```verilog
/*-------------------------------------------------------------------------------------------------
Name Lamin Jammeh
CLass: EE417 Summer 2024
FINAL PROJECT: Datapath
Group: Ron Kalin/ Lamin Jammeh
Project Description: The Datapath module computes Filters the input Sample by Multiplying and Accumulating
-------------------------------------------------------------------------------------------------*/

module Pipeline_FIR_DataPath (FIR_out, Sample_in, clock, reset);

//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     = Sample_size + weight_size + 3;    //log2(2^2 * 2^5 * (order+1))

//define output
output reg [word_size_out -1:0]     FIR_out;

//define inputs
input    [Sample_size -1:0]      Sample_in;
input                         clock, reset;

//define the filter coefficients
parameter   b0 = 5'd3;
parameter   b1 = 5'd7;
parameter   b2 = 5'd20;
parameter   b3 = 5'd7;
parameter   b4 = 5'd3;

reg         [Sample_size -1:0]      Sample_Array[1:FIR_order];       //5th coefficient multiplied by Data_in
integer     k;

//define PR0 to PR3 as registers
reg      [word_size_out -1:0] PR0 [0:FIR_order];
reg      [word_size_out -1:0] PR1 [1:FIR_order];
reg      [word_size_out -1:0] PR2 [2:FIR_order];
reg      [word_size_out -1:0] PR3 [3:FIR_order];

//define the transition logic
always @ (posedge clock)
   if (reset == 1)
   /*---------------------------------------
   if reset is high do the following
   ****set all Pipeline registers to zero
   *************IR = 0     Input register
   *************PR[0:order-1] = 0     Pipeline register
   *************OR = 0     Output register
   ----------------------------------------*/
      begin
      //The input shift register
         for (k=1; k <= FIR_order; k = k + 1)
            Sample_Array[k]  <= 0;

      //The pipeline register
         for (k = 0; k <= FIR_order; k = k + 1)
            PR0[k]  <= 0;
      //The pipeline register
         for (k = 1; k <= FIR_order; k = k + 1)
            PR1[k]  <= 0;

      //The pipeline register
         for (k = 2; k <= FIR_order; k = k + 1)
            PR2[k]  <= 0;
      //The pipeline register
```

```verilog
            for (k = 3; k <= FIR_order; k = k + 1)
               PR3[k] <= 0;

         //The outpput register
               FIR_out <= 0;
         end
      else
      /*--------------------------------------
      if reset is low do the following
      *************1 => move the Sample in into a cutset (Input register) to reduce idle time of the input
      *************2 => insert the PR at the input of the add and perform x[n] * b(n) and save in Pipeline register (PRn[n-1])
      *************3 => add all the PR registers at the input of the output register and save in the Output register
      ----------------------------------------*/
         begin
         //The input shift register
            Sample_Array[1] <= Sample_in;
            for (k = 2; k <= FIR_order; k = k + 1)
               Sample_Array[k] <= Sample_Array[k-1];

         //The pipeline register at PR0
              PR0[0] <= b0 * Sample_in;
              PR0[1] <= b1 * Sample_Array[1];
              PR0[2] <= b2 * Sample_Array[2];
              PR0[3] <= b3 * Sample_Array[3];
              PR0[4] <= b4 * Sample_Array[4];

         //The pipeline register at PR1
              PR1[1] <= b1 * Sample_Array[1] + PR0[1];
              PR1[2] <= b2 * Sample_Array[2];
              PR1[3] <= b3 * Sample_Array[3];
              PR1[4] <= b4 * Sample_Array[4];

         //The pipeline register at PR2
              PR2[2] <= b2 * Sample_Array[2] + PR1[2];
              PR2[3] <= b3 * Sample_Array[3];
              PR2[4] <= b4 * Sample_Array[4];

         //The pipeline register at PR3
              PR3[3] <= b3 * Sample_Array[3] + PR2[3];
              PR3[4] <= b4 * Sample_Array[4];

         //The outpput register
               FIR_out <= PR3[3] + PR3[4];
         end

endmodule

```