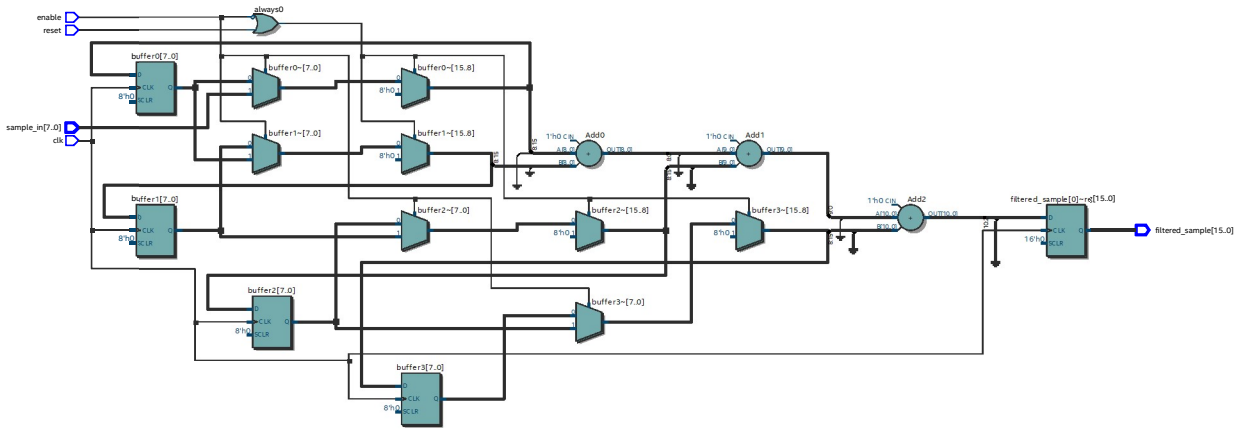
```
1  // ee417 lesson 9 Assignment 1 L9A1
2  // Name: Ron Kalin, Date: 07-10-24  Group: Kalin/Jammeh
3  // Design: moving average FIR filter of overlapping windows of 4 input samples
4  // FIR order will be a power of 2, paramterized with word_size
5  // top level module
6  module moving_average_filter #(parameter word_size=8, order=4, n=2)(
7      output reg [2*word_size-1:0] filtered_sample,
8      input      [word_size-1:0]   sample_in,
9      input                        enable, clk, reset
10 );
11     // Coefficient values for a 4-tap moving average filter
12     //reg [word_size-1:0] coeff [0:order] = {16'h1000, 16'h1000, 16'h1000, 16'h1000};
13     reg [word_size-1:0] coeff0 = 4'd1; //coefficients of one were selected
14     reg [word_size-1:0] coeff1 = 4'd1; //because for moving average they aren't needed
15     reg [word_size-1:0] coeff2 = 4'd1;
16     reg [word_size-1:0] coeff3 = 4'd1;
17     reg [word_size-1:0] tap_outputs [0:3];
18
19     // Circular buffer to store input samples
20     reg [word_size-1:0] buffer0, buffer1, buffer2, buffer3;
21     reg [word_size-1:0] buffer [0:order-1];
22     always @(posedge clk) begin
23         if (reset  || ~enable) begin
24             buffer0 = 0;
25             buffer1 = 0;
26             buffer2 = 0;
27             buffer3 = 0;
28         end else if (enable) begin
29             buffer3 = buffer2;
30             buffer2 = buffer1;
31             buffer1 = buffer0;
32             buffer0 = sample_in;
33         end
34       // Multipliers and accumulator
35       tap_outputs[0] = buffer0 * coeff0; //coeff[0];
36       tap_outputs[1] = buffer1 * coeff1; //coeff[1];
37       tap_outputs[2] = buffer2 * coeff2; //coeff[2];
38       tap_outputs[3] = buffer3 * coeff3; //coeff[3];
39
40       filtered_sample = (buffer0 + buffer1 + buffer2 + buffer3); // >> 2;
41     //filtered_sample = buffer[0] + buffer[1] + buffer[2] + buffer[3];
42     //filtered_sample <= (tap_outputs[0] + tap_outputs[1] + tap_outputs[2] + tap_outputs[3])
43     // Right shift by 2 bits to approximate division by 4
44       filtered_sample = (filtered_sample >> n); //shift n places to divide by 2^n
45       //shift n places requires 2^n samples (buffers and coefficients)
46     end
47 endmodule
48
49
```

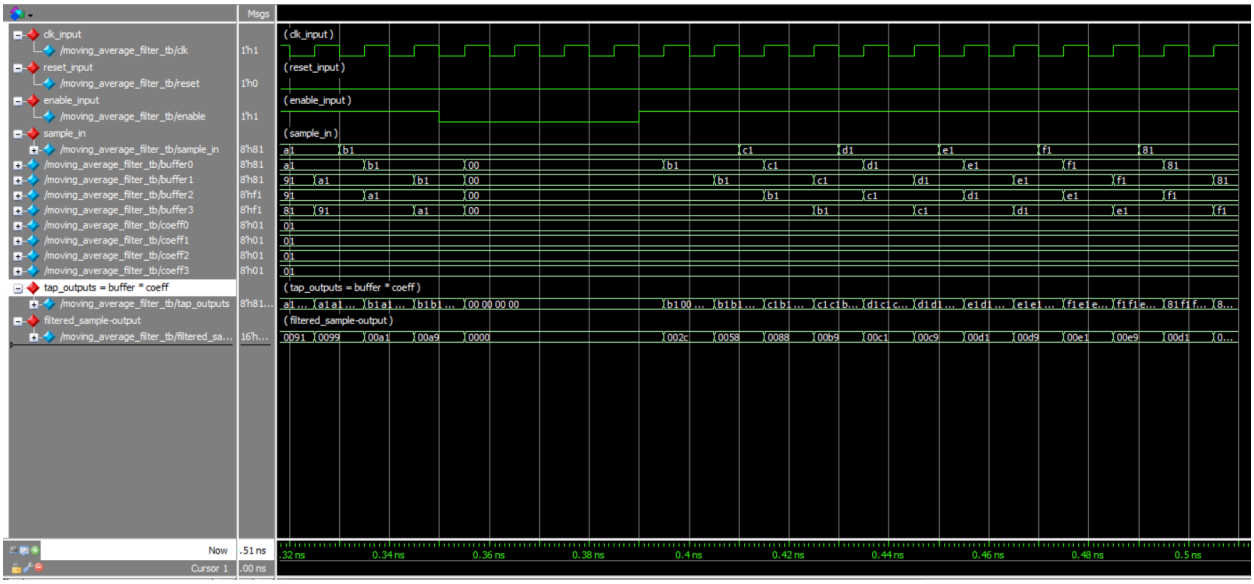

enable
reset
sample_in[7..0]
clk
always0
buffer0[7..0]
buffer0~[7..0]
buffer0~[15..8]
buffer1~[7..0]
buffer1~[15..8]
buffer1[7..0]
buffer2~[7..0]
buffer2~[15..8]
buffer2[7..0]
buffer3~[7..0]
buffer3[7..0]
buffer3~[15..8]
Add0
Add1
Add2
filtered_sample[0]~reg[15..0]
filtered_sample[15..0]

```
/*-------------------------------------------------------------------------------
Name Ron Kalin
Class: EE417 Summer 2024
Lesson 09 HW Question 01
Group: Ron Kalin/ Lamin Jammeh
Project Description: test-bench for moving average FIR filter
-------------------------------------------------------------------------------*/
module moving_average_filter_tb();
    // Parameters
    parameter DATA_WIDTH =  8;
    parameter FILT_LENGTH = 4;

    // Signals
    reg clk, reset, enable;
    reg [DATA_WIDTH-1:0] sample_in;
    wire [2*DATA_WIDTH-1:0] filtered_sample;

    //wires monitor internal variables
    wire [DATA_WIDTH-1:0] buffer0, buffer1, buffer2, buffer3;
    wire [DATA_WIDTH-1:0] coeff0, coeff1,coeff2,coeff3;
    wire [DATA_WIDTH-1:0] tap_outputs [0:3];

    // Instantiate the moving_average_filter module
    moving_average_filter UUT (
        .filtered_sample(filtered_sample),
        .sample_in(sample_in),
        .enable(enable),
        .clk(clk),
        .reset(reset)
    );
    assign buffer0 =UUT.buffer0;
    assign buffer1 =UUT.buffer1;
    assign buffer2 =UUT.buffer2;
    assign buffer3 =UUT.buffer3;
    assign coeff0  =UUT.coeff0;
    assign coeff1  =UUT.coeff1;
    assign coeff2  =UUT.coeff2;
    assign coeff3  =UUT.coeff3;
    assign tap_outputs[0]=UUT.tap_outputs[0];
    assign tap_outputs[1]=UUT.tap_outputs[1];
    assign tap_outputs[2]=UUT.tap_outputs[2];
    assign tap_outputs[3]=UUT.tap_outputs[3];

    // Clock generation
    always #5 clk = ~clk;

    // Test stimulus
    initial begin
        clk = 0;
        reset = 1;
        #10 reset = 0; enable =1; // Release reset, enable
        #20 sample_in = 8'b0000_0000;  // input sample of 0
        #20 sample_in = 8'b0000_0000;  // input sample of 0
        #20 sample_in = 8'b0000_0100;  // input sample of 4
        #20 sample_in = 8'b0000_0000;  // input sample of 0
        #20; // Wait for some cycles

        // Applying additional test vectors
        #20 sample_in = 8'b0001_0001;  // input sample of 17
        #20 sample_in = 8'b0010_0001;  // input sample of 33
        #20 sample_in = 8'b0011_0001;  // input sample of 49
        #20 sample_in = 8'b0100_0001;  // input sample of 65
        #20 sample_in = 8'b0101_0001;  // input sample of 81
        #20 sample_in = 8'b0110_0001;  // input sample of 97
        #20 sample_in = 8'b0111_0001;  // input sample of 113
        #20 sample_in = 8'b1000_0001;  // input sample of 129
        #20 sample_in = 8'b1001_0001;  // input sample of 145
        #20 sample_in = 8'b1010_0001;  // input sample of 161
        #20 sample_in = 8'b1011_0001;  // input sample of 177
```

```
70            #20 enable =0;
71            #40 enable =1;
72            #20 sample_in = 8'b1100_0001;  // input sample of 193
73            #20 sample_in = 8'b1101_0001;  // input sample of 209
74            #20 sample_in = 8'b1110_0001;  // input sample of 225
75            #20 sample_in = 8'b1111_0001;  // input sample of 241
76            #20 sample_in = 8'b1000_0001;  // input sample of 129a
77            #20; // Wait for some cycles
78
79            $display("Filtered Output: %h", filtered_sample);
80            //$finish;
81            $stop;
82        end
83
84    // Display the results
85    always @(posedge clk)
86       begin
87          $display("Filtered Output: %h", filtered_sample);
88       end
89
90    endmodule
```

```
# Filtered Output: xxxx
# Filtered Output: 0000
# Filtered Output: Xxxx
# Filtered Output: Xxxx
# Filtered Output: Xxxx
# Filtered Output: Xxxx
# Filtered Output: Xxxx
# Filtered Output: 0000
# Filtered Output: 0001
# Filtered Output: 0002
# Filtered Output: 0002
# Filtered Output: 0002
# Filtered Output: 0001
# Filtered Output: 0000
# Filtered Output: 0004
# Filtered Output: 0008
# Filtered Output: 0010
# Filtered Output: 0019
# Filtered Output: 0021
# Filtered Output: 0029
# Filtered Output: 0031
# Filtered Output: 0039
# Filtered Output: 0041
# Filtered Output: 0049
# Filtered Output: 0051
# Filtered Output: 0059
# Filtered Output: 0061
# Filtered Output: 0069
# Filtered Output: 0071
# Filtered Output: 0079
# Filtered Output: 0081
# Filtered Output: 0089
# Filtered Output: 0091
# Filtered Output: 0099
# Filtered Output: 00a1
# Filtered Output: 00a9
# Filtered Output: 0000
# Filtered Output: 0000
# Filtered Output: 0000
# Filtered Output: 0000
# Filtered Output: 002c
# Filtered Output: 0058
# Filtered Output: 0088
# Filtered Output: 00b9
# Filtered Output: 00c1
# Filtered Output: 00c9
# Filtered Output: 00d1
# Filtered Output: 00d9
# Filtered Output: 00e1
# Filtered Output: 00e9
# Filtered Output: 00d1
# Filtered Output: 00b9

```

Bitwave


Msgs
clk_input
/moving_average_filter_tb/clk
reset_input
/moving_average_filter_tb/reset
enable_input
/moving_average_filter_tb/enable
sample_in
/moving_average_filter_tb/sample_in
/moving_average_filter_tb/buffer0
/moving_average_filter_tb/buffer1
/moving_average_filter_tb/buffer2
/moving_average_filter_tb/buffer3
/moving_average_filter_tb/coeff0
/moving_average_filter_tb/coeff1
/moving_average_filter_tb/coeff2
/moving_average_filter_tb/coeff3
tap_outputs = buffer * coeff
/moving_average_filter_tb/tap_outputs
filtered_sample-output
/moving_average_filter_tb/filtered_sa...
Now
Cursor 1
0.34 ns
0.36 ns
0.38 ns
0.4 ns
0.42 ns
0.44 ns
0.46 ns
0.48 ns
0.5 ns