ELCT 501: Digital System Design

Course Project Report

Submission Date: 14/01/2021

		Member Name	Member ID	Tutorial #	Team Number
Compression	**1**	Mohammed Ahmed Shaban	46-10273	T-60	L2
	2	sherif Khalid	46-10638	t-60	L2
	3	Omar Nasser	46-10512	T-55	L1
	4	Mohammed Osama Nafea	46-2168	T-60	L2
	5	Youssef Salama	46-2334	T-60	L2
Decompression	**1**	Hadi ElNemr	46-6804	T-62	L2
	2	Ahmed Essam	46-5804	T-62	L2
	3	Mohammed Ashraf	46-13903	T-62	L2
	4	Moaz Elsayed	46-11674	T-62	L2
	5	Mohamed Khaled	46-2388	T-60	L2

TEAM NUMBER:13

1.Overview

- **Project Description and features:**

A satellite and a ground station can communicate only few times a day. The data collected by the satellite should be sent and received by the ground station in order to be analyzed and studied. However, due to the huge size of the data collected by the satellite, it is difficult for the ground station to receive as much data as possible from the satellite in a small period. The idea of data compression and decompression is used and implemented through chips. The data compression chip will be implemented in the satellite to compress the data received from the telemetry subsystem in the satellite to reduce the communication time with the ground station. The data decompression chip will be implemented in the ground station in order to decompress the data received in the ground station, then it will be analyzed in the computer. **The proposed data compression in this work is the LZ78 data compression algorithm.**

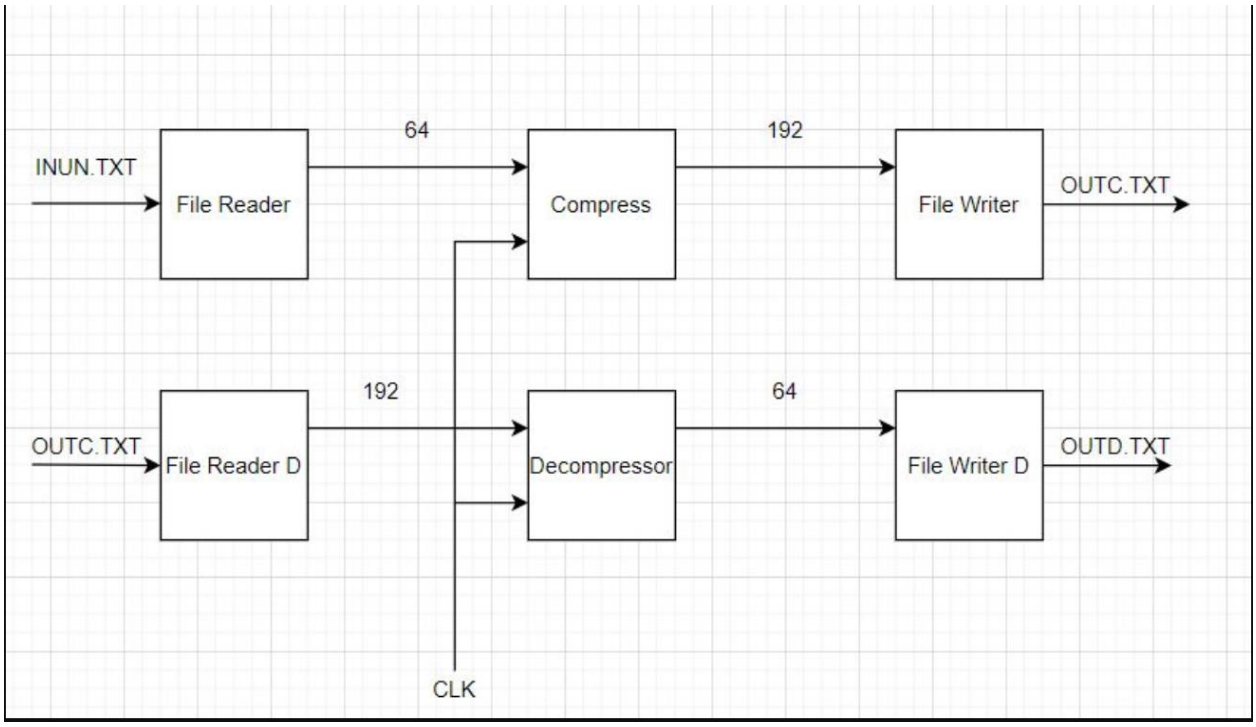
- **How to run the project:**

The compressor and the decompressor are implemented using the hardware language VHDL and wrote and run by the IDE (Xilinx). The data received from the telemetry subsystem is read by the compression chip via a file reader (.txt type) and outputs an array of strings. The ground station receives the data sent from the satellite and is read by a file reader in the decompression chip present in the ground station The decompressor takes the array of strings as input and decompresses it using the order of array which was in the input stream and outputs the decompressed array. Then, the decompressed array is written as lines in a new file and is presented in order to be analyzed.

***** When using simulation run 20000000 ns *****

1. Design Methodology

RTL DIAGRAM:



Resources:

- Converting a string into an integer: https://www.edaboard.com/threads/vhdl-convert-string-to-integer.280036/
- Text I/O: https://vhdlguide.com/2017/09/22/textio/
- Syntax: VHDL handbook by: HARDI Electronics

2. Implementation

- ***Module description:***

The project was divided into 6 modules and they are defined below

i) FileReader: The file reader of the compressor reads a file of type (.txt). This works as the File reader code written in VHDL reads each line in the input file and stores the data in a String. The string is made from 64 characters where each character is represented by 1 Byte which results in a total of 64 Bytes.

ii) Compressor: The Compressor loops on the string we get from our file reader. The Loop checks each character which has a size of one Byte. If the character is not present in our Dictionary which is based an array of strings. The array length is 64 and String has a length of 11. The 64 represents each time it checks a character it adds it to the dictionary. However, before adding it to the dictionary it checks if its present before as we mentioned earlier. If it's not present; it stores our character or characters in a new string stored in our Dictionary. Moreover, the string has a maximum size of 11 as we found through applying different examples and calculating the size difference each time. We found that if we an input which made from the same character and a size of 64 byte; The string will not exceed 11 characters

iii) FileWriter: The File Writer code takes the output from our Compressor where it will store it in a string which will be later on used as the input of the Decompressor's file reader.

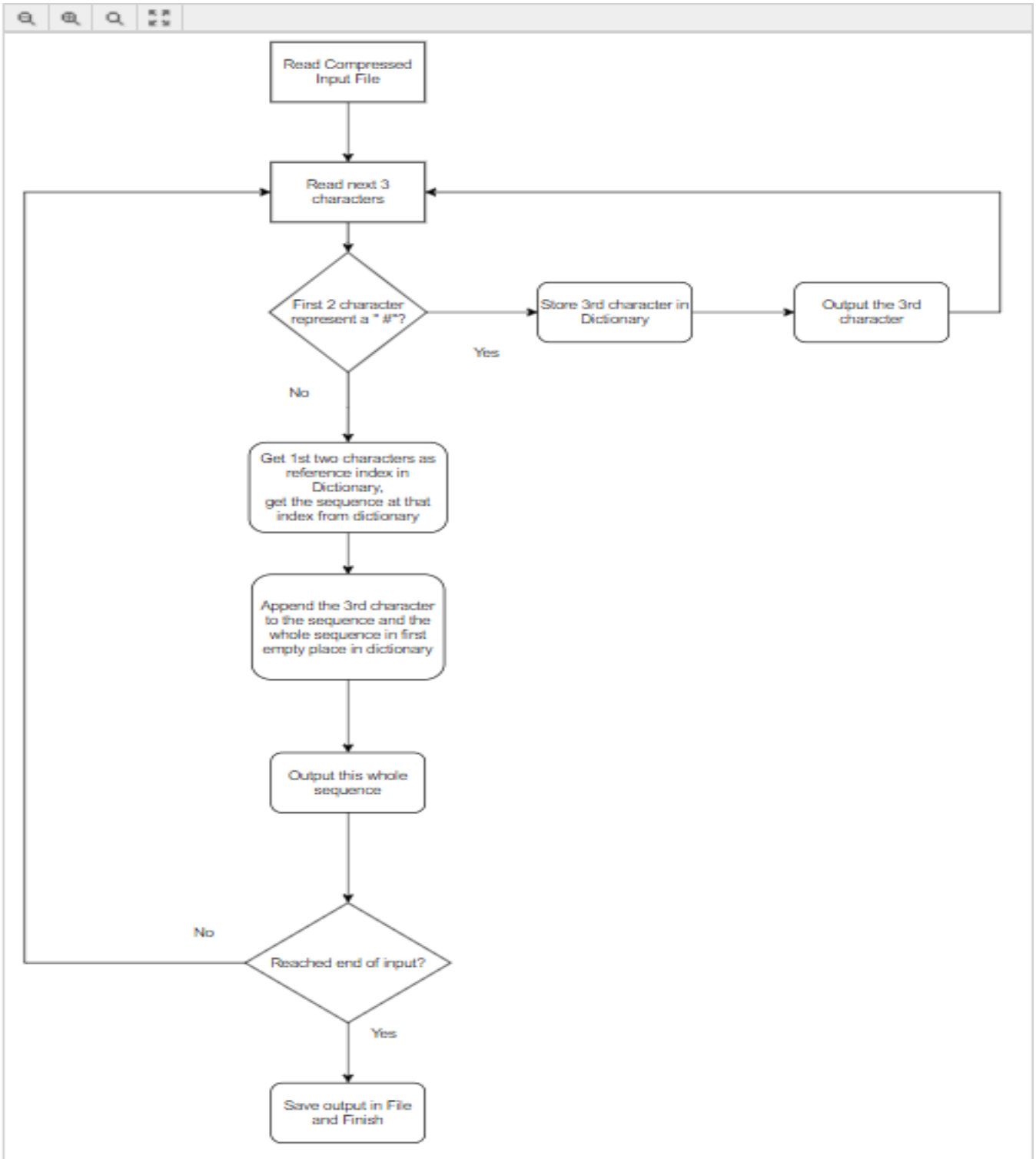
iv) FilerReaderD (decompression): It is the file reader of the decompression process that takes in the compressed file generated by the file writer of the compressor by reading each line in the file and gives out an array of strings where each string contains 3 consecutive characters from the word input where the first two characters represent the reference of the string in the dictionary and the third character which represents the end of the string.

v) Decompressor: This is the module that carries out the decompression process. It takes in the compressed input which is read by the FileReaderW and gives an output which is the decompressed version of the initial input given and sends it to the FileWriterW by translating the first two characters that resemble the reference of the string in the dictionary that was made by the sequence of the input stream given.

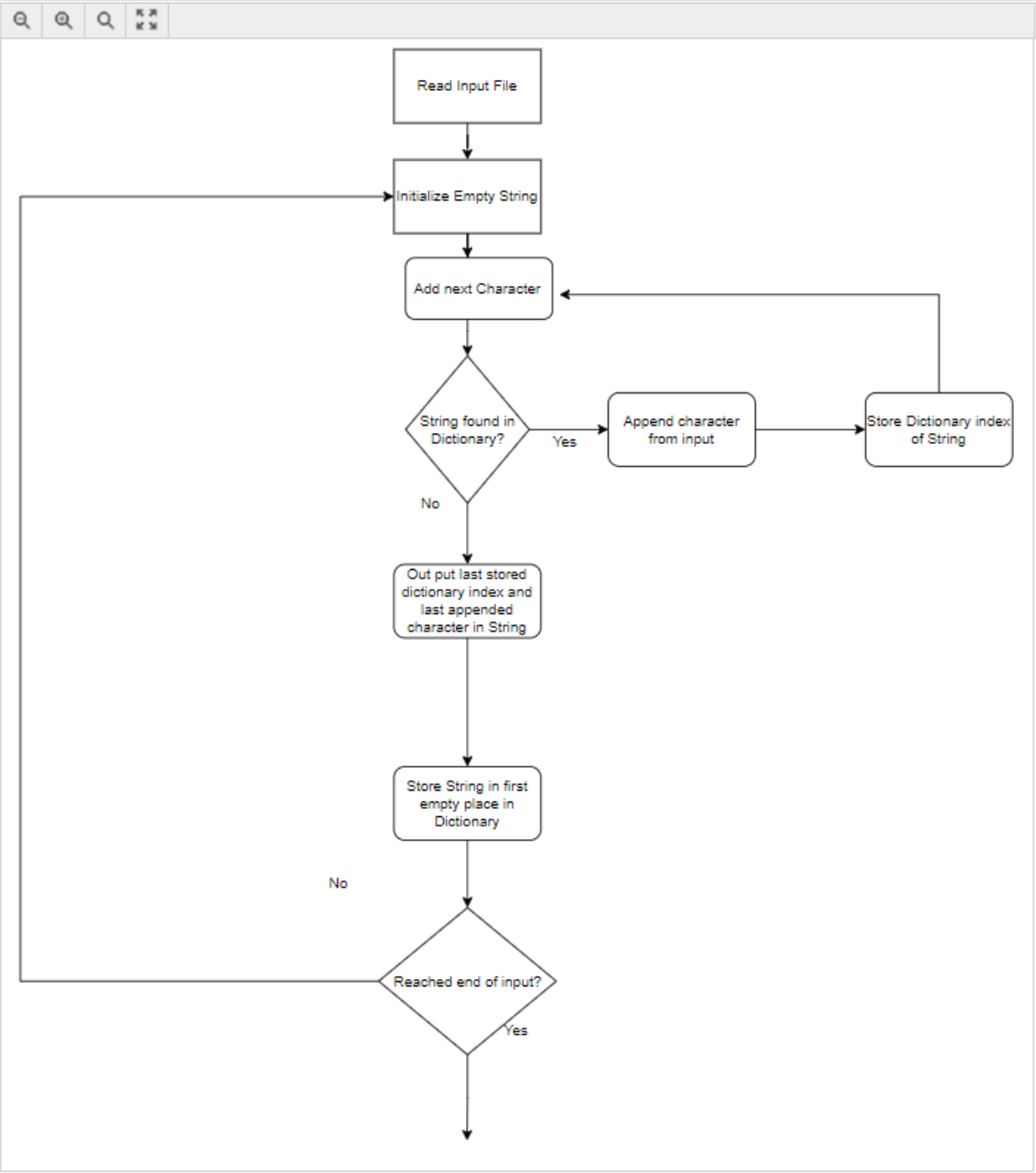
vi) FileWriterD (decompression): It is the file writer that takes the output of the decompressor as the input and generates lines that are written into the output file as a string.

- ***Circuit Diagram:***

Decompression flow chart:



Compression flow chart:



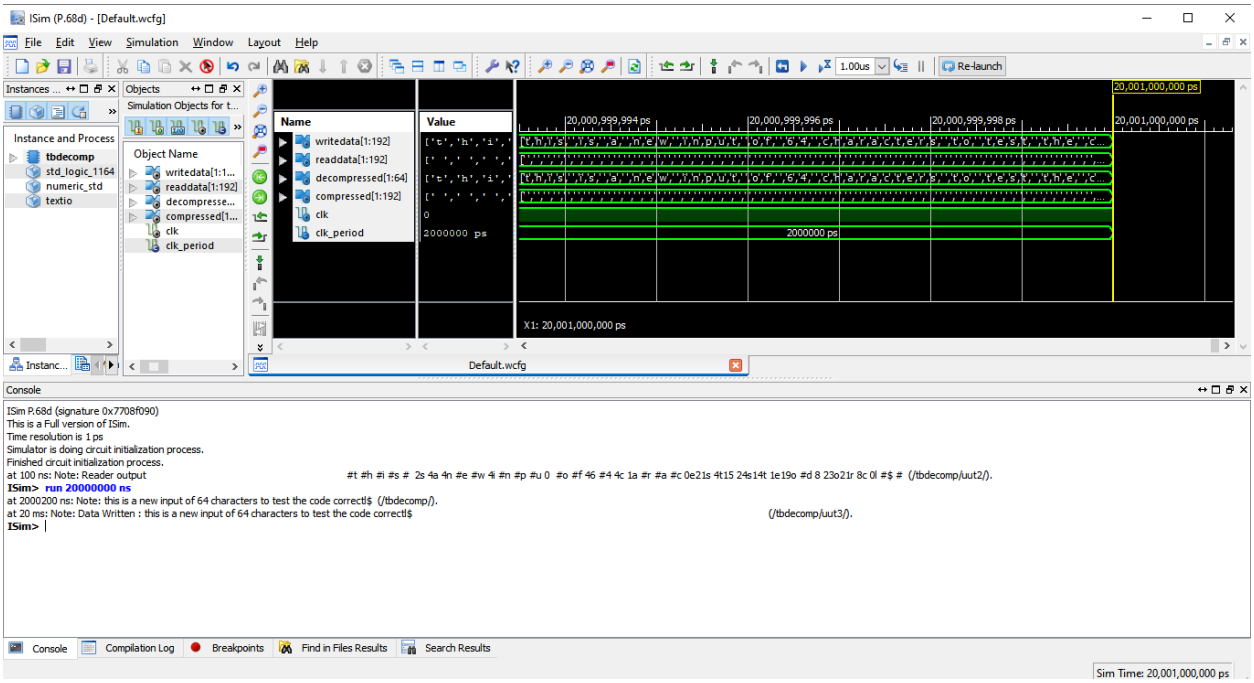
4. Results and Limitations

Limitations:

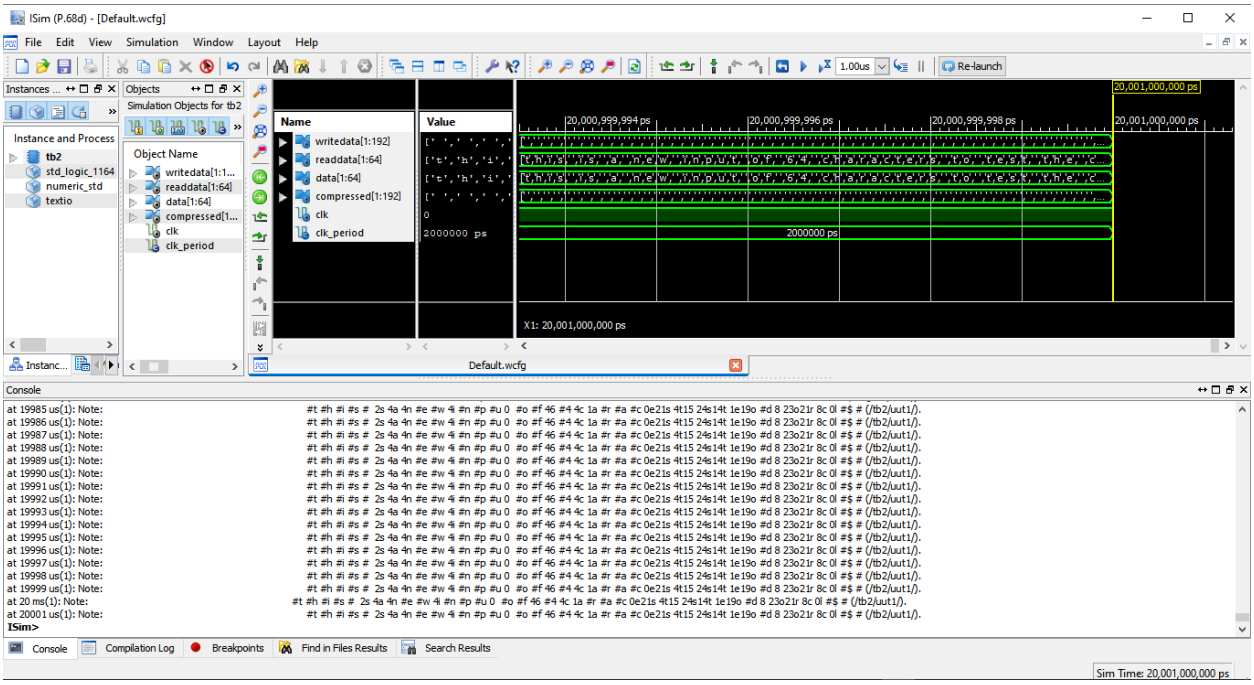
- The compressor doesn't work on any inputs except the ones that has extensions(.txt)
- for the worst-case scenario if all the 64 characters in input are unique, a restraint is put where the output of the compression must be 192 because each unique character is represented in 3 spaces for the decompressor to work correctly
- Sometimes the output contains a lot of spaces at the beginning of the String output.

Results:

Decompression result:



Compression results:



5. Code Listing

- File reader:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use std.textio.all;
use ieee.numeric_std.all;
--use ieee.std_logic_textio.all;


entity FileReader is
    Port (  ReadData : out  String (1 to 64));
end FileReader;

architecture Behavioral of FileReader is
```

```
begin

        p_read : process

                file File_toRead            : TEXT open read_mode is "INUN.txt";
                variable row                : LINE;
                variable Char          : Character;
                variable output: String (1 to 64);
                variable idx: integer:=1;

                begin
                wait for 100 ns;
                        while not endfile(File_toread) loop
                                readline(File_toread,row);
                                for k in row'range loop
--                              for i in 1 to 64 loop
                                        read(row,output(idx));
                                        idx := idx+1;
                                end loop;
                        end loop;
                        ReadData <= output;
                        report "Data to Read: " & output;
                        file_close( File_toRead);

                        wait;
        end process p_read;
```

```
end Behavioral;
```

- File Writer:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use STD.TEXTIO.all;
entity FileWriter is
	port(
			WriteData: in String(1 to 192)
	);
end FileWriter;

architecture Behavioral of FileWriter is

begin

	process
```

```
   file F: TEXT open WRITE_MODE is "OUTC.txt";
   variable L: LINE;
         variable temp: String (1 to 192):= "
";


  begin
         wait for 2000000 ns;
         temp:= WriteData;
         for i in temp'range loop
               if(temp(1) = '*') then
                     temp := temp(2 to 192) & " ";
               end if;
         end loop;
         for i in temp'range loop
                WRITE (L, temp(i), Left);
         end loop;

         report("Data to be written : " & WriteData);

         WRITELINE (F, L);
         file_close(F);
         wait;
         end process;

end Behavioral;
```

- FileReaderD:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use std.textio.all;
use ieee.numeric_std.all;
--use ieee.std_logic_textio.all;

entity FileReaderD is
   Port (  ReadData : out  String (1 to 192));
end FileReaderD;

architecture Behavioral of FileReaderD is

begin

        p_read : process

                file File_toRead            : TEXT open read_mode is "OUTC.txt";
                variable row                : LINE;
                variable Char          : Character;
                variable output: String (1 to 192):= "
";
                variable idx: integer:=1;

                begin
                        wait for 100 ns;
```

```
                while not endfile(File_toread) loop

                        readline(File_toread,row);
                        for k in row'range loop
                                if(idx > 192)then
                                        exit;
                                end if;
                                read(row,output(idx));
                                idx := idx+1;
                        end loop;
                end loop;
                ReadData <= output;
                report "Reader output" & output;
                file_close(File_toRead);

                wait;
        end process p_read;

end Behavioral;
```

- FileWriterD:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use STD.TEXTIO.all;
entity FileWriterD is
	port(
			WriteData: in String(1 to 192)
	);
end FileWriterD;

architecture Behavioral of FileWriterD is

begin

	 process

   file F: TEXT open WRITE_MODE is "OUTD.txt";
   variable L: LINE;
   variable temp: String (1 to 192):= "
";

  begin
	wait for 20000000 ns;
	temp:= WriteData;
	for i in temp'range loop
		if(temp(1) = '*') then
```

```
                    temp := temp(2 to 192) & " ";
                end if;
        end loop;
        for i in temp'range loop
                WRITE (L, temp(i), Left);
        end loop;

        report("Data Written : " & temp);

        WRITELINE (F, L);
        file_close(F);
        wait;
        end process;

end Behavioral;
```

- Compressor:

```
library IEEE;

use IEEE.STD_LOGIC_1164.ALL;

USE ieee.numeric_std.ALL;


entity Compressor is

   Port (              Clk: in std_logic;

                       Data : in string (1 to 64);

                       Compressed : inout  string(1 to 192) );

end Compressor;


architecture Behavioral of Compressor is



begin

                       compress: process(clk)

                       type StringArray is array(0 to 63) of string(1 to 3);

                       type DictionaryStringArray is array(0 to 63) of string(1 to 11);

--                     variable Dictionary : DictionaryStringArray := ("           " , "           " , "           " ,
"           " , "           " , "           " , "           " , "           " , "           " , "           " , "           " , "           " , "
" , "           " , "           " , "           " , "           " , "           " , "           " , "           " , "           " , "           " , "
" , "           " , "           " , "           " , "           " , "           " , "           " , "           " , "           " , "           " , "
" , "           " , "           " , "           " , "           " , "           " , "           " , "           " , "           " , "           " , "
" , "           " , "           " , "           " , "           " , "           " , "           " , "           " , "           " , "           " , "
" , "           " , "           " , "           " , "           " , "           " , "           " , "           " , "           " , "           " , "
" , "           " ); --6 chars (bytes)

                       variable Dictionary : DictionaryStringArray := ("***********",
"***********", "***********", "***********", "***********", "***********",
"***********", "***********", "***********", "***********", "***********",
```

```
"**********","**********","**********","**********","**********",
"**********","**********","**********","**********","**********",
"**********","**********","**********","**********","**********",
"**********","**********","**********","**********","**********",
"**********","**********","**********","**********","**********",
"**********","**********","**********","**********","**********",
"**********","**********","**********","**********","**********",
"**********","**********","**********","**********","**********",
"**********","**********","**********","**********","**********",
"**********","**********","**********" ); --6 chars (bytes)

                variable str : String(1 to 11) := "          ";

                variable dIdx: integer := 0;

                variable outIdx: integer := 0;

                variable refIdx : Integer := 0;

                variable ts : string(1 to 3) := "   ";

                variable i : Integer:= 1;

                variable outString : StringArray := ("   ","   ","   ","   ","   ","   ","   ","   ","   ","   ","
","   ","   ","   ","   ","   ","   ","   ","   ","   ","   ","   ","   ","   ","   ","   ","   ","   ","   ","
","   ","   ","   ","   ","   ","   ","   ","   ","   ","   ","   ","   ","   ","   ","   ","   ","   ","   ","
","   ","   ","   ","   ","   ","   ","   ","   ","   ","   ","   ","   ","   ","   ","   ","   ","   ","   ","
","   ","   ","   ","   ","   ","   ","   ","   ","   ","   ","   ","   ","   ","   ","   ","   ","   ","   ");

                variable final : string(1 to 192) :=
"************************************************************************
************************************************************************
************************************";

                variable j : integer := 0;

                begin
--              wait for 400 ns;
                 if( clk'event and clk = '1') then
                 if i < 65 then
--                                  str := "          ";
                                    str := "**********";
```

```
                                        refIdx := -1;
                                        j:=0;
                                        while j < 64 loop
                                                if (i >= 65) then
                                                        exit;
                                                end if;

                                                if (Dictionary(j) = str(2 to 11) & Data(i)) then
                                                        str := str(2 to 11) & Data(i);
                                                        refIdx := j;
--                                                      j := -1;
                                                        i := i + 1;
                                                end if;
                                                j:=j+1;
                                        end loop;
                                        if( i <65) then
                                                if refIdx = -1   then
                                                        outString(outIdx) := " #" & Data(i) ;
                                                        Dictionary(dIdx) := "**********" & Data(i);
                                                else
                                                        if(refIdx > 9) then
                                                                outString(outIdx) :=
integer'image(refIdx) & Data(i) ;
                                                        else
                                                                outString(outIdx) := " " &
integer'image(refIdx) & Data(i) ;
                                                        end if;
```

```
                                        Dictionary(dIdx) := Dictionary(refIdx)(2 to
11) & Data(i);
                                    end if;
                --                      ts := integer'image(refIdx);
                                    dIdx := dIdx + 1;
                                    outIdx := outIdx + 1;
                                    i := i+1;
                            end if;
                        end if;
--                      if str /= "         " then
--                          outString(outIdx) := Integer'image(refIdx) & " ";
--                      end if;
                        final := "
";
                        for i in outString'range loop
                            if outString(i) = "   " then
                                exit;
                            end if;
                            final := (final(4 to 192) & outString(i));
                        end loop;
                        end if;
                        Compressed <= final;
                        report final;
                end process;
end Behavioral;
```

- Decompressor:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
USE ieee.numeric_std.ALL;
USE STD.TEXTIO.ALL;


entity Decompress is
                        port( CLK : in std_logic;
                                Compressed : in string(1 to 192);
                                Decompressed : out string(1 to 64)
                         );
end Decompress;


architecture Behavioral of Decompress is
 ------------------------------------------------------------------
  function string_to_int(x_str : string; radix : positive range 2 to 36 := 10) return integer is
    constant STR_LEN         : integer := x_str'length;

    variable chr_val         : integer;
    variable ret_int         : integer := 0;
    variable do_mult         : boolean := true;
    variable power           : integer := 0;
  begin
```

```
for i in STR_LEN downto 1 loop
  case x_str(i) is
    when '0'      =>   chr_val := 0;
    when '1'      =>   chr_val := 1;
    when '2'      =>   chr_val := 2;
    when '3'      =>      chr_val := 3;
    when '4'      =>   chr_val := 4;
    when '5'      =>   chr_val := 5;
    when '6'      =>   chr_val := 6;
    when '7'      =>   chr_val := 7;
    when '8'      =>   chr_val := 8;
    when '9'      =>   chr_val := 9;
    when 'A' | 'a' =>   chr_val := 10;
    when 'B' | 'b' =>   chr_val := 11;
    when 'C' | 'c' =>   chr_val := 12;
    when 'D' | 'd' =>   chr_val := 13;
    when 'E' | 'e' =>   chr_val := 14;
    when 'F' | 'f' =>   chr_val := 15;
    when 'G' | 'g' =>   chr_val := 16;
    when 'H' | 'h' =>   chr_val := 17;
    when 'I' | 'i' =>   chr_val := 18;
    when 'J' | 'j' =>   chr_val := 19;
    when 'K' | 'k' =>   chr_val := 20;
    when 'L' | 'l' =>   chr_val := 21;
    when 'M' | 'm' =>   chr_val := 22;
    when 'N' | 'n' =>   chr_val := 23;
```

```
      when 'O' | 'o' =>   chr_val := 24;
      when 'P' | 'p' =>   chr_val := 25;
      when 'Q' | 'q' =>   chr_val := 26;
      when 'R' | 'r' =>   chr_val := 27;
      when 'S' | 's' =>   chr_val := 28;
      when 'T' | 't' =>   chr_val := 29;
      when 'U' | 'u' =>   chr_val := 30;
      when 'V' | 'v' =>   chr_val := 31;
      when 'W' | 'w' =>   chr_val := 32;
      when 'X' | 'x' =>   chr_val := 33;
      when 'Y' | 'y' =>   chr_val := 34;
      when 'Z' | 'z' =>   chr_val := 35;
      when '-' =>
        if i /= 1 then
          report "Minus sign must be at the front of the string" severity failure;
        else
          ret_int          := 0 - ret_int;
          chr_val          := 0;
          do_mult          := false;    --Minus sign - do not do any number manipulation
        end if;

      when others => report "Illegal character for conversion for string to integer" severity
failure;
    end case;

    if chr_val >= radix then report "Illagel character at this radix" severity failure; end if;
```

```
    if do_mult then
      ret_int             := ret_int + (chr_val * (radix**power));
    end if;

    power                 := power + 1;

  end loop;

  return ret_int;

 end function;
begin
Decompress: process(CLK)
                        type StringArray is array(0 to 63) of string(1 to 3);
                        type DictionaryStringArray is array(0 to 63) of string(1 to 11);
                        variable Dictionary : DictionaryStringArray := ("**********",
"**********","**********","**********","**********","**********",
"**********","**********","**********","**********","**********",
"**********","**********","**********","**********","**********",
"**********","**********","**********","**********","**********",
"**********","**********","**********","**********","**********",
"**********","**********","**********","**********","**********",
"**********","**********","**********","**********","**********",
"**********","**********","**********","**********","**********",
"**********","**********","**********","**********","**********",
"**********","**********","**********","**********","**********",
"**********","**********","**********" ); --6 chars (bytes)
                        variable dIdx: integer := 0;
                        variable outIdx: integer := 0;
```

```
                        variable refIdx : Integer := 0;
--                      variable final : string(1 to 64) := "                                                ";
                        variable final : string(1 to 64) :=
"****************************************************************";
--                      variable shortfinal : string(1 to 64) := "
";
                        variable strt : integer range 0 to 200 := 1;
                        variable refcom : Integer := 0;
--                      variable temp : string (1 to 2):= "  ";


                        begin
                        if(now > 300 ns and clk'event and clk='1') then
--                       while strt<127 loop
                         if strt<191 then
                                 if compressed(strt + 1) = ' ' then
                                         strt := strt + 1;
                                 else
                                         if Compressed(strt) = ' ' then


                                                 if      Compressed(strt + 1) = '#' then
                                                         Dictionary(dIdx) := "**********" &
Compressed(strt + 2);
                                                         final := final( 2 to 64) & Compressed(strt +
2);
                                                 else
                                                         refcom := string_to_int("" &
Compressed(strt + 1));
```

```
                                        Dictionary(dIdx) :=  Dictionary(refcom)(2 to
11) & Compressed(strt + 2);
                                        for n in Dictionary(dIdx)'range loop
                                            if ( Dictionary(dIdx)(n) /= '*') then
                                                final := final( 2 to 64)
& Dictionary(dIdx)(n);
                                            end if;
                                        end loop;
                                    end if;
                                else
                                        refcom := string_to_int("" & Compressed(strt) &
Compressed(strt + 1));
                                        Dictionary(dIdx) :=  Dictionary(refcom)(2 to 11) &
Compressed(strt + 2);
                                        for n in Dictionary(dIdx)'range loop
                                            if ( Dictionary(dIdx)(n) /= '*') then
                                                final := final( 2 to 64) &
Dictionary(dIdx)(n);
                                            end if;
                                        end loop;
                                end if;
                                strt := strt+3;
                                didx := dIdx+1;


--                              end loop;
                                end if; -- of space character
                                end if; -- of loop
                                Decompressed <= final;
```

```
                        end if;
end process;
end Behavioral;


-- al4ea55alfal2345srf53rtaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa$
-- [ #d #f #g #s 0g 0s #y #3 #6 #8 #5 72 #7 8t128 74 # 16 17 18 19 20 21 22]
```

- Test Bench Compressor(TB2):

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
use std.textio.all;
ENTITY TB2 IS
END TB2;

ARCHITECTURE behavior OF TB2 IS

   -- Component Declaration for the Unit Under Test (UUT)



                        COMPONENT Compressor
   PORT(
                                CLK : in std_logic;
      Data : IN  string(1 to 64);
      Compressed : INOUT  string(1 to 192)
     );
   END COMPONENT;

                        COMPONENT FileWriter
   PORT(
                                WriteData: in String (1 to 192)
                         );
```

```
  END COMPONENT;

                        COMPONENT FileReader
  PORT(
     ReadData : OUT  String(1 to 64)
    );
  END COMPONENT;
                        ----------------------------------------------------------------------

                        signal writeData : String(1 to 192):= "
";

                        signal ReadData :  String(1 to 64):= "                                                  ";

  --Inputs
  signal Data : string(1 to 64) :=     "                                                  ";

                        --BiDirs
  signal Compressed : string(1 to 192) := "
";

                        signal clk : std_logic := '1';

                        constant clk_period : time := 2000 ns;
BEGIN

                        -- Instantiate the Unit Under Test (UUT)
  uut1: Compressor PORT MAP (
```

```
                                clk => clk,
    Data => Data,
    Compressed => Compressed
  );

                    uut2: FileReader PORT MAP (
    ReadData => ReadData
  );

                    uut3: FileWriter PORT MAP (
                            WriteData => WriteData
  );

                    clk_proc: process
                    begin
                     clk <= '1';
                     wait for clk_period/2;
                     clk <= '0';
                     wait for clk_period/2;
end process;
stim_proc: process

begin
                     wait for 300 ns;
                     Data <= ReadData;  --&"                                                ";
                     wait for 200000 ns;
                     WriteData <= Compressed;
```

```
                    wait;
  end process;

END;
```

- Test Bench decompressor:

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
use std.textio.all;
ENTITY TBDecomp IS
END TBDecomp;

ARCHITECTURE behavior OF TBDecomp IS

   -- Component Declaration for the Unit Under Test (UUT)



        COMPONENT Decompress
   PORT(
                        Clk : in std_logic;
      Compressed : IN  string(1 to 192);
      Decompressed : out  string(1 to 64)
     );
   END COMPONENT;

        COMPONENT FileWriterD
   PORT(
                        WriteData: in String (1 to 192)
                );
   END COMPONENT;
```

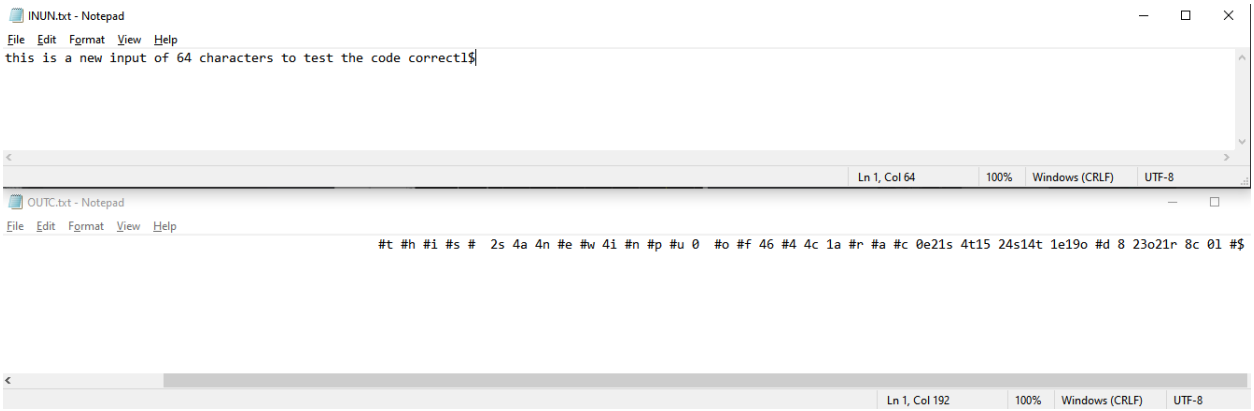
```
            COMPONENT FileReaderD
   PORT(
      ReadData : OUT  String(1 to 192)
     );
   END COMPONENT;
          ----------------------------------------------------------------------

          signal writeData : String(1 to 192):= "
";

          signal ReadData :  String(1 to 192):= "
";

  --Inputs
  signal Decompressed : string(1 to 64) :=     "                                                    ";

          --BiDirs
  signal Compressed : string(1 to 192) := "
";

          signal clk : std_logic := '1';

          constant clk_period : time := 2000 ns;

BEGIN

          -- Instantiate the Unit Under Test (UUT)
```

```
uut1: DeCompress PORT MAP (
      clk => clk,
                              Compressed => Compressed,
      deCompressed => deCompressed
    );


         uut2: FileReaderD PORT MAP (
      ReadData => ReadData
    );


        uut3: FileWriterD PORT MAP (
                              WriteData => WriteData
    );


        clk_process: process
        begin
                wait for clk_period/2;
                clk<= '0';
                wait for clk_period/2;
                clk<= '1';
        end process;


stim_proc: process


begin
                wait for 200 ns;
                Compressed <= ReadData;  --&"                                                    ";
```

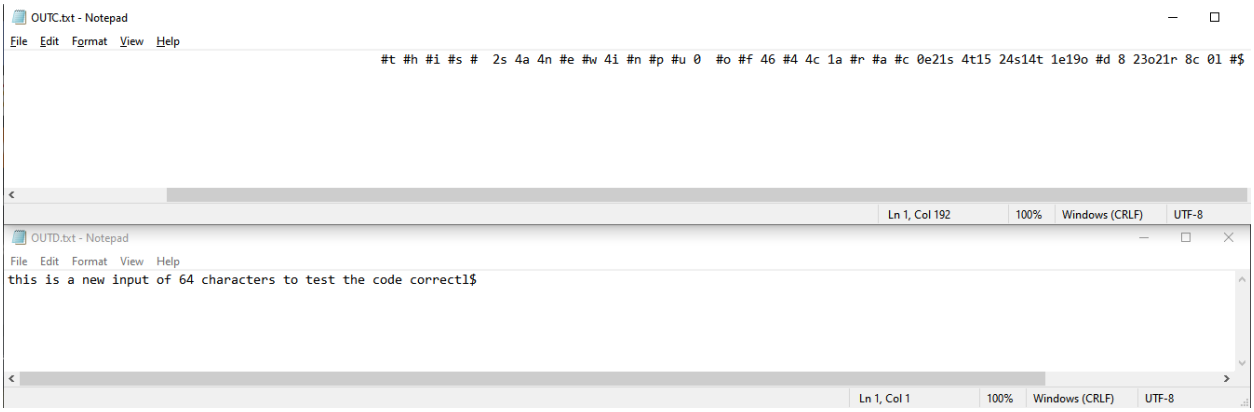
```
                wait for 2000000 ns;
                report Decompressed;
                WriteData <= Decompressed & "
";
                wait;
  end process;

END;
```

Results:

Compression:



Decompression:



Link : [LZCompression - Google Drive]