CS232-Lab6

Hardik Rajpal - 200050048

March 2022

1 Run Length Compressor

1.1 Component

The main circuit component responsible for compressing the input accepts four inputs:

- charIn : an 8-bit input for the character being read
- clk : the clock signal
- rst : this signal is used to reset the circuit for the first cycle.
- inputOver: this signal is used to indicate that the input array is exhausted.

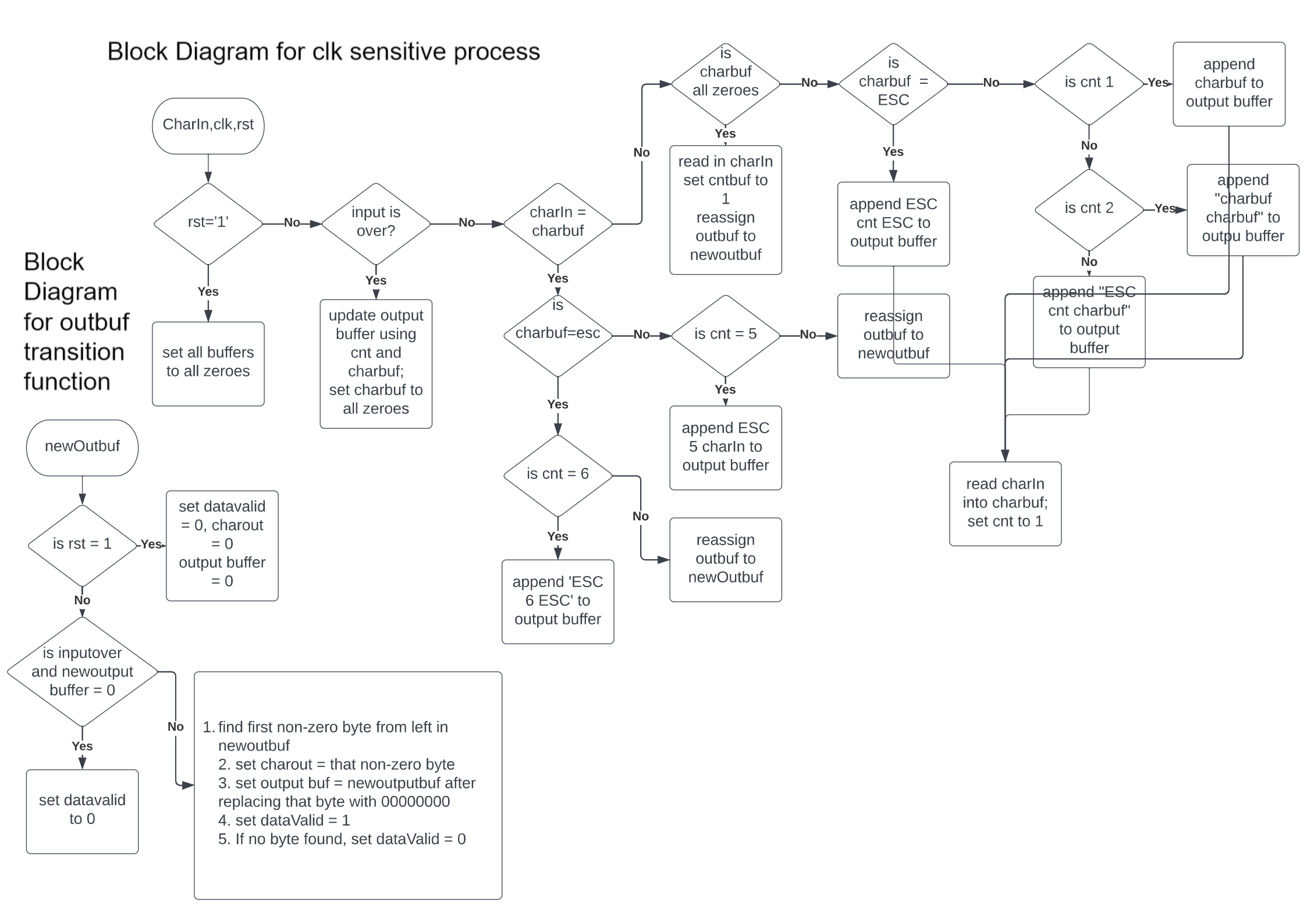
The outputs are the specified dataValid bit and an 8-bit vector, charOut.
The architecture maintains a local signals to keep a track of the current character being read (charbuf), the state of the number of times the current character has appeared (cntbuf and nextcnt), a state of the output buffer (using outbuf and newOutBuf), a constant bit vector for ESC and a signal for indicating that the input is over.

The first clock cycle, we reset the variables to 0s. Every clock cycle, we read into charIn, update the output buffer and count appropriately. If the character count has reached the limit (assumed to be 5), or a new character has been encountered while count ≥ 3 we append "ESC [count] [char]" to the output buffer. If the count ≤ 2 and a new character is found, we append "[char]" *count* times to the output buffer. Similarly, the rules described for ESC are followed. Even if the output buffer doesn't have to be updated, we still reassign outbuf to newOutBuf, to call the process sensitive to it.
Every time newOutBuf is assigned a value, (excluding reset conditions) we loop through newOutbuf to find the first non-zero byte form the left. We assign this byte to charOut and set dataValid = 1. If there are no non-zero bytes and the inputOverStorage is 1, meaning the input has been completed, we set dataValid = 0, as it is not possible for the buffer to output any meaningful values from here onward.
Finally there are two more processes to update count with nextCount and to keep track of inputOverStorage.

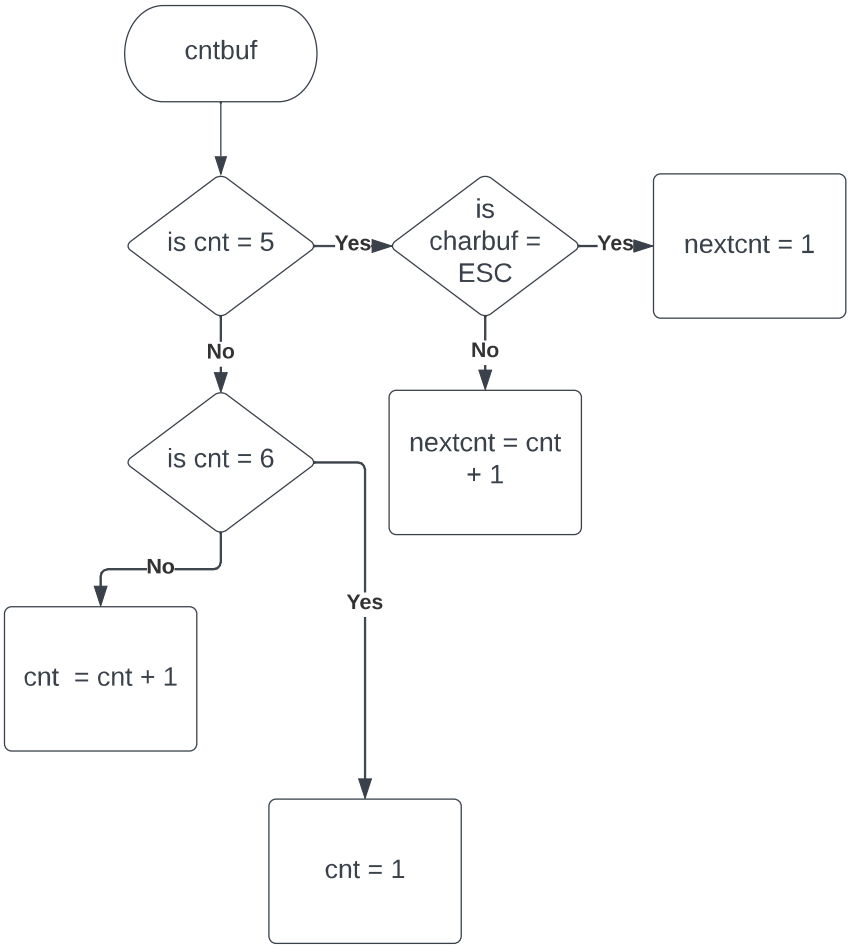
1.2 TestBench

The testbench uses one instance of the component designed previously and has corresponding signals which it manipulates to use it. CharInSg is read from the input file (input.txt) while clkSg,rst and inputOverSg are manually set to 0 or 1 suitably. We have used a clock period of 2ns. We use two while loops to compress the input.
The first one goes through input.txt, reading each character into the component and updating output.txt if dataValid = 1.
The second loop runs until dataValidSg is counted to have been 0 for 8 cycles, and since the input is over, it means the output buffer is empty. The testbench ends execution on exiting the second loop.
Please see block diagrams for futher clarity

Block Diagram for clk sensitive process



Block Diagram for cntbuf transition function



Block Diagram for TestBench