

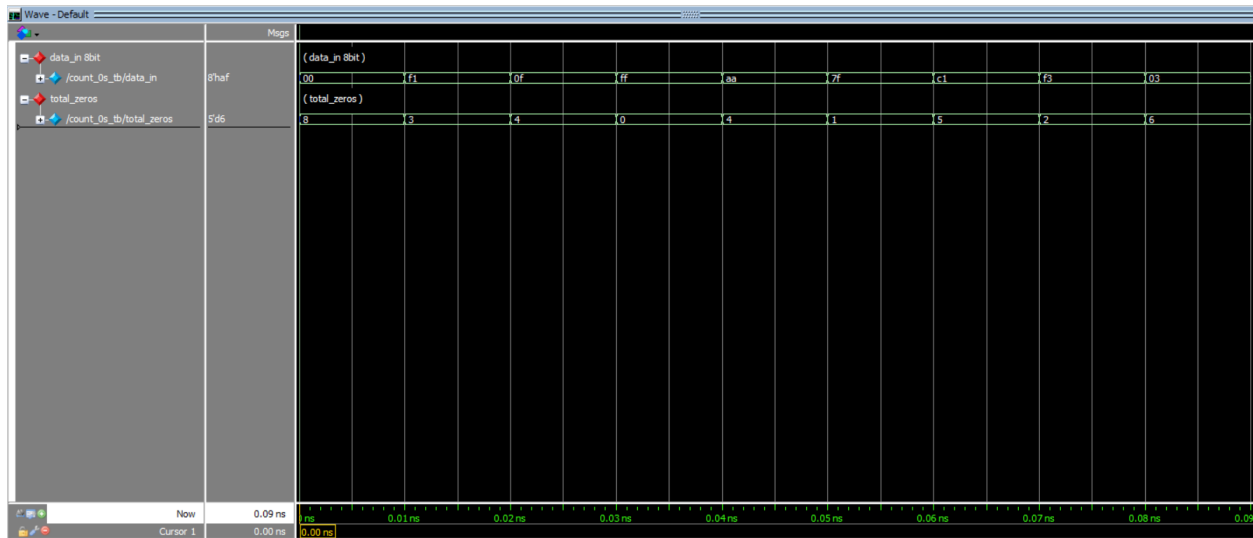
```
1  /*-----*
2  Name Lamin Jammeh
3  Class: EE417 Summer 2024
4  Lesson 06 HW Question 1
5  Group: Ron Kalin/ Lamin Jammeh
6  Project Description: This is the main module, it counts the number of 0s in the input
7  data and displays the results
8  Note that the word_size is Parameterize so it can change without causing an error in the
9  code
10 -----*/
11 /*-----16 bits word_size Trial-----*/
12 module count_0s #(
13     parameter word_size = 8,           //change as you wish
14     parameter count_size = 5           //count size should be at
15     least 16 to accomodate for both 8bit and 16bit word_size
16 )
17     input      [word_size-1: 0] data_in,           //data_in =
18     [15:0]
19     output reg [count_size-1:0] total_zeros
20 );
21 integer index;    //define index as integer to shift through the data_in
22
23 //define an always block and write the conditions for the output or behavior of the system
24 always @ (data_in) //look for change in data_in
25     begin: count_zeros //name the begin block as count_ones
26         total_zeros = 0; //start total_zeros at 0
27         for (index = 0; index < word_size; index = index + 1) //start index from
28             0 to word_size and increment by 1
29             begin
30                 if (data_in[index] == 0) //check each
31                     index of data for 0s
32                     begin
33                         total_zeros = total_zeros + 1;
34                     //increment count by 1 once a 0 is encounter at an index in data_in
35                     end
36                 end
37             end
38         end
39     end
40 endmodule
```

```

1  /*-----
2  Name Lamin Jammeh
3  Class: EE417 Summer 2024
4  Lesson 06 HW Question 1
5  Group: Ron Kalin/ Lamin Jammeh
6  Project Description: This is the test-bench for the count_0s code
7  -----*/
8
9  /*-----16 bits word_size Trial-----*/
10 module count_0s_tb ();
11
12 //define the parameters, registes and wires
13 parameter word_size = 8; //can be change to any
14 desired word_size
15 parameter count_size = 5;
16 reg [word_size -1:0] data_in;
17 wire [count_size -1:0] total_zeros;
18
19 //define the unit under test UUT
20 count_0s UUT (data_in, total_zeros);
21
22 //simulate different data_in and observe the outputs to validate the design
23
24 //-----16-bit data_in word_size-----//
25 //initial
26 // begin
27 // #0 data_in = 16'b0000_0000_0000_0000;
28 // #10 data_in = 16'b1111_0000_0000_0001;
29 // #10 data_in = 16'b1111_0111_1111_1111;
30 // #10 data_in = 16'b1010_1010_1000_1110;
31 // #10 data_in = 16'b0111_0000_0011_1111;
32 // #10 data_in = 16'b0011_1111_0011_1001;
33 // #10 data_in = 16'b0000_0011_0000_0011;
34 // #10 data_in = 16'b1010_1111_1101_0011;
35 // end
36
37 //-----8-bit data_in word_size-----//
38 initial
39 begin
40 #0 data_in = 8'b0000_0000;
41 #10 data_in = 8'b1111_0001;
42 #10 data_in = 8'b0000_1111;
43 #10 data_in = 8'b1111_1111;
44 #10 data_in = 8'b1010_1010;
45 #10 data_in = 8'b0111_1111;
46 #10 data_in = 8'b1100_0001;
47 #10 data_in = 8'b1111_0011;
48 #10 data_in = 8'b0000_0011;
49 #10 data_in = 8'b1010_1111;
50 end
51 //monitor the results
52 initial
53 begin
54 $monitor ($time,, "data_in = %b: total_zeros = %d", data_in, total_zeros);
55 end
56 endmodule

```

Bit wave



Output table

```
#          0 data_in = 00000000: total_zeros = 8
#          10 data_in = 11110001: total_zeros = 3
#          20 data_in = 00001111: total_zeros = 4
#          30 data_in = 11111111: total_zeros = 0
#          40 data_in = 10101010: total_zeros = 4
#          50 data_in = 01111111: total_zeros = 1
#          60 data_in = 11000001: total_zeros = 5
#          70 data_in = 11110011: total_zeros = 2
#          80 data_in = 00000011: total_zeros = 6
#          90 data_in = 10101111: total_zeros = 2
```

Summary for 8bit word_size

- This design takes in 8-bit word_size as data_in and checks the number of 0s in the data.
- The total number of 0s in the data_in is reported in a register called total_zeros