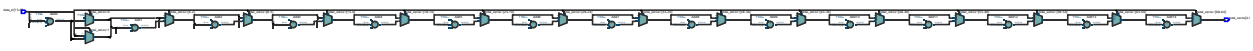
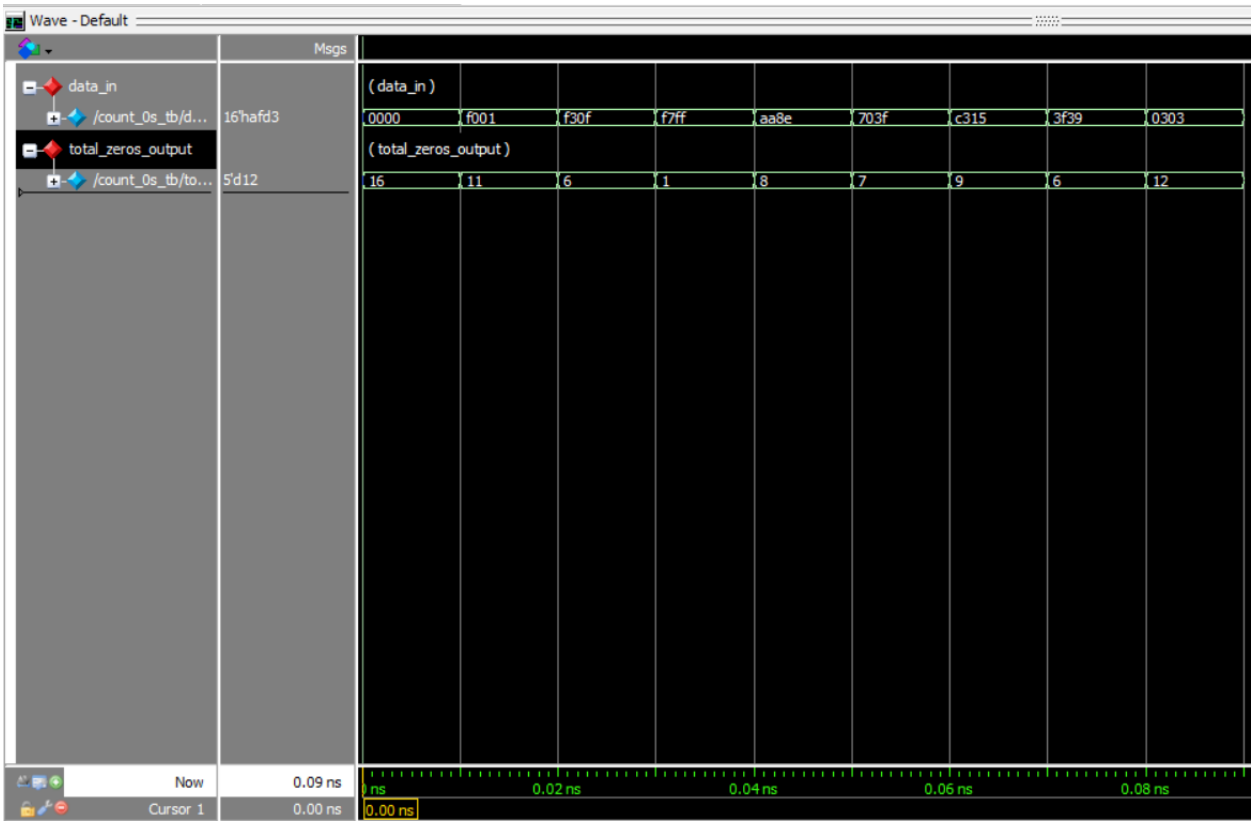
```
1  /*-------------------------------------------------------------------------------
2  Name Lamin Jammeh
3  CLass: EE417 Summer 2024
4  Lesson 06 HW Question 1
5  Group: Ron Kalin/ Lamin Jammeh
6  Project Description: This is the main module, it counts the number of 0s in the input
7  data and displays the results
8  Note that the word_size is Parameterize so it can change without causing an error in the
   code
9  -------------------------------------------------------------------------------*/
10
11 /*----------16 bits Word_size Trial--------------*/
12 module count_0s #(
13                         parameter   word_size  = 16,           //change as you wish
14                         parameter   count_size = 5             //count size should be at
   least 16 to accomodate for both 8bit and 16bit word_size
15                         )
16                        (
17                         input    [word_size -1: 0] data_in,                  //data_in =
   [15:0]
18                         output   reg   [count_size -1:0] total_zeros
19                         );
20 integer  index;     //define index as integer to shift through the data_in
21
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 begine 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
   0 to word_size and increment by 1
28          begin
29             if (data_in[index] == 0)                                    //check each
   index of data for 0s
30                begin
31                total_zeros = total_zeros + 1;
   //increment count by 1 once a 0 is encounter at an index in data_in
32                end
33          end
34    end
35 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  = 16;              //can be change to any
   desirded word_size
14 parameter                       count_size = 5;
15 reg         [word_size -1:0]     data_in;
16 wire        [count_size -1:0]    total_zeros;
17
18 //define the unit under test UUT
19 count_0s UUT      (data_in, total_zeros);
20
21 //simulate different data_in and observe the outputs to validate the design
22
23 //-------------16-bit data_in word_size-------------//
24 initial
25    begin
26       #0   data_in = 16'b0000_0000_0000_0000;
27       #10  data_in = 16'b1111_0000_0000_0001;
28       #10  data_in = 16'b1111_0111_1111_1111;
29       #10  data_in = 16'b1010_1010_1000_1110;
30       #10  data_in = 16'b0111_0000_0011_1111;
31       #10  data_in = 16'b0011_1111_0011_1001;
32       #10  data_in = 16'b0000_0011_0000_0011;
33       #10  data_in = 16'b1010_1111_1101_0011;
34    end
35
36 //-------------8-bit data_in word_size-------------//
37 //initial
38 // begin
39 //    #0  data_in = 8'b0000_0000;
40 //    #10  data_in = 8'b1111_0001;
41 //    #10  data_in = 8'b0000_1111;
42 //    #10  data_in = 8'b1111_1111;
43 //    #10  data_in = 8'b1010_1010;
44 //    #10  data_in = 8'b0111_1111;
45 //    #10  data_in = 8'b1100_0001;
46 //    #10  data_in = 8'b1111_0011;
47 //    #10  data_in = 8'b0000_0011;
48 //    #10  data_in = 8'b1010_1111;
49 // end
50 //monitor the results
51 initial
52    begin
53       $monitor ($time,, "data_in = %b: total_zeros = %d", data_in, total_zeros);
54    end
55 endmodule
```

Bit wave



Output table

```
#                  0 data_in = 0000000000000000: total_zeros = 16
#                 10 data_in = 1111000000000001: total_zeros = 11
#                 20 data_in = 1111001100001111: total_zeros =  6
#                 30 data_in = 1111011111111111: total_zeros =  1
#                 40 data_in = 1010101010001110: total_zeros =  8
#                 50 data_in = 0111000000111111: total_zeros =  7
#                 60 data_in = 1100001100010101: total_zeros =  9
#                 70 data_in = 0011111100111001: total_zeros =  6
#                 80 data_in = 0000001100000011: total_zeros = 12
#                 90 data_in = 1010111111010011: total_zeros =  5
```

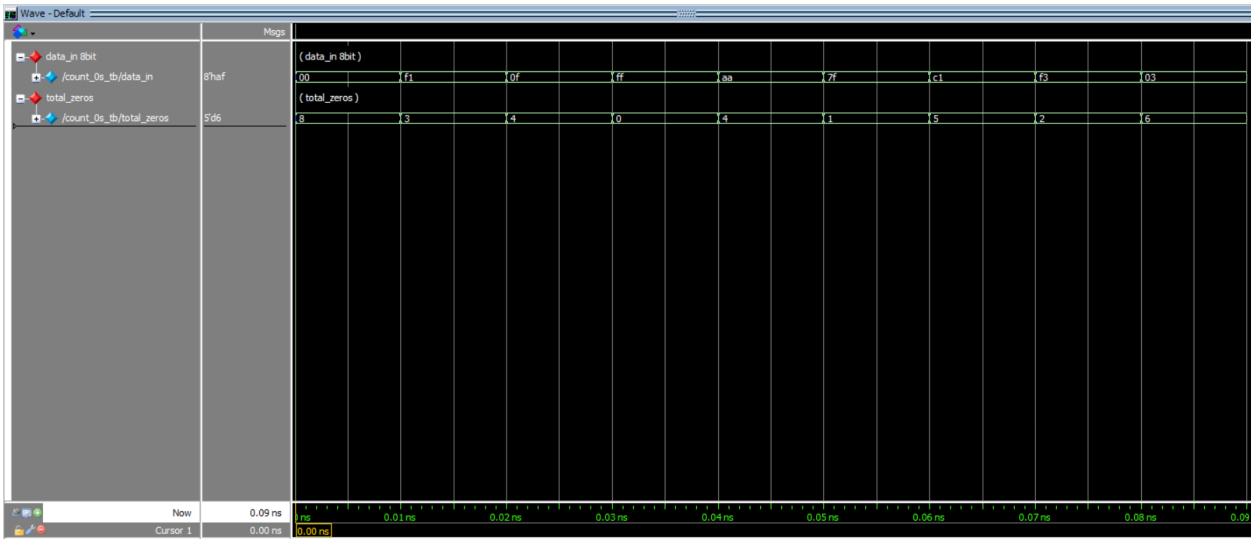
Summary for 16bit word_size

- This design takes in 16-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

```
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
   code
9  -------------------------------------------------------------------------------*/
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
   least 16 to accomodate for both 8bit and 16bit word_size
15                         )
16                        (
17                         input    [word_size -1: 0] data_in,                  //data_in =
   [15:0]
18                         output   reg   [count_size -1:0] total_zeros
19                         );
20 integer  index;     //define index as integer to shift through the data_in
21
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 begine 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
   0 to word_size and increment by 1
28          begin
29             if (data_in[index] == 0)                                    //check each
   index of data for 0s
30                begin
31                total_zeros = total_zeros + 1;
   //increment count by 1 once a 0 is encounter at an index in data_in
32                end
33          end
34    end
35 endmodule
```

```
/*-------------------------------------------------------------------------------
Name Lamin Jammeh
CLass: EE417 Summer 2024
Lesson 06 HW Question 1
Group: Ron Kalin/ Lamin Jammeh
Project Description: This is the test-bench for the count_0s code
-------------------------------------------------------------------------------*/

/*----------16 bits Word_size Trial--------------*/
module count_0s_tb ();

//define the parameters, registes and wires
parameter                       word_size  = 8;              //can be change to any desirded word_size
parameter                       count_size = 5;
reg         [word_size -1:0]    data_in;
wire        [count_size -1:0]   total_zeros;

//define the unit under test UUT
count_0s UUT      (data_in, total_zeros);

//simulate different data_in and observe the outputs to validate the design

//-------------16-bit data_in word_size-------------//
//initial
// begin
//    #0   data_in = 16'b0000_0000_0000_0000;
//    #10  data_in = 16'b1111_0000_0000_0001;
//    #10  data_in = 16'b1111_0111_1111_1111;
//    #10  data_in = 16'b1010_1010_1000_1110;
//    #10  data_in = 16'b0111_0000_0011_1111;
//    #10  data_in = 16'b0011_1111_0011_1001;
//    #10  data_in = 16'b0000_0011_0000_0011;
//    #10  data_in = 16'b1010_1111_1101_0011;
// end

//-------------8-bit data_in word_size-------------//
initial
   begin
      #0  data_in = 8'b0000_0000;
      #10  data_in = 8'b1111_0001;
      #10  data_in = 8'b0000_1111;
      #10  data_in = 8'b1111_1111;
      #10  data_in = 8'b1010_1010;
      #10  data_in = 8'b0111_1111;
      #10  data_in = 8'b1100_0001;
      #10  data_in = 8'b1111_0011;
      #10  data_in = 8'b0000_0011;
      #10  data_in = 8'b1010_1111;
   end
//monitor the results
initial
   begin
      $monitor ($time,, "data_in = %b: total_zeros = %d", data_in, total_zeros);
   end
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