Data Compression Circuit using RLE

Abstract—**This document covers designing and implementation of data compression circuit using run-length encoding (RLE). It's work is to replace continuously repeated occurrences of a byte by a repeat count and the byte value.**

I. General idea

I divided the given problem into three parts / components :

(a) Taking inputs on each rising edge of clock.
(b) Processing the data and preparing the output format.
(c) Printing the output by giving extra rising edges of clock.

II. Implementation

A. *Taking inputs on each rising edge of clocks*

I defined data structure *buffers* which is an array of vector (length 1 byte) of length 100 and a counter *input_count* which stores the number of inputs stored in array at any time. For every rising edge of clock, input is given to the program from the *test.txt* file. The given input is stored in the array and the the counter *input_count* is incremented. This process is repeated until all inputs from file are taken. After taking all inputs a sentinel *("00000000")* is sent as input to stop the process and move towards next step of the program.

B. *Processing the data and preparing the output format*

After taking inputs next step is to process the given data and generate output. As per the given conditions/constraints I used the input data to generate the output and stored it in the array *output_array*. Handled last input as special case because after that there were no inputs. Used variable *repetitions* to count the repetitions of a specific byte in the sequence.
The given conditions/constraints are as follows:

1) If any character 'c' repeats 'n' times in the input stream such that 2<n<16 then we output the three-byte sequence "ESC n c".
2) If 'n' number of 'ESC' characters arrive contiguously in the input stream, we output the 3-byte sequence "ESC n ESC", where n can be from 0<n<16. Otherwise, we just output the received characters without any change.
3) Only 1 byte is output per clock cycle and whenever a byte is being output, the Data Valid output line must go from low to high.
4) If the repeat count is more than 15, we handle the first 15 characters as above and treat the 16^{th} occurrence onwards as if a new character has been received.
5) The reading and writing operations are done in the test bench itself.

Using these above conditions/rules output is generated in *output_array* and for each rising edge of the clock it should be given as output in *out.txt* one by one. The count of number of elements in *output_array* is stored in variable *output_count* which is incremented when a new output is added into the array.

C. *Printing the output by giving extra rising edges of clock*

After generating and storing output in the *output_array* the next task is give the output out for printing it in *out.txt*. As for each rising edge we can give only 1 output so to get all outputs I provided some extra rising edges. When all output is given out at last one extra output (sentinel *"00000000"*) is given which stops the rising edges of clock.

III. Algorithm

The general implementation idea of the code can be seen from below algorithm which covers only abstract features of the code.

Algorithm 1: Algorithm implementation

```
Input: Bytes sequence
Output: Compressed Bytes sequence
 1: if RISING_EDGE(clk) then
 2:    if ($input != "00000000"$) then
 3:       save input into input_arrays
 4:    else if ($process\_output = 0$) then
 5:       if element='ESC' then
 6:          store it into output_array as per the condition 1
 7:       else
 8:          store it into output_array as per the condition 2
 9:       end if
          $process\_output = 1$
10:    else
11:       send the output out
12:    end if
13: end if
```

IV. Assumptions/ Usage

1) Used sentinel *"00000000"* to inform end of the process.
2) Used ieee.numeric_std.all to convert integer to 8 bit binary numbers.
3) Assumed standard assumptions mentioned in the question

References

(a) Array in vhdl: https://surf-vhdl.com/vhdl-array/
(b) Conversion of integer into binary number of 8 bits: https://electronics.stackexchange.com/questions/4482/vhdl-converting-from-an-integer-type-to-a-std-logic-vector
(c) Initialising variables and signals: https://stackoverflow.com/questions/41681715/signal-is-unable-to-initialise-to-integer-value-in-vhdl