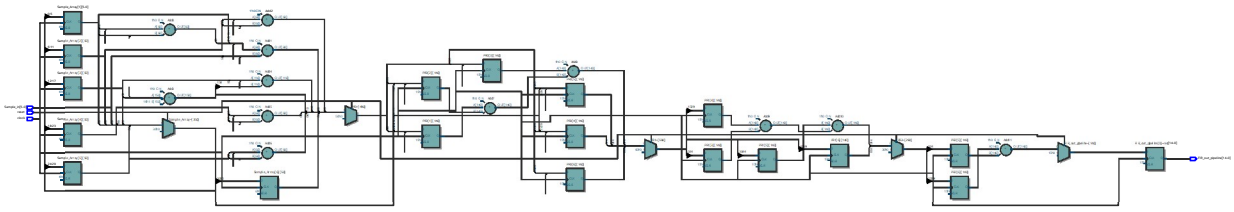
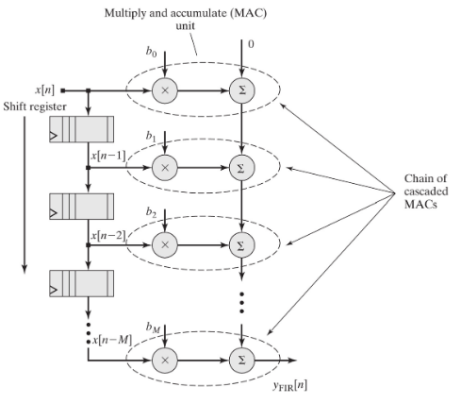
```
1  /*-----------------FIR Pipeline MAC-------------------------------------------------------------------------*/
2  /*-------------------------------------------------------------------------------------------------------------
   ---------
3  Name Lamin Jammeh
4  CLass: EE417 Summer 2024
5  Lesson 10 HW Question 3
6  Group: Ron Kalin/ Lamin Jammeh
7  Project Description: In this module the Pipeline registers are placed at the output of the
   adders
8  note thet Current_adder = Prev_adder + b(n) * Sample_Array[n]
9  -------------------------------------------------------------------------------------------------
   --------*/
10
11 module FIR_Pipeline_MAC (FIR_out_pipeline, Sample_in, clock, reset);
12
13 //define the parameter sets for the design
14 parameter FIR_order        = 6;
15 parameter Sample_size      = 6;                 //maximum sample value is 63
16 parameter weight_size      = 4;                 //maximum value may be 31
17 parameter word_size_out    = 2 * Sample_size + 3;     //log2(2^2 * 2^5 * (order+1))
18 parameter product_size     = Sample_size + weight_size + 3;
19
20 //define output
21 output reg [word_size_out -1:0]     FIR_out_pipeline;
22
23 //define inputs
24 input    [Sample_size -1:0]      Sample_in;
25 input                            clock, reset;
26
27 //define the filter coefficients
28 parameter   b0 = 4'd2;
29 parameter   b1 = 4'd5;
30 parameter   b2 = 4'd9;
31 parameter   b3 = 4'd14;
32 parameter   b4 = 4'd9;
33 parameter   b5 = 4'd5;
34 parameter   b6 = 4'd2;
35
36
37 reg         [Sample_size -1:0]      Sample_Array[1:FIR_order];       //5th coefficient
   multiplied by Data_in
38 integer     k;
39
40 //define PR0 to PR3 as registers
41 reg      [word_size_out -1:0] PR0 [0:FIR_order];
42 reg      [word_size_out -1:0] PR1 [1:FIR_order];
43 reg      [word_size_out -1:0] PR2 [2:FIR_order];
44
45 //define the transition logic
46 always @ (posedge clock)
47    if (reset == 1)
48    /*---------------------------------------
49    if reset is high do the following
50    ****set all Pipeline registers to zero
51    *************IR = 0     Input register or Sample_Array
52    *************PR = 0     Pipeline register or PR
53    *************OR = 0     Output register or FIR_out_pipeline
54    ---------------------------------------*/
55       begin
56       //The input shift register
57          for (k=1; k <= FIR_order; k = k + 1)
58             Sample_Array[k] <= 0;
59
60       //The pipeline register
61          for (k = 0; k <= FIR_order; k = k + 1)
62             PR0[k] <= 0;
63       //The pipeline register
64          for (k = 1; k <= FIR_order; k = k + 1)
```

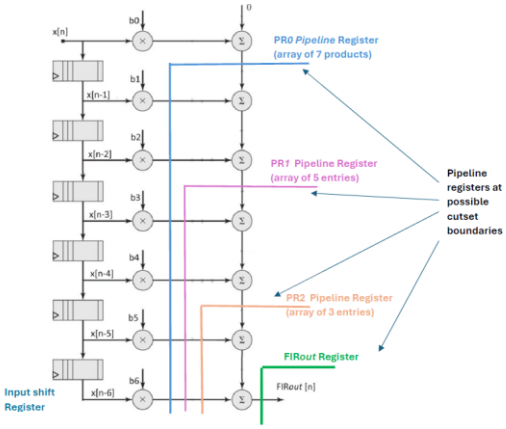
```
65              PR1[k] <= 0;
66
67          //The pipeline register
68             for (k = 2; k <= FIR_order; k = k + 1)
69                PR2[k] <= 0;
70
71          //The outpput register
72                 FIR_out_pipeline <= 0;
73          end
74
75       else
76       /*---------------------------------------
77       if reset is low do the following
78       **************1 => move the Sample in into a cutset (Input register) to reduce idle time
     of the input
79       **************2 => PR = Prev_PR + b(n) * x[n] since the first PR is the at the output of
     the first adder
80       **************3 => add all the products to generate the output
81       ---------------------------------------*/
82          begin
83          //The input shift register
84             Sample_Array[1] <= Sample_in;
85             for (k = 2; k <= FIR_order; k = k + 1)
86                Sample_Array[k] <= Sample_Array[k-1];
87
88          //@ input of add_0 there is no Pipeline Register
89              //PR0[0] <= b0 * Sample_in;              //ignore the other input (c_in = 0) to
     the summation (add_0)
90
91          //@ input of add_1 or output of add_0 there is PR0 and b1*Sample_in[1]
92          //make sure the countour line with pipeline registers is extended to rest of the
     coefficients from b2...bn for data coherency
93              PR0[1] <= b1 * Sample_Array[1] + (b0 * Sample_in);
94              PR0[2] <= b2 * Sample_Array[2];
95              PR0[3] <= b3 * Sample_Array[3];
96              PR0[4] <= b4 * Sample_Array[4];
97              PR0[5] <= b5 * Sample_Array[5];
98              PR0[6] <= b6 * Sample_Array[6];
99
100         //@ input of add_2 or output of add_1 there is PR1 and b2*Sample_in[2]
101             //PR2[2] <= PR0[1] + PR0[2];
102
103         //@ input of add_3 or output of add_2 there is PR2 and b3*Sample_in[3]
104         //make sure the countour line with pipeline registers is extended to rest of the
     coefficients from b4...bn for data coherency
105             PR1[3] <= PR0[3] + (PR0[1] + PR0[2]);
106             PR1[4] <= PR0[4];
107             PR1[5] <= PR0[5];
108             PR1[6] <= PR0[6];
109
110         //@ input of add_4 or output of add_3 there is PR3 and b4*Sample_in[4]
111             //PR4[4] <= PR0[4] + PR1[3];
112
113         //@ input of add_5 or output of add_4 there is PR4 and b5*Sample_in[5]
114         //make sure the countour line with pipeline registers is extended to rest of the
     coefficients from b6...bn for data coherency
115             PR2[5] <= PR1[5] + (PR1[4] + PR1[3]);
116             PR2[6] <= PR1[6];
117
118         //The outpput register
119                FIR_out_pipeline <= PR2[6] + PR2[5];
120         end
121
122   endmodule
```



```
/*-------------------------------------------------------------------------------------------------
---------
Name Lamin Jammeh
CLass: EE417 Summer 2024
Lesson 10 HW Question 3
Group: Ron Kalin/ Lamin Jammeh
Project Description: testbench
-------------------------------------------------------------------------------------------------
--------*/

module FIR_Pipeline_MAC_tb ();

//define the parameter sets for the design
parameter FIR_order        = 4;
parameter Sample_size      = 6;                  //maximum sample value is 63
parameter weight_size      = 5;                  //maximum value may be 31
parameter word_size_out    = 2 * Sample_size + 2;     //maximum possible output 63*31*(4+1)
parameter product_size     = Sample_size + weight_size;

//define the wires and registers for the test bench
wire  [word_size_out -1:0] FIR_out_pipeline;

reg      [Sample_size -1:0]      Sample_in;
reg                          clock, reset;

//define the unit under test UUT
FIR_Pipeline_MAC  UUT   (FIR_out_pipeline, Sample_in, clock, reset);

//instantiate the clock signal
initial
   begin
      clock = 0;
      forever     #5 clock = ~clock;
   end

//instantiate and toggle the reset signal
initial
   begin
      reset = 1;
      #40 reset = 0;
   end

//aplly different input Sample and observe the outputs
initial
   begin
      Sample_in = 0;
      #100  Sample_in = 1;       //impulse response
      #10   Sample_in = 0;
      #100  Sample_in = 10;      //same input over 5 clock cycles
      #50   Sample_in = 0;
      #100  Sample_in = 1;
      #10   Sample_in = 2;
      #10   Sample_in = 8;
      #10   Sample_in = 2;
      #10   Sample_in = 1;
      #10   Sample_in = 0;
      #100  Sample_in = 63;
      #100  Sample_in = 0;

      #100;
      $stop;
   end
endmodule
```

Data Flow graph (DFG)





Bitwave