LZSS Circuit

By Eliat Avidan

CPE 405

LZSS is one of many techniques of decoding and encoding losslessly. This is a simple compression technique that based on two windows: search buffer and lookahead buffer. Search buffer contain the recently encoded sequence while lookahead buffer contain the next part to be encoded. To minimize the time, it takes to encode the whole sequence- a very large buffer is being used (in the assignment we used a small buffer, 7 and 5 symbols, but this was just for learning purposes).

The process: there is a pointer that goes back in the search buffer until there is a match to the symbol in the lookahead buffer. The distance of the pointer from the lookahead buffer is called the **offset**. After finding the first match, the pointer continues searching for more symbols followed by the first one to match a sequence in the lookahead buffer. The number of symbols matched between the search buffer to the lookahead buffer is called the **length of the match**. Every step we will shift to the left the sequence and have new symbol in the lookahead buffer to compare. In case of finding length of the match, we need to shift as the length of the match symbols.

Encoding steps:

Step 1-> Initialize the buffers to zero (search and lookahead).
Step 2-> shift char by char from known string until filling lookahead buffer.
Step 3->Search for the longest matching string in the buffer.
Step 4.->If a match is found, set all the data needed (match, length and distance).

Otherwise, set match to zero (match did not find) and the first un-coded symbol to the encoded output.

Step 5.->Shift the buffers as the number of matched char found, otherwise, shift one time.
Step 6.->Repeat from Step 3, until all the entire input has been encoded.

Eilat Avidan
CPE405
Final project

Verilog code:

```
module lzss (clk, reset, data_in,o_match_out,o_distance_out, o_length_out,o_char_send);

input clk;
input reset;
input [7:0] data_in;            //8 bit char
output o_match_out;             //either 1/0 depend of a match found
output [3:0] o_distance_out;    //the distance of the char found in the search buffer
output [3:0] o_length_out;      //the number of char found
output [7:0] o_char_send;       //if there is no match, return char
parameter bits = 8;             //size of buff as constent
parameter numChar = 8;


reg [numChar*bits-1:0] search;      //reg to hold 8 char
reg [numChar*bits-1:0] lookahead;   //reg to hold 8 char
reg [3:0] distance_out;             //the distance of the char found in the search buffer
reg [3:0] length_out;               //the number of char found
reg [7:0] char_send;                //if there is no match, return char
reg match_out;

integer count,i,j;
integer count_match = 0;            //keep truck of the # of matched char found
integer temp;
integer length_til_first_match;     //distance from the begining of search buffer
integer shift_count = 1;            //keep truck of shifting

assign o_distance_out = distance_out;
assign o_length_out = length_out;
assign o_char_send = char_send;
assign o_match_out = match_out;

initial
begin
    count = 0;
    for (i=1; i<=8 ; i = i+1)       //initialize both buffers to zeros
    begin
        search[i*bits-1-:bits] = 0;
        lookahead[i*bits-1-:bits] = 0;
    end                             //end for loop
end                                 //end initial


always @(posedge clk) begin
if (lookahead[numChar*bits-1-:bits] == 01)      //stop condition
    $stop;
    if (shift_count != 0) begin
        search = search << 8;                                   //shift buffer by one char (total of 8 shifting)
        search[bits-1-:bits] = lookahead[numChar*bits-1-:bits]; //copy MSB in lookahead buffer to LSB in search buffer
        lookahead = lookahead << 8;                             //shift lookahead buffer by one char
        #1 lookahead[bits-1-:bits] = data_in;                   //new char to the lookahead buffer
        shift_count = shift_count - 1;                          //decreament shift count
        length_out = 0;
        match_out = 0;
        char_send = 0;

    end                                                         //end if statement (shift_count != 0)
    if (shift_count == 0) begin
    count = count +1;                                           //keep truck of num of shifts
    temp = 0;
    count_match = 0;
    length_til_first_match = 1;
    if (count > numChar) begin                  //as long as we did not reach the end of the buffer, ifl
        for (i = 1; i < numChar; i = i + 1) begin
            if (search[bits*i-1-:bits] == lookahead[numChar*bits-1-:bits] && (i != 1)) begin    //if 2 check if there is any bit in search buff = to MSB in look
                count_match = count_match + 1;              //if there is a match, increament count match
                length_til_first_match = i;                 //keep truch of the distance from the beginning of search buffer
```

Eilat Avidan
CPE405
Final project

```
67                  //$display ("%c,%c", search[bits*i-1-:bits], lookahead[numChar*bits-1-:bits]);
68                  for (j = i-1; j > 0; j = j - 1) begin       //if match found, go back to check rest of char
69                      if (search[bits*j-1-:bits] == lookahead[numChar*bits-(i-j)-:bits] && (temp == 0)) begin
70                          count_match = count_match + 1;      //if other char matched, increament count_match
71                      end                                     //end for loop (j)
72                      else
73                          temp = 1;                           //distinguish between first char in the buffer compared, to the rest of char
74                      end                                     //end for loop
75                  end                                         //end if2
76              else if ( search[bits*i-1-:bits] == lookahead[numChar*bits-1-:bits])    //else to if 2, just first char match
77                  count_match = 1;
78              if (count_match == 1 ) begin
79                  length_out = 1'b1;                          //after exiting the loop, copy data needed
80                  distance_out = length_til_first_match;
81                  match_out = 1'b1;
82              end                                             // end (count_match == 1 )
83              else if (count_match > 1 ) begin
84                  length_out = count_match;
85                  distance_out = length_til_first_match;      //changed for i
86                  match_out = 1'b1;
87                  shift_count = count_match - 1;
88                  end                                         // end else if
89              else begin                                      //if we did not find match
90                  match_out = 1'b0;                           //set match_out to zero
91                  char_send = lookahead[numChar*bits-1-:bits];//send the char
92                  end                                         //end else
93
94              end                                             // end big for loop
95          end                                                 //end if
96          shift_count = shift_count + 1;
97      end                                                     // end (shift_count == 0)
98  end                                                         //end always
99
100 endmodule
```

Testbench:

Eilat Avidan
CPE405
Final project

```
module testB();
//================================================
//Declartions
//================================================
reg clk,reset;
reg [7:0] data_in;          //8 bit char
reg [7:0] mem [0:19];
wire o_match_out;
wire [3:0] o_distance_out;
wire [3:0] o_length_out;
wire [7:0] o_char_send;
parameter bits = 8;
parameter numChar = 8;

integer i = 0;

//================================================
//Calling top modoule
//================================================

lzss test (clk, reset, data_in,o_match_out,o_distance_out, o_length_out,o_char_send);

//================================================
//Initial Declerations
//================================================

initial
begin
    clk <= 0;
    reset <= 1;

    #5 reset <= 0;
    $display("Match not found:(Match found, Char)");
    $display("Match found:(Match found, Distance, Length)");

end
```

```
always @(posedge clk) begin
    $readmemh ("data.dat", mem);        //reading data from a file
    data_in = mem[i];                   //copy the data to data_in
    i = i + 1;                          //move to the other char
    if (o_match_out == 0 && o_char_send != 0) begin     //print if n o match found
        $display("(0, %c)",o_char_send);
    end
    if (o_match_out == 1) begin                         //print if match has been found
        $display("(1,%d,%d)",o_distance_out, o_length_out);
    end
end     //end always

always begin
        clk <= 0;#5;
        clk <= 1;#5;
end

endmodule
```

Clarifications:
Using the string given in the homework- hillibilly_hill_bill (total of 20 char).
Size of search buffer – 8 char (64 bits)

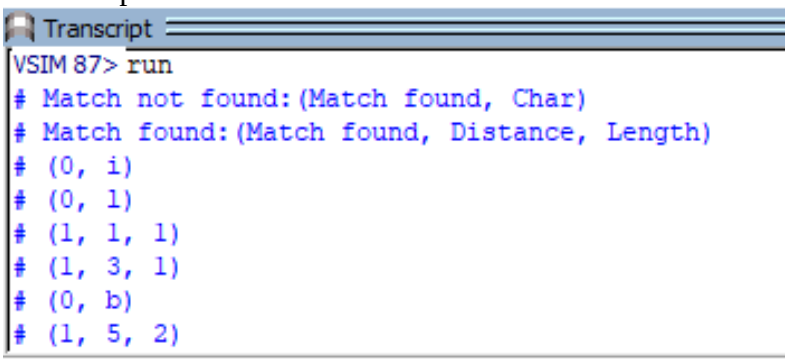
Eilat Avidan
CPE405
Final project
Size of lookahead buffer – 8 char (64 bits)

Search buffer								Lookahead buffer							
							h	I	L	l	i	b	i	l	l
						h	i	L	L	i	b	i	l	l	y
					h	i	l	L	I	b	i	l	l	y	_
				h	i	l	l	I	B	i	l	l	y	_	h
			h	i	l	l	i	b	I	l	l	y	_	h	i
		H	i	l	l	i	b	I	L	l	y	_	h	i	L

Match found- (1, distance , length of string found)
Match not found- (0, char sent)

Expected output-
(0,i)
(0,l)
(1,1,1)
(1,3,1)
(0,b)
(1,5,3)

Code output:



LZSS module:

```
reg [numChar*bits-1:0] search;          //reg to hold 8 char
reg [numChar*bits-1:0] lookahead;       //reg to hold 8 char
reg [3:0] distance_out;                 //the distance of the char found in the search buffer
reg [3:0] length_out;                   //the number of char found
reg [7:0] char_send;                    //if there is no match, return char
reg match_out;
```

Eilat Avidan
CPE405
Final project

```
integer count,i,j;
integer count_match = 0;
integer temp;
integer length_til_first_match;
integer shift_count = 1;
```

The two buffers declared using parameters that their size can be changed. In my case, I chose 64 bit long for each buffer. There are 4 main outputs: 1/0- if a match found, the character sent in case no match has been found, the distance from the beginning of search buffer of the char found, and lastly, the length of the string of matched characters.

In order to achieve it, I used some variables to keep truck of the counting.

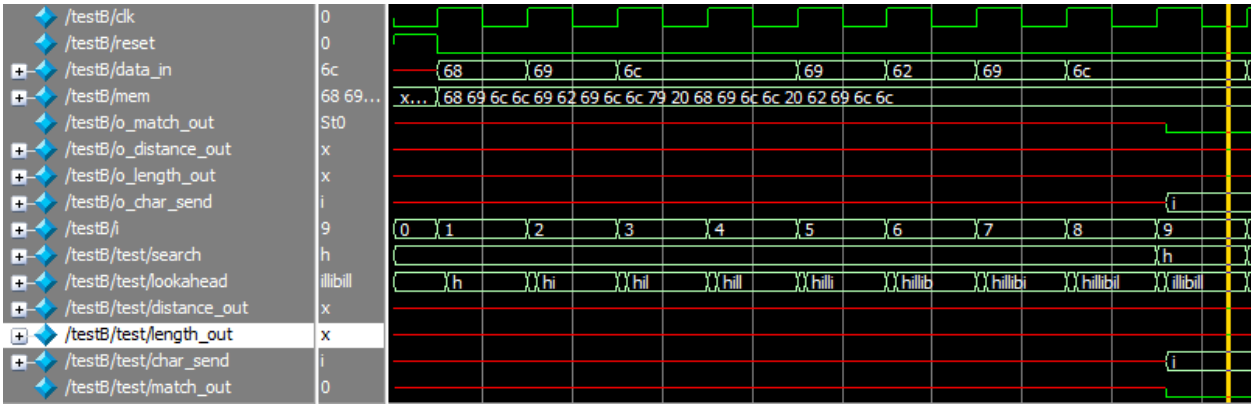
Count_match will count the length of the string match, after shift will reset to zero.

Length_til_first_match- count the distance from the beginning of search string until the first match char found.

Shift_count- we always shift one time until more than one characters found

Simulation results:

➔ Shifting 8 times until lookahead buffer is full and search buffer gets first char.



Eilat Avidan
CPE405
Final project

➔ Output after shifting when the algorithm is looking for match.