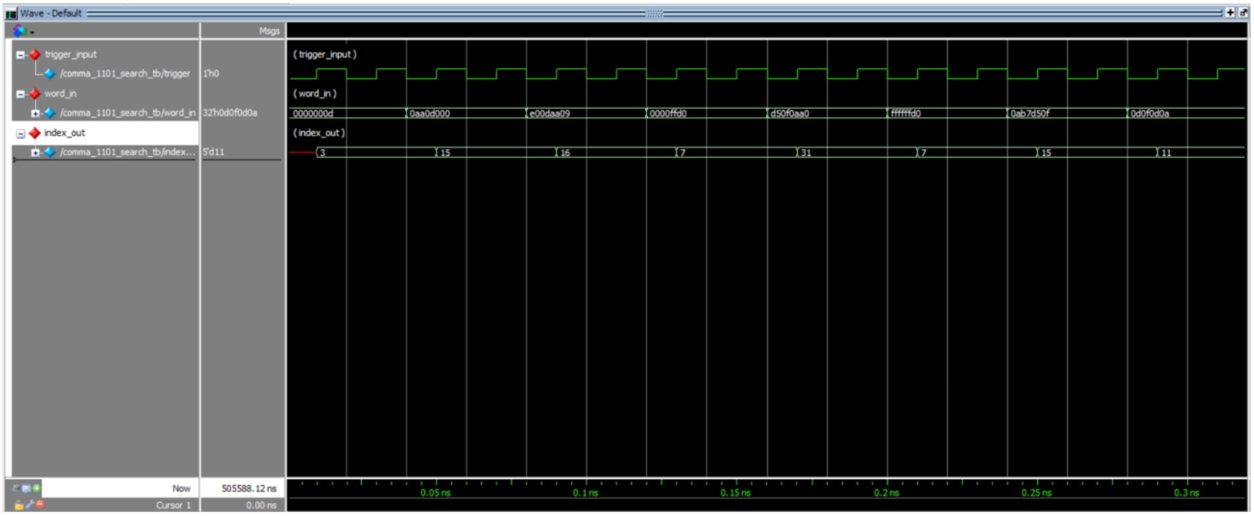
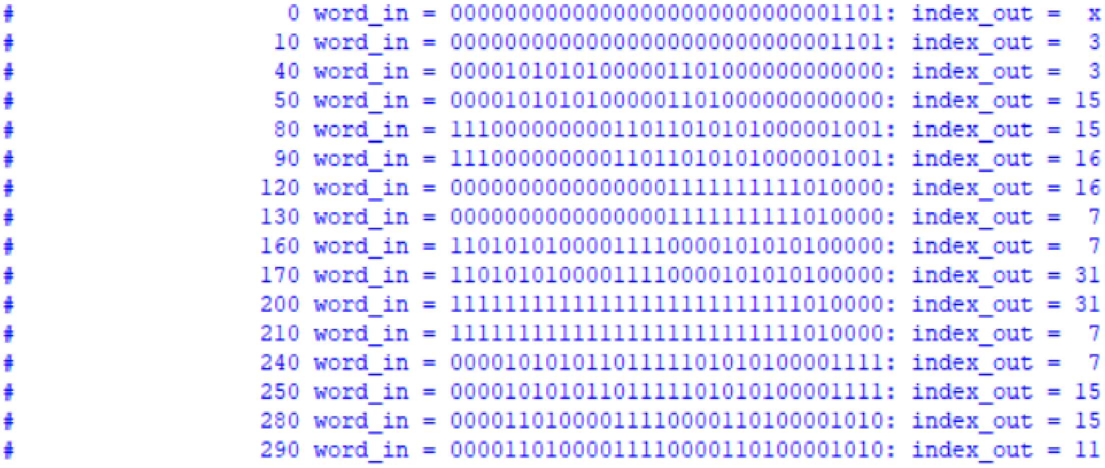
```verilog
/*----------------------------------------------------------------------------------
Name Lamin Jammeh
CLass: EE417 Summer 2024
Lesson 06 HW Question 2
Group: Ron Kalin/ Lamin Jammeh
Project Description: This is the main module, it detects a comma symbol 4b'1101 once
the symbol is detect the index of the MSB of the symbol and send it as an output
----------------------------------------------------------------------------------*/
module comma_1101_search #(parameter word_size = 32,
                       index_size = 4)
               (output  reg [index_size:0]   index_out,     //4 bits index [4:0]=16+8+4+2+1=31, possible index_out = 0:31
               input    [word_size - 1:0]    word_in,       //32 bits input [31:0]
               input                         trigger);      //triggers the count
               reg [word_size - 1:0]   temp_reg;            //Temporary register to locate the comma code

always @ (posedge trigger)
         begin: search_code
            temp_reg = word_in;                       //copying the result into the temporary register
            for (index_out = 3;  index_out < (word_size);   index_out = (index_out + 1))
               begin
                  if (temp_reg[3:0] == 4'b1101) disable search_code; //always check the 3LSBs
                  else begin
                        temp_reg = temp_reg >> 1;                       //shift the temp_reg to the right
                        if ((temp_reg == 0) | (index_out == (word_size - 1)))    //the temp_reg is all 0s
                               begin
                               index_out = 4'b0000;
                               disable search_code;
                               end
                        end
                  end
         end
endmodule
```

```
/*----------------------------------------------------------------------------------
Name Lamin Jammeh
CLass: EE417 Summer 2024
Lesson 06 HW Question 2
Group: Ron Kalin/ Lamin Jammeh
Project Description: This is the test-bench for the comma symbol 4b'1101 detector
-------------------------------------------------------------------------------------*/
module comma_1101_search_tb ();

//set the parameters wires and registers
parameter                 word_size   = 32;
parameter                 index_size  = 4 ;
wire       [index_size:0]   index_out;
reg         [word_size  - 1:0]   word_in;
reg                         trigger;

//define the unit under test UUT
comma_1101_search UUT (index_out, word_in, trigger);

//instantiate the trigger signal
initial begin
      trigger     = 1'b0;
      forever
         #10      trigger  = ~trigger;
      end

//define the word inut and observe the outputs
initial fork
        word_in = 32'b_0000_0000_0000_0000_0000_0000_0000_1101;
   #40  word_in = 32'b_0000_1010_1010_0000_1101_0000_0000_0000;
   #80  word_in = 32'B_1110_0000_0000_1101_1010_1010_0000_1001;
   #120 word_in = 32'b_0000_0000_0000_0000_1111_1111_1101_0000;
   #160 word_in = 32'b_1101_0101_0000_1111_0000_1010_1010_0000;
   #200 word_in = 32'b_1111_1111_1111_1111_1111_1111_1101_0000;
   #240 word_in = 32'b_0000_1010_1011_0111_1101_0101_0000_1111;
   #280 word_in = 32'b_0000_1101_0000_1111_0000_1101_0000_1010;
        join

////monitor and display the output
initial
      $monitor ($time,, "word_in = %b: index_out = %d", word_in, index_out);
endmodule
```

Bit wave



Output Table


```
#                      0 word_in = 00000000000000000000000000001101: index_out =  x
#                     10 word_in = 00000000000000000000000000001101: index_out =  3
#                     40 word_in = 00001010101000001101000000000000: index_out =  3
#                     50 word_in = 00001010101000001101000000000000: index_out = 15
#                     80 word_in = 11100000000011011010101000001001: index_out = 15
#                     90 word_in = 11100000000011011010101000001001: index_out = 16
#                    120 word_in = 00000000000000001111111111010000: index_out = 16
#                    130 word_in = 00000000000000001111111111010000: index_out =  7
#                    160 word_in = 11010101000011110000101010100000: index_out =  7
#                    170 word_in = 11010101000011110000101010100000: index_out = 31
#                    200 word_in = 11111111111111111111111111010000: index_out = 31
#                    210 word_in = 11111111111111111111111111010000: index_out =  7
#                    240 word_in = 00001010101101111101010100001111: index_out =  7
#                    250 word_in = 00001010101101111101010100001111: index_out = 15
#                    280 word_in = 00001101000011110000110100001010: index_out = 15
#                    290 word_in = 00001101000011110000110100001010: index_out = 11
```

Summary

- The design takes in a 32 bits input as word_in
- Checks for a 4-bit set of 1101 and marks the index of the MSB of the set
- If the set is not found in the first 4 bits the index shifts one bit to the right and looks at the next 4 bits
- Once the set is found the index of the MSB is reported as index_out
- If all 32 bits are checked OR a set of 1101 is not found index_out becomes zero