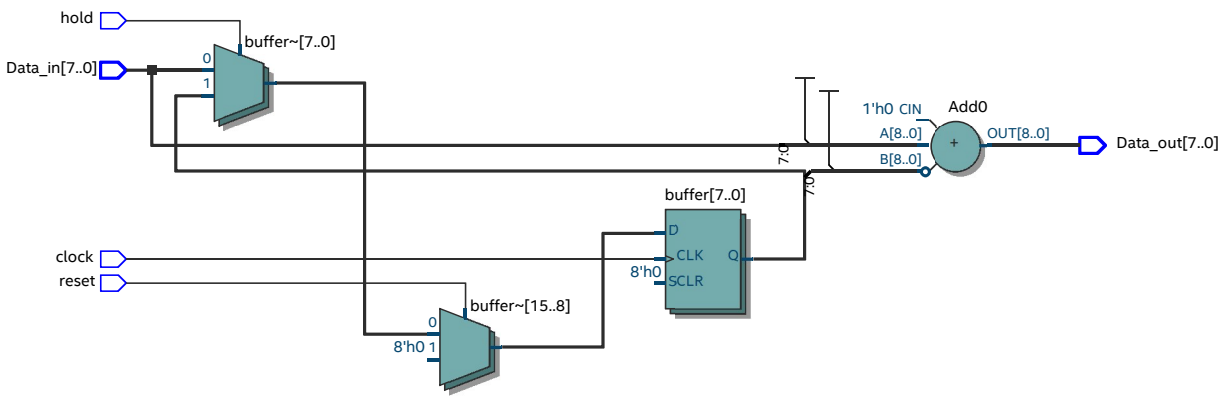
```
1   /*-------------------------------------------------------------------------------
2   Name Lamin Jammeh
3   CLass: EE417 Summer 2024
4   Lesson 09 HW Question 2
5   Group: Ron Kalin/ Lamin Jammeh
6   Project Description: This module is Differentiator, it finds the difference between
7   adjacent sample
8   -------------------------------------------------------------------------------*/
9
10  module Differentiator #(parameter word_size = 8)
11                         (
12                          output  [word_size-1:0] Data_out,
13                          input   [word_size-1:0] Data_in,
14                          input                   hold, clock, reset
15                         );
16
17  // Internal registers and wires
18  reg [word_size-1:0] buffer; // Internal register for buffer
19
20  // Assign the formula for Data_out
21  assign Data_out = Data_in - buffer;
22
23  always @(posedge clock)
24     begin
25        if (reset)
26
    /*-------------------------------------------------------------------------------------
    --------
27           @ high reset the following will occur
28           ** buffer goes to zero
29           ** Data_in will keep its value since it is an external input but the buffer will
    remain zero
30           ** Data_out will still be [Data_in - buffer] but since buffer is zero therefore
    Data_out will be Data_in
31           ** This Means that no filtering is occurring at high reset
32
    -------------------------------------------------------------------------------------
    --------*/
33
34            buffer <= 0; // When reset is high, and clock is rising, buffer <= 0
35        else if (hold)
36           /*-------------------------------------------------
37           @ low rest and hold = 1 the following will happen
38           **buffer Keeps it current value
39           ** Data_in is not transfered to buffer
40           ** Data_out is still equal to [Data_in - buffer] but
41           ** therefore @ hold = 1 the buffer is not updated
42           -------------------------------------------------*/
43            buffer <= buffer; // When hold is high, reset is low, and clock is rising, buffer
    remains unchanged
44        else
45            /*-------------------------------------------------------------
46            @ hold = 0 and reset = 0 the following will occur
47            ** buffer will store the previous Data_in value
48            ** Data_out will be (Data_in - buffer)
49            ** therefore the buffer is update when hold is low and reset is low
50            -------------------------------------------------------------*/
51            buffer <= Data_in; // When hold is low, reset is low, and clock is rising, buffer
    <= Data_in
52     end
53
54  endmodule
```


hold
buffer~[7..0]
Data_in[7..0]
0
1
1'h0 CIN
Add0
A[8..0]
B[8..0]
OUT[8..0]
Data_out[7..0]
7:0
7:0
buffer[7..0]
D
CLK
Q
8'h0
SCLR
clock
reset
buffer~[15..8]
0
8'h0 1

```
1   /*-------------------------------------------------------------------------------
2   Name Lamin Jammeh
3   CLass: EE417 Summer 2024
4   Lesson 09 HW Question 2
5   Group: Ron Kalin/ Lamin Jammeh
6   Project Description: Test-Bench for the Differentiator Module
7   -------------------------------------------------------------------------------*/
8   module Differentiator_tb ();
9
10  // Define the registers and wires for the signals to monitor
11  reg         clock, reset, hold;
12  reg  [7:0]  Data_in;
13  wire [7:0]  Data_out;
14
15  //define the internal probes in the testbench for buffer
16  wire [7:0] buffer; // New wire for observing buffer
17
18  // Instantiate the unit under test (UUT)
19  Differentiator #(8) UUT (
20                           .Data_out(Data_out),
21                           .Data_in(Data_in),
22                           .hold(hold),
23                           .clock(clock),
24                           .reset(reset)
25                          );
26
27  // Assign buffer in testbench to buffer in the Unit under test
28  assign buffer = UUT.buffer;
29
30
31  // Instantiate the clock cycle
32  always
33     begin
34        clock = 0;
35        forever #5 clock = ~clock;
36     end
37
38  // Update Data_in at negative edge of the clock
39  always @(posedge clock)
40     begin
41        Data_in = Data_in + 1; // Change this as needed
42     end
43
44  initial
45     begin
46        // Initialize all the inputs
47        Data_in = 8'sd5;
48        reset = 0;
49        hold = 0;
50
51        // Create a test scenario with the reset function
52        #10 reset = 1;
53
54        // Turn off reset and create a test scenario to check the buffer
55        #10 reset = 0;
56        Data_in = 8'sd15;
57        #10 hold = 1;
58
59        // Turn off hold and try different Data_in values to see Data_out
60        #10 hold = 0;
61        Data_in = -8'sd18;
62        #10 Data_in = 8'sd23;
63
64        // Reset to bring Data_out back to zero
65        #10 reset = 1;
66        #10 reset = 0;
67
68        // Final test case to ensure differentiation works correctly after reset
69        Data_in = 8'sd12;
```

```
70          #10 Data_in = -8'sd22;
71          #10 Data_in = 8'sd13;
72          #10 Data_in = 8'sd7;
73
74          //stop the simulation
75          #10;
76          $stop;
77       end
78
79    // Display the results
80    always @(posedge clock)
81       begin
82          $display("@ Time = %t      Data_in = %d      Buffer = %d    Data_out = %d", $time,
      Data_in, buffer, Data_out);
83       end
84
85    endmodule
86
```

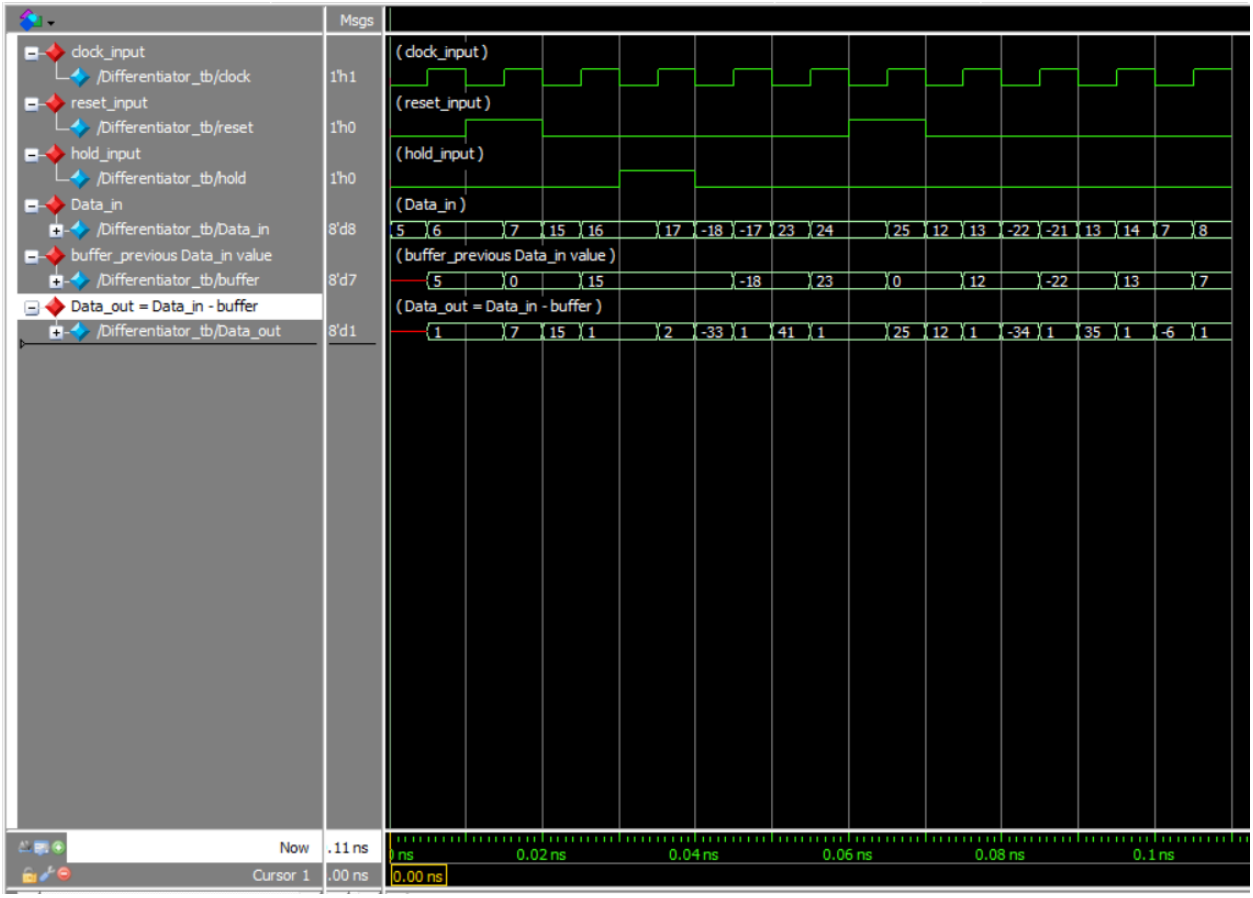
Output table

```
# @ Time =                    5  Data_in =   6  Buffer =   x  Data_out =   x
# @ Time =                   15  Data_in =   7  Buffer =   5  Data_out =   1
# @ Time =                   25  Data_in =  16  Buffer =   0  Data_out =  15
# @ Time =                   35  Data_in =  17  Buffer =  15  Data_out =   1
# @ Time =                   45  Data_in = 239  Buffer =  15  Data_out = 223
# @ Time =                   55  Data_in =  24  Buffer = 238  Data_out =  41
# @ Time =                   65  Data_in =  25  Buffer =  23  Data_out =   1
# @ Time =                   75  Data_in =  13  Buffer =   0  Data_out =  12
# @ Time =                   85  Data_in = 235  Buffer =  12  Data_out = 222
# @ Time =                   95  Data_in =  14  Buffer = 234  Data_out =  35
# @ Time =                  105  Data_in =   8  Buffer =  13  Data_out = 250
```

Note that some of the Data_in values and buffer values are missing. Below is the Bitwave show all Data_in values and buffer values. The table above is a little misleading because of the equation below

- buffer = Previous Data_in
- Data_out = Data_in – previous Data_in

Bitwave



Summary:

@ high reset the following will occur

- buffer goes to zero
- Data_in will keep its value since it is an external input but the buffer will remain zero
- Data_out will still be [Data_in - buffer] but since buffer is zero therefore Data_out will be Data_in
- This Means that no filtering is occurring at high reset

@ low rest and hold = 1 the following will happen

- buffer Keeps it current value
- Data_in is not transferred to buffer
- Data_out is still equal to [Data_in - buffer] but
- therefore @ hold = 1 the buffer is not updated

@ hold = 0 and reset = 0 the following will occur

- buffer will store the previous Data_in value
- Data_out will be (Data_in - buffer)
- therefore, the buffer is update when hold is low and reset is low