Test Bench Project

Members:

Bruno Hernández López

Jose Alberto Gomez Diaz

Josue Isaias Gomez Cosme

Christian Aaron Ortega Blanco

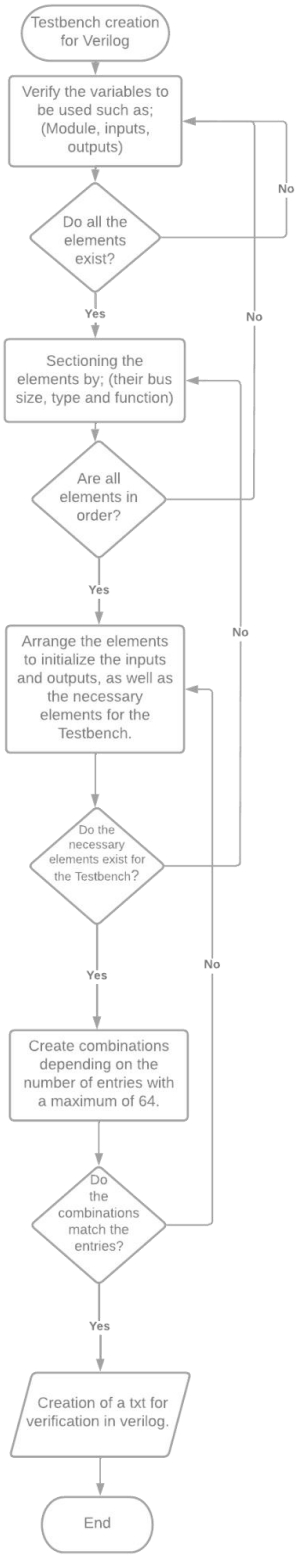
Index

Summary

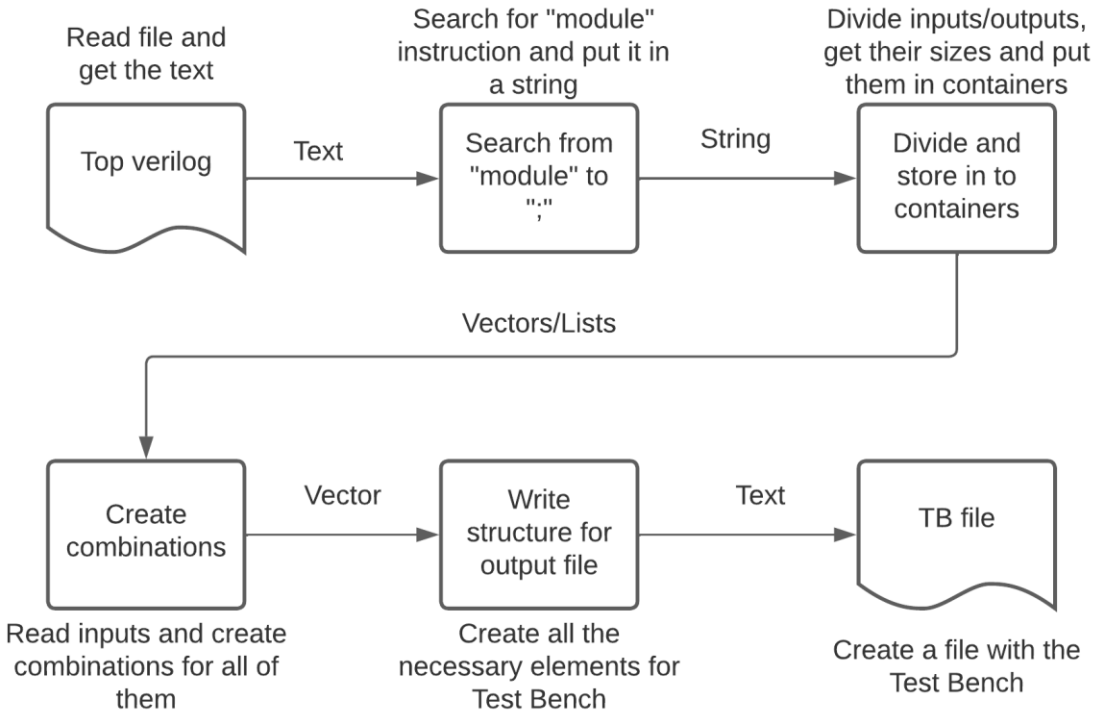
The objective of this document is to create a program that makes it easier for Verilog programmers to create a Testbench, to reduce and save time when testing programs in Verilog. The program that will be presented in this document is a support that does not eliminate the interaction of the programmer with the program, it simply reduces the need to register within the program each of the variables, so they will only focus on how to use the program and what elements will be tested according to the needs of each programmer.

Methodology

Program design methodology



Program flow diagram



Description of what it does and how it does it (inputs, outputs, types of statements, etc.)

Obtaining the necessary project information

Initially the program receives as input a file corresponding to the design of a program in verilog. This file is opened and read line by line until one of the identifiers "module", "input" or "output" is found. Each time one of these identifiers is found, the line is read. To ensure that only the necessary information is extracted, each time a line is read, an identifier is searched that indicates the existence of a comment and deletes everything to its right, the words “reg” and “wire” are also removed to avoid problems when separating the variables. Once the unnecessary information is removed from the line, it is concatenated with the rest of the lines to form a single string.

Identification and separation of variables

In this part of the code, some containers were used to separate the different necessary variables. These containers were the following:

var_input container where inputs are stored
var_output container where outputs are stored

input_list list of inputs without identifying or size of buses
output_list list of outputs without identifying or size of buses

var_list names of the inputs and outputs without taking into consideration the size of the bus

inputs number of bits per input
module_name module name

Once the module instruction is obtained with all the necessary information, said instruction is processed to separate it into inputs and outputs with their respective bus size. For this, the string was initially trimmed by removing everything outside the parentheses. This information is assigned to a new string called "t" which, by means of a while loop, will be processed until it has finished separating all the inputs and outputs. Next, the character “,” is searched for to identify if it is a single variable or if it is more than one. If the character “,” is not found, the algorithm assumes that it is the last variable and takes it as the last instruction to separate by assigning. If it finds the character "," then the program takes from the start of the string to a position before the first "," and takes it as an instruction. Then that instruction is removed from the original string and the first word of this instruction is read and compared with the words "input" and "output" to know if it is an input or output. In case of matching one of these two words, the algorithm will replace the word “input” with “reg” or “output” with “wire” and it is added to its respective container. After this (in the case that it is an input) the character “]” is searched to identify whether it is a data bus or a 1-bit variable. If this character is found, then the number contained between the characters “[“ and “:” is taken and the same is done with the number between the characters “:” and “]” to then add a “1” to the difference between the magnitudes of these numbers and add it to the “inputs” container where the sizes of each entry are stored. After this, “reg” and the bus size of the instruction are trimmed and added to the input list as well as raising a flag to identify that the last instruction read was an input. In the case of an "output", a similar procedure is carried out except for the part in which the size of the bus is obtained. If the words “input” or “output” are not found at the beginning of the instruction, the program proceeds to identify what type the last instruction read was. It immediately accesses this last instruction and copies both the type and the size of the bus (if it is one).

Filling the testbench structure

With the information of the different containers explained in the previous section, the testbench file can be created by filling the information into a testbench structure.

Depending on the programming language, the first thing to do is create or open the text file where the data wants to be stored. After that, with a function of the class the information can be written according to the structure. In the case of the code presented in this work, the first part that will be filled does not depend on the information of the module since it will be the timescale, which was assigned as a standard value. Then, the name of the testbench will be written, this name will also be generated automatically by the program so it will be nondependent on the Verilog module.

The next information to be filled is the inputs and outputs of the testbench, this information are loaded from containers “inputs” and “outputs”, while doing this, the code ensures that the both inputs and outputs will have the same size as the signals of the Verilog module so it won’t cause any problem.

Following the structure of the testbench, the next part is the signal assignments within the Verilog module and the testbench. The program will write this part by creating an implicit

assignation and to do that it fills the information of the module name container, followed by a code generated word “DUT” and filling the information of the container “inouts”, which has only the names of the inputs and outputs.

Other information written are the $dumpvars and $dumpfile elements needed to generate a waveform simulation and the monitor to see the signals in the Kernel.

One of the most important elements of the testbench is the combinations of the inputs to generate the outputs, to do this, an accumulator with the name “combinaciones” is generated with performs the addition of the elements of the container “bitsint”, which has the values of lengths of the inputs, and the total combinations are calculated by the 2^n pattern, where n is equal to the result of the accumulator “combinaciones”. To assign those combinations to the different inputs, for cycle is implemented, where a concatenation of the value of i is assigned to the different inputs and incremented every cycle to see the different combinations.

Some final statements required for the testbench are written and the generation of the txt file for the testbench is ready and filled with the information extracted from the Verilog module.

CODES

Code in Python with Regular Expressions

```
from pickle import FALSE, TRUE

import array as arr
import math

import re                                                   #The library is imported
#######################################################Josue
man = open('design.txt')                                    #The file opens
patterns=['module ','input ','output ']                     #The regular expressions that
will be used are created.
Lines= []                                                   #A variable is created as a
vector
Final=''                                                    #A variable is created as a
string

for linea in man:                                           #A for is created to be able to
read the content in lines
    linea = linea.rstrip()                                  #Command to read content in lines
    for pattern in patterns:                                #A for is created to search for
the words set in the for in each line of the document.
        if re.search(pattern,linea):                        #An if is created to search for
regular expressions.
            linea = linea.strip()                           #Line breaks are removed by
default in python
            #linea = linea.rstrip(",")                      #Commas are removed to the right
            linea = linea.replace("reg","")                 #Replace variables "reg",
"wire"," and ";" by none
            linea = linea.replace("wire","")
            linea = linea.replace(";","")
            linea = linea.replace(")","")
            if re.search('\,',linea):                       #Each line is searched until the
end of the line.
                 linea = linea.rstrip(linea[1])
            Lines.append(linea)
            break
print(Lines)

Final= ''.join(Lines)                                      #The vector is converted to a
string
Final += ')'                                               #The missing elelemnt is
concatenated
#Final=re.sub(',','',Final,1)                              #The excess element is removed
print(Final)
print(len(Lines))                                          #Printed

#################################################Jose
```

```
#module="module    buses(input intp,input[5:1] entrada0,prueba,output[1:0]
salida,salida4,input [3:0] entrada1,input [12:0] entrada2,prueba2,output
sal,sali);//comentarios"
module=Final
var_input=[]        #inputs**
var_output=[]       #outputs**

input_list=[]         #list of entries without bus identifier and size**
output_list=[]        #list of outputs without bus identifier and size**.

var_list=""        #names of inputs and outputs regardless of bus size**
inputs=[]          #number of bits per input**
module_name=""       #module name**


#The program begins

module_name=module[6:module.find("(")]           #The name of the module is obtained
module_name = re.sub("\ ","",module_name)        #The spaces " " are eliminated to leave
only the name
t=module[module.find("(")+1:module.find(")")]    #you get what is between "( )" by
cutting out the rest

isinput=FALSE
while(t!=""):
    instruction=""
    if "," in t:
        instruction=t[0:t.find(",")]         #is taken as an instruction from position 0
to the decimal point
        t=t[t.find(",")+1:]                  #the instruction is removed from the string
t

    else:   #if the "," character is no longer found
        instruction=t   #all that is left in t is taken as instruction
        t=""            #string is emptied to complete the process

#extraer entradas

    if re.match("input",instruction)!=None : #if instruction begins with word input
        instruction=instruction.replace("input","reg",1)     #input is replaced by reg
        var_input.append(instruction)  #entry is added to the list of entries

        #add to variable string without identifiers or bus size

        if "]"  in instruction: #DETECTS IF IT IS A BUS
            r1=int(instruction[instruction.find("[")+1:instruction.find(":")])  #look
for the number on the left side of the bus
            r2=int(instruction[instruction.find(":")+1:instruction.find("]")])  #look
for the number on the right side of the bus
            if r1>r2:                            #the bus order is identified
                inputs.append(r1-r2+1)           #the size of the bus is added to the
list of inputs
            elif r2>=r1:                         #the bus order is identified
```

```
                inputs.append(r2-r1+1)          #the size of the bus is added to the
list of inputs
                                                                #number of buses per inlet is
added
            instruction=instruction[instruction.find("]")+1:] #"reg" and buse size are
cut off

        else:                   #if it is not a bus
            inputs.append(1) #1 is added to list of input sizes
            instruction=instruction[instruction.find("g")+1:] #"reg" is cut
        var_list+=instruction+"," #is added to the list of variables
        instruction = re.sub("\ ","",instruction)   #blanks are removed " "
        input_list.append(instruction) #is added to list of entries without identifier
or bus

        isinput=TRUE


#extraer salidas

    elif re.match("output",instruction)!=None : #if instruction begins with word output
        instruction=instruction.replace("output","wire",1)    #output is replaced by reg
        var_output.append(instruction)  #entry is added to the output list

        #add to variable string without identifiers or bus size

        if "]"  in instruction: #DETECTS IF IT IS A BUS
            instruction=instruction[instruction.find("]")+1:] #wire" and bus size are
cut down
        else:
            instruction=instruction[instruction.find("e")+1:] #"wire" is cut
        var_list+=instruction+","   #is added to the list of variables
        instruction = re.sub("\ ","",instruction)   #blanks are removed " "
        output_list.append(instruction) #is added to list of entries without identifier
or bus


        isinput=FALSE


#extract inputs or outputs without identifier

    elif isinput==TRUE:         #if the first word does not specify whether it is input
or output it is analyzed if the last element was input or output
        temp=var_input[-1]      #if it was input the last item in the input list is
taken
        if "]" in temp:         #if "]" character is found, it means it is a bus.
            temp=temp[0:temp.find("]")+1]       #copy "reg" of the input and its bus
size
            #tamaño del bus
            r1=int(temp[temp.find("[")+1:temp.find(":")])  #look for the number on the
left side of the bus
```

```
            r2=int(temp[temp.find(":")+1:temp.find("]")])  #search for the number to the
right of bus
            if r1>r2:                           #the bus order is identified
                inputs.append(r1-r2+1)          #the size of the bus is added to the
list of inputs
            elif r2>=r1:                        #the bus order is identified
                inputs.append(r2-r1+1)          #the size of the bus is added to the
list of inputs
                                                        #number of buses per entry is
added
        else:                   #if it is not a bus
            inputs.append(1) #1 is added to list of input sizes
            temp="reg"      #just copy the "reg"
        temp=temp+" "+instruction               #concatenate "reg" + " " + the name of
the instruction
        var_input.append(temp)                  #is added to the list of entries

        #add to variable string without identifiers or bus size
        var_list+=instruction+","
        instruction = re.sub("\ ","",instruction)   #blanks are removed " "
        input_list.append(instruction) #is added to list of entries without identifier
or bus

    elif isinput==FALSE:        #if last variable was output

        temp=var_output[-1]     #if it was output the last item in the output list is
taken
        if "]" in temp:         #if "]" character is found, it means it is a bus.
            temp=temp[0:temp.find("]")+1]       #wire copy of the output and its bus
size
        else:                   #if it is not a bus
            temp="wire"      #just copy the wire
        temp=temp+" "+instruction               #concatenates "wire" + " " + the name of
the instruction
        var_output.append(temp)                 #is added to the list of departures
        #add to variable string without identifiers or bus size
        var_list+=instruction+","
        instruction = re.sub("\ ","",instruction)   #blanks are removed " "
        output_list.append(instruction) #is added to the output list without identifier
or bus

if "," in var_list:
    var_list=var_list[:-1]  #the last "," is deleted
var_list = re.sub("\ ","",var_list)       #Spaces " " are deleted

print("LISTAS")
print("ENTRADAS: ")
print(var_input)
print("SALIDAS: ")
print(var_output)
print("LISTA DE VARIABLES: ")
print(var_list)
print("INPUT_LIST: ")
```

```
print(input_list)
print("OUTPUT_LIST: ")
print(output_list)
print("INPUTS: ")
print(inputs)
print("MODULO")
print(module_name)
##############################Bruno
# The possible solutions are created,
# in case there are more than 64 possible combinations, the first 10,
# the middle 10 and the last 10 are given.
#bits=arr.array('i',[1,2,1,1])

lista=input_list
outputs=output_list


TotalInputs=len(input_list)
TotalOutputs=len(output_list)
accum=0
for w in inputs:
    accum=accum+w

pot = (2**accum)
limpot=(pot/2)
limpotint=int(limpot)

limpotm=limpotint-10
limpotM=limpotint+10
pot_s=str(pot)
limpot_s=str(limpot)
limpotm_s=str(limpotm)
limpotM_s=str(limpotM)
potm=pot-10
potm_s=str(potm)


lines = ['`timescale 1ns/100ps', 'module mux_TB ();']

for i in range(len(var_input)):
    lines.append(var_input[i]+';')

for i in range(len(var_output)):
    lines.append(var_output[i]+';')


module_name=module_name+" DUT"+'('+var_list+')'
lines.append(module_name+';')
lines.append('initial begin')
lines.append('  $dumpvars;')
lines.append('  $dumpfile("dump.vcd");')
##//////////////////////////////////////////////
# The necessary elements are created to perform the Testbench,
```

```
# using all the variables and information above as a vector.
print (pot)
if pot<=64:


    for i in range (0,TotalInputs):
        lines.append(lista[i]+'<=0;')

    monitor=('$monitor("')
    monitor+='in: '
    for i in range (0,len(input_list)):
        monitor+='%b '
    monitor+='out: '
    for i in range (0,len(output_list)):
        monitor+='%b '
    monitor+='" '

    for i in range (0,TotalInputs):
        monitor+=','
        monitor+=lista[i]
    for i in range (0,TotalOutputs):
        monitor+=','
        monitor+=outputs[i]
    monitor+=');'
    lines.append(monitor)
    lines.append('for (int i=0;i<'+pot_s+';i=i+1) begin')
    s = ','.join(lista)
    lines.append('   {'+s+'}=i;')
    lines.append('   #10;')
    lines.append('end')


elif pot>64:
    for i in range (0,TotalInputs):
        lines.append(lista[i]+'<=0;')

    monitor=('$monitor("')
    monitor+='in: '
    for i in range (0,len(input_list)):
        monitor+='%b '
    monitor+='out: '
    for i in range (0,len(output_list)):
        monitor+='%b '
    monitor+='" '

    for i in range (0,TotalInputs):
        monitor+=','
        monitor+=lista[i]
    for i in range (0,TotalOutputs):
        monitor+=','
        monitor+=outputs[i]
    monitor+=');'
    lines.append(monitor)
```

```
    lines.append('for (int i=0;i<10;i=i+1) begin')
    s = ','.join(lista)
    lines.append('   {'+s+'}=i;')
    lines.append('   #10;')
    lines.append('end')

    lines.append('for (int i='+limpotm_s+';i<'+limpotM_s+';i=i+1) begin')
    s = ','.join(lista)
    lines.append('   {'+s+'}=i;')
    lines.append('   #10;')
    lines.append('end')

    lines.append('for (int i='+potm_s+';i<'+pot_s+';i=i+1) begin')
    s = ','.join(lista)
    lines.append('   {'+s+'}=i;')
    lines.append('   #10;')
    lines.append('end')
##////////////////////////////////////////////
lines.append('  $finish;')
lines.append('end')
#\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\
lines.append('endmodule')
with open('automatic_TB.txt', 'w') as f:
    f.write('\n'.join(lines))
```

Captures and changes in code execution

Code in C++

```
// basic file operations
#include <iostream>
#include <fstream>

#include <string>
#include <vector>
#include <math.h>
using namespace std;


string module_name;
////////////////////////////////////////////////////////////////////////////////////////
josue
string lectura() {
    ifstream archivo;
    string texto;                                       //The string Text is created
    archivo.open("prueba_mux_design.txt", ios::in);       //The file is opened in read
mode
    string Prueba;                                      //The global variable Test is
created
    string Apend;
    if (archivo.fail()) {                               //If the file cannot be opened
```

```
        cout << "No se pudo abrir el archivo";         //The message is sent
        exit(1);                                    //Exits the program
    }
    while (!archivo.eof()) {                            //As long as it is not the end of
the file
        getline(archivo, texto);                   //You get the text
        if (texto.find("//")!=string::npos){        //If is used to remove comments
within the text.
            int ecom = texto.find("//");
            texto.erase(ecom,t1-ecom);
        }
        if (texto.find('`')!=string::npos){        //If is used to remove comments within
the text.
            int t1 = texto.length();                    //The size of each line of text is
searched to find the size of each line of text, to place the comments in the necessary
places.
            int ecom = texto.find('`');             //
            texto.erase(ecom,t1-ecom);
        }
        if (texto.find('#')!=string::npos){        //If is used to remove comments within
the text.
            int t1 = texto.length();                    //The size of each line of text is
searched to find the size of each line of text, to place the comments in the necessary
places.
            int ecom = texto.find('#');
            texto.erase(ecom,t1-ecom);
        }

        Prueba += texto;                                //The text is concatenated
    }
    archivo.close();                                    //The file is closed

    string text = Prueba;
    string temp=Prueba.substr(Prueba.find("module"),Prueba.find('('));
    temp.erase(0,Prueba.find("module")+6);
    module_name.append(temp);
    int a = Prueba.find("(");                        //Brackets are found
    //cout << a << endl;
    int b = a + 1;
    int c = Prueba.find(")", a);
    //cout << c << endl;

                                            //Assigns the value of the usable text to a
string
    for (int i = a; i <= c; i++) {
        Apend.push_back(Prueba[i]);
    }

    //To remove the wire and reg text, proceed as follows
    while(Apend.find(" wire ")!=string::npos){
        int e = Apend.find(" wire ");
        Apend.erase(e,6);
    }
```

```
    while(Apend.find(" wire[")!=string::npos){
        int e = Apend.find(" wire[");
        Apend.erase(e,5);
    }

    while(Apend.find(" reg ")!=string::npos){
        int f = Apend.find(" reg ");
        Apend.erase(f,5);
    }

    while(Apend.find(" reg[")!=string::npos){
        int f = Apend.find(" reg[");
        Apend.erase(f,4);
    }

  return Apend;


}
////////////////////////////////////////////////////////////////////////////////////////
jose
string module=lectura();

vector<string> inputs;  //input vector
vector<string> outputs; //output vector

vector<int> bitsint;       //vector with the number of bits per input
vector<int> bitsout;        //vector with the number of bits per output

vector<string> bitrgint;       //vector with range of input buses
vector<string> bitrgout;       //vector with range of output buses
string inouts="";
int main()
{
    cout<<module;

    int par=module.find('(');
    string t=module.substr(par+1,module.find(")")-par-1);//take what is in brackets and
paste in variable t

    bool linout=false;    //true si the last variable due intput or false si the last
variable fue output

    while(t.length()>0){ //extract all instructions from the string
        string instruction="";//we create string to store the instruction

        if(t.find(',')!=string::npos){
            instruction=t.substr(0,t.find(','));//is taken as an instruction from the
first character up to the decimal point

            t.erase(0,t.find(',')+1);//the instruction including the comma is deleted
        }
```

```
        else{
            instruction=t; //everything left in the string is taken as an instruction
            t.erase(0,t.back());//to eliminate the string
        }

        if(instruction.find("input")!=-1){
            instruction.replace(0,instruction.find("input")+5,"reg ");//the word "input"
is replaced by "reg".
            inputs.push_back(instruction);//is added to vector
            string temp=instruction;
            temp.replace(0,(temp.find(']')+1),"");
            if (temp.substr(0,3)=="reg")temp.replace(0,3,"");
            inouts+=temp;
            inouts+=",";
            linout=true; //flag was input
        }
        else if(instruction.find("output")!=-1){
            instruction.replace(0,instruction.find("output")+6,"wire ");//the word
"output" is replaced by "wire".
            outputs.push_back(instruction);//is added to vector
            string temp=instruction;
            temp.replace(0,(temp.find(']')+1),"");
            if (temp.substr(0,4)=="wire")temp.replace(0,4,"");
            inouts+=temp;
            inouts+=",";
            linout=false; //flag was output

        }
        else{           //if the first characters are neither input nor output

                //access output or input vector
                //copy last element of vector
                //delete variable name and overwrite new name
                //add to vector
            if(linout){ //if the last variable was input
                string temp=inputs.back();
//////////////////////////////////////////////////////
                if(temp.find(']')!=string::npos){
                    temp=temp.substr(0,temp.find(']')+1);
                }
                else{
                    temp=temp.substr(0,2);
                }
//////////////////////////////////////////////


                if(temp.find(']')!=-1) temp=temp.substr(0,temp.find(']')+1);
                else temp="reg ";
                inouts+=instruction;
                inouts+=",";
                temp+=" "+instruction; //concatenate variable name and its values
                inputs.push_back(temp);
            }
```

```
            else {      //if the last variable was output


                string temp=outputs.back();
                temp=temp.substr(0,temp.find(']')+1);
                ///////////////////////
                if(temp.find(']')!=string::npos){
                    temp=temp.substr(0,temp.find(']')+1);
                }
                else{
                    temp="wire ";
                }
              ///////////////////////
                inouts+=instruction;
                inouts+=",";
                temp+=" "+instruction; //concatenate variable name and its values
                outputs.push_back(temp);
            }

        }


    }
    inouts.replace(inouts.length()-1,inouts.length(),"");
    //obtain bus ranges inputs
    for(int i=0;i<inputs.size();i++){ //traverse the entire vector of entries
        if(inputs.at(i).find('[')!=string::npos){//if "[" character is found
            string st;
            int posi=inputs.at(i).find('[');
            int posf=inputs.at(i).find(']');
            st=inputs.at(i).substr(posi+1,posf-posi-1); //the characters in the "[ ]"
are extracted.

            bitrgint.push_back(st);//is added to range vector

            //get size in integer from cad bus
            int rango=0;//variable where the range will be stored
            int r1=stoi(st.substr(0,st.find(':')));
            int r2=stoi(st.substr(st.find(':')+1,st.back()));

            if(r1>r2){ //if the number on the left is greater than the number on the
right
                rango=r1-r2+1;
                bitsint.push_back(rango);

            }
            else{ //if the number on the left is less than the number on the right
                rango=r2-r1+1;
                bitsint.push_back(rango);
            }

        }
        else{                           //if no "[" character is found
```

```
            bitrgint.push_back("1"); //the range is assumed to be 1
            bitsint.push_back(1);   //the range is assumed to be 1
        }
    }

    //obtain bus ranges outputs
    for(int i=0;i<outputs.size();i++){ //traverse the entire output vector
        if(outputs.at(i).find('[')!=string::npos){
            string st;
            int posi=outputs.at(i).find('[');
            int posf=outputs.at(i).find(']');
            st=outputs.at(i).substr(posi+1,posf-posi-1);//the characters in the "[ ]"
are extracted.
            bitrgout.push_back(st);

            //
            int rango=0;//variable where the range will be stored
            int r1=stoi(st.substr(0,st.find(':')));
            int r2=stoi(st.substr(st.find(':')+1,st.back()));

            if(r1>r2){ //if the number on the left is greater than the number on the
right
                rango=r1-r2+1;
                bitsout.push_back(rango);

            }
            else{ //if the number on the left is less than the number on the right
                rango=r2-r1+1;
                bitsout.push_back(rango);
            }
            //
        }
        else{
            bitrgout.push_back("1");
            bitsout.push_back(1);
        }
    }
    //get string of variables

////////////////////////////////////////////////////////////////////////////////////////
bruno
// The possible solutions are created,
// in case there are more than 64 possible combinations, the first 10,
// the middle 10 and the last 10 are given.
    int combinaciones=0;
    string input_comb;
    vector<string> arr= inputs;
     for (int i=0; i<arr.size(); i++) {
        arr[i].erase(0,arr[i].find("reg")+3);
        if(arr[i].find(']')!=-1)arr[i].erase(0,arr[i].find(']')+1);
        input_comb.append(arr[i]);
        input_comb.append(",");
    }
```

```
    input_comb.erase(input_comb.length()-1,input_comb.length());

    for (int i=0; i<bitsint.size(); i++) {
        combinaciones+=bitsint[i];
    }
    combinaciones=pow (2, combinaciones);
    if (combinaciones>64)combinaciones=64;
////////////////////////////////////////////////////////////////////////////////////////
// The necessary elements are created to perform the Testbench,
// using all the variables and information above as a vector.
  ofstream myfile;
  myfile.open ("testBench.txt");
  myfile << "`timescale 1ns/100ps\n";

  myfile <<"module automatic_TB ();\n";
  myfile << "/*        variables          */\n";

  for(int i=0;i<inputs.size();i++){
    myfile << inputs[i]<<";\n";
  }
    for(int i=0;i<outputs.size();i++){
    myfile <<  outputs[i]  <<";\n";
  }
  myfile << "\n";
  myfile << module_name <<"\n";  //module name
  myfile << "DUT("<< inouts <<");\n";  /// module variables
  myfile << "initial begin\n";
  myfile << "$dumpvars; $dumpfile(\"dump.vcd\");\n";
  myfile << "\n$monitor(\"";
  for (int i=0;i<inputs.size();i++) myfile << "%b ";
  for (int i=0;i<outputs.size();i++) myfile << "%b ";
  myfile << "\",";
  myfile << inouts;
  myfile << ");\n";
  myfile << "\nfor(int i=0;i<"<<combinaciones<<";i++)begin";
  myfile << "\n     {"<<input_comb<<"}=i;";
  myfile << "\n      #1;";
  myfile << "\nend";
  myfile << "\n";
  myfile << "\n";
  myfile << "\n";
  myfile << "$finish();\n";
  myfile << "end\n";
  myfile << "endmodule\n";
  myfile << "/*  ---------------module end------------------       */\n";
  myfile << "\n";


  myfile.close();
  return 0;
}
```

Captures and changes in code execution

Code in C++ with Regular Expressions

```
// basic file operations
#include <iostream>
#include <fstream>

#include <string>
#include <vector>
#include <math.h>
#include <regex>
using namespace std;


string module_name;
string Prueba;
////////////////////////////////////////////////////////////////////////////////////////
josue
string lectura() {
    ifstream archivo;
    string texto;                                        //The string Text is created
    archivo.open("prueba_mux_design.txt", ios::in);        //The file is opened in read
mode
                                            //The global variable Prueba is created
    string Apend;
    if (archivo.fail()) {                                //If the file cannot be opened
        cout << "No se pudo abrir el archivo";           //The message is sent
        exit(1);                                     //Se sale del programa
    }
    while (!archivo.eof()) {                            //While it's not the end of the
file
        getline(archivo, texto);                   //get the text
        if (texto.find("//")!=string::npos){        //If is used to remove comments
within the text
            int t1 = texto.length();                    //It seeks to find the size of
each line of text, to remove the comments in the necessary places.
            int ecom = texto.find("//");             //
            texto.erase(ecom,t1-ecom);
        }
        if (texto.find('`')!=string::npos){        //If is used to remove libraries
within the text
            int t1 = texto.length();                    //It seeks to find the size of
each line of text, to remove the libraries in the necessary places.
            int ecom = texto.find('`');             //
            texto.erase(ecom,t1-ecom);
        }
        if (texto.find('#')!=string::npos){        //If is used to remove parameters
within the text
            int t1 = texto.length();                    //It seeks to find the size of
each line of text, to remove the parameters in the necessary places.
            int ecom = texto.find('#');             //
```

```
            texto.erase(ecom,t1-ecom);
        }

        Prueba += texto;                                //The text is concatenated
    }
    archivo.close();                                    //The file is closed


    smatch m;

    string temp=Prueba;
    regex e ("module\\s+(\\w+)\\s*+\\(([^)]+)");


    while (regex_search (temp,m,e)) {
        cout<<m[1];
        module_name.append(m[1]);
        cout << endl;
        temp = m.suffix().str();
    }



    int a = Prueba.find("(");                         //brackets are found

    int b = a + 1;
    int c = Prueba.find(")", a);


                                        //Assigns the usable text value to a string
    for (int i = a; i <= c; i++) {
        Apend.push_back(Prueba[i]);
    }

    //To remove the wire and reg text, the following is done
    while(Apend.find(" wire ")!=string::npos){
        int e = Apend.find(" wire ");
        Apend.erase(e,6);
    }

    while(Apend.find(" wire[")!=string::npos){
        int e = Apend.find(" wire[");
        Apend.erase(e,5);
    }

    while(Apend.find(" reg ")!=string::npos){
        int f = Apend.find(" reg ");
        Apend.erase(f,5);
    }

    while(Apend.find(" reg[")!=string::npos){
        int f = Apend.find(" reg[");
        Apend.erase(f,4);
```

```
    }

  return Apend;


}
////////////////////////////////////////////////////////////////////////////////////////
jose
string module=lectura();

vector<string> inputs;  //inputs vector
vector<string> outputs; //outputs vector

vector<int> bitsint;       //vector with the number of bits per input
vector<int> bitsout;        //vector with the number of bits per output

vector<string> bitrgint;       //vector with range of input buses
vector<string> bitrgout;       //vector with range of output buses
string inouts="";
int main()
{

    string t;
    smatch m;

    string temp=Prueba;
    regex e ("module\\s+(\\w+)\\s*+\\(([^)]+)");
    while (regex_search (temp,m,e)) {
        cout<<m[2];
        t.append(m[2]);
        cout << endl;
        temp = m.suffix().str();
    }





    bool linout=false;    //true if the last variable was intput or false if the last
variable was output

    while(t.length()>0){ //extract all instructions from the string
        string instruction="";//we create string to store the instruction

        if(t.find(',')!=string::npos){
            instruction=t.substr(0,t.find(','));//It is taken as an instruction from the
first character to the comma

            t.erase(0,t.find(',')+1);//the instruction including the comma is removed
        }
        else{
```

```
            instruction=t; //everything that remains in the string is taken as an
instruction
            t.erase(0,t.back());//the string is removed
        }

        if(regex_match ( instruction , regex("(input)(.*)") )){
            instruction.replace(0,instruction.find("input")+5,"reg ");//word "input" is
replaced by "reg"
            inputs.push_back(instruction);//adds to vector
            string temp=instruction;
            temp.replace(0,(temp.find(']')+1),"");
            if (temp.substr(0,3)=="reg")temp.replace(0,3,"");
            inouts+=temp;
            inouts+=",";
            linout=true; //flag was input
        }
        else if(regex_match ( instruction , regex("(output)(.*)") )){
            instruction.replace(0,instruction.find("output")+6,"wire ");//word "output"
is replaced by "wire"
            outputs.push_back(instruction);//adds to vector
            string temp=instruction;
            temp.replace(0,(temp.find(']')+1),"");
            if (temp.substr(0,4)=="wire")temp.replace(0,4,"");
            inouts+=temp;
            inouts+=",";
            linout=false; //flag was output

        }
        else{           //if the first characters are neither input nor output

                //access vector output or input
                 // copy last element of vector
                 //delete variable name and overwrite new name
                 //add to vector
            if(linout){ //if the last variable was input
                string temp=inputs.back();
/////////////////////////////////////////////////////
                if(temp.find(']')!=string::npos){
                    temp=temp.substr(0,temp.find(']')+1);
                }
                else{
                    temp=temp.substr(0,2);
                }
/////////////////////////////////////////////


                if(temp.find(']')!=-1) temp=temp.substr(0,temp.find(']')+1);
                else temp="reg ";
                inouts+=instruction;
                inouts+=",";
                temp+=" "+instruction; //concatenate variable name and its values
                inputs.push_back(temp);
            }
```

```
            else {      //if the last variable was output


                string temp=outputs.back();
                temp=temp.substr(0,temp.find(']')+1);
                ///////////////////////
                if(temp.find(']')!=string::npos){
                    temp=temp.substr(0,temp.find(']')+1);
                }
                else{
                    temp="wire ";
                }
              ///////////////////////
                inouts+=instruction;
                inouts+=",";
                temp+=" "+instruction; //concatenate variable name and its values
                outputs.push_back(temp);
            }

        }


    }
    inouts.replace(inouts.length()-1,inouts.length(),"");
    //get input bus ranges
    for(int i=0;i<inputs.size();i++){ //traverse the entire vector of inputs
        if(inputs.at(i).find('[')!=string::npos){//if character "[" is found
            string st;
            int posi=inputs.at(i).find('[');
            int posf=inputs.at(i).find(']');
            st=inputs.at(i).substr(posi+1,posf-posi-1); //the characters between the "[
]" are extracted

            bitrgint.push_back(st);//adds to vector of ranges

            //get integer size of each bus
            int rango=0;//variable where the range will be stored
            int r1=stoi(st.substr(0,st.find(':')));
            int r2=stoi(st.substr(st.find(':')+1,st.back()));

            if(r1>r2){ //If the number on the left is greater than the number on the
right
                rango=r1-r2+1;
                bitsint.push_back(rango);

            }
            else{ //if the number on the left is less than the number on the right
                rango=r2-r1+1;
                bitsint.push_back(rango);
            }

        }
        else{                           //if no character "[" is found
```

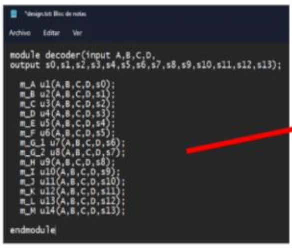
```
            bitrgint.push_back("1"); //the rank is assumed to be 1
            bitsint.push_back(1);   //the rank is assumed to be 1
        }
    }

    //get output bus ranges
    for(int i=0;i<outputs.size();i++){ //traverse the entire vector of outputs
        if(outputs.at(i).find('[')!=string::npos){
            string st;
            int posi=outputs.at(i).find('[');
            int posf=outputs.at(i).find(']');
            st=outputs.at(i).substr(posi+1,posf-posi-1);//the characters between the "[
]" are extracted
            bitrgout.push_back(st);

            //
            int rango=0;//variable where the range will be stored
            int r1=stoi(st.substr(0,st.find(':')));
            int r2=stoi(st.substr(st.find(':')+1,st.back()));

            if(r1>r2){ //If the number on the left is greater than the number on the
right
                rango=r1-r2+1;
                bitsout.push_back(rango);

            }
            else{ //if the number on the left is less than the number on the right
                rango=r2-r1+1;
                bitsout.push_back(rango);
            }
            //
        }
        else{
            bitrgout.push_back("1");
            bitsout.push_back(1);
        }
    }
    //get string of variables

////////////////////////////////////////////////////////////////////////////////////////
bruno
    int combinaciones=0;
    string input_comb;
    vector<string> arr= inputs;
     for (int i=0; i<arr.size(); i++) {
        arr[i].erase(0,arr[i].find("reg")+3);
        if(arr[i].find(']')!=-1)arr[i].erase(0,arr[i].find(']')+1);
        input_comb.append(arr[i]);
        input_comb.append(",");
    }
    input_comb.erase(input_comb.length()-1,input_comb.length());

    for (int i=0; i<bitsint.size(); i++) {
```

```
        combinaciones+=bitsint[i];
    }
    combinaciones=pow (2, combinaciones);
    if (combinaciones>64)combinaciones=64;
////////////////////////////////////////////////////////////////////////////////////////////////

  ofstream myfile;
  myfile.open ("testBench.txt");
  myfile << "`timescale 1ns/100ps\n";

  myfile <<"module automatic_TB ();\n";
  myfile << "/*        variables          */\n";

  for(int i=0;i<inputs.size();i++){
    myfile << inputs[i]<<";\n";
  }
    for(int i=0;i<outputs.size();i++){
    myfile <<  outputs[i]  <<";\n";
  }
  myfile << "\n";
  myfile << module_name <<"\n";  //module name
  myfile << "DUT("<< inouts <<");\n";  /// module variables
  myfile << "initial begin\n";
  myfile << "$dumpvars; $dumpfile(\"dump.vcd\");\n";
  myfile << "\n$monitor(\"";
  for (int i=0;i<inputs.size();i++) myfile << "%b ";
  for (int i=0;i<outputs.size();i++) myfile << "%b ";
  myfile << "\",";
  myfile << inouts;
  myfile << ");\n";
  myfile << "\nfor(int i=0;i<"<<combinaciones<<";i++)begin";
  myfile << "\n     {"<<input_comb<<"}=i;";
  myfile << "\n      #1;";
  myfile << "\nend";
  myfile << "\n";
  myfile << "\n";
  myfile << "\n";
  myfile << "$finish();\n";
  myfile << "end\n";
  myfile << "endmodule\n";
  myfile << "/*  --------------module end-----------------       */\n";
  myfile << "\n";


  myfile.close();
  return 0;
}
```

Captures and changes in code execution

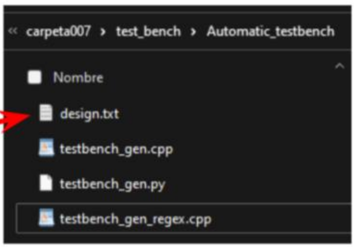
Results

To create a new Testbench file, it's necessary to add a Design.txt file with the top-level code in the program folder.



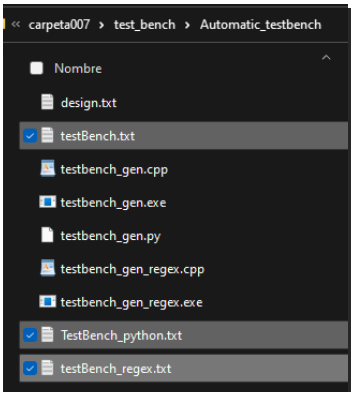
```
module decoder(input A,B,C,D,
output s0,s1,s2,s3,s4,s5,s6,s7,s8,s9,s10,s11,s12,s13);

  m_A u1(A,B,C,D,s0);
  m_B u2(A,B,C,D,s1);
  m_C u3(A,B,C,D,s2);
  m_D u4(A,B,C,D,s3);
  m_E u5(A,B,C,D,s4);
  m_F u6(A,B,C,D,s5);
  m_G_1 u7(A,B,C,D,s6);
  m_G_2 u8(A,B,C,D,s7);
  m_H u9(A,B,C,D,s8);
  m_I u10(A,B,C,D,s9);
  m_J u11(A,B,C,D,s10);
  m_K u12(A,B,C,D,s11);
  m_L u13(A,B,C,D,s12);
  m_M u14(A,B,C,D,s13);

endmodule
```

a) TopLevel example

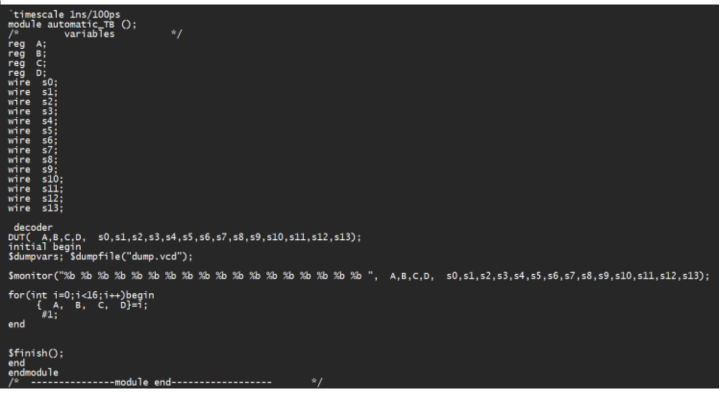


b) Programs folder

The execution of the .exe files will return an automatic generated TestBench file of the given design.txt located on the programs folder.



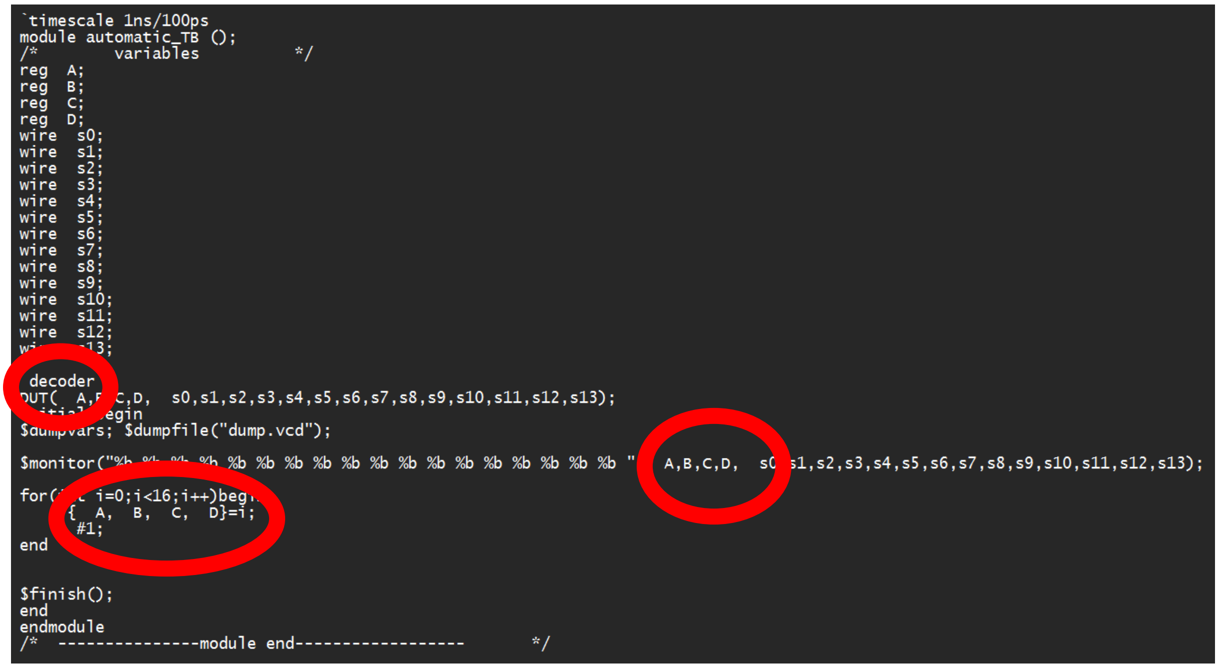
C) Program results



```
`timescale 1ns/100ps
module automatic_TB ();
/*         variables          */
reg  A;
reg  B;
reg  C;
reg  D;
wire  s0;
wire  s1;
wire  s2;
wire  s3;
wire  s4;
wire  s5;
wire  s6;
wire  s7;
wire  s8;
wire  s9;
wire  s10;
wire  s11;
wire  s12;
wire  s13;

 decoder
DUT(  A,B,C,D,  s0,s1,s2,s3,s4,s5,s6,s7,s8,s9,s10,s11,s12,s13);
initial begin
$dumpvars; $dumpfile("dump.vcd");

$monitor("%b %b %b %b %b %b %b %b %b %b %b %b %b %b %b %b %b %b ",  A,B,C,D,  s0,s1,s2,s3,s4,s5,s6,s7,s8,s9,s10,s11,s12,s13);

for(int i=0;i<16;i++)begin
     {  A,  B,  C,  D}=i;
      #1;
end


$finish();
end
endmodule
/*  ---------------module end------------------       */
```

D) test bench result example

The difference between resulting TestBenches is slightly different, it's possible to find some tabulations or blank spaces between variables or in some cases realize that one program doesn't make the testbench that the others do. The following images are the resulting testbench of the different programs.



```
`timescale 1ns/100ps
module automatic_TB ();
/*         variables          */
reg  A;
reg  B;
reg  C;
reg  D;
wire  s0;
wire  s1;
wire  s2;
wire  s3;
wire  s4;
wire  s5;
wire  s6;
wire  s7;
wire  s8;
wire  s9;
wire  s10;
wire  s11;
wire  s12;
wire  s13;

 decoder
DUT(  A,B,C,D,  s0,s1,s2,s3,s4,s5,s6,s7,s8,s9,s10,s11,s12,s13);
initial begin
$dumpvars; $dumpfile("dump.vcd");

$monitor("%b %b %b %b %b %b %b %b %b %b %b %b %b %b %b %b %b %b "  A,B,C,D,  s0,s1,s2,s3,s4,s5,s6,s7,s8,s9,s10,s11,s12,s13);

for(int i=0;i<16;i++)begin
     {  A,  B,  C,  D}=i;
      #1;
end


$finish();
end
endmodule
/*  ---------------module end------------------       */
```

e) Resulting TestBench of the C++ program : some blank spaces have been found.

```
`timescale 1ns/100ps
module mux_TB ();
reg A;
reg B;
reg C;
reg D;
wire s0;
wire s1;
wire s2;
wire s3;
wire s4;
wire s5;
wire s6;
wire s7;
wire s8;
wire s9;
wire s10;
wire s11;
wire s12;
wire s13;
decoder DUT(A,B,C,D,s0,s1,s2,s3,s4,s5,s6,s7,s8,s9,s10,s11,s12,s13);
initial begin
  $dumpvars;
  $dumpfile("dump.vcd");
A<=0;
B<=0;
C<=0;
D<=0;
$monitor("in: %b %b %b %b out: %b %b %b %b %b %b %b %b %b %b %b %b %b %b " ,A,B,C,D,s0,s1,s2,s3,s4,s5,s6,s7,s8,s9,s10,s11,s12,s13);
for (int i=0;i<16;i=i+1) begin
   {A,B,C,D}=i;
   #10;
end
  $finish;
end
endmodule
```

f) Resulting TestBench of the python program: Perfect format

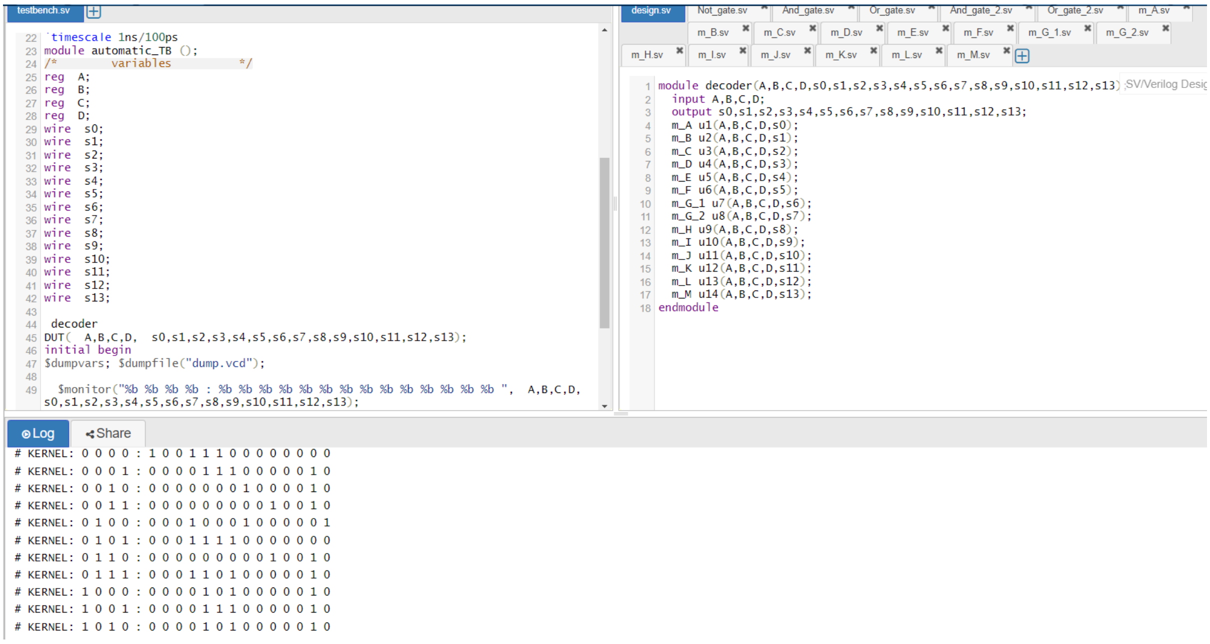
```
`timescale 1ns/100ps
module automatic_TB ();
/*        variables          */
reg  A;
reg  B;
reg  C;
reg  D;
wire  s0;
wire  s1;
wire  s2;
wire  s3;
wire  s4;
wire  s5;
wire  s6;
wire  s7;
wire  s8;
wire  s9;
wire  s10;
wire  s11;
wire  s12;
wire  s13;

decoder
DUT(  A,B,C,D,  s0,s1,s2,s3,s4,s5,s6,s7,s8,s9,s10,s11,s12,s13);
initial begin
$dumpvars; $dumpfile("dump.vcd");

$monitor("%b %b %b %b %b %b %b %b %b %b %b %b %b %b %b %b %b %b %b ",  A,B,C,D,  s0  1,s2,s3,s4,s5,s6,s7,s8,s9,s10,s11,s12,s13);

for(i  i=0;i<16;i++)beg
       A,  B,  C,  D}=i;
      #1;
end


$finish();
end
endmodule
/*  ---------------module end------------------       */
```

g) Resulting TestBench of the C++ (regex) program: some blank spaces have been found.

EDA Results

Since the testbench file is generated, we can use it on a verilog project like the given example which is only a program that spells the name “Christian” on a 14-segment format. if the top-level design file works with other modules on different files it is necessary to manually include those files at the beginning of the TB file.

advise: the testbench is going to make all the possible combinations of the given inputs by default, so if you don't need to make all the possible combinations just set manually the number of combinations on the for loop.



```
`timescale 1ns/100ps
module automatic_TB ();
/*          variables          */
reg  A;
reg  B;
reg  C;
reg  D;
wire  s0;
wire  s1;
wire  s2;
wire  s3;
wire  s4;
wire  s5;
wire  s6;
wire  s7;
wire  s8;
wire  s9;
wire  s10;
wire  s11;
wire  s12;
wire  s13;

 decoder
DUT(  A,B,C,D,  s0,s1,s2,s3,s4,s5,s6,s7,s8,s9,s10,s11,s12,s13);
initial begin
$dumpvars; $dumpfile("dump.vcd");

  $monitor("%b %b %b %b : %b %b %b %b %b %b %b %b %b %b %b %b %b %b ",  A,B,C,D, s0,s1,s2,s3,s4,s5,s6,s7,s8,s9,s10,s11,s12,s13);
```

```
module decoder(A,B,C,D,s0,s1,s2,s3,s4,s5,s6,s7,s8,s9,s10,s11,s12,s13);
  input A,B,C,D;
  output s0,s1,s2,s3,s4,s5,s6,s7,s8,s9,s10,s11,s12,s13;
  m_A u1(A,B,C,D,s0);
  m_B u2(A,B,C,D,s1);
  m_C u3(A,B,C,D,s2);
  m_D u4(A,B,C,D,s3);
  m_E u5(A,B,C,D,s4);
  m_F u6(A,B,C,D,s5);
  m_G_1 u7(A,B,C,D,s6);
  m_G_2 u8(A,B,C,D,s7);
  m_H u9(A,B,C,D,s8);
  m_I u10(A,B,C,D,s9);
  m_J u11(A,B,C,D,s10);
  m_K u12(A,B,C,D,s11);
  m_L u13(A,B,C,D,s12);
  m_M u14(A,B,C,D,s13);
endmodule
```

```
# KERNEL: 0 0 0 0 : 1 0 0 1 1 1 0 0 0 0 0 0 0 0
# KERNEL: 0 0 0 1 : 0 0 0 0 1 1 1 0 0 0 0 0 1 0
# KERNEL: 0 0 1 0 : 0 0 0 0 0 0 0 1 0 0 0 0 1 0
# KERNEL: 0 0 1 1 : 0 0 0 0 0 0 0 0 0 1 0 0 1 0
# KERNEL: 0 1 0 0 : 0 0 0 1 0 0 0 1 0 0 0 0 0 1
# KERNEL: 0 1 0 1 : 0 0 0 1 1 1 1 0 0 0 0 0 0 0
# KERNEL: 0 1 1 0 : 0 0 0 0 0 0 0 0 0 1 0 0 1 0
# KERNEL: 0 1 1 1 : 0 0 0 1 1 0 1 0 0 0 0 0 1 0
# KERNEL: 1 0 0 0 : 0 0 0 0 1 0 1 0 0 0 0 0 1 0
# KERNEL: 1 0 0 1 : 0 0 0 0 1 1 1 0 0 0 0 0 1 0
# KERNEL: 1 0 1 0 : 0 0 0 0 1 0 1 0 0 0 0 0 1 0
```

EDA_playground example:

Differences between codes

The most different program is the one that is written in a different language. Besides that, the structure and the behavior were almost the same on the tree variants. For example, on the first part of the program the first task is to make a container which has all the variable statements on it. In the python program we use a list container that has the module name and the variables on it, including the size of the buffer for each one. The c++ and c++(regex) uses a String to work with the content of the module statements.

Recommendations

It is possible to find issues in the generated file, they could happen if the format of the Design file were full of module instantiations, or the module variables were wrongly declared. Depending on which program we chose, the input format will have more or less flexibility, for example, the python script could fail if the declarations of the variables begin with a tabulation or a blank space, otherwise, the simple C++ script is going to successfully make the Testbench no matter how many blank spaces are. The C++ script with regular expressions will return the most polished format.

Conclusions

The main problem of the project realization is the beginning, on this beginning there's no preconception of how to do it properly, we must learn to the march and on this case, it was kind of difficult to planning and distribute tasks considering the personal abilities of each member since it was the first time working together as a team. One tool that helps to find all the things to do was a simple flux diagram who has the procedure from the beginning of the execution to the wanted result. The next step was to allow that everyone choose which step they want to perform. At the end, the results were successfully obtained considering the current abilities at the beginning of the project and the limit time to deliver.

Personal conclusion and contribution

Christian Ortega

I set up the output file code that the program will return after the execution, in addition I integrate the scripts that my team gives me and optimize some parts of the code that where easy to compress. I think that the way that we fallow to work and find problems was pretty optimal because every code elaboration doesn't take more time that the class extension and at the end, all of us learn about different programing structures and even learn about each other.

References

1. https://regex101.com/
2. https://en.cppreference.com/w/cpp/regex
3. https://sg.com.mx/content/view/545
4. https://pythex.org/
5. https://www.regextester.com/102475
6. https://docs.python.org/3/howto/regex.html
7. https://docs.python.org/3/library/re.html