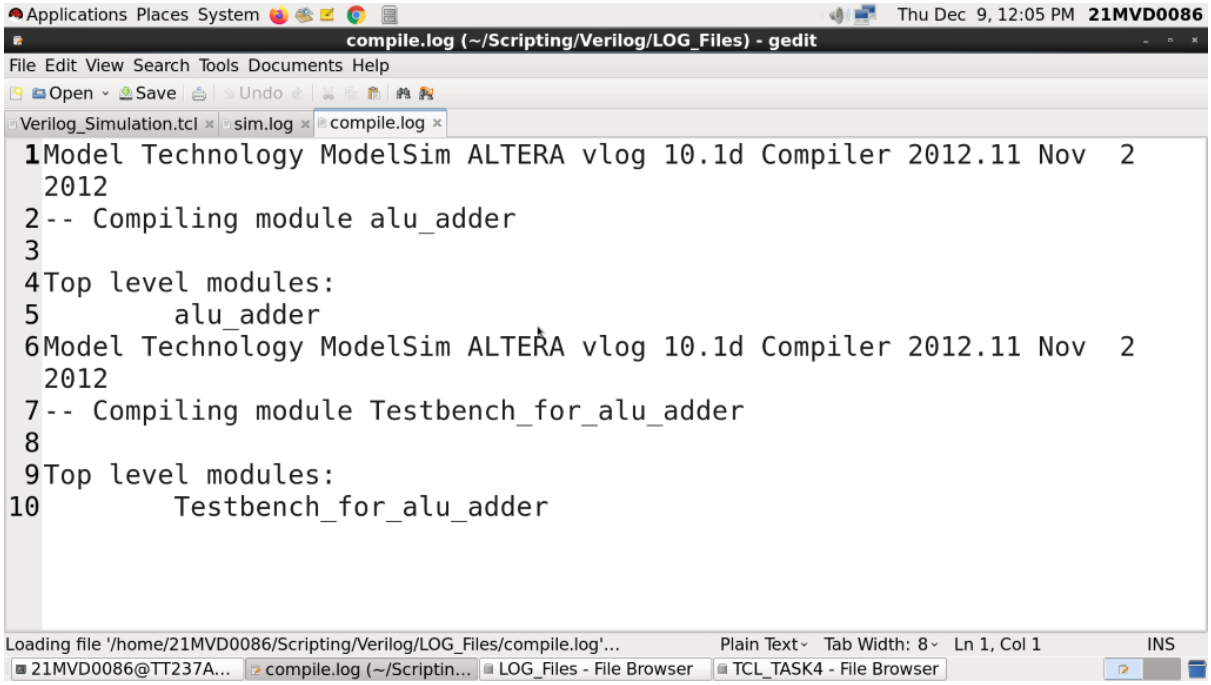


Figure 4.5 The compile log shows the design module name and testbench module name.

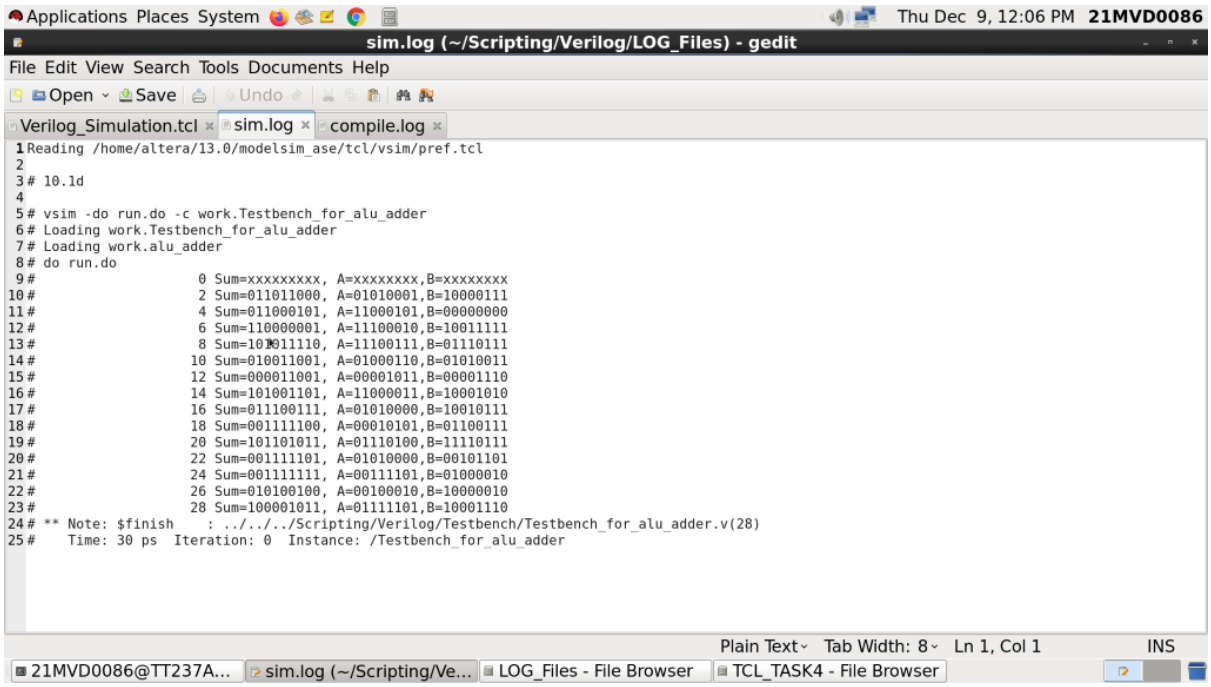


Figure 4.6 The Sim log shows the simulation output.

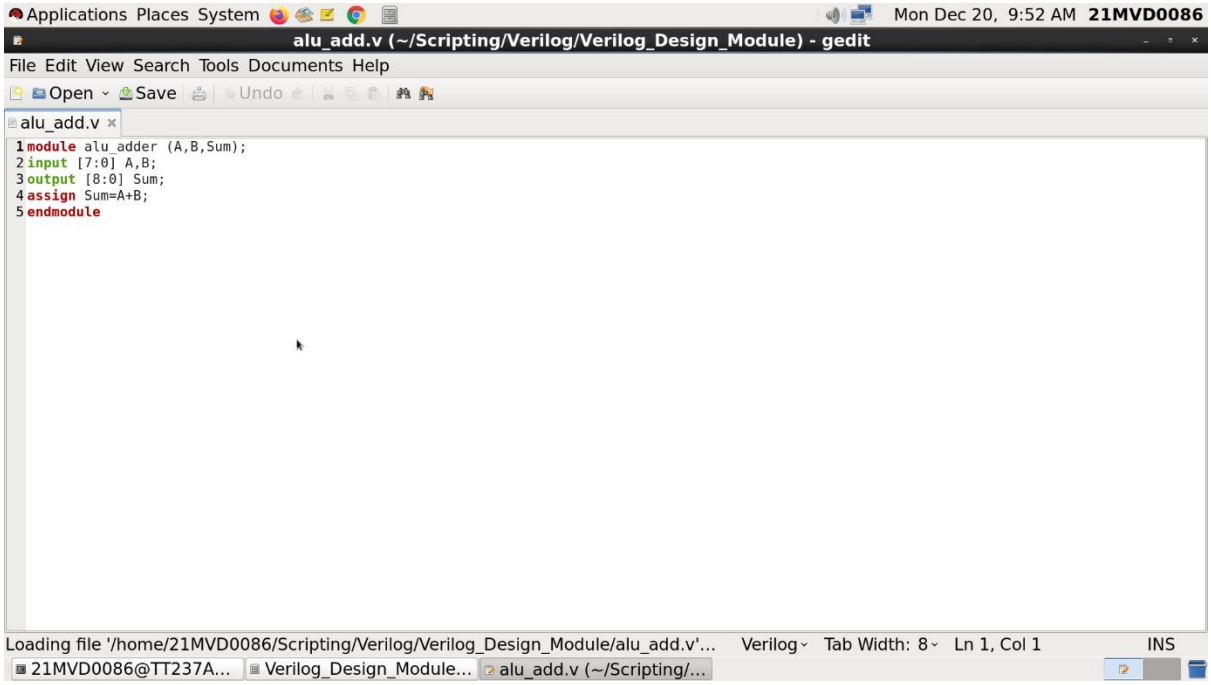


Figure 4.7 The Verilog Design Module File.

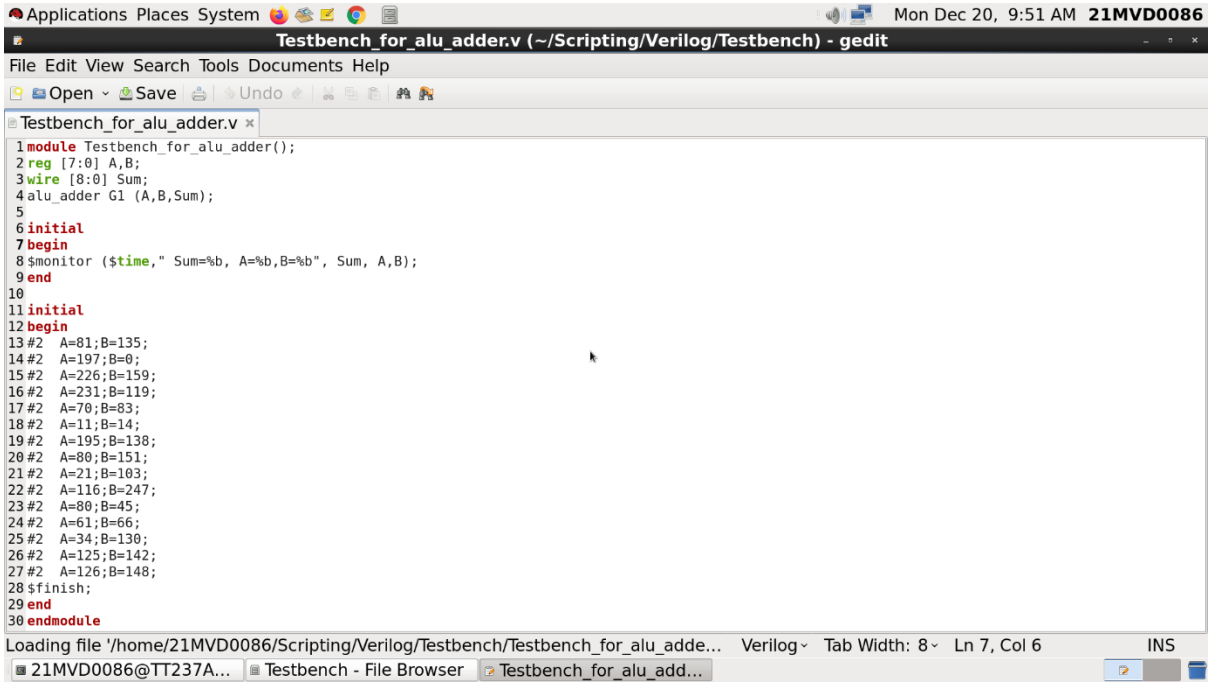


Figure 4.8 The Verilog Testbench Module File.

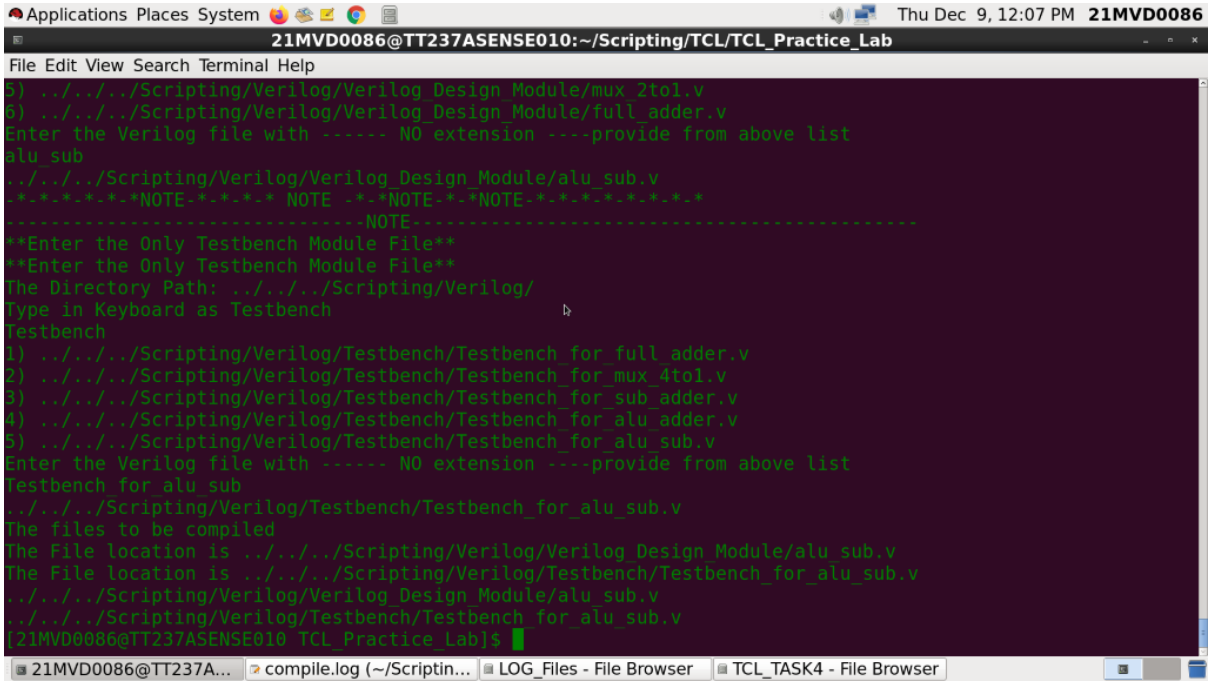


Figure 4.9 The User has selected Alu Subtractor as design module and testbench file of it.

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

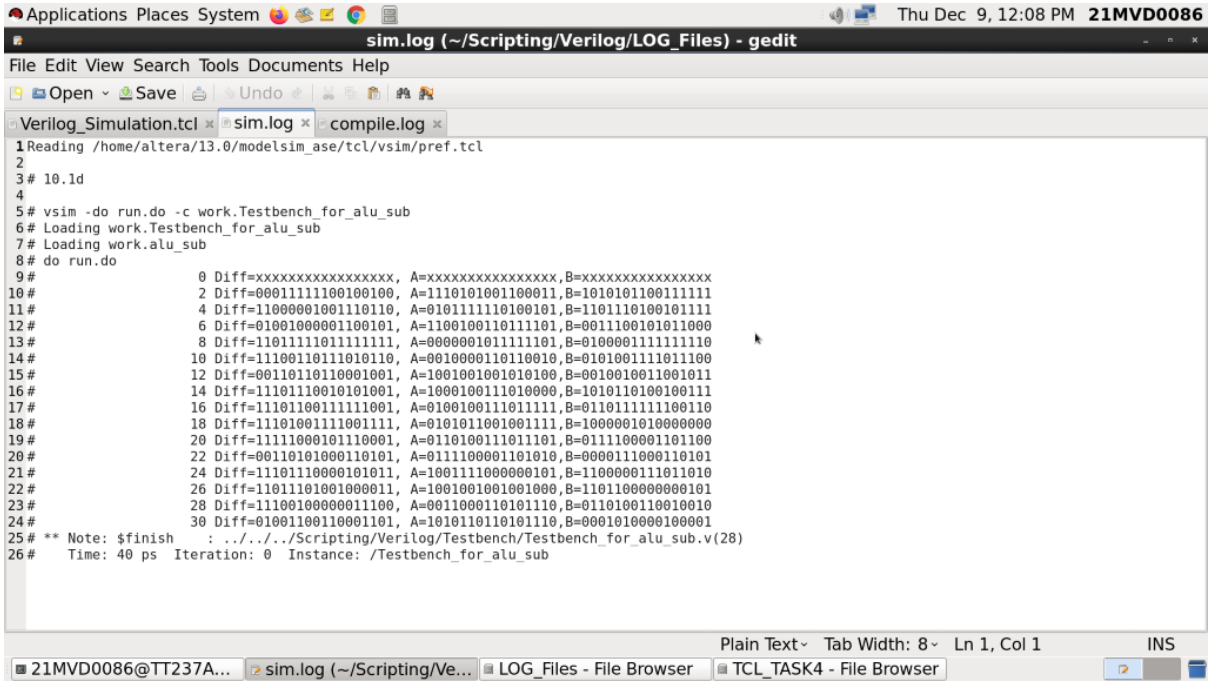


Figure 4.11 The Sim log shows the simulation output.

Figure 4.12 The Verilog Design Module File.

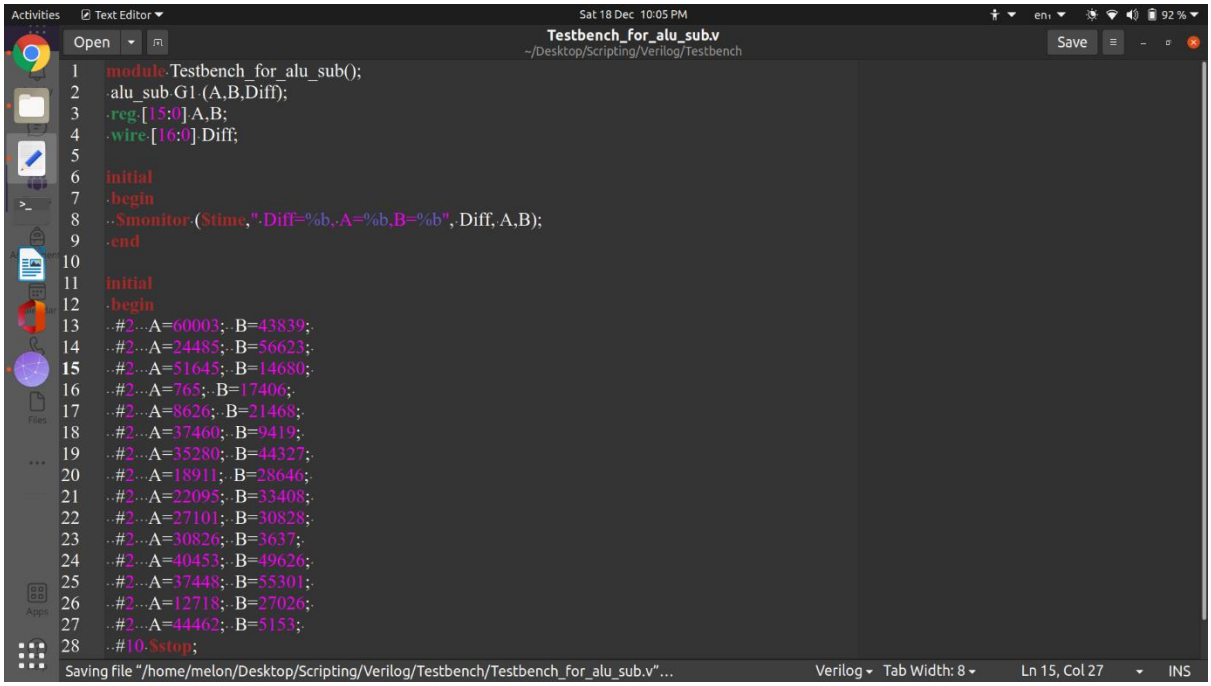


Figure 4.13 The Verilog Testbench Module File.

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.