Date:22/10/2021

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:22/10/2021

Lab Task 02

Random Number Generation and Testbench Automation

Section 1 Random Number Generation

Aim: To generate random test vectors for a given Verilog design.

Source Code or PERL Script code:

```
#! /usr/bin/perl -w
# PERL interpreter included and enabled for Warnings

print"How many Test Vector to be generated ? \n Please Enter Integer value: ";
$Test_Range=<STDIN>;
chomp($Test_Range);

@Arr=(A .. Z); # Variable Name array
print("Variable List :@Arr\n");
print"Specify the Number of variables used in Design:";
$Variable=<STDIN>;
chomp($Variable);

@Range_LL=(); # An empty array created to store Lower limit
@Range_UL=(); # An empty array created to store Lower limit

for($j=0;$j<$Variable;$j++)
{
print("Rand number for Array $Arr[$j]\n");
# Store the Upper limit and Lower limit value sepeartely in the array.
print"Specify the Lower limit for $Arr[$j] Variable:";
$Range_LL[$j]=<STDIN>;
chomp($Range_LL[$j]);

print"Specify the Upper limit for $Arr[$j] Variable:";
$Range_UL[$j]=<STDIN>;
chomp($Range_UL[$j]);
if($Range_LL[$j]>$Range_UL[$j])
{
print"Please enter Proper Limit range.";
print"\nLower limit must always be less than Upper limit\n";
print"------------------------------";
print"\nExecute once again with Proper Limit range\n";
print"------------------------------\n";
exit();
}
}
for($u=0;$u<$Variable;$u++)
```

```
{
print("Rand number for Array $Arr[$u]\n");
print"\nThe Lower limit for $Arr[$u] Variable:";
print"$Range_LL[$u]";
print"\nThe Upper limit for $Arr[$u] Variable:";
print"$Range_UL[$u]\n";
}
#print"\n@Range_LL";
#print"\n@Range_UL\n";
# Random Number Generation
$X=0;
for($k=0;$k<$Variable;$k++)
{
print("\nRand number for Array $Arr[$k]:\n");
for($l=0;$l<$Test_Range;$l++)
{
$Z=1;
while($Z)
{
#Storing random number
$X=int(rand$Range_UL[$k]);
if($X>$Range_LL[$k])
{
print"\t$X";
$Z=0;
}
}
}
print"\n Next loop\n";
}
```

Date:22/10/2021

Screenshots of Perl Script Written in Gedit window:

LAB_TASK_PERL.pl
~/Desktop/Scripting/PERL/PERL_Practice_Lab

```perl
#!/usr/bin/perl
#.PERL.Library.included

print"How many Test Vector to be generated ?.\n Please Enter Integer value:.";
$Test_Range=<STDIN>;
chomp($Test_Range);

@Arr=(A..Z); #.Variable.Name.array
print("Variable List :@Arr\n");
print"Specify the Number of variables used in Design:";
$Variable=<STDIN>;
chomp($Variable);

@Range_LL=();
@Range_UL=();

for($j=0;$j<$Variable;$j++)
{
print("Rand number for Array $Arr[$j]\n");
#.Store the Upper limit and Lower limit value sepeartely in the array.
print"Specify the Lower limit for $Arr[$j] Variable:";
$Range_LL[$j]=<STDIN>;
chomp($Range_LL[$j]);

print"Specify the Upper limit for $Arr[$j] Variable:";
$Range_UL[$j]=<STDIN>;
chomp($Range_UL[$j]);
if($Range_LL[$j]>$Range_UL[$j])
{
print"Please enter Proper Limit range.";
print"\nLower limit must always be less than Upper limit\n";
print"----------";
print"\nExecute once again with Proper Limit range\n";
print"----------\n";
exit();
}
}
for($u=0;$u<$Variable;$u++)
{
print("Rand number for Array $Arr[$u]\n");
print"\nThe Lower limit for $Arr[$u] Variable:";
print"$Range_LL[$u]";
print"\nThe Upper limit for $Arr[$u] Variable:";
print"$Range_UL[$u]\n";
}
#print"\n@Range_LL";
#print"\n@Range_UL\n";
#.Random Number Generation
```

LAB_TASK_PERL.pl
~/Desktop/Scripting/PERL/PERL_Practice_Lab

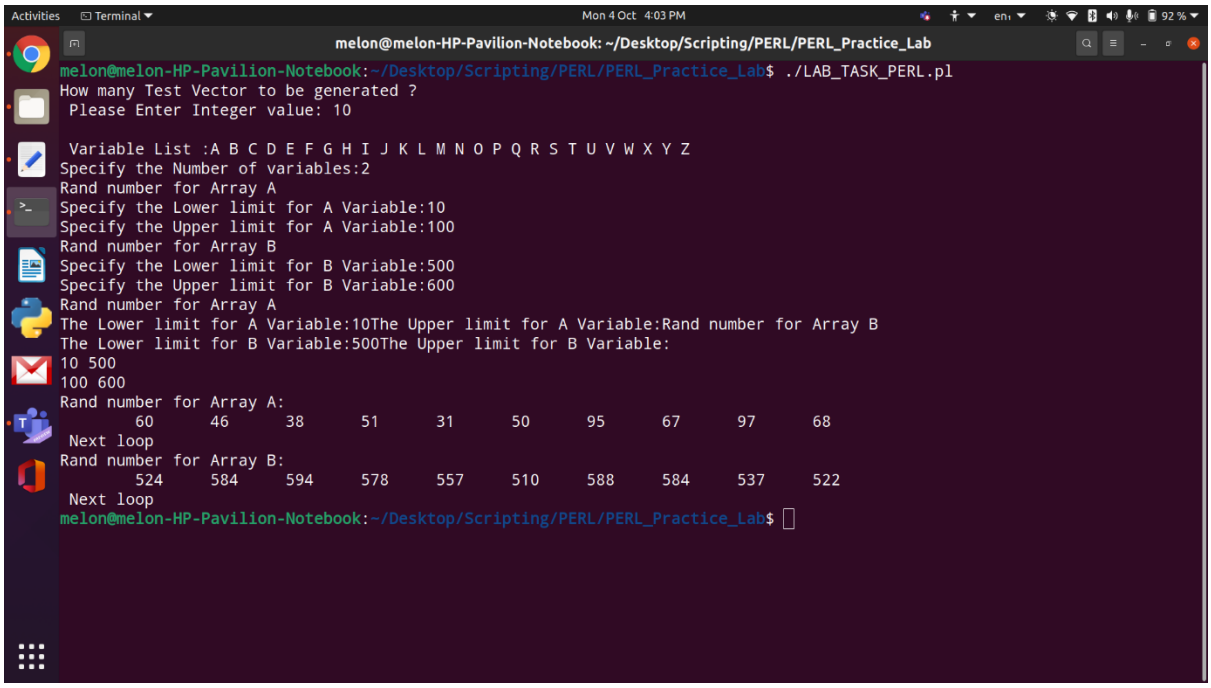
```perl
}
#print"\n@Range_LL";
#print"\n@Range_UL\n";
#.Random Number Generation
$X=0;
for($k=0;$k<$Variable;$k++)
{
print("\nRand number for Array $Arr[$k]:\n");
for($l=0;$l<$Test_Range;$l++)
{
$Z=1;
while($Z)
{
#Storing random number
$X=int(rand$Range_UL[$k]);
if($X>$Range_LL[$k])
{
print"\t$X";
$Z=0;
}
}
}
print"\n Next loop\n";
}
```

Perl Tab Width: 8 Ln 14, Col 14 INS

Date:22/10/2021

Output Screenshots

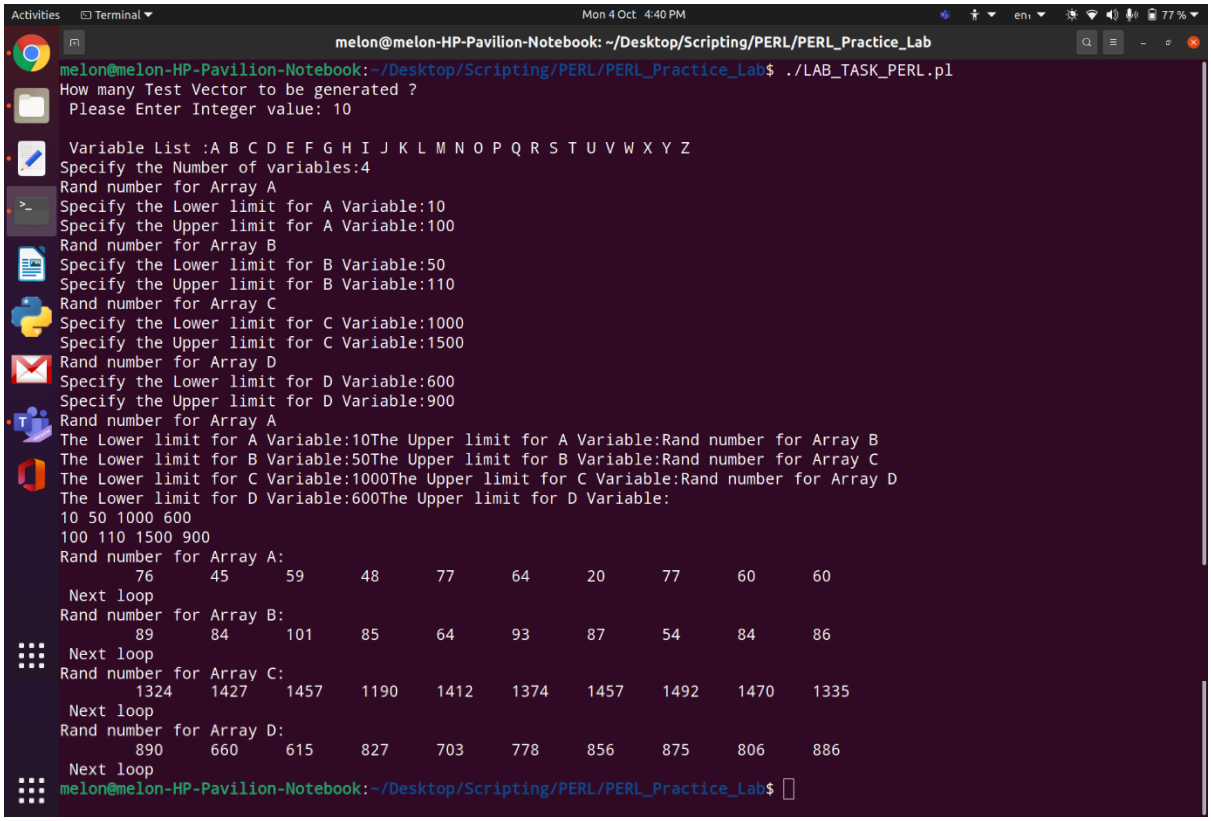
Example 1 : User has given input to generate 10 Test Vector for 2 Variable



```
melon@melon-HP-Pavilion-Notebook:~/Desktop/Scripting/PERL/PERL_Practice_Lab$ ./LAB_TASK_PERL.pl
How many Test Vector to be generated ?
 Please Enter Integer value: 10

 Variable List :A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
Specify the Number of variables:2
Rand number for Array A
Specify the Lower limit for A Variable:10
Specify the Upper limit for A Variable:100
Rand number for Array B
Specify the Lower limit for B Variable:500
Specify the Upper limit for B Variable:600
Rand number for Array A
The Lower limit for A Variable:10The Upper limit for A Variable:Rand number for Array B
The Lower limit for B Variable:500The Upper limit for B Variable:
10 500
100 600
Rand number for Array A:
        60      46      38      51      31      50      95      67      97      68
 Next loop
Rand number for Array B:
        524     584     594     578     557     510     588     584     537     522
 Next loop
melon@melon-HP-Pavilion-Notebook:~/Desktop/Scripting/PERL/PERL_Practice_Lab$
```

Figure 1.1 In this screenshot shows the random number for Variable A and Variable B. The user has given upper and lower limit for each variable separate.

Example 2 : User has given input to generate 10 Test Vector for 4 Variable



```
melon@melon-HP-Pavilion-Notebook:~/Desktop/Scripting/PERL/PERL_Practice_Lab$ ./LAB_TASK_PERL.pl
How many Test Vector to be generated ?
 Please Enter Integer value: 10

 Variable List :A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
Specify the Number of variables:4
Rand number for Array A
Specify the Lower limit for A Variable:10
Specify the Upper limit for A Variable:100
Rand number for Array B
Specify the Lower limit for B Variable:50
Specify the Upper limit for B Variable:110
Rand number for Array C
Specify the Lower limit for C Variable:1000
Specify the Upper limit for C Variable:1500
Rand number for Array D
Specify the Lower limit for D Variable:600
Specify the Upper limit for D Variable:900
Rand number for Array A
The Lower limit for A Variable:10The Upper limit for A Variable:Rand number for Array B
The Lower limit for B Variable:50The Upper limit for B Variable:Rand number for Array C
The Lower limit for C Variable:1000The Upper limit for C Variable:Rand number for Array D
The Lower limit for D Variable:600The Upper limit for D Variable:
10 50 1000 600
100 110 1500 900
Rand number for Array A:
        76      45      59      48      77      64      20      77      60      60
 Next loop
Rand number for Array B:
        89      84      101     85      64      93      87      54      84      86
 Next loop
Rand number for Array C:
        1324    1427    1457    1190    1412    1374    1457    1492    1470    1335
 Next loop
Rand number for Array D:
        890     660     615     827     703     778     856     875     806     886
 Next loop
melon@melon-HP-Pavilion-Notebook:~/Desktop/Scripting/PERL/PERL_Practice_Lab$
```

Figure 1.2 In this screenshot shows the random number for Variable A, B, C and D.

Date:22/10/2021

Example 3 If Lower limit is less than Upper limit prints with the message to user to give proper input.

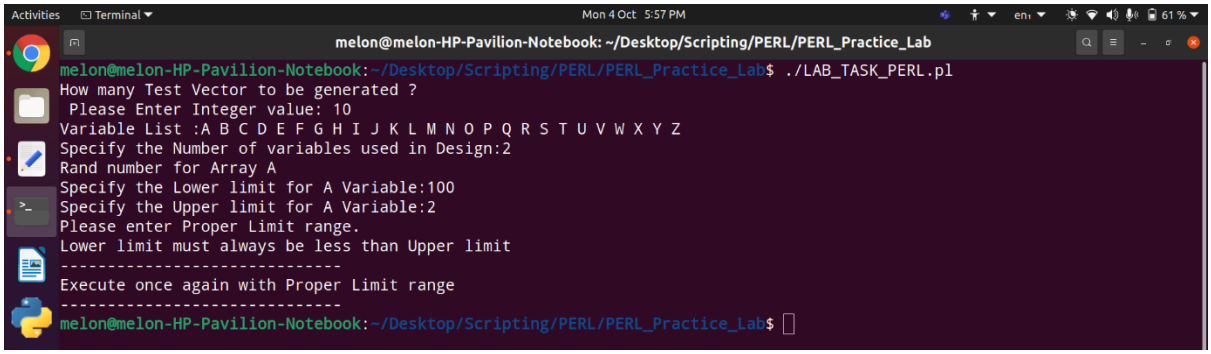


Figure 1.3 In this screenshot shows user to provide proper upper and lower limits.

Inference:

1. Writing PERL Script and how to execute it in Terminal window.
2. Get familiarised with Scalar Data, Arrays and List Data, Control Structure, Hashes syntax and using in the script.
3. The Random testvector generation helps to generate testvector for 'N' number input by specifying the limits to the variable.
4. Automation of VLSI Verification. This Random Generation module is used in next section how it is achieved.
5. Learning inbuilt functions like chomp(), chop(), rand(), print(), shift(), unshift(), pop(), push(), sort(), reverse() and many more.

Date:22/10/2021

Section 2 : Testbench Automation

Aim: Read Verilog file and generate Testbench file in Verilog.

Source Code or PERL Script code:

```
#! /usr/bin/perl -w

$Dirname=$ARGV[0];
print"#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*";
print"#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*\n";
print"\n\nThe Directory path is $Dirname\n";
opendir (DIR1,$Dirname) or die "Error in opening: $!";
print"\nThe Verilog Files present in $Dirname Directory are:\n";
# Filtering the Verilog files
my $cnt=1;
foreach(sort grep(/^.*\.v$/,readdir(DIR1)))
{
print"$cnt. $_\n";
$cnt++;
}
print"\n";
print"#*#*#*#*#*#*#*#*#*#*#*#*#*#*##*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*
#*#*#*#*#*#*#*#*#*#*";
print"\n\n-*-*--*--*--*-NOTE-*-NOTE-*-*--*--*--*-NOTE-*-NOTE-*-*--*--*--*-NOTE-*-
NOTE---*--*--*--*--*-\n";
print"\n-----------CAREFULL ENTRY-----------CAREFULL ENTRY---------CAREFULL
ENTRY----------CAREFULL ENTRY-------------\n\n";
print"For example the verilog file name is ABC.v then type ABC.v as itself\n";
print"\n\nSelect filename with extension of Verilog file:";
$Veri_File=<STDIN>;
chomp($Veri_File);
print"\n";
open (FH1,"$Dirname/$Veri_File") or die "Error in Opening Verilog File:$!";
@Veri_Mod=<FH1>;
close (FH1);
#The below command gives number of lines present in Verilog Module
$Line_Count=@Veri_Mod;
print"The Total Number of Lines in Design Module is :$Line_Count\n";
print"The Design Module Contents\n";
print"*****************************************************************\n";
print"*********** Module Definition *****************\n";
print"@Veri_Mod\n";
print"*****************************************************************\n";
print"Verilog Module Selected\n";
print"Generation of Testbench and Testvectors using PERL Script\n";
#Now Searching for Module name, Input Port, Output Port
open (FH1,"<$Dirname/$Veri_File") or die "Error in Opening Verilog File:$!";
open FH2,">$Dirname/Testbench.v" or die "Error in Writing to file $!";
```

```
@Input_List =();
@Output_List=();
@Range_LL_Input=();
@Range_UL_Input=();
$NI=0;  #This denotes total Number of Input bits.
$NO=0; #This denotes total Number of Output bits.
$len_updated_input_count=0;
##################################################################
############### Main Code Starting here #########################
while(<FH1>)
{
chomp;
######### Getting Module Name and Port list #######################
if($_=~ m/^m.*e\s/)
{
my $module_nameport=$';
&get_module_name_portlist($module_nameport);
}
################### To get Input Ports ############################
elsif ($_=~ m/^input\s/)
{
my $Module_Input_Port=$';
my @Input_List_Check=();
@Input_List_Check= &get_input_ports($Module_Input_Port);
my $len_inputlist_update=@Input_List;
if($len_inputlist_update == -1)
{
@Input_List=@Input_List_Check;
}
else
{
push (@Input_List, @Input_List_Check);
print "\nUpdate in Input List because of mixture of Multibit and Singlebit Input\n";
# The values are stored in Input_List;
}
print "The Input List: @Input_List\n";
$len_updated_input_count=@Input_List;
}
#################To get Output Ports#############################
elsif ($_=~ m/^output\s/)
{
my $Module_Output_Port=$';
my @Output_List_Check=();
@Output_List_Check= &get_output_ports($Module_Output_Port);
my $len_outputlist_update=@Output_List;
if($len_outputlist_update == -1)
{
@Output_List=@Output_List_Check;
}
else
```

```
{
push (@Output_List, @Output_List_Check);
# The values are stored in Output_List;
print "Update in Output List because of mixture of Multibit and Singlebit Output\n";
}
print "The Output List: @Output_List\n";
}
elsif($Line_Count==1)
{
my @Monitor_List_print=();
push(@Monitor_List_print,@Output_List);
push(@Monitor_List_print,@Input_List);
&monitor_print(@Monitor_List_print);
&rand_test_input(); # To Generate Random Test Vector for Input Line
}
$Line_Count=$Line_Count-1;
####### End of Module Reading and While Loop ####################
}
############# Closing The File of Main Module and Testbench File#####
close (FH1);
close(FH2);
#Renaming the file from testbench to with testbench_modulename addition
$New_name="Testbench_for_".$Module_Name;
$Testbench_loc="/home/melon/Desktop/Scripting/Verilog/Testbench";
rename ("$Dirname/Testbench.v", "$Testbench_loc/$New_name.v") or die "Error in
renaming : $!";
##################### Subroutine Declaration #####################
####Step 1. Getting Module Name and Port List Declaration ############
sub get_module_name_portlist
{
print"The Module Name and port list:\t\t", $_[0],"\n";
my $module_name_port = $_[0];
@mod_nam_port=split /\s+/,$module_name_port;
$Module_Name=$mod_nam_port[0];
my $Module_Portlist=$mod_nam_port[1];
print"The Module Name is:\t\t$Module_Name\n";
print"The Module Portlist is:\t\t $Module_Portlist\n";
# Defining Module for Testbench
print FH2"module test_$Module_Name();\n";
# Instantiation of Design Module in Testbench with Instatation given as G1
print FH2" $Module_Name G1 $Module_Portlist\n";
}
###########Step 2. Getting Input Portlist #######################
sub get_input_ports
{
my $Module_Inputs=$_[0];
print"The Input ports are:\t\t";
print"$Module_Inputs";
print"\n";
print FH2 " reg $Module_Inputs\n";
```

```
# Check for Multi-bit Input used \d+
if($Module_Inputs =~ m/^\[(\d+):(\d+)\]/)
{
my @Input_List_Multi=();
print"*************************************************";
print"\nMultibit Input\n";
print"MSB Size of Multibit Input:\t\t$1";
print"\n";
my $Input_Multi_MSB=$1;
$NI=$Input_Multi_MSB+1;
$Input_Multi_MSB=$1-$2+1;
print"Input Field Bit Length:\t\t$NI\n";
print"LSB Size of Multibit Input:\t\t$2 \n";
$Input_Multi_LSB=$2;
# To remove [MSB:LSB] format for futher processing;
$Module_Inputs =~ s/^\[(\d+):(\d+)\]//;
@Input_List_Multi = &input_port_list_sepearting($Module_Inputs);
$Variable_multi_count = @Input_List_Multi;
foreach $Var(@Input_List_Multi)
{
push (@Range_UL_Input,((2**$Input_Multi_MSB)-1));
push(@Range_LL_Input,0);
}
print"*************************************************";
return @Input_List_Multi;
}
# Check for Single-bit Input
else
{
my @Input_List_Single=();
my $Input_Single_MSB=0;
my $Input_Single_LSB=0;
$NI=$Input_Single_MSB;
print"*******************************";
print "\nInput is Single bit\n";
@Input_List_Single = &input_port_list_sepearting($Module_Inputs);
$Variable_single_count = @Input_List_Single;
foreach $Var1(@Input_List_Single)
{
push (@Range_UL_Input,2**$Input_Single_MSB);
push(@Range_LL_Input,2**$Input_Single_LSB-1);
}
print"*******************************";
return @Input_List_Single;
}
}
sub input_port_list_sepearting
{
# To Remove ";" at end of the input port list
my $Module_Inputs_Separate=$_[0];
```

```
my @Input_List_Separate=();
$Module_Inputs_Separate =~ s/;//;
print"Module Input List is:\t\t$Module_Inputs_Separate\n";
#split(', ', $str);
@Input_List_Separate= split (',',$Module_Inputs_Separate);
$Input_Port_Count = @Input_List_Separate;
print"Input port name is:\t\t@Input_List_Separate\n";
return @Input_List_Separate; # Return Input_List_Separated
}
##################################################################
######Step 3. Getting Output Portlist ###########################
sub get_output_ports
{
my $Module_Outputs=$_[0];
print"The Output ports are:\t\t";
print"$Module_Outputs";
print"\n";
print FH2 " wire $Module_Outputs\n";
# Check for Multi-bit Output
if($Module_Outputs =~ m/^\[(\d+):(\d+)\]/)
{
my @Output_List_Multi=();
print"&*&*&*&*&*&*&*&*&*&*&*&*&*&*&*&*&*&*";
print"\nMultibit Output \n";
print"MSB Size : $1\n";
my $Output_Multi_MSB=$1;
$NI=$Output_Multi_MSB+1;
print"Output Field Bit Length: $NI\n";
print"LSB Size : $2 \n";
$Output_Multi_LSB=$2;
# To remove [MSB:LSB] format for futher processing;
$Module_Outputs =~ s/^\[(\d+):(\d+)\]//;
@Output_List_Multi = &output_port_list_sepearting($Module_Outputs);
print"&*&*&*&*&*&*&*&*&*&*&*&*&*&*&*&*&*&*";
return @Output_List_Multi;
}

# Check for Single-bit Input
else
{
my @Output_List_Single=();
my $Output_Multi_MSB=0;
my $Output_Multi_LSB=0;
$NO=$Output_Multi_MSB;
print"&*&*&*&*&*&*&&*&*&*&*&*&*&*&*&*&*&*&*&*&*&*&*&*";
print "\nOutput is Single bit\n";
@Output_List_Single = &output_port_list_sepearting($Module_Outputs);
return @Output_List_Single;
print"&*&*&*&*&*&*&*&&*&*&*&*&*&*&*&*&*&*&*&*&*&*&*&*";
}
```

```
}
sub output_port_list_sepearting
{
# To Remove ";" at end of the input port list
my $Module_Outputs_Separate=$_[0];
my @Output_List_Separate=();
$Module_Outputs_Separate =~ s/;//;
print"Module Input List is $Module_Outputs_Separate\n";
#split(', ', $str); Syntax
@Output_List_Separate= split (',',$Module_Outputs_Separate);
$Output_Port_Count = @Output_List_Separate;
print"Output port name is @Output_List_Separate\n";
return @Output_List_Separate; # Return Output_List_Separated
}
##############################################################
##############################################################
####Step 4. Writing Monitor block and test input block #################
sub monitor_print
{
###############Initial Block Statement Declaration #################
print FH2 "\ninitial\n";
print FH2 " begin\n";
### Input Monitor Declaration ###
my @Monitor_List = @_;
$MONITOR=0;
foreach $MONITOR(@Monitor_List)
{
print"\n Input Monitor List :$MONITOR\n";
}
$Total_Port_Count = @Monitor_List;
print"\n The Monitor List is updated with Input port and Output port\n";
print"\n\nMonitor_List =@Monitor_List\n";
print"\nTotal Ports present in Design Module = \t\t$Total_Port_Count\n";

print"*#@#\$\**#@#\$\**#@#\$\**#@#\$\**#@#\$\**#@#\$\**#@#\$\**#@#\$\**#@#\$\
**#@#\$\**#@#\$\**#@#\$\**#@#\$\**#@#\$\*";
print FH2 "  \$monitor (\$time,\"";
print "\n\n\n\$monitor (\$time,\"";
$CNT_PORT=0;
#$Total_Port_Count
foreach $name(@Monitor_List)
{
$CNT_PORT+=1;
if($CNT_PORT == $Total_Port_Count)
{
print FH2 "$name=\%b";
}
else
{
print FH2 "$name=\%b,";
```

```
print "$name=\%b,";
}
}
print FH2 "\"\,";
print "\b\"\,";

$CNT_PORT=0;
foreach $name(@Monitor_List)
{
$CNT_PORT+=1;
if($CNT_PORT == $Total_Port_Count)
{
print FH2 "$name";
}
else
{
print FH2 "$name,";
print " $name,";
}
}
print FH2 "\);\n";
print "\b);\n\n\n";
print"*#@#\$\**#@#\$\**#@#\$\**#@#**#@#\$\**#@#\$\**#@#\$\**#@#\$\**#@#\$\**
#@#\$\*\n";
######## Inital Block end Statement #######################
print FH2 " end\n";
return 0;
&rand_test_input ($len_updated_input_count);
}

#################################################################

#################################################################
########## Step 5 Random Generation of Input #####################
############Random Number Generation code ####################
#########This block used from Last week Assignment ###############
sub rand_test_input()
{
print "\n*******************#####Random input Generation
#####************************\n";
print
">>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>><<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
<<<<\n";
print"\n\n-----------CAREFULL ENTRY---------CAREFULL ENTRY--------------------\n\n";
print "\n******************** Please Enter Integer value
****************************\n";
print"How many Test Vector to be generated ?";
my $Test_Range=<STDIN>;
chomp($Test_Range);
# Input Port count. So generating those many numbers
```

```
$Variable_count = @Input_List;
print "The Input Variable count $Variable_count\n";
push(@Range_LL,@Range_LL_Input);
push(@Range_UL,@Range_UL_Input);
for($u=0;$u<$Variable_count;$u++)
{
print
"<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<>>>>>>>>>>>>>>>>>>>>>>>>>>>\n";
print("Rand number for Array $Input_List[$u]\n");
print"\nThe Lower limit for $Input_List[$u] Variable:";
print"$Range_LL[$u]";
print"\nThe Upper limit for $Input_List[$u] Variable:";
print"$Range_UL[$u]\n";
print
"<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<>>>>>>>>>>>>>>>>>>>>>>>>>>>\n";
}
my $X=0;
my @Input_List_Var=[];
for($k=0;$k<$Variable_count;$k++)
{
print "\n##<>##---##<>##---##<>##---##<>##---##<>##---##<>##---##<>##---##<>##---
##<>##---##<>##---\n";
print("\nRand number for Array $Input_List[$k]:\n");
for($l=0;$l<$Test_Range;$l++)
{
my $Z=1;
while($Z)
{
#Storing random number
$X=int(rand($Range_UL[$k]-$Range_LL[$k]+1))+$Range_LL[$k];
if($X>=$Range_LL[$k])
{
$Input_List_Var[$k][$l]=$X;
$Z=0;
}
}
print"$Input_List_Var[$k][$l]\t";
}
print "\n##<>##---##<>##---##<>##---##<>##---##<>##---##<>##---##<>##---##<>##---
##<>##---##<>##---\n";
}

########## Storing the Random number generated for Input Variable ####
#######Initial Block Statement for Input Declaration ##################
print FH2 "\ninitial\n";
print FH2 " begin\n";
for($l=0;$l<$Test_Range;$l++)
{
print FH2 "  #2 ";
for($k=0;$k<$Variable_count;$k++)
```

```
{
#print"$Input_List_Var[$k][$l]\t";
print FH2 " $Input_List[$k]=$Input_List_Var[$k][$l]; ";
}
print FH2 "\n";
}
######## Input Inital Block end Statement #######################
print FH2 "  #10 \$stop;\n";
print FH2 " end\n";
print FH2 "endmodule\n";
}
```

Important Points

1. In Verilog Folder there are two Directories **Testbench** and **Verilog_Design_Module**.
   Note:
   There should a space between module modulename (port list);
2. The **Verilog_Testbench_Gen.pl** source file present in PERL Directory.
3. Execute the file and provide Verilog module file location using command arguments.
4. Provide Integer value for Test input Generation. The code is implemented using Random Number Generation.
5. After Generation of testbench the file is stored in the Testbench folder.

Output Screenshots

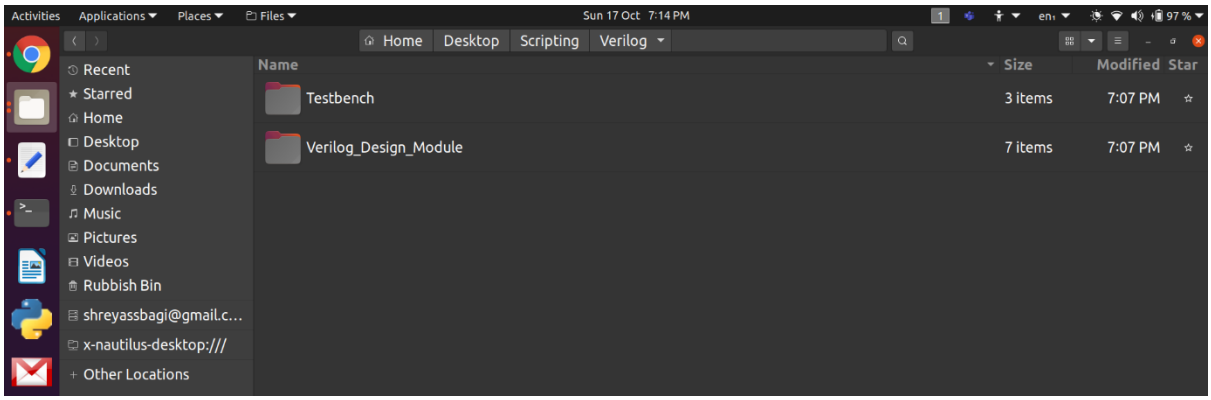


Figure 2.1 The source code is present in Verilog_Design_Module and testbench generated is stored in Testbench folder.

Example 1: 1 bit Full adder Design

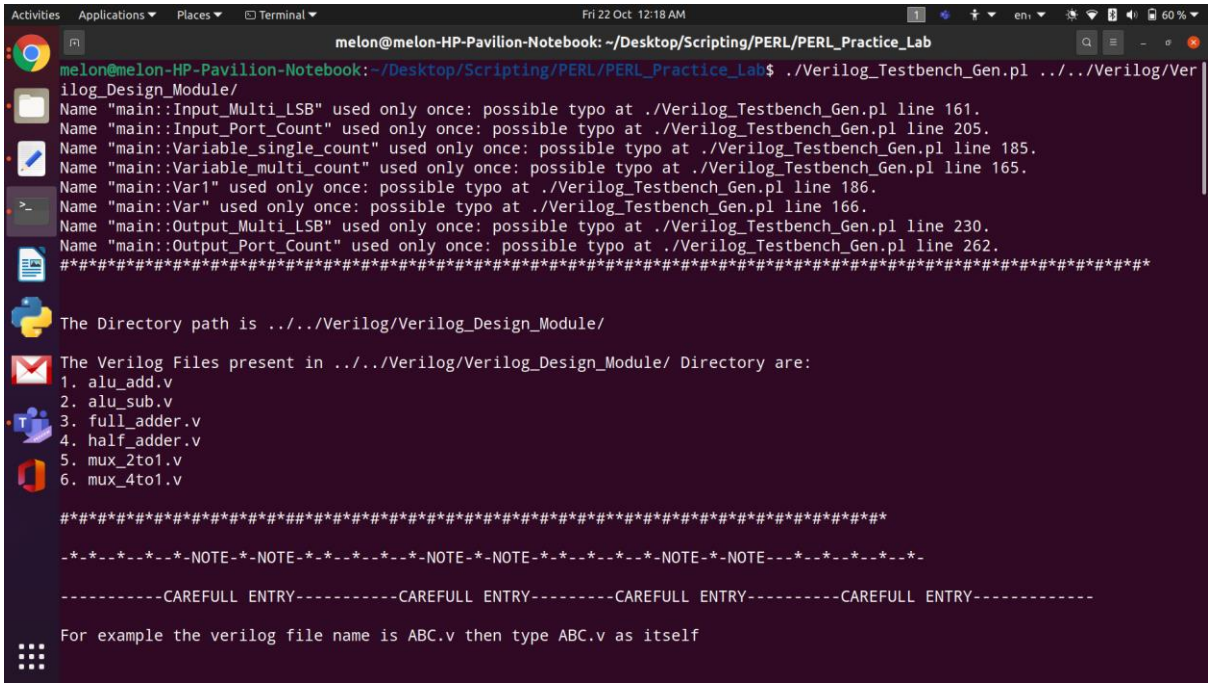


Figure 2.2 The user needs to give Verilog file as input to the terminal. The terminal is displaying the list of available Verilog files under Design Directory.

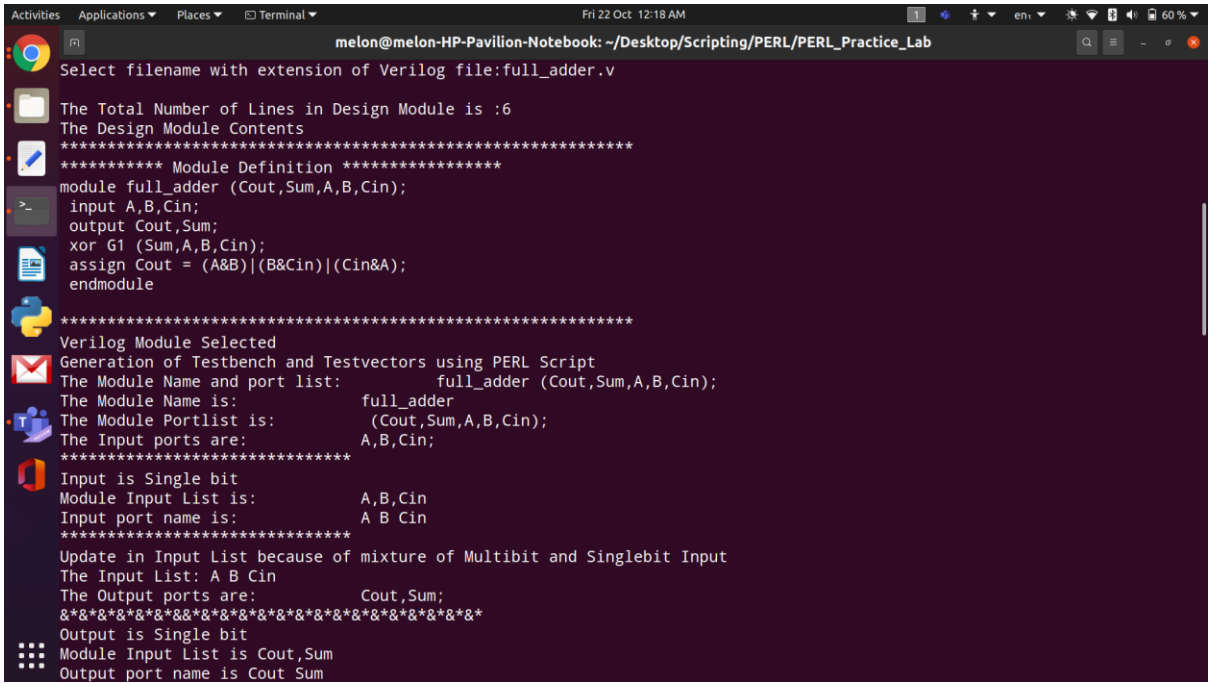
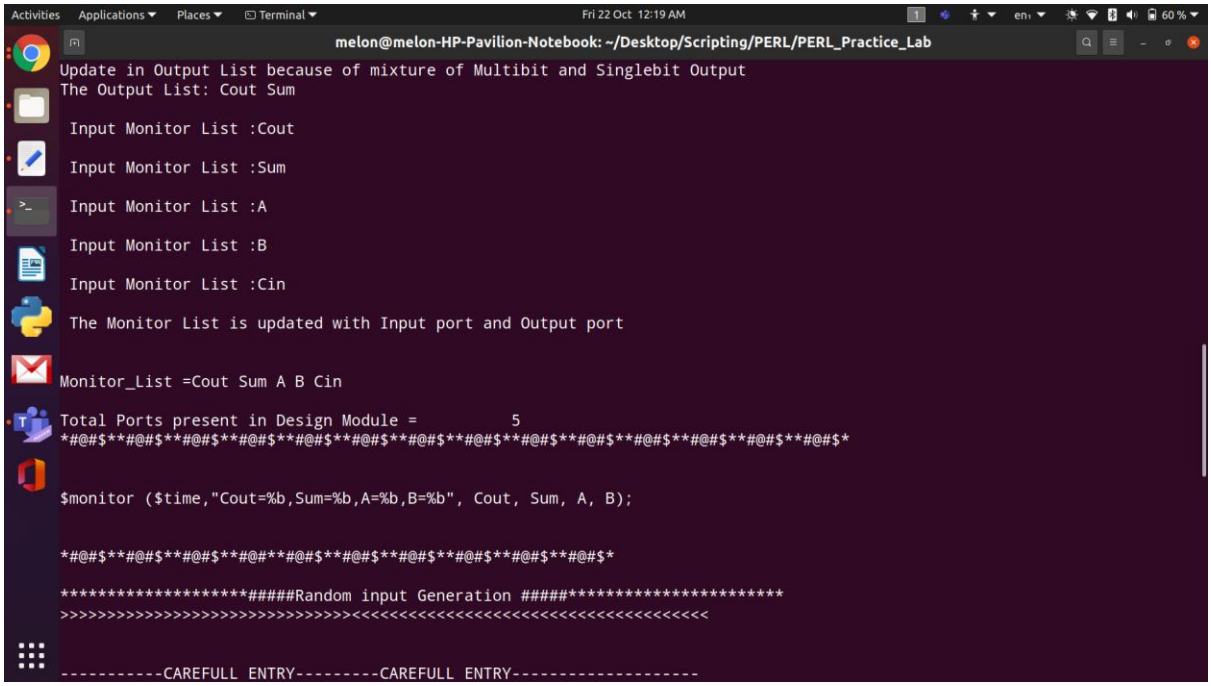


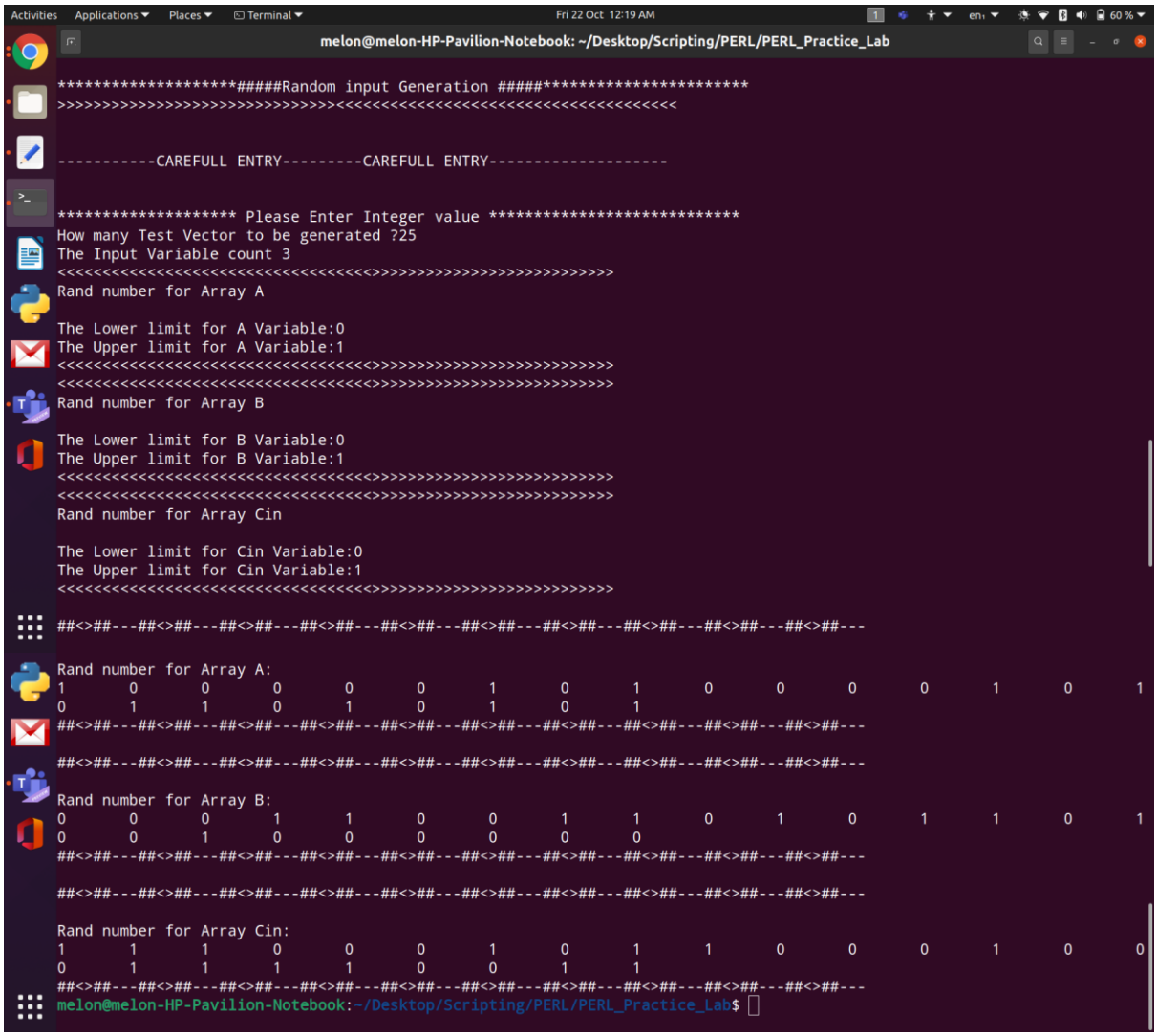
Figure 2.3 The Terminal window displays the Verilog design module and now it is processing file Line by Line.



```
Update in Output List because of mixture of Multibit and Singlebit Output
The Output List: Cout Sum

 Input Monitor List :Cout

 Input Monitor List :Sum

 Input Monitor List :A

 Input Monitor List :B

 Input Monitor List :Cin

 The Monitor List is updated with Input port and Output port


Monitor_List =Cout Sum A B Cin

Total Ports present in Design Module =          5
*#@#$**#@#$**#@#$**#@#$**#@#$**#@#$**#@#$**#@#$**#@#$**#@#$**#@#$**#@#$**#@#$**#@#$*


$monitor ($time,"Cout=%b,Sum=%b,A=%b,B=%b", Cout, Sum, A, B);


*#@#$**#@#$**#@#$**#@#**#@#$**#@#$**#@#$**#@#$**#@#$**#@#$*

********************#####Random input Generation #####***********************
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>><<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<


-----------CAREFULL ENTRY---------CAREFULL ENTRY--------------------
```

Figure 2.4 The Terminal window shows the intermediate text useful in debugging.

Date:22/10/2021



```
********************#####Random input Generation #####***********************
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>><<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<


-----------CAREFULL ENTRY---------CAREFULL ENTRY--------------------


******************** Please Enter Integer value ****************************
How many Test Vector to be generated ?25
The Input Variable count 3
<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<>>>>>>>>>>>>>>>>>>>>>>>>>>>>
Rand number for Array A

The Lower limit for A Variable:0
The Upper limit for A Variable:1
<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<>>>>>>>>>>>>>>>>>>>>>>>>>>>>
<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<>>>>>>>>>>>>>>>>>>>>>>>>>>>>
Rand number for Array B

The Lower limit for B Variable:0
The Upper limit for B Variable:1
<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<>>>>>>>>>>>>>>>>>>>>>>>>>>>>
<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<>>>>>>>>>>>>>>>>>>>>>>>>>>>>
Rand number for Array Cin

The Lower limit for Cin Variable:0
The Upper limit for Cin Variable:1
<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<>>>>>>>>>>>>>>>>>>>>>>>>>>>>

##<>##---##<>##---##<>##---##<>##---##<>##---##<>##---##<>##---##<>##---##<>##---##<>##---

Rand number for Array A:
1       0       0       0       0       0       1       0       1       0       0       0       0       1       0       1
0       1       1       0       1       0       1       0       1
##<>##---##<>##---##<>##---##<>##---##<>##---##<>##---##<>##---##<>##---##<>##---##<>##---

##<>##---##<>##---##<>##---##<>##---##<>##---##<>##---##<>##---##<>##---##<>##---##<>##---

Rand number for Array B:
0       0       0       1       1       0       0       1       1       0       1       0       1       1       0       1
0       0       1       0       0       0       0       0       0
##<>##---##<>##---##<>##---##<>##---##<>##---##<>##---##<>##---##<>##---##<>##---##<>##---

##<>##---##<>##---##<>##---##<>##---##<>##---##<>##---##<>##---##<>##---##<>##---##<>##---

Rand number for Array Cin:
1       1       1       0       0       0       1       0       1       1       0       0       0       1       0       0
0       1       1       1       1       0       0       1       1
##<>##---##<>##---##<>##---##<>##---##<>##---##<>##---##<>##---##<>##---##<>##---##<>##---
melon@melon-HP-Pavilion-Notebook:~/Desktop/Scripting/PERL/PERL_Practice_Lab$ 
```

Figure 2.5 The Terminal window is enabling user to be cautious to enter how many test vector are needed to be generated.

Screenshot of Testbench Generated for adder:

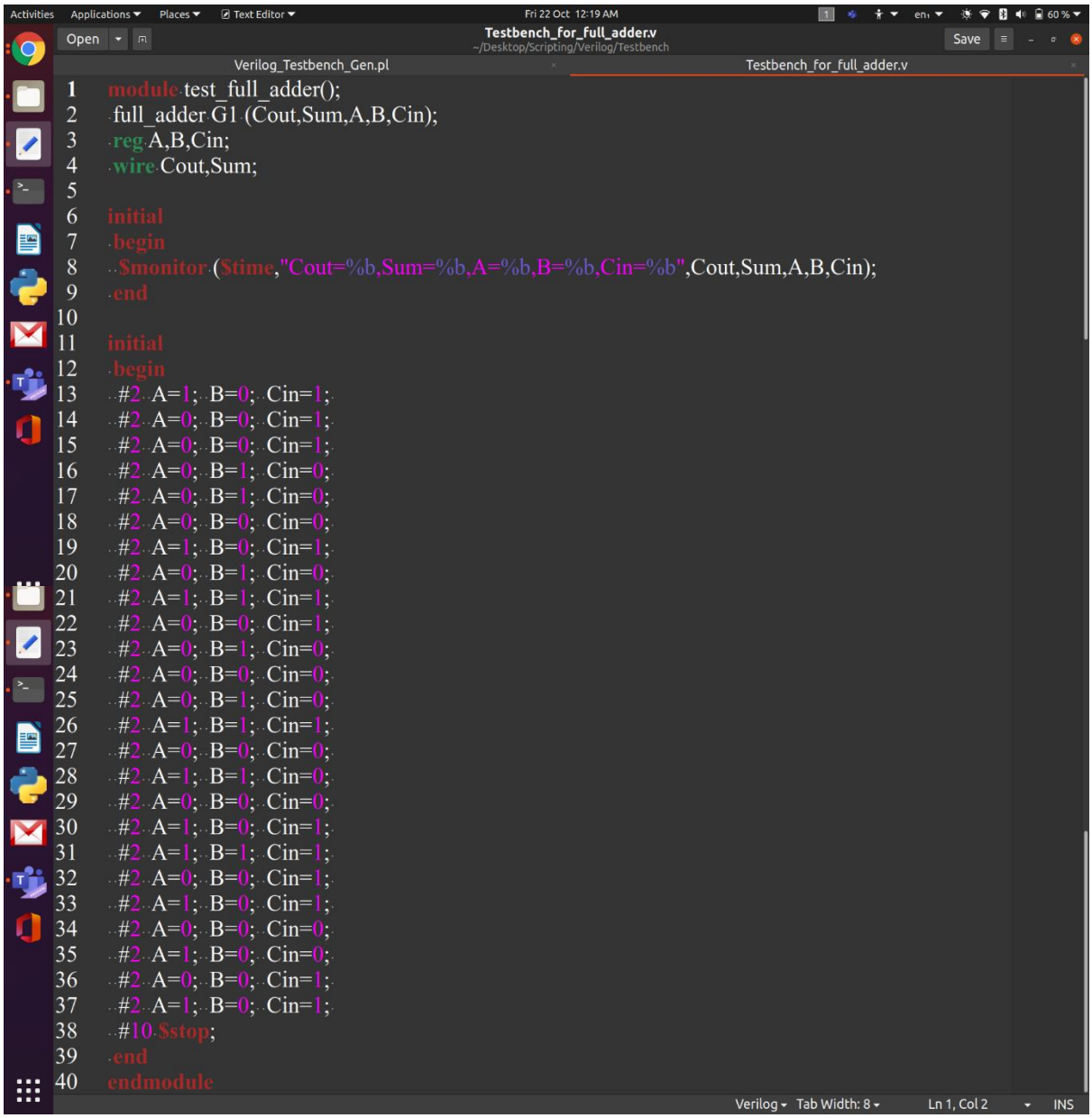


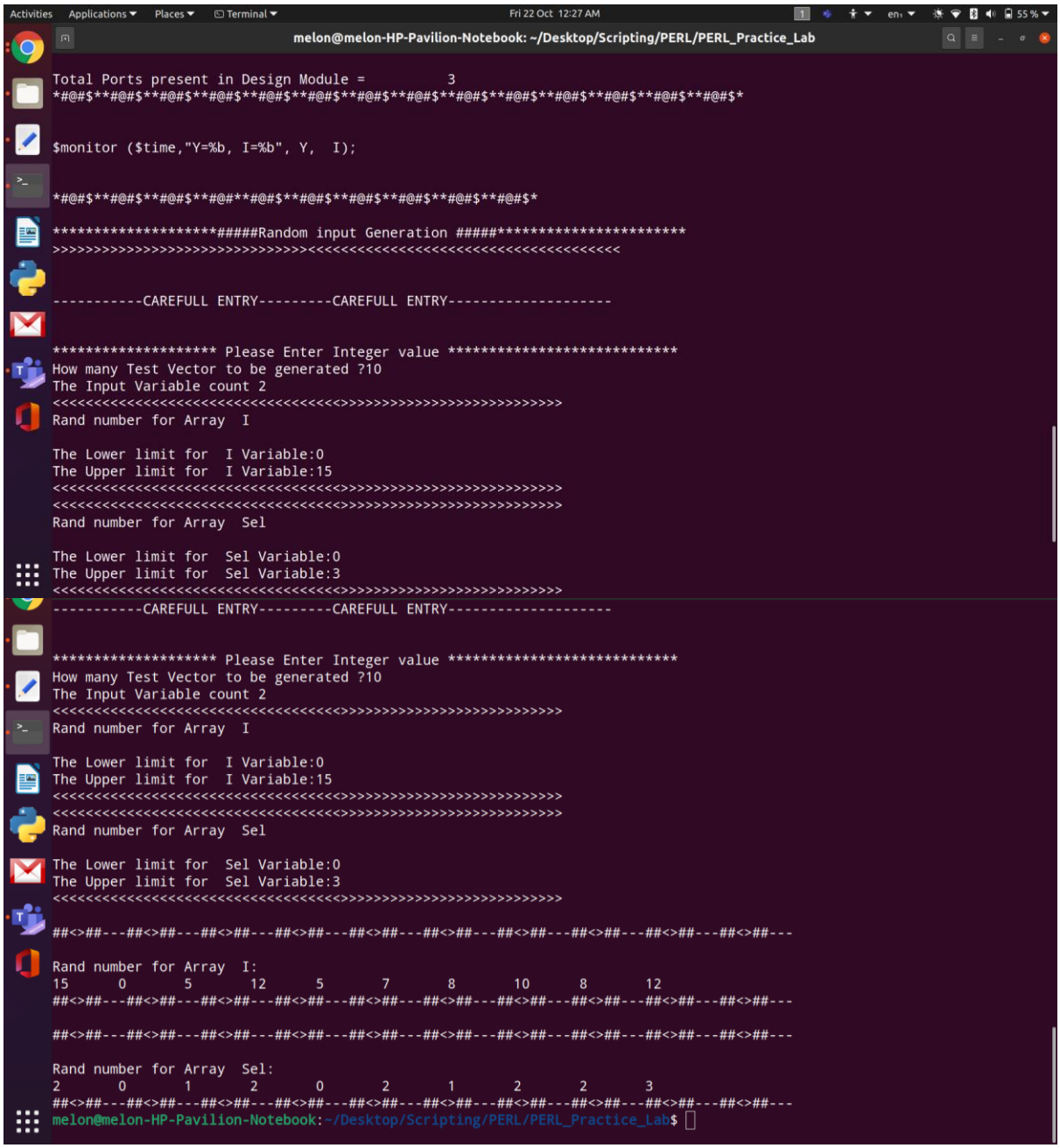
Figure 2.6 The testbench generated for 1-bit adder.

Date:22/10/2021

Example 2: Mux 4to1

```
melon@melon-HP-Pavilion-Notebook:~/Desktop/Scripting/PERL/PERL_Practice_Lab$ ./Verilog_Testbench_Gen.pl ../../Verilog/Ver
ilog_Design_Module/
Name "main::Input_Multi_LSB" used only once: possible typo at ./Verilog_Testbench_Gen.pl line 161.
Name "main::Output_Port_Count" used only once: possible typo at ./Verilog_Testbench_Gen.pl line 262.
Name "main::Variable_single_count" used only once: possible typo at ./Verilog_Testbench_Gen.pl line 185.
Name "main::Var" used only once: possible typo at ./Verilog_Testbench_Gen.pl line 166.
Name "main::Variable_multi_count" used only once: possible typo at ./Verilog_Testbench_Gen.pl line 165.
Name "main::Var1" used only once: possible typo at ./Verilog_Testbench_Gen.pl line 186.
Name "main::Output_Multi_LSB" used only once: possible typo at ./Verilog_Testbench_Gen.pl line 230.
Name "main::Input_Port_Count" used only once: possible typo at ./Verilog_Testbench_Gen.pl line 205.
#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*


The Directory path is ../../Verilog/Verilog_Design_Module/

The Verilog Files present in ../../Verilog/Verilog_Design_Module/ Directory are:
1. alu_add.v
2. alu_sub.v
3. full_adder.v
4. half_adder.v
5. mux_2to1.v
6. mux_4to1.v

#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*#*

-*-*--*--*--*-NOTE-*-NOTE-*-*--*--*--*-NOTE-*-NOTE-*-*--*--*--*-NOTE-*-NOTE---*--*--*--*--*-

-----------CAREFULL ENTRY-----------CAREFULL ENTRY---------CAREFULL ENTRY----------CAREFULL ENTRY-------------

For example the verilog file name is ABC.v then type ABC.v as itself


Select filename with extension of Verilog file:mux_4to1.v

The Total Number of Lines in Design Module is :11
The Design Module Contents
**************************************************************
*********** Module Definition *****************
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

**************************************************************
Verilog Module Selected
Generation of Testbench and Testvectors using PERL Script
The Module Name and port list:          mux_4to1 (Y,I,Sel);
The Module Name is:             mux_4to1
The Module Portlist is:          (Y,I,Sel);
The Input ports are:            [3:0] I;
************************************************
Multibit Input
MSB Size of Multibit Input:             3
Input Field Bit Length:         4
LSB Size of Multibit Input:             0
Module Input List is:            I
Input port name is:              I
************************************************
Update in Input List because of mixture of Multibit and Singlebit Input
The Input List:  I
The Input ports are:            [1:0] Sel;
************************************************
Multibit Input
MSB Size of Multibit Input:             1
Input Field Bit Length:         2
LSB Size of Multibit Input:             0
Module Input List is:            Sel
Input port name is:              Sel
************************************************
Update in Input List because of mixture of Multibit and Singlebit Input
The Input List:  I  Sel
The Output ports are:           Y;
&*&*&*&*&*&*&&*&*&*&*&*&*&*&*&*&*&*&*&*&*&*&*&*
Output is Single bit
Module Input List is Y
Output port name is Y
Update in Output List because of mixture of Multibit and Singlebit Output
The Output List: Y

 Input Monitor List :Y

 Input Monitor List : I

 Input Monitor List : Sel

 The Monitor List is updated with Input port and Output port


Monitor_List =Y  I  Sel
```

Figure 2.7: The user now selecting Mux_4to1 Module and the perl script is processing to generate tesbench



```
Total Ports present in Design Module =          3
*#@#$**#@#$**#@#$**#@#$**#@#$**#@#$**#@#$**#@#$**#@#$**#@#$**#@#$**#@#$**#@#$**#@#$*


$monitor ($time,"Y=%b, I=%b", Y,  I);


*#@#$**#@#$**#@#$**#@#**#@#$**#@#$**#@#$**#@#$**#@#$**#@#$*

*******************#####Random input Generation #####**********************
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>><<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<


-----------CAREFULL ENTRY---------CAREFULL ENTRY--------------------


******************** Please Enter Integer value ****************************
How many Test Vector to be generated ?10
The Input Variable count 2
<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
Rand number for Array  I

The Lower limit for  I Variable:0
The Upper limit for  I Variable:15
<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
Rand number for Array  Sel

The Lower limit for  Sel Variable:0
The Upper limit for  Sel Variable:3
<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
-----------CAREFULL ENTRY---------CAREFULL ENTRY--------------------


******************** Please Enter Integer value ****************************
How many Test Vector to be generated ?10
The Input Variable count 2
<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
Rand number for Array  I

The Lower limit for  I Variable:0
The Upper limit for  I Variable:15
<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
Rand number for Array  Sel

The Lower limit for  Sel Variable:0
The Upper limit for  Sel Variable:3
<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>

##<>##---##<>##---##<>##---##<>##---##<>##---##<>##---##<>##---##<>##---##<>##---##<>##---

Rand number for Array  I:
15      0       5       12      5       7       8       10      8       12
##<>##---##<>##---##<>##---##<>##---##<>##---##<>##---##<>##---##<>##---##<>##---##<>##---

##<>##---##<>##---##<>##---##<>##---##<>##---##<>##---##<>##---##<>##---##<>##---##<>##---

Rand number for Array  Sel:
2       0       1       2       0       2       1       2       2       3
##<>##---##<>##---##<>##---##<>##---##<>##---##<>##---##<>##---##<>##---##<>##---##<>##---
melon@melon-HP-Pavilion-Notebook:~/Desktop/Scripting/PERL/PERL_Practice_Lab$ 
```

Figure 2.8 The Terminal window is enabling user to be cautious to enter how many test vector are needed to be generated. The upper limit and lower limit is specified by the Verilog module like [3:0] means 4 bit (0 to 15) for I and [1:0] for Sel.

Screenshot of Testbench Generated for Mux_4to1

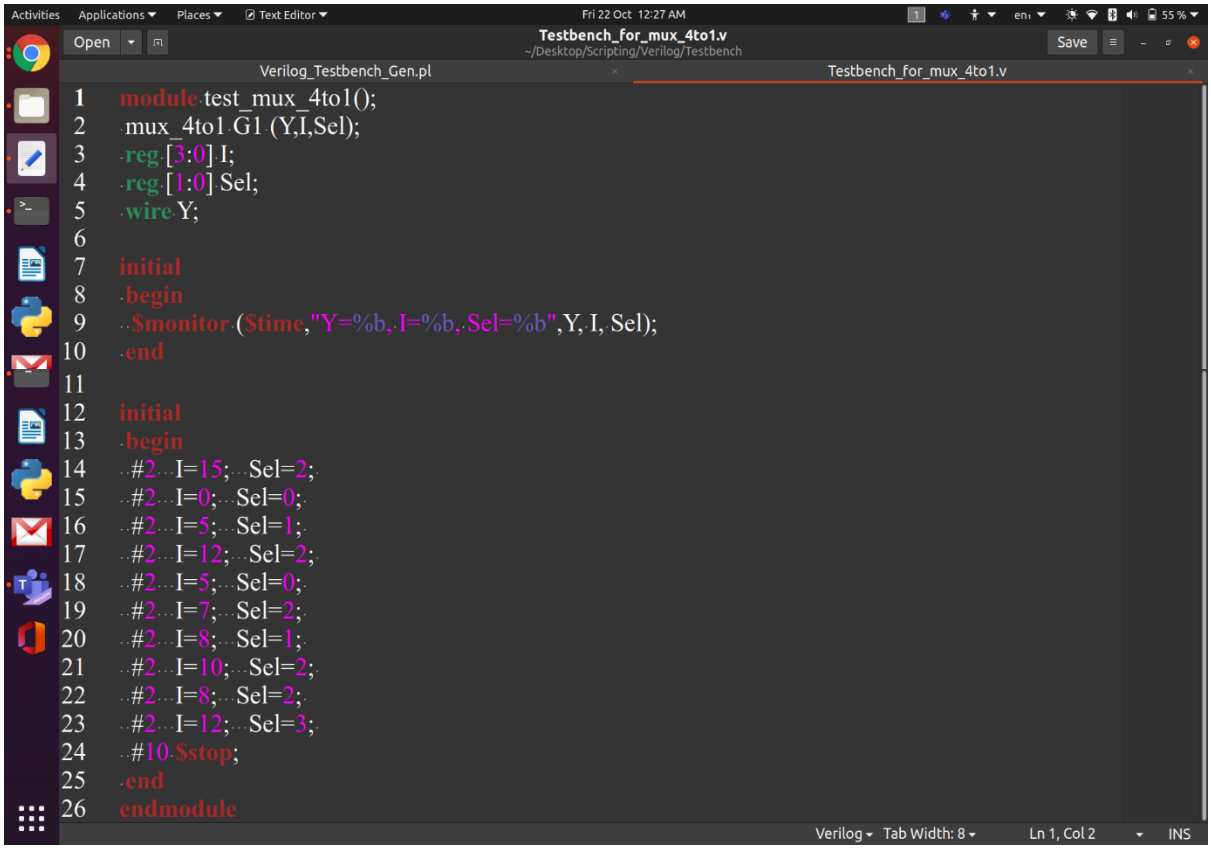


Figure 2.9 The testbench generated for Multiplexer 4 to 1.

Example 3: Subtractor

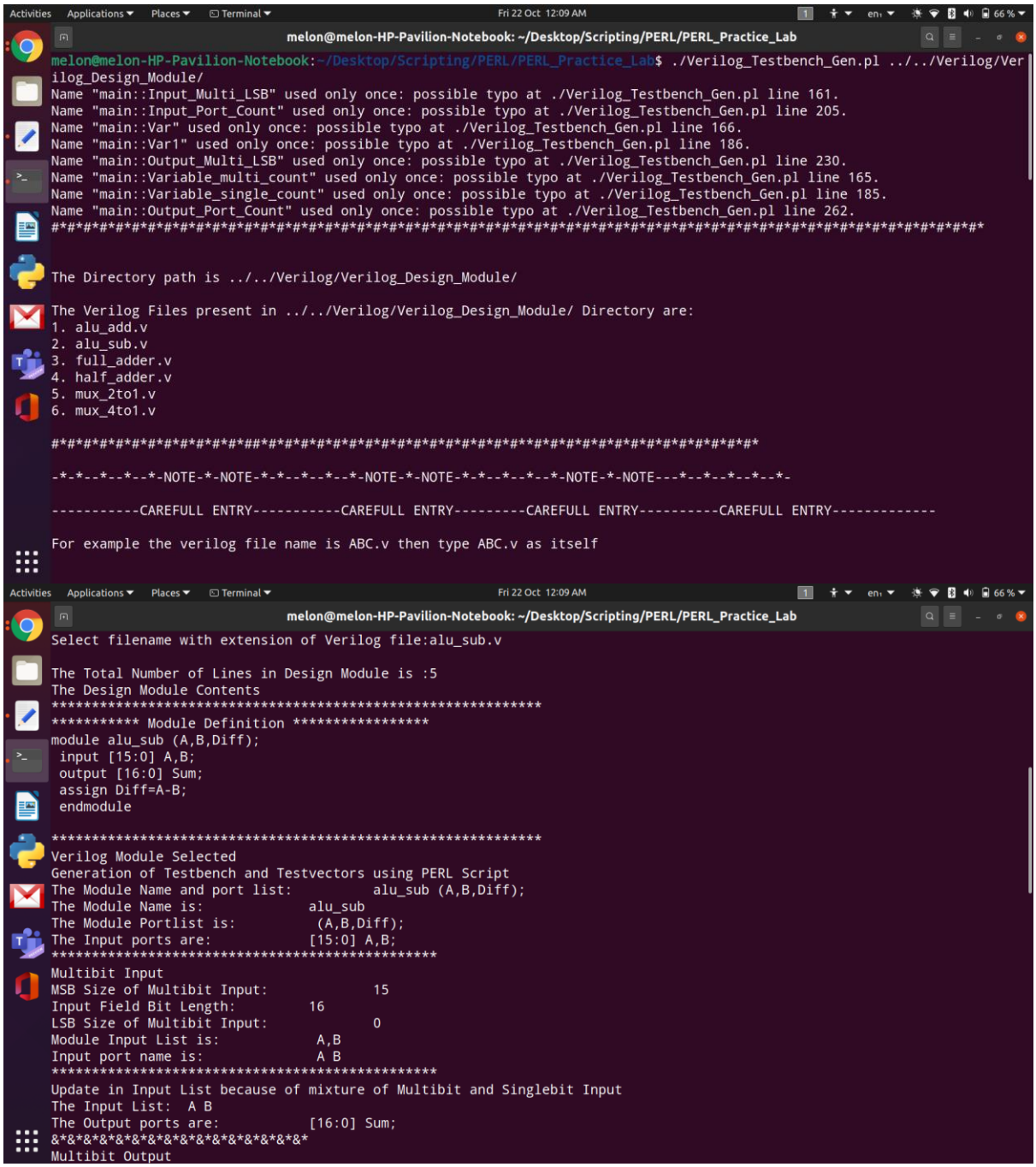
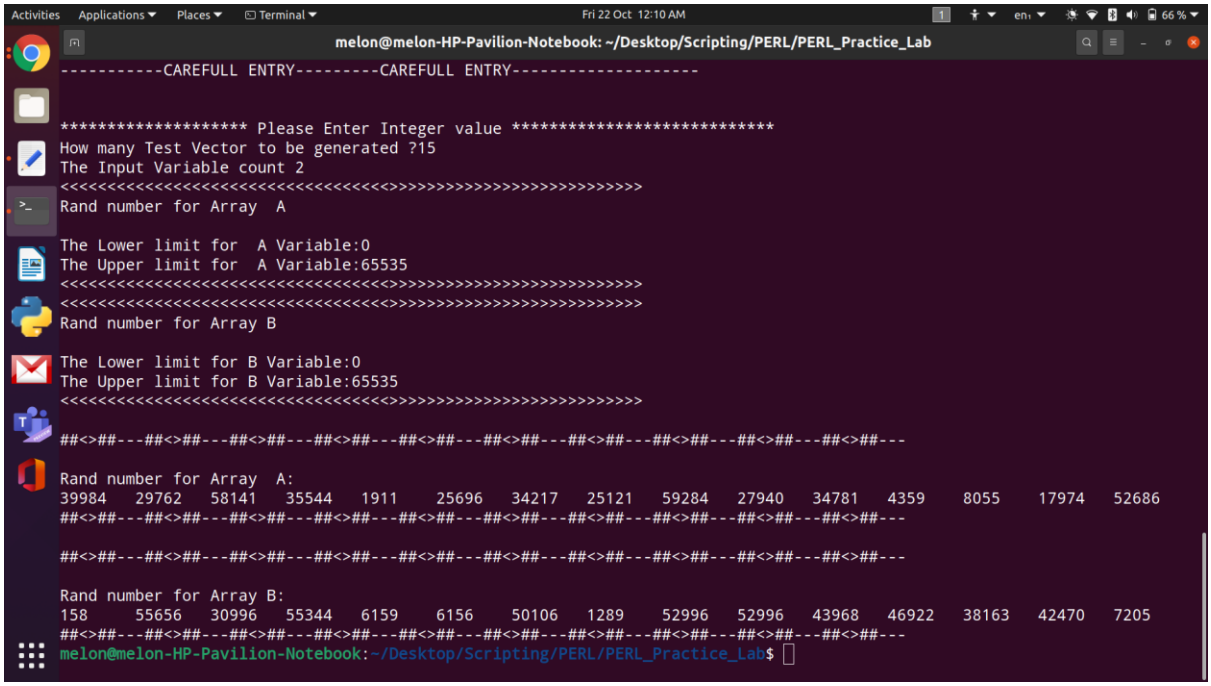


Figure 2.10: The user now selecting sub_add.v Module file and the perl script is processing to generate tesbench**.**



```
-----------CAREFULL ENTRY---------CAREFULL ENTRY--------------------


******************** Please Enter Integer value ****************************
How many Test Vector to be generated ?15
The Input Variable count 2
<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<>>>>>>>>>>>>>>>>>>>>>>>>>>>
Rand number for Array  A

The Lower limit for  A Variable:0
The Upper limit for  A Variable:65535
<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<>>>>>>>>>>>>>>>>>>>>>>>>>>>
<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<>>>>>>>>>>>>>>>>>>>>>>>>>>>
Rand number for Array B

The Lower limit for B Variable:0
The Upper limit for B Variable:65535
<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<>>>>>>>>>>>>>>>>>>>>>>>>>>>

##<>##---##<>##---##<>##---##<>##---##<>##---##<>##---##<>##---##<>##---##<>##---##<>##---

Rand number for Array  A:
39984   29762   58141   35544   1911    25696   34217   25121   59284   27940   34781   4359    8055    17974   52686
##<>##---##<>##---##<>##---##<>##---##<>##---##<>##---##<>##---##<>##---##<>##---##<>##---

##<>##---##<>##---##<>##---##<>##---##<>##---##<>##---##<>##---##<>##---##<>##---##<>##---

Rand number for Array B:
158     55656   30996   55344   6159    6156    50106   1289    52996   52996   43968   46922   38163   42470   7205
##<>##---##<>##---##<>##---##<>##---##<>##---##<>##---##<>##---##<>##---##<>##---##<>##---
melon@melon-HP-Pavilion-Notebook:~/Desktop/Scripting/PERL/PERL_Practice_Lab$ 
```

Figure 2.11: The screenshot shows the random number generated for input A and B. In the screenshot user has entered 15 and A and B input are of 16-bit input. Therefore, the upper and lower limit range from 65535 to 0.

Date:22/10/2021

Screenshot of Testbench generated:

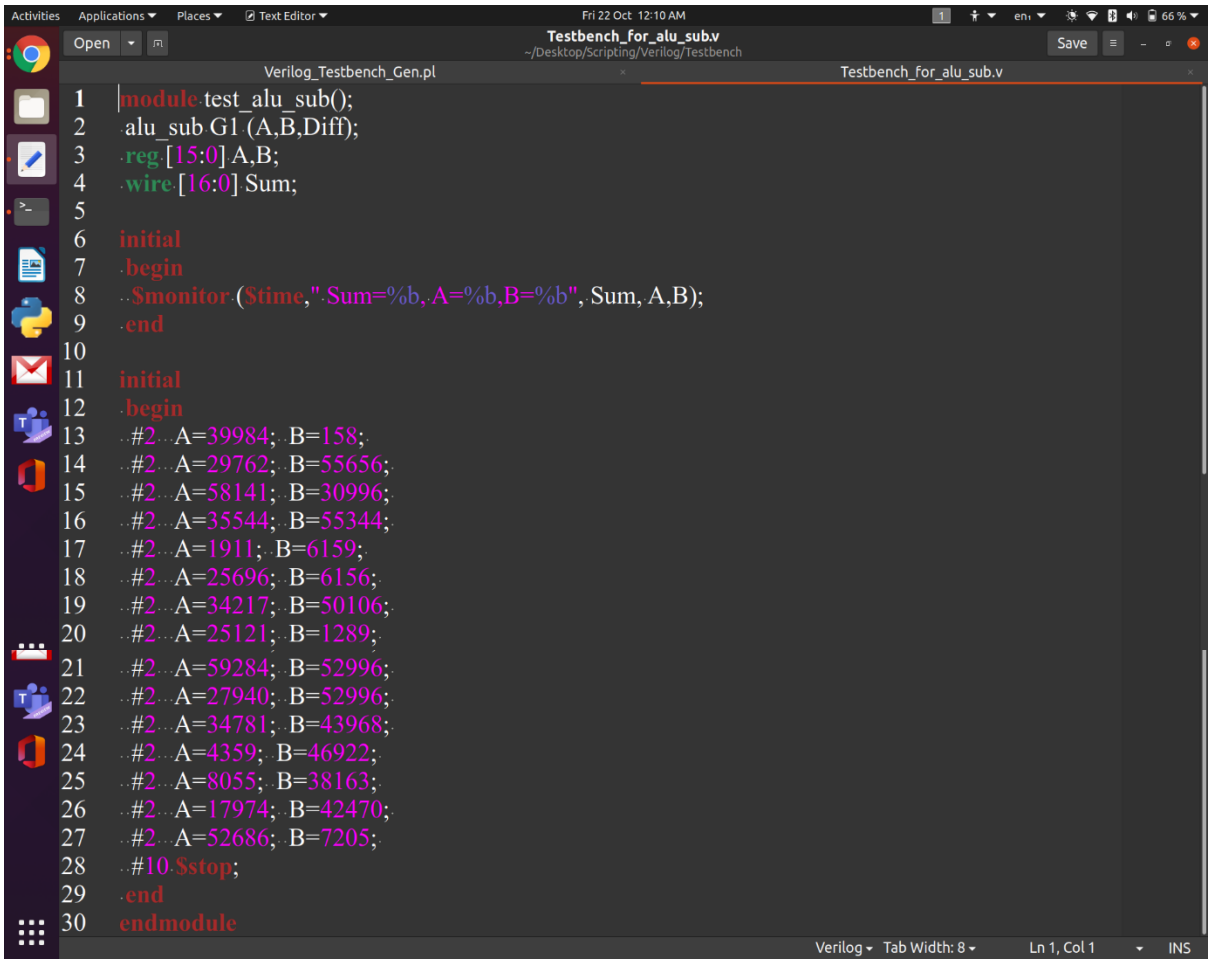


Figure 2.12 : The testbench generated for 16bit-Subtraction.

Inference:

1. Writing PERL Script and how to execute it in Terminal window.
2. Getting familiarised with Scalar Data, Arrays and List Data, Control Structure, Hashes syntax and using in the script.
3. Learning inbuilt functions like chomp(), chop(), rand(), print(), shift(), unshift(), pop(), push(), sort(), reverse() and many more.
4. Learning File handling, text manipulation of files. In built functions of file handling open(), close() and file modes.
5. Learning Directories handling. In built functions of directories handling opendir(), closedir() and readdir().
6. Writing PERL Script for VLSI Automation.