```
#! /usr/bin/perl -w
# -w to turn on the warnings
use warnings;

$Dirname=$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 "end\n";
}
```