```
1   // Name Lamin Jammeh / Ron Kalin
2   // EE417 Date: 08-03-2024, Group: Ron Kalin/ Lamin Jammeh
3   // FINAL PROJECT: FIR MAC Datapath
4   // Description: Datapath module computes/filters input Sample by Multiplying and Accumulating
5   module Pipeline_FIR_DataPath (FIR_out, Sample_in, clock, reset);
6
7   //define the parameter sets for the design
8   parameter FIR_order        = 4;
9   parameter Sample_size      = 6;                  //maximum sample value is 63
10  parameter weight_size      = 5;                  //maximum value may be 31
11  parameter word_size_out    = Sample_size + weight_size + 3; //log2(2^2 * 2^5 * (order+1))
12
13  //define output
14  output reg [word_size_out -1:0]  FIR_out;
15
16  //define inputs
17  input    [Sample_size -1:0]      Sample_in;
18  input                            clock, reset;
19
20  //define the filter coefficients
21  parameter   b0 = 5'd3;
22  parameter   b1 = 5'd7;
23  parameter   b2 = 5'd20;
24  parameter   b3 = 5'd7;
25  parameter   b4 = 5'd3;
26
27  reg         [Sample_size -1:0]   Sample_Array[1:FIR_order]; //5th coefficient multiplied by
    Data_in
28  integer     k;
29
30  //define PR0 to PR3 as registers
31  reg      [word_size_out -1:0] PR0 [0:FIR_order];
32  reg      [word_size_out -1:0] PR1 [1:FIR_order];
33  reg      [word_size_out -1:0] PR2 [2:FIR_order];
34  reg      [word_size_out -1:0] PR3 [3:FIR_order];
35
36  //define the transition logic
37  always @ (posedge clock)
38     if (reset == 1) // reset high
39     // set all Pipeline registers to zero: IR = 0       Input register
40     //                                     PR[0:order-1] = 0  Pipeline register
41     //                                     OR = 0       Output register
42        begin
43           for (k=1; k <= FIR_order; k = k + 1)
44              Sample_Array[k] <= 0; // input shift register
45           for (k = 0; k <= FIR_order; k = k + 1)
46              PR0[k] <= 0;   // pipeline register 0
47           for (k = 1; k <= FIR_order; k = k + 1)
48              PR1[k] <= 0;   // pipeline register 1
49           for (k = 2; k <= FIR_order; k = k + 1)
50              PR2[k] <= 0;   // pipeline register 2
51           for (k = 3; k <= FIR_order; k = k + 1)
52              PR3[k] <= 0;   // pipeline register 3
53
54              FIR_out <= 0;  // output register
55        end
56     else
57     // reset low
58     // 1 => move Sample_in into a cutset (Input register) to reduce input idle time
59     // 2 => insert the PR at the input of the add and perform x[n] * b(n) and save in
    Pipeline register (PRn[n-1])
60     // 3 => add all the PR registers at the input of the output register and save in the
    Output register
61        begin
62        //The input shift register
63           Sample_Array[1] <= Sample_in;
64           for (k = 2; k <= FIR_order; k = k + 1)
65              Sample_Array[k] <= Sample_Array[k-1];
66
67        //The pipeline registePR0
```

```
68              PR0[0] <= b0 * Sample_in;        // find products
69              PR0[1] <= b1 * Sample_Array[1];
70              PR0[2] <= b2 * Sample_Array[2];
71              PR0[3] <= b3 * Sample_Array[3];
72              PR0[4] <= b4 * Sample_Array[4];
73
74          // pipeline register PR1
75              PR1[1] <= PR0[0] + PR0[1];
76              PR1[2] <= PR0[2];
77              PR1[3] <= PR0[3];
78              PR1[4] <= PR0[4];
79
80          // pipeline register PR2
81              PR2[2] <= PR1[1] + PR1[2];
82              PR2[3] <= PR1[3];
83              PR2[4] <= PR1[4];
84
85          // pipeline register PR3
86              PR3[3] <= PR2[2] + PR2[3];
87              PR3[4] <= PR2[4];
88
89          // outpput register, sum products
90             FIR_out <= PR3[3] + PR3[4];
91          end
92    endmodule
93
```