```
/*-------------------------------------------------------------------------------
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
```