Assignment-6 (CS 232) by Utkarsh Ranjan

Data Compression Circuit using Run-Length Encoding (RLE)

Introduction

- **Run-length encoding** (**RLE**) is a form of **lossless data compression** in which *runs* of data (sequences in which the same data value occurs in many consecutive data elements) are stored as a single data value and count, rather than as the original run.
- In this assignment we have implemented this encoding using a data compression circuit. At every rising edge of the clock the circuit receives a fresh byte.
- These sequence of byte inputs are compressed using the following rule:-

  a. If any character 'c' repeats 'n' times in the input stream such that 2<n<6 then we output the three-byte sequence "ESC n c".

  b. 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<7. Otherwise, we just output the received characters without any change.

  c. If the repeat count is more than 5, we handle the first 5 characters as above and treat the
  6th occurrence onwards as if a new character has been received.

Circuit Component
RLE encoder => It has an array of bytes which acts as the buffer in this case.

Entity Description

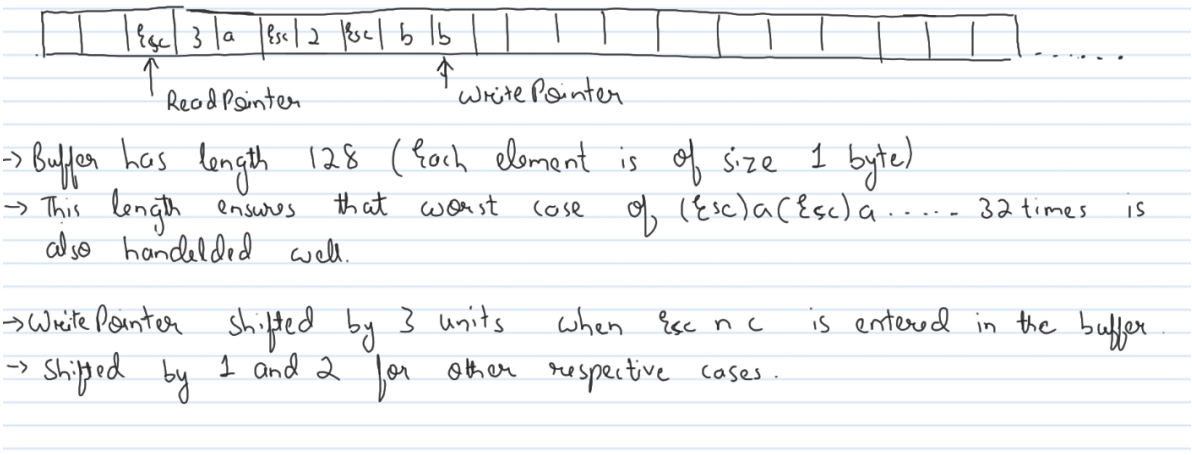
```
entity RLE_encoder is
port ( clk, rst: in std_logic;
         input: in std_logic_vector (7 downto 0);
         DataValid: out std_logic := '0' ;
         output: out std_logic_vector (7 downto 0));
end entity;
```

- The entity runs a single process which maintains two state of the byte input being given to the encoder. These state are respectively `state` and `prev_state` . It also has a variable `count` which stores the streak of the current input byte in the input array.

  ```
  variable state, prev_state: std_logic_vector(7 downto 0) := (others => '-');
  variable count : std_logic_vector(2 downto 0) := "001";
  ```

- The size of the buffer is kept as 128. There are two pointers associated with the buffer `WritePointer` and `ReadPointer` which are updated in accordance to the condition of output being fed to the buffer or is being extracted out

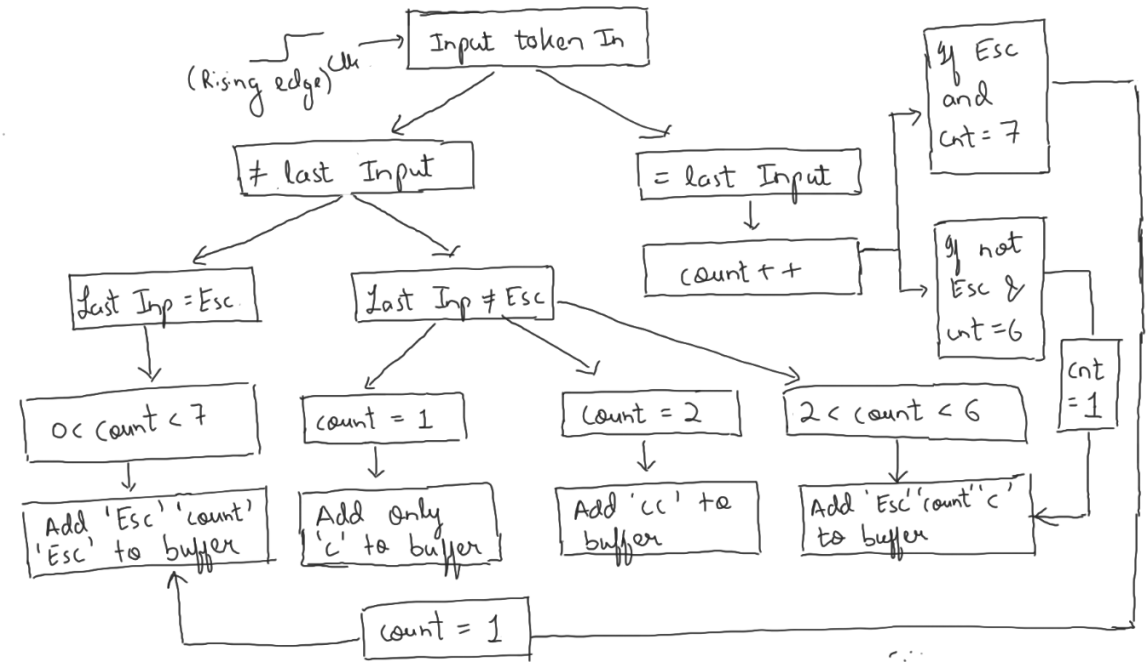
```
// Buffer
Type Mem is array ( 127 downto 0) of std_logic_vector( 7 downto 0);
variable Memory : Mem;

// Pointers
variable ReadPointer : std_logic_vector(7 downto 0) := (others => '0');
variable WritePointer : std_logic_vector(7 downto 0) := (others => '0');
```



-> Buffer has length 128 (Each element is of size 1 byte)
-> This length ensures that worst case of (Esc)a(Esc)a..... 32 times is also handelded well.

->WritePointer shifted by 3 units when Esc n c is entered in the buffer.
-> Shifted by 1 and 2 for other respective cases.

Block Diagram



Testbench

- The testbench is almost same as the one discussed in the class with a slight difference in the implementation of the clock.
- Clock signal is a std_logic which is updated as follows:-

```
// Declaration
signal clk : std_logic := '0';
```

```
while Line_count < 129 loop

    if(clk = '1') then
        clk <= not clk;
        wait for 20 ns;
        next;
    end if;

    // read input

    clk <= not clk;
    wait for 20 ns;

    // print output
end loop;
```