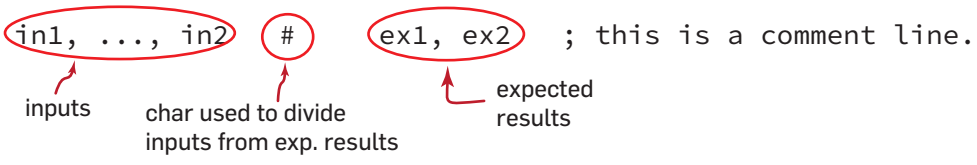
VER Assignment

Pietro Pennestrì - s2382660

The architecture **file_io3** is an upgraded architecture for the **tvc_siso_gen** entity. It is possible to recognize the following new features:

1. The architecture can handle input files with a more complex structure. In particular, expected results are part of the input file. The architecture can handle a varying number of inputs and expected results. An example of an input line is given in the following:



2. **file_io3** compares the expected result with data read from data_out ports. If these values do not match, the simulation is stopped with failure, and a warning message is given in the shell of ModelSim. Stopping the simulation when an error occurs will give the designer the possibility to debug the case where the bug was found.

3. **file_io** architecture stops the simulation before displaying results from the last input data. This bug was fixed in **file_io3** architecture.

The speed up the debug process, two python scripts were developed:

-)**genRandInFile**: generates random input with expected results.
The script outputs an input file compatible with the **file_io3** architecture.

-)**genInFile**: given a file with english keywords, it generates an input file for the **file_io3** architecture.

```
add max max ; check addition for max input
mul max max ; check multiplication for max input
add min min ; check addition for min input
mul min min ; check multiplication for min input
add max min ; check commutative property for addition
add min max ; check commutative property for addition
mul max min ; check commutative property for multiplication
mul min max ; check commutative property for multiplication
add rand 0  ; check neutral element of addition
mul rand 1  ; check neutral element of multiplication
mul -1 rand ; check if multiplication with -1 inverts sign of the random input
mul 0 rand  ; check multiplication with 0 and random number
mul 0 0     ; check multiplication if both input are 0s
add 0 0     ; check addition if both input are 0s
mul rand_pos rand_pos ; check multiplication for two random positive int.
mul rand_neg rand_neg ; check multiplication for two random negative int.
add rand_pos rand_pos ; check addition for two random positive int.
add rand_neg rand_neg ; check addition for two random negative int.
null rand rand ; check null opcode for random inputs
null max max ; check null for max input
```

add : addition opcode
mul: multiplication opcode
nul: null opcode
max: maximum input
min: minimum input
rand: random number between min and max value
rand_pos: random number between 1 and max
rand_neg: random number between min and -1

generated with the genInFile Script

```
1  127 127 # -2 ;  check addition for max input
2  127 127 # 63 1 ;  check multiplication for max input
1  -128 -128 # 0 ;  check addition for min input
2  -128 -128 # 64 0 ;  check multiplication for min input
1  127 -128 # -1 ;  check commutative property for addition
1  -128 127 # -1 ;  check commutative property for addition
2  127 -128 # -64 -128 ;  check commutative property for multiplication
2  -128 127 # -64 -128 ;  check commutative property for multiplication
1  -105 0 # -105 ;  check neutral element of addition
2  -62 1 # -1 -62 ;  check neutral element of multiplication
2  -1 -41 # 0 41 ;  check if multiplication with -1 inverts sign of the random input
2  0 26 # 0 0 ;  check multiplication with 0 and random number
2  0 0 # 0 0 ;  check multiplication if both input are 0s
1  0 0 # 0 ;  check addition if both input are 0s
2  17 -110 # -8 -78 ;  check multiplication for two random positive int.
2  49 -58 # -12 -26 ;  check multiplication for two random negative int.
1  -104 -34 # 118 ;  check addition for two random positive int.
1  -58 15 # -43 ;  check addition for two random negative int.
0  -21 63 # 0 ;  check null opcode for random inputs
0  127 127 # 0 ;  check null for max input
```

Simulation Results

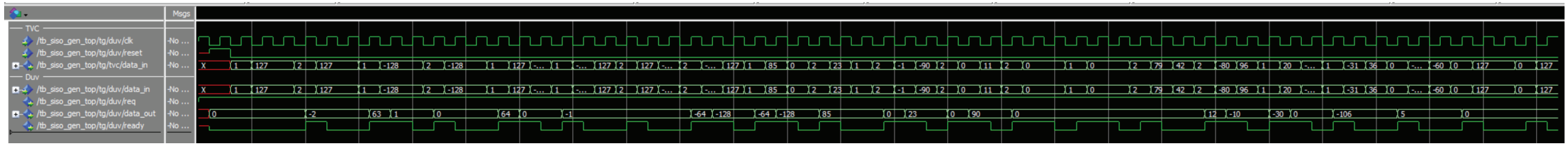


Figure 1: Simulation result for the **calc** architecture.

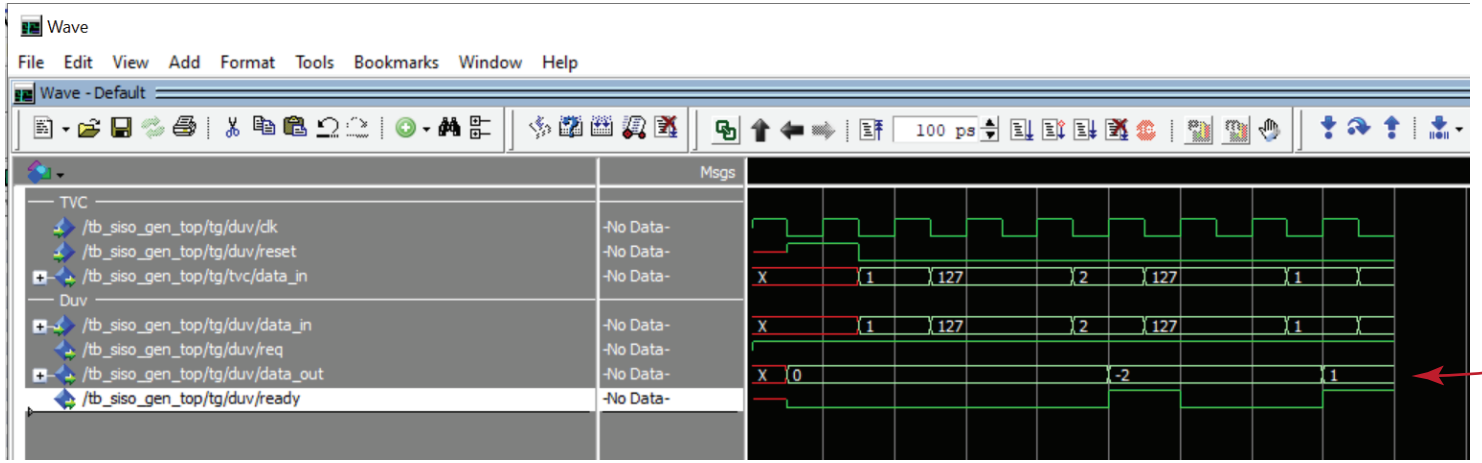
```
** Note: Result is as expected
   Time: 9600 ns  Iteration: 1  Instance: /tb_siso_gen_top/tg/tvc
** Note: Result is as expected
   Time: 9800 ns  Iteration: 1  Instance: /tb_siso_gen_top/tg/tvc
** Note: Result is as expected
   Time: 10200 ns  Iteration: 1  Instance: /tb_siso_gen_top/tg/tvc
** Note: Result is as expected
   Time: 10400 ns  Iteration: 1  Instance: /tb_siso_gen_top/tg/tvc
** Note: Result is as expected
   Time: 10800 ns  Iteration: 1  Instance: /tb_siso_gen_top/tg/tvc
** Note: Result is as expected
   Time: 11400 ns  Iteration: 1  Instance: /tb_siso_gen_top/tg/tvc
** Note: Result is as expected
   Time: 12 us  Iteration: 1  Instance: /tb_siso_gen_top/tg/tvc
** Note: Result is as expected
   Time: 12600 ns  Iteration: 1  Instance: /tb_siso_gen_top/tg/tvc
** Failure: Simulation completed successfully!
   Time: 12700 ns  Iteration: 1  Process: /tb_siso_gen_top/tg/tvc/ProcesswatchdogTime File: C:/Users/pietr/Documents/twente/courses/soc/assignments/h6/modelsim2/tvc_siso_gen_file_io3_arch.vhd
```

If simulation end succesfully a positive feedback is returned in the Modelsim log.

Figure 2: Modelsim log.

Figures 1 and 2 depict the simulation results for the **calc** architecture with the input file reported at page 1. No errors are found during the simulation The simulation is successfully terminated at the end of the input file.

The capability of error detection of **file_io3**, is tested by creating a new architecture **calc_bug**. This architecture behaves almost the same as **calc**, except that the result of the multiplication is displayed in reverse order.
Simulations results of calc_bug are depicted in Figure 3 and Figure 4.



The simulation is interrupted when the error is found

Figure 3: Simulation result for the **calc_bug** architecture.

An error message warns the user that the architecture is not behaving as expected.

```
VSIM 27> run -all
# ** Note: Result is as expected
#    Time: 1200 ns  Iteration: 1  Instance: /tb_siso_gen_top/tg/tvc
# ** Failure: Error data_out. Expected:63 got 1
#    Time: 1800 ns  Iteration: 1  Process: /tb_siso_gen_top/tg/tvc/readfile File: C:/Users/pietr/Documents/twente/courses/soc/assignments/h6/modelsim2/tvc_siso_gen_file_io3_arch.vhd
# Break in Process readfile at C:/Users/pietr/Documents/twente/courses/soc/assignments/h6/modelsim2/tvc_siso_gen_file_io3_arch.vhd line 231
```

Figure 4: Modelsim log.

Submitted Files

conf_tb3_siso_gen_calc_8: configuration file to test the **file_io3** architecture
conf_tb3_siso_gen_calc_8_bug: configuration file to test the **file_io3** architecture with the bugged **cal_bug** architecture.
genInFile: python script to generate random inputs.
genRandInFile: given a file with english keywords, it generates an input file for the **file_io3** architecture.
siso_gen_calc_bug_arch: bugged version **calc** architecture.
tvc_siso_gen_file_io3_arch: the new version of **file_io** architecture.

```
1
2  configuration conf_tb3_siso_gen_calc_bug_8_mix of tb_siso_gen_top is
3    for top
4      for tg: tb_siso_gen use entity work.tb_siso_gen(structure)
5              generic map (word_length => 8);
6        for structure
7          for duv: siso_gen use entity work.siso_gen(calc_bug);
8          end for;
9          for tvc: tvc_siso_gen use entity work.tvc_siso_gen(file_io3)
10             generic map (word_length => 8, -- code does not compile w/o this
11                          in_file_name =>  "mix.in",
12                          out_file_name => "mix.out");
13         end for;
14       end for;
15     end for;
16   end for;
17 end conf_tb3_siso_gen_calc_bug_8_mix;
18
```

```python
# -*- coding: utf-8 -*-
"""
Created on Sat Sep 26 18:08:32 2020

@author: pietro pennestri
"""
# This scripts generates given a file (infile) with inputs for siso_gen the
# expected output for the cal architecture. The script result is a
# text file (inputfileName)  which respects the syntax required by the
# architecture file_io3  of tvc_siso_gen.

import random

word_length = 8
opcodes = [0,1,2]

input2generate = 30

infile = "pre.in"
inputfileName =
"C:/Users/pietr/Documents/twente/courses/soc/assignments/h6/modelsim2/mix.in"

max_input = 2**(word_length -1) -1
min_input = -2**(word_length -1);


def toDecimal(bits):
    if bits!=None:
        if(bits[0]=='1'):
            return -2**(len(bits)-1)+int(bits[1:len(bits)],2)
        else:
            return int(bits, 2)
    else:
        return None

def twocomplement(val, bits):
    max_input = 2**(bits -1) -1
    min_input = -2**(bits -1);
    if(val>= min_input and val<= max_input):
        if val>=0 :
            return str(format(val, '#0'+str(bits+2)+'b')).replace("0b","")
        else:
            tempVal = 2**bits + val
            return str(format(tempVal, '#0'+str(bits+2)+'b')).replace("0b","")
    else:
        return None

# this function has the same beahaviour of calc architecture
def calc(opcode,input1,input2):
    if(opcode==0):
        return 0
    elif(opcode==1):
        return input1 + input2
    elif (opcode==2):
        return input1*input2
    else:
        return None

f0 = open(infile, "r")
content = f0.readlines()
```

```
f = open(inputfileName, "w")
for line in content:
    inputList=line.split(' ')
    inputList=' '.join(inputList).split()

    if("mul" in inputList[0]):
        opcode =2
    elif("null" in inputList[0]):
        opcode = 0
    elif("add" in inputList[0]):
        opcode = 1
    else:
        opcode =int(inputList[0])

    if("max" in inputList[1]):
        input1 =max_input
    elif("min" in inputList[1]):
        input1 =min_input
    elif("rand" in inputList[1]):
        input1=random.randint(min_input, max_input)
    elif("rand_neg" in inputList[1]):
        input1=random.randint(min_input, -1)
    elif("rand_pos" in inputList[2]):
        input1=random.randint(1, max_input)
    else:
        input1 =int(inputList[1])

    if("max" in inputList[2]):
        input2 =max_input
    elif("min" in inputList[2]):
        input2 =min_input
    elif("rand" in inputList[2]):
        input2=random.randint(min_input, max_input)
    elif("rand_neg" in inputList[2]):
        input2=random.randint(min_input, -1)
    elif("rand_pos" in inputList[2]):
        input2=random.randint(1, max_input)
    else:
        input2 =int(inputList[2])

    if(';' in line):
        comment=line.split(';')[1].replace('\n','')
    else:
        comment=""

    str2write=str(opcode)+ " "+ " "+ str(input1)+" "+str(input2)+" # "
    result = calc(opcode,input1,input2)
    if(opcode==0):
        str2write=str2write+str(0)
    elif(opcode==2):
        temp= twocomplement(result, 2*word_length)
        tempRes1 = temp[:word_length]
        tempRes2 = temp[word_length:]
        tempRes1= toDecimal(tempRes1)
        tempRes2= toDecimal(tempRes2)
        str2write=str2write+str(tempRes1)+" "+str(tempRes2)
    else:
        temp= twocomplement(result, word_length+1)
        temp= toDecimal(temp[1:len(temp)])
        str2write=str2write+str(temp)
```

```
120
121     f.write(str2write+" ; "+ comment+'\n')
122     print(str2write)
123     #print(opcode,", ",input1,", ", input2)
124
125 f.close()
126 f0.close()
```

```python
# -*- coding: utf-8 -*-
"""
Created on Sat Sep 26 18:08:32 2020

@author: pietr
"""
# This scripts generates random inputs with the respective expected output
# for the cal architecture of siso_gen. The input file respects the syntax
# required by the architecture file_io3 of tvc_siso_gen.

import random

word_length = 8
opcodes = [0,1,2]

input2generate = 30
inputfileName = "mix.in"

max_input = 2**(word_length -1) -1
min_input = -2**(word_length -1);

def toDecimal(bits):
    if bits!=None:
        if(bits[0]=='1'):
            return -2**(len(bits)-1)+int(bits[1:len(bits)],2)
        else:
            return int(bits, 2)
    else:
        return None

def twocomplement(val, bits):
    max_input = 2**(bits -1) -1
    min_input = -2**(bits -1);
    if(val>= min_input and val<= max_input):
        if val>=0 :
            return str(format(val, '#0'+str(bits+2)+'b')).replace("0b","")
        else:
            tempVal = 2**bits + val
            return str(format(tempVal, '#0'+str(bits+2)+'b')).replace("0b","")
    else:
        return None

# this function has the same beahaviour of calc architecture
def calc(opcode,input1,input2):
    if(opcode==0):
        return 0
    elif(opcode==1):
        return input1 + input2
    elif (opcode==2):
        return input1*input2
    else:
        return None

f = open(inputfileName, "w")
for i in range(input2generate):
    opcode = opcodes[random.randint(0, len(opcodes)-1)]
    input1=random.randint(min_input, max_input)
    input2=random.randint(min_input, max_input)
    str2write=str(opcode)+ " "+ " "+ str(input1)+" "+str(input2)+" # "
    result = calc(opcode,input1,input2)
```

```
    if(opcode==0):
        str2write=str2write+str(0)+" ;"
    elif(opcode==2):
        temp= twocomplement(result, 2*word_length)
        tempRes1 = temp[:word_length]
        tempRes2 = temp[word_length:]
        tempRes1= toDecimal(tempRes1)
        tempRes2= toDecimal(tempRes2)
        str2write=str2write+str(tempRes1)+" "+str(tempRes2)+" ;"
    else:
        temp= twocomplement(result, word_length+1)
        temp= toDecimal(temp[1:len(temp)])
        str2write=str2write+str(temp)+" ;"

    f.write(str2write+'\n')
    print(str2write)
    #print(opcode,", ",input1,", ", input2)

f.close()


```

```vhdl
 -- this architecture needs arithmetic functions
library ieee;
use ieee.numeric_std.all;

architecture calc_bug of siso_gen is
  -- state for ALU
  -- two "read opcode (opc)" states, the first does not raise "ready"
  type state is (read_opc_init, read_opc_ready1, read_opc_ready2,
                 read_left1,    read_left2,      read_right);
  signal cur_state: state;
  signal nxt_state: state;

  -- internal registers for left and right ALU input and ALU
  -- output, all having the same width
  signal left_in_reg:  signed(word_length-1 downto 0);
  signal right_in_reg: signed(word_length-1 downto 0);
  signal result_reg:   signed(2*word_length-1 downto 0);
  -- at most 16 different operations are supported by this ALU
  signal opcode_reg:   std_logic_vector(3 downto 0);

  -- and their next values
  signal left_in_nxt:  signed(word_length-1 downto 0);
  signal right_in_nxt: signed(word_length-1 downto 0);
  signal result_nxt:   signed(2*word_length-1 downto 0);
  signal opcode_nxt:   std_logic_vector(3 downto 0);

  -- output of adder
  signal adder_out: signed(word_length-1 downto 0);

  -- output of multiplier
  signal mult_out: signed(2*word_length-1 downto 0);

  -- next value for ready
  signal ready_nxt: std_logic;

begin
  -- the next process is sequential and only sensitive to clk and reset
  seq: process(clk, reset)
  begin
    if (reset = '1')
    then
      left_in_reg  <= (others => '0');
      right_in_reg <= (others => '0');
      opcode_reg   <= (others => '0');
      result_reg   <= (others => '0');
      ready        <= '0';
      cur_state    <= read_opc_init;
    elsif rising_edge(clk)
    then
      left_in_reg  <= left_in_nxt;
      right_in_reg <= right_in_nxt;
      opcode_reg   <= opcode_nxt;
      result_reg   <= result_nxt;
      ready        <= ready_nxt;
      cur_state    <= nxt_state;
    end if;
  end process seq;

  -- combinational next-value process
  nxt: process(data_in, cur_state, left_in_reg, right_in_reg,
```

```
61                opcode_reg, adder_out, mult_out)
62    begin
63      case cur_state is
64        when read_opc_init =>
65          nxt_state    <= read_left1;
66          left_in_nxt  <= left_in_reg;
67          right_in_nxt <= right_in_reg;
68          opcode_nxt   <= data_in(3 downto 0);
69          result_nxt   <= result_reg;
70          ready_nxt    <= '0';
71        when read_left1 =>
72          nxt_state    <= read_right;
73          left_in_nxt  <= signed(data_in);
74          right_in_nxt <= right_in_reg;
75          opcode_nxt   <= opcode_reg;
76          result_nxt   <= result_reg;
77          ready_nxt    <= '0';
78        -- read_left2 is identical to read_left1 except for ready_nxt
79        when read_left2 =>
80          nxt_state    <= read_right;
81          left_in_nxt  <= signed(data_in);
82          right_in_nxt <= right_in_reg;
83          opcode_nxt   <= opcode_reg;
84          result_nxt   <= result_reg;
85          ready_nxt    <= '1';
86        when read_right =>
87          -- multiplication needs two cycles to output result
88          if (opcode_reg = "0010")
89          then
90             nxt_state    <= read_opc_ready2;
91          else
92             nxt_state    <= read_opc_ready1;
93          end if;
94          left_in_nxt  <= left_in_reg;
95          right_in_nxt <= signed(data_in);
96          opcode_nxt   <= opcode_reg;
97          result_nxt   <= result_reg;
98          ready_nxt    <= '0';
99        when read_opc_ready1 | read_opc_ready2 =>
100         -- multiplication needs two cycles to output result
101         if (cur_state = read_opc_ready2)
102         then
103            nxt_state    <= read_left2;
104         else
105            nxt_state    <= read_left1;
106         end if;
107         left_in_nxt  <= left_in_reg;
108         right_in_nxt <= right_in_reg;
109         case opcode_reg is
110           when "0000" => -- the null result
111             result_nxt <= (others => '0');
112           when "0001" => -- addition
113             result_nxt(word_length - 1 downto 0)             <= adder_out;
114             result_nxt(2*word_length - 1 downto word_length) <= (others => '0'); --
BUG!!
115           when "0010" => -- multiplication
116             result_nxt <= mult_out(word_length - 1 downto 0) &
mult_out(2*word_length - 1 downto word_length);
117           when others => -- non-implemented codes behave like null command
118             result_nxt <= (others => '0');
```

```
119         end case;
120         opcode_nxt   <= data_in(3 downto 0);
121         ready_nxt    <= '1';
122       end case;
123   end process nxt;
124 
125   -- adder, wrap around in case of overflow, so discard carry
126   adder_out <= left_in_reg + right_in_reg;
127 
128   -- multiplier
129   mult_out <= left_in_reg * right_in_reg;
130 
131     data_out <= std_logic_vector(result_reg(2*word_length - 1 downto word_length))
132                 when cur_state = read_left2
133               else std_logic_vector(result_reg(word_length - 1 downto 0));
134 
135   -- this block should receive data in every clock cycle
136   req <= '1';
137 end calc_bug;
138 
```

```vhdl
architecture file_io3 of tvc_siso_gen is

    ---------------------------------- Define Functions ------------------------
-----
    function max(LEFT, RIGHT: INTEGER) return INTEGER is
        begin
        if LEFT > RIGHT then
            return LEFT;
        else
            return RIGHT;
        end if;
    end;

    function char2Digit(chr : character) return integer is
        begin
            case chr is
                when '0' =>     return 0;
                when '1' =>     return 1;
                when '2' =>     return 2;
                when '3' =>     return 3;
                when '4' =>     return 4;
                when '5' =>     return 5;
                when '6' =>     return 6;
                when '7' =>     return 7;
                when '8' =>     return 8;
                when '9' =>     return 9;
                when others =>
                    return -1;
                    assert false
                    report "Error in char2Digit function" severity failure;
            end case;
    end function;

    function isDigit(chr : character) return boolean is
        begin
            case chr is
                when '0' =>     return true;
                when '1' =>     return true;
                when '2' =>     return true;
                when '3' =>     return true;
                when '4' =>     return true;
                when '5' =>     return true;
                when '6' =>     return true;
                when '7' =>     return true;
                when '8' =>     return true;
                when '9' =>     return true;
                when others =>      return false;
            end case;
    end function;


    function string2integer(str : string) return integer is
        --variable str_len : integer := str'length;

        variable convertedNum : integer := 0;
        variable  isneg : boolean;
        variable k :integer:= -1;

        begin
            isneg := false;
```

```
60              for i in str'reverse_range loop
61                  if ( isDigit(str(i))) then
62                      k:=k+1;
63                      convertedNum := convertedNum + char2Digit(str(i))*10**k;
64                  else
65                      isneg := true;
66                  end if;
67              end loop ;
68
69              if (isneg) then
70                  convertedNum :=-1*convertedNum;
71              end if ;
72
73              return convertedNum;
74      end function;
75
76      ------------------------------ Define Signals & Constants ----------------------
--------
77
78      --internal clock and reset signals
79      signal clk_i, rst_i : std_logic;
80      signal line_read : integer := 0;
81      signal line_processed: integer := 0;
82      -- define input and output files
83      file in_file : text open Read_mode is in_file_name;
84      file out_file : text open Write_mode is out_file_name;
85
86      -- constants watch_dog and input_mem_size and expected_result_mem_size
87      -- can be added to generic.
88      constant watch_dog                : integer := 2000;  -- define the maximum
number of clock cycles
89      constant input_mem_size           : integer := 3;   -- define the maximum
number of inputs in line
90      constant expected_result_mem_size  : integer := 2;   -- define the maximum
number of expected result in line
91
92      constant max_input : integer := 2**(word_length -1) -1;
93      constant min_input : integer := -2**(word_length -1);
94      -- max_char is the expected maximum number of char in input text line and it is
given by the following formula:
95      --    max_char = (input_mem_size + expected_result_mem_size) * (
max(n_char_for_max_input + 1, n_char_for_min_input + 1) ) + 2
96      -- note that the +1 is due to the white space used to divide two numbers. The +2
at the of the formula is due to chars
97      -- "# " used to divide input from expected results.
98      constant max_char : integer := (input_mem_size + expected_result_mem_size) *
max(integer'image(max_input)'length +1 , integer'image(min_input)'length +1) +2;
99
100 begin
101   clk <= clk_i;
102   --reset <= rst_i;
103
104     Processclock : process
105     begin
106         clk_i <= '1';
107     wait for half_clock_period;
108     clk_i <= '0';
109     wait for half_clock_period;
110     end process ; -- clock
111
```

```
112     readfile : process(clk_i)
113 
114     type input_mem_type is array (input_mem_size-1 downto 0) of integer;
115     variable input_memory : input_mem_type; -- store inputs read from input line
116 
117     type expected_mem_type is array (expected_result_mem_size-1  downto 0) of
    integer;
118     type fifo_expected_mem_type is array (1  downto 0) of expected_mem_type;
119 
120     variable tmp_expected_memory : expected_mem_type ; -- store expected results
    read form input line
121     variable fifo_expected_memory : fifo_expected_mem_type;
122 
123     variable inline             : line;      -- used to store the line read from
    input_file
124     variable temp_char          : character; -- used to store temp char read from
    line
125     variable temp_string        : string(1 to max_char); -- used to store chars read
    from line
126     variable i                  : integer; -- iteration variable for reading char in
    line
127     variable j                  : integer := -1; -- iteration variable for checking
    result
128     variable n_inputs           : integer; -- stores the n. of inputs
129     variable n_expected_results : integer; -- stores the n. of expected results
130     variable inputs_sent        : integer := 0; -- store the n. of inputs sent to
    duv (for a single input line)
131     variable start_int          : integer; -- used to store position of the inial
    char of integer
132     variable end_int            : integer; -- used to store position of the last
    char of integer
133     variable search_int_start   : boolean;
134     variable digit_found        : boolean;
135     variable hashtag_found      : boolean;
136     variable isFirst            : boolean := true; -- check if first iteration
137     variable canRead            : boolean := true; -- check if can read
138     variable convInt            : integer;
139     variable line2write         : line; -- line to write
140     begin
141         if falling_edge(clk_i) then
142             if(not endfile(in_file) and canRead) then
143                 fifo_expected_memory(1) := fifo_expected_memory(0) ;
144                 line_read <= line_read +1;
145                 readline(in_file, inline);
146                 canRead := false;
147                 i:= 0;
148                 search_int_start := true;
149                 digit_found := false;
150                 hashtag_found := false;
151                 n_expected_results := 0;
152                 n_inputs := 0;
153                 inputs_sent := 0;
154                 for k in inline'range loop
155                     i:= i+1;
156                     read(inline, temp_char );
157                     temp_string(i to i) := "" & temp_char;
158 
159                     if ( (isDigit(temp_char) or temp_char='-' ) and
    search_int_start) then
160                         search_int_start := false;
```

```
161                         start_int:=i;
162                     end if ;
163 
164                     if( (isDigit(temp_char) and search_int_start) or ((not
    search_int_start) and isDigit(temp_char) ) )then
165                         digit_found :=true;
166                     end if ;
167 
168                     if(not search_int_start and not isDigit(temp_char) and
    i/=start_int and digit_found) then
169                         end_int := i-1;
170                         search_int_start := true;
171                         convInt:= string2integer(temp_string(start_int to end_int));
172 
173                         if (temp_char/='#' and hashtag_found) then
174                             n_expected_results:=  n_expected_results +1;
175                             tmp_expected_memory(n_expected_results-1) :=  convInt;
176                         else
177                             n_inputs := n_inputs +1;
178                             input_memory(n_inputs-1) :=  convInt;
179                         end if ;
180 
181                     end if;
182 
183                     if temp_char='#' then
184                         hashtag_found := true;
185                     end if;
186 
187                     if temp_char =';' then
188                         exit;
189                     end if ;
190 
191                 end loop;
192                 fifo_expected_memory(0) := tmp_expected_memory;
193 
194             elsif(canRead) then
195                 inputs_sent:=0;
196                 canRead := false;
197                 fifo_expected_memory(1) := fifo_expected_memory(0) ;
198             end if;
199 
200             if (req='1' and (not isFirst)) then
201                 data_in <=   std_logic_vector(to_signed(input_memory(inputs_sent) ,
    word_length));
202                 inputs_sent:=inputs_sent+1;
203             end if ;
204 
205             if(inputs_sent = n_inputs) then
206                 canRead := true;
207             end if;
208 
209             if (isFirst) then
210                 isFirst := false;
211                 rst_i  <= '1';
212                 reset <= '1';
213             else
214                 reset <= '0';
215             end if ;
216 
217         end if ; -- end falling_edge
```

```
218
219
220         if(rising_edge(clk_i)) then
221             if(ready='1') then
222                 j:= j+1;
223
224                 if (fifo_expected_memory(1)(j)= to_integer(signed(data_out)) ) then
225                     report "Result is as expected" severity note;
226                     write(line2write, to_integer(signed(data_out)));
227                     writeline(out_file, line2write);
228                 else
229                     report "Error data_out. Expected:" &
integer'image(fifo_expected_memory(1)(j)) & " got " &
integer'image(to_integer(signed(data_out))) severity failure;
230                 end if ;
231
232             else
233                 j:= -1;
234             end if;
235         end if;
236     end process ; -- readfile
237
238
239
240     ProcessLine : process(ready)
241     begin
242         if falling_edge(ready) then
243             line_processed <= line_processed+1;
244         end if ;
245     end process ; -- ProcessLine
246
247
248     ProcesswatchdogTime : process(clk_i)
249     variable timer : integer := 0;
250     begin
251         -- fix me
252         if falling_edge(clk_i) then
253             if(endfile(in_file) and line_processed=line_read) then
254                 assert false
255                 report "Simulation completed successfully!" severity failure;
256             end if;
257
258             if timer=watch_dog then
259                 assert false
260                 report "Error: Watch Dog Timer!" severity failure;
261             end if;
262
263             if ready='1' then
264                 -- the watch dog timer is reset every
265                 -- time the reset signal is raised high
266                 timer := 0;
267             else
268                 timer := timer +1;
269             end if ;
270         end if ;
271     end process ; -- watchdogTime
272
273 end file_io3;
```

```
1
2  configuration conf_tb3_siso_gen_calc_8_mix of tb_siso_gen_top is
3    for top
4      for tg: tb_siso_gen use entity work.tb_siso_gen(structure)
5              generic map (word_length => 8);
6        for structure
7          for duv: siso_gen use entity work.siso_gen(calc);
8          end for;
9          for tvc: tvc_siso_gen use entity work.tvc_siso_gen(file_io3)
10             generic map (word_length => 8, -- code does not compile w/o this
11                          in_file_name =>  "mix.in",
12                          out_file_name => "mix.out");
13         end for;
14       end for;
15     end for;
16   end for;
17 end conf_tb3_siso_gen_calc_8_mix;
18
```