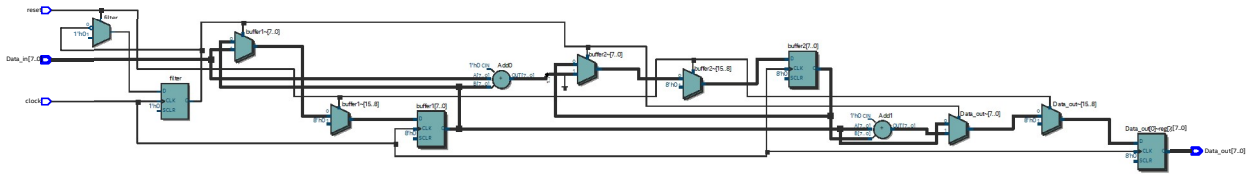
```
/*-------------------------------------------------------------------------------
Name Lamin Jammeh
CLass: EE417 Summer 2024
Lesson 09 HW Question 3
Group: Ron Kalin/ Lamin Jammeh
Project Description: Interpolator Filter, the filter takes Data_in and sending through
bufffer1 and buffer2 to double the rate at the output
-------------------------------------------------------------------------------*/

module Linear_Interpolator #(parameter word_size = 8)
                           (
                            output reg [word_size-1:0] Data_out,
                            input  [word_size-1:0]     Data_in,
                            input                      clock, reset
                            );

// internal registers and wires
reg [word_size-1:0] buffer1; // internal register for storing Data_in
reg [word_size-1:0] buffer2; // internal register for storing average
reg filter;

always @ (posedge clock)
   begin
      if (reset)
      /*-----------------------------------------------
        @ reset everything goes to zero exception Data_in
        since Data_in is coming from an external source
        -----------------------------------------------*/
         begin
            buffer1  <= 0;
            buffer2  <= 0;
            filter   <= 0;
            Data_out <= 0;
         end
      else
         begin
         /*-----------------------------------------------
         @ reset low the internal filter truns on and the
         filtering logic is carry out below
         -----------------------------------------------*/
         filter <= ~filter;
            if (filter)
            /*-------------------------------------------------------------------
             when the filter is high perform the following
             ** set buffer1 to store the Data_in values
             ** set buffer2 to interpolate (average) current Data_in with buffer1
             ** set Data_out to be combination of buffer1 and buffer2
             -------------------------------------------------------------------*/
               begin
                  buffer1  <= Data_in;
                  buffer2  <= (Data_in + buffer1) >> 1;
                  Data_out <= buffer1 + buffer2;
               end
            else
            /*-----------------------------------------------------
             when the filter is low or not active
             ** set Data_out to be the current value of buffer1
             -----------------------------------------------------*/
               begin
                  Data_out <= buffer1;
               end
            /*-----------------------------------------------------
            the above logic shows that Data_in is always filter before
            getting to Data_out
            Data_out only reads from buffer1 and buffer2
            -----------------------------------------------------*/
         end
   end

```

```
70    endmodule
71
```



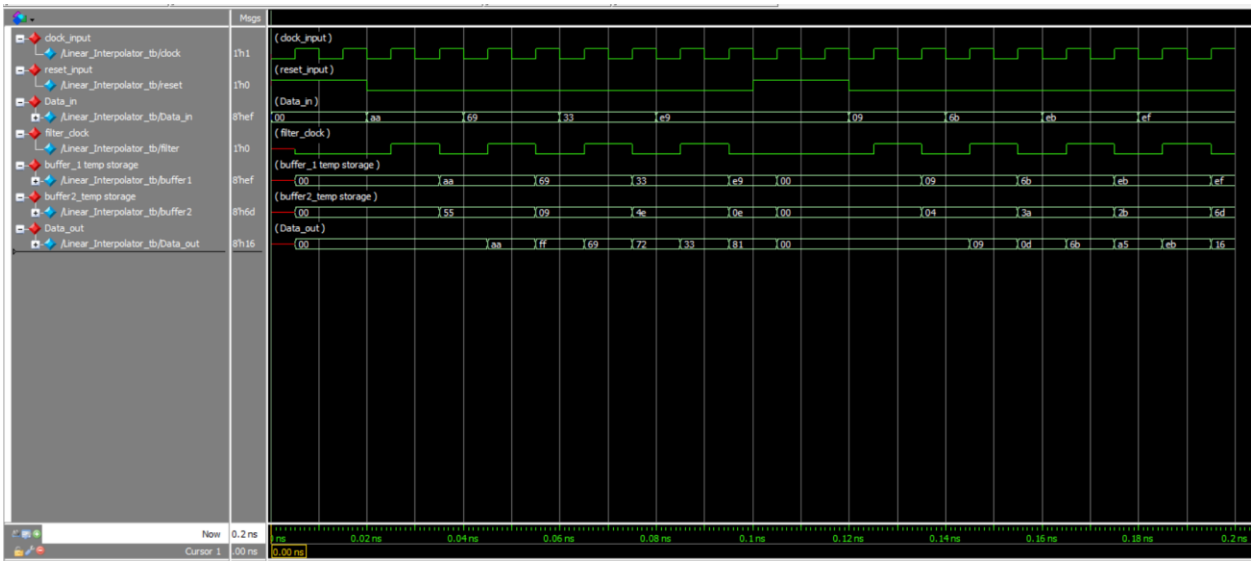
```
1  /*-------------------------------------------------------
2  Name Lamin Jammeh
3  CLass: EE417 Summer 2024
4  Lesson 09 HW Question 3
5  Group: Ron Kalin/ Lamin Jammeh
6  Project Description: TestBench for Interpolator Filter
7  -------------------------------------------------------*/
8  module Linear_Interpolator_tb ();
9
10 //define the registers and wire for the signals to monitor
11 reg         clock, reset;
12 reg   [7:0] Data_in;
13 wire  [7:0] Data_out;
14
15 //define the internal probes in the testbench for buffer1 and buffer2
16 wire       filter;  // New wire for observing filter
17 wire [7:0] buffer1; // New wire for observing buffer1
18 wire [7:0] buffer2; // New wire for observing buffer2
19
20 //Instantiate the unit under test UUT
21 Linear_Interpolator #(8)   UUT (
22                           .Data_out(Data_out),
23                           .Data_in(Data_in),
24                           .clock(clock),
25                           .reset(reset)
26                           );
27
28 // Assign buffer1 and buffer2 in the testbench to buffer1 and buffer2 in the Unit under test
29 assign filter  = UUT.filter;
30 assign buffer1 = UUT.buffer1;
31 assign buffer2 = UUT.buffer2;
32
33 //instantiate the clock cycle
34 initial
35    begin
36       clock = 0;
37       forever #5 clock = ~clock;
38    end
39
40 initial
41    begin
42       //Initialize all the inputs
43       reset = 1;
44       Data_in = 8'b0;
45       #20 reset = 0;
46
47       //multiple Data_in samples and observe the buffer and Data_out
48           Data_in = 8'b10101010;
49       #20 Data_in = 8'b01101001;
50       #20 Data_in = 8'b00110011;
51       #20 Data_in = 8'b11101001;
52
53       //take everything back Data_out back to zero by setting reset to high
54       #20  reset = 1;
55       #20  reset = 0;
56
57       // Final test case to ensure the interpolator works correctly after reset
58           Data_in = 8'b00001001;
59       #20 Data_in = 8'b01101011;
60       #20 Data_in = 8'b11101011;
61       #20 Data_in = 8'b11101111;
62
63       // stop the simulation
64       #20;
65       $stop;
66    end
67
68 // Display the results
69 always @(posedge clock)
```

```
70      begin
71         $display("@ Time = %t   Data_in = %d   Filter = %d    Buffer1 = %d   Buffer2 = %d
    Data_out = %d", $time, Data_in, filter, buffer1, buffer2, Data_out);
72      end
73
74   endmodule
```

Output table

```
# @ Time =                5 Data_in =   0 Filter = x  Buffer1 =   x  Buffer2 =   x  Data_out =   x
# @ Time =               15 Data_in =   0 Filter = 0  Buffer1 =   0  Buffer2 =   0  Data_out =   0
# @ Time =               25 Data_in = 170 Filter = 0  Buffer1 =   0  Buffer2 =   0  Data_out =   0
# @ Time =               35 Data_in = 170 Filter = 1  Buffer1 =   0  Buffer2 =   0  Data_out =   0
# @ Time =               45 Data_in = 105 Filter = 0  Buffer1 = 170  Buffer2 =  85  Data_out =   0
# @ Time =               55 Data_in = 105 Filter = 1  Buffer1 = 170  Buffer2 =  85  Data_out = 170
# @ Time =               65 Data_in =  51 Filter = 0  Buffer1 = 105  Buffer2 =   9  Data_out = 255
# @ Time =               75 Data_in =  51 Filter = 1  Buffer1 = 105  Buffer2 =   9  Data_out = 105
# @ Time =               85 Data_in = 233 Filter = 0  Buffer1 =  51  Buffer2 =  78  Data_out = 114
# @ Time =               95 Data_in = 233 Filter = 1  Buffer1 =  51  Buffer2 =  78  Data_out =  51
# @ Time =              105 Data_in = 233 Filter = 0  Buffer1 = 233  Buffer2 =  14  Data_out = 129
# @ Time =              115 Data_in = 233 Filter = 0  Buffer1 =   0  Buffer2 =   0  Data_out =   0
# @ Time =              125 Data_in =   9 Filter = 0  Buffer1 =   0  Buffer2 =   0  Data_out =   0
# @ Time =              135 Data_in =   9 Filter = 1  Buffer1 =   0  Buffer2 =   0  Data_out =   0
# @ Time =              145 Data_in = 107 Filter = 0  Buffer1 =   9  Buffer2 =   4  Data_out =   0
# @ Time =              155 Data_in = 107 Filter = 1  Buffer1 =   9  Buffer2 =   4  Data_out =   9
# @ Time =              165 Data_in = 235 Filter = 0  Buffer1 = 107  Buffer2 =  58  Data_out =  13
# @ Time =              175 Data_in = 235 Filter = 1  Buffer1 = 107  Buffer2 =  58  Data_out = 107
# @ Time =              185 Data_in = 239 Filter = 0  Buffer1 = 235  Buffer2 =  43  Data_out = 165
# @ Time =              195 Data_in = 239 Filter = 1  Buffer1 = 235  Buffer2 =  43  Data_out = 235
```

Bitwave



Summary:

- @ reset everything goes to zero exception Data_in, since Data_in is coming from an external source
- when the filter is high perform the following
    - set buffer1 to store the Data_in values
    - set buffer2 to interpolate (average) current Data_in with buffer1
    - set Data_out to be combination of buffer1 and buffer2
- when the filter is low or not active
    - set Data_out to be the current value of buffer1

- The above logic shows that Data_in is always filtered before getting to Data_out. Data_out only reads from buffer1 and buffer2