Fall
2024

FIR Filter Design

PROJECT 2 CE6325 VLSI DESIGN: SYNOPSYS PROJECT

LAMIN JAMMEH NET-ID: DAL852207

Project Description:

The Finite Impulse Response (FIR) filter designed in Project 1 had to be scaled to reach the project specifications. The original design had 386 cells after analysis and elaboration using Synopsys EDA tool.

The filter tap (order) was changed to 15 and both the input (Data_in) and output (Data_out) sizes were increased. These changes increased the latency in the simulation. To improve this, Pipeline registers were added to the output of each Multiply node and input of each addition node to reduce latency and idle time.

Design Specification:

Filter order: 15

Filter coefficients: [7, 8, 9, 12, 4, 7, 8, 9, 12, 4]

Data_in size: 8

Data_out size: 20

PR_mul and PR_add: Pipeline registers for multiplication results and addition results respectively

Data flow of the Design

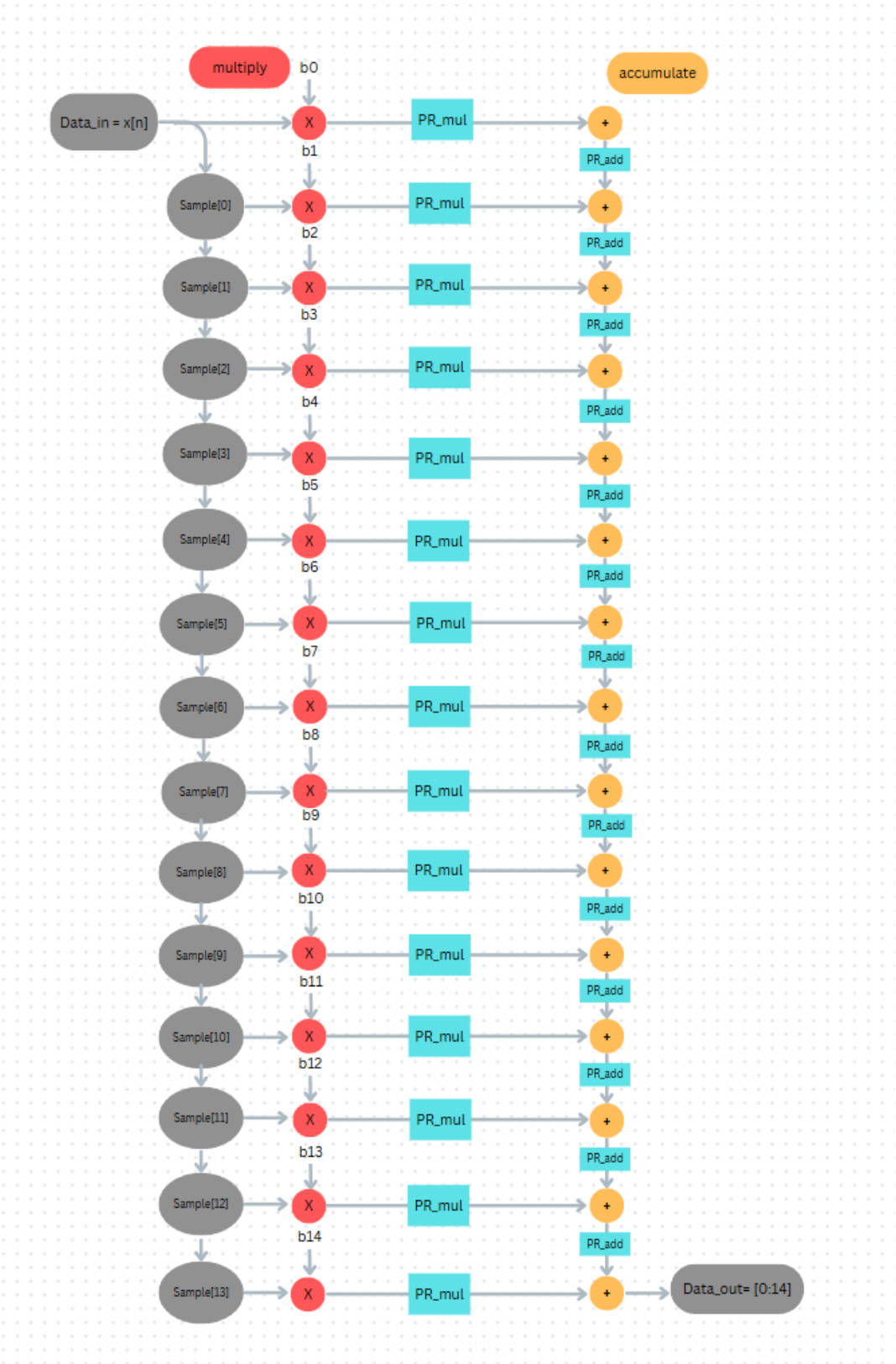


Figure1: Finite Impulse Response (FIR) Filter using Multiply and Accumulate with Pipeline Registers

Testbench Process Flow

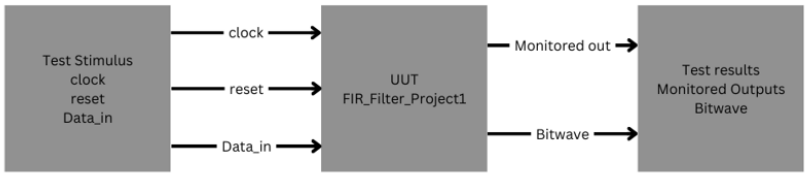


Figure 2: Shows the Testbench Process flow

Table 1 showing Filter out with when Data_in = decimal (6) in this case

Data_in		Sample[k]	Filter Coefficient (bn)		bn * Sample[k]		Acc @ filter stage
		6	7		42	acc0	42
		6	8		48	acc1	90
		6	9		54	acc2	144
		6	12		72	acc3	216
		6	4		24	acc4	240
		6	7		42	acc5	282
	Data_in	6	8	bn * Sample[k]	48	acc6	330
6		6	9		54	acc7	384
		6	12		72	acc8	456
		6	4		24	acc9	480
		6	7		42	acc10	522
		6	8		48	acc11	570
		6	9		54	acc12	624
		6	12		72	acc13	696
		6	4		24	Data_out/acc14	720

Table 1 shows that Data_in will go shift through all of Sample[k] when enough time is allowed before changing the value at the input. The value in each register of Sample[k] is multiplied by the filter

Design Simulation Results from Behavioral Model

Netlist from Verilog Model using Intel Quartus Prime and Questa

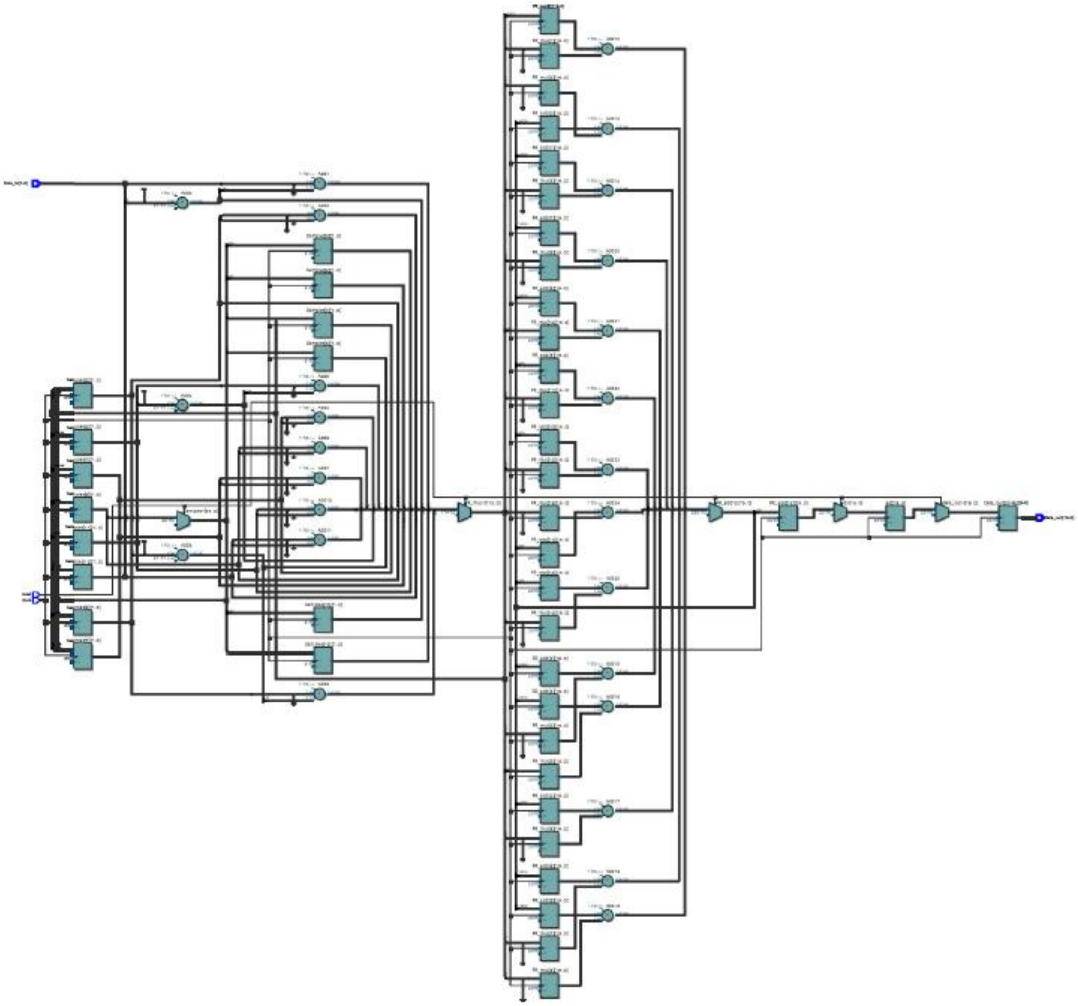


Figure 3: Netlist for the FIR filter

Table2 shows the observed values from the Behavioral Model Transcript Report

```
# run -all
# Time:    0  | Data_in: 0  | Data_out:   x  | reset: 1
# Time:    5  | Data_in: 0  | Data_out:   0  | reset: 1
# Time:   10  | Data_in: 0  | Data_out:   0  | reset: 0
# Time:  160  | Data_in: 6  | Data_out:   0  | reset: 0
# Time:  310  | Data_in: 3  | Data_out:   0  | reset: 0
# Time:  325  | Data_in: 3  | Data_out:  42  | reset: 0
# Time:  335  | Data_in: 3  | Data_out: 720  | reset: 0
# Time:  460  | Data_in: 2  | Data_out: 720  | reset: 0
# Time:  475  | Data_in: 2  | Data_out: 699  | reset: 0
# Time:  485  | Data_in: 2  | Data_out: 360  | reset: 0
# Time:  610  | Data_in: 5  | Data_out: 360  | reset: 0
# Time:  620  | Data_in: 5  | Data_out: 360  | reset: 1
# Time:  625  | Data_in: 5  | Data_out:   0  | reset: 1
# Time:  630  | Data_in: 5  | Data_out:   0  | reset: 0
# Time:  780  | Data_in: 2  | Data_out:   0  | reset: 0
# Time:  795  | Data_in: 2  | Data_out:  35  | reset: 0
# Time:  805  | Data_in: 2  | Data_out: 600  | reset: 0
# Time:  930  | Data_in: 0  | Data_out: 600  | reset: 0
# Time:  945  | Data_in: 0  | Data_out: 579  | reset: 0
# Time:  955  | Data_in: 0  | Data_out: 240  | reset: 0
# Time: 1080  | Data_in: 2  | Data_out: 240  | reset: 0
# Time: 1095  | Data_in: 2  | Data_out: 226  | reset: 0
# Time: 1105  | Data_in: 2  | Data_out:   0  | reset: 0
# Time: 1230  | Data_in: 4  | Data_out:   0  | reset: 0
# Time: 1245  | Data_in: 4  | Data_out:  14  | reset: 0
# Time: 1255  | Data_in: 4  | Data_out: 240  | reset: 0
# ** Note: $stop    : C:/Users/lmnjm/OneDrive/Documents/Grad School/UTD/Fall 2024/EEDG 6325/Projects/Project 2/Project Design/FIR_Filter_Project1_tb.v(66)
#    Time: 1380 ps  Iteration: 0  Instance: /FIR_Filter_Project1_tb
```

Simulation Waveform for the Behavioral Model using Intel Quartus Prime and Questa

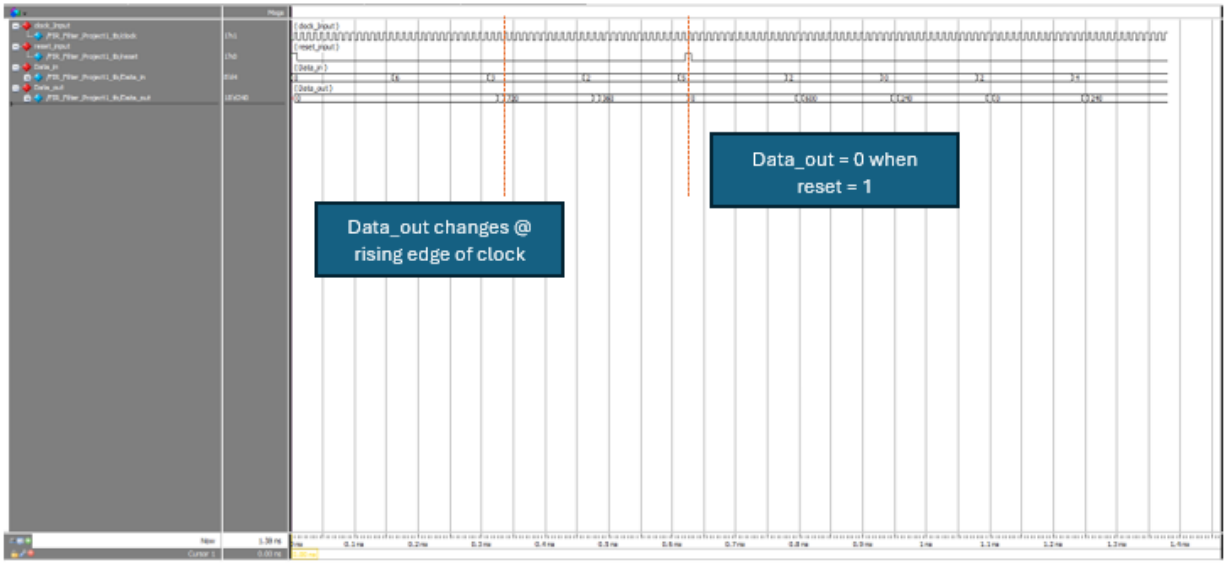
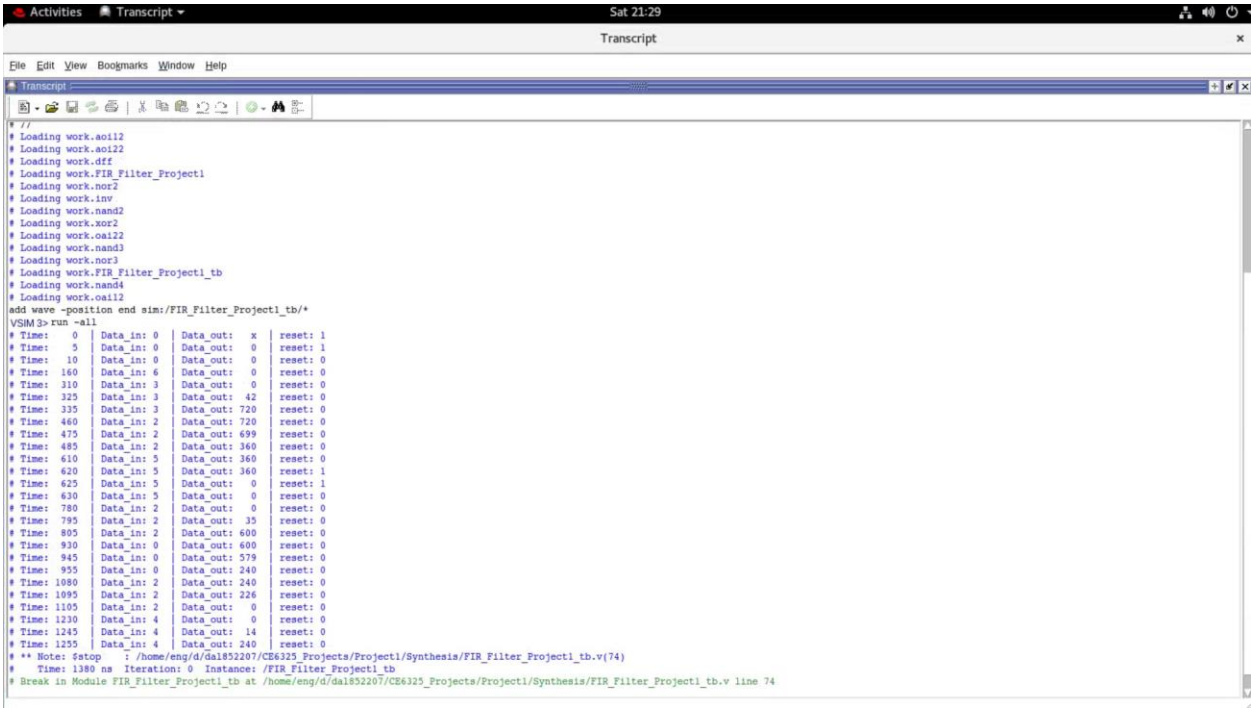


Figure 4: Behavioral Waveform showing that that the Data_out is dependent on reset and rising edge of the clock signal.

Design Simulation Results from Structural Model

Table3 shows the observed values from the Structural Model Transcript Report which is identical to behavioral Model's Trancript report



Simulation Waveform for the Structural model

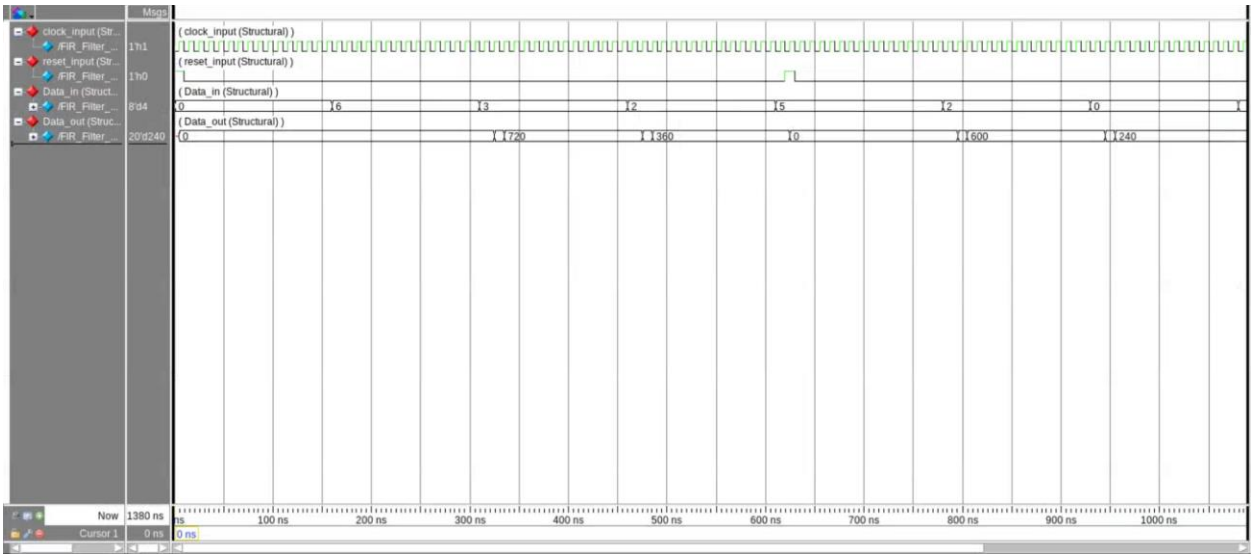


Figure 5: Structural Waveform showing that that the Data_out is dependent on reset and rising edge of the clock signal.

Cell Report from Synopsys (note some of the cells have been deleted)

```
****************************************
Report : cell
Design : FIR_Filter_Project1
Version: O-2018.06-SP1
Date   : Sun Sep  8 21:30:47 2024
****************************************

Attributes:
    b - black box (unknown)
    h - hierarchical
    n - noncombinational
    r - removable
    u - contains unmapped logic

Cell                    Reference       Library              Area  Attributes
--------------------------------------------------------------------------------
Data_out_reg[0]         dff             library          7.000000  n
Data_out_reg[1]         dff             library          7.000000  n
Data_out_reg[2]         dff             library          7.000000  n
Data_out_reg[3]         dff             library          7.000000  n
Data_out_reg[4]         dff             library          7.000000  n
Data_out_reg[5]         dff             library          7.000000  n
Data_out_reg[6]         dff             library          7.000000  n
Data_out_reg[7]         dff             library          7.000000  n
Data_out_reg[8]         dff             library          7.000000  n
Data_out_reg[9]         dff             library          7.000000  n
U2865                   inv             library          1.000000
U2866                   inv             library          1.000000
U2867                   inv             library          1.000000
U2868                   inv             library          1.000000
U2869                   nand3           library          1.000000
U2870                   nor3            library          1.000000
U2871                   inv             library          1.000000
--------------------------------------------------------------------------------
Total 3694 cells                                         8304.000000
```

Header.v module (contains all the different cell types use to create the structural Model)

```
module inv(in, out);
input in;
output out;
assign out = ~in;
endmodule

module nand2(a, b, out);
input a, b;
output out;
assign out = ~(a & b);
endmodule

module nand3(a, b, c, out);
input a, b, c;
output out;
assign out = ~(a & b & c);
endmodule

module nand4(a, b, c, d, out);
input a, b, c, d;
output out;
assign out = ~(a & b & c & d);
endmodule

module nor2(a, b, out);
input a, b;
output out;
assign out = ~(a | b);
endmodule

module nor3(a, b, c, out);
input a, b, c;
output out;
assign out = ~(a | b | c);
endmodule

module xor2(a, b, out);
input a, b;
output out;
assign out = (a ^ b);
endmodule

module aoi12(a, b, c, out);
input a, b, c;
output out;
assign out = ~(a | (b & c));
endmodule

module aoi22(a, b, c, d, out);
input a, b, c, d;
output out;
assign out = ~((a & b) | (c & d));
endmodule

module oai12(a, b, c, out);
input a, b, c;
output out;
assign out = ~(a & (b | c));
endmodule

module oai22(a, b, c, d, out);
input a, b, c, d;
output out;
assign out = ~((a | b) & (c | d));
endmodule

module dff( d, gclk, rnot, q);
input d, gclk, rnot;
output q;
reg q;
always @(posedge gclk or negedge rnot)
  if (rnot == 1'b0)
    q = 1'b0;
  else
    q = d;
endmodule
```

Summary:

The initial design from Project1 had to be scaled to meet the project Specification of at least 3000 cells

The structural Model from Synopsys and the header file were simulated on Model sim to obtain the transcript report and the waveform

The Table from the behavioral model was identical to the table from the structural model, the waveforms are also identical. These concludes that the structural model and the behavioral model are identical

The header.v file shows the different types of cells (inv, nan2, nan3, aoi12, oai12, dff, etc.) that will be need in the design library to create the FIR filter

The design will have a total of 3694 cells.

```
/*--------------------------------------------------------------------------
Name:    Lamin Jammeh
CLass:   CE6325 Fall_2024
Project: 1 Scaled up
Project Description: this is a Finite Impulse Response (FIR) filter design using Verilog HDL
The design follows the concepts below
**The filter order is selected and parameterized so the design can be scaled in the fututre
**The filter cooefficents are pre-determined
**Data_in samples will be provided in the testbench to determine the filter behavior
**The Data_in sample is Multiplied and accumalated through the diffrent filter stages/taps
**The Data_out word_size = word_in + coeff_size + [log2[N]]  where N= # of taps in the filte

--------------------------------------------------------------------------*/
module FIR_Filter_Project1
    #(parameter order = 15,   // Filter order [N=15] coeff size =8 [Max_coeff_size = 2^8 -1 = 255]
      parameter word_size_in = 8,   // Size of data_in [Max_Data_in = 2^8 - 1 = 255]
      parameter word_size_out = 20) // Output word size = log_2 (N * Max_Data_in * Max_coeff) = roughly 20

    // Declare inputs and outputs
    (
    output reg [word_size_out - 1:0]   Data_out,
    input      [word_size_in - 1:0]    Data_in,
    input                              clock, reset
    );

    reg [word_size_in  -1:0] Samples[order-1:0];   // Temporary storage for input samples (x(n))
    reg [word_size_out -1:0] acc;                  // Temporary storage for output data
    reg [word_size_out -1:0] PR_mul[order-1:0];    // Storage for multiplication results
    reg [word_size_out -1:0] PR_add[order-1:0];    // Pipeline registers for add operations
    integer k;

    // Filter Coefficients
    parameter b0 = 8'd7;
    parameter b1 = 8'd8;
    parameter b2 = 8'd9;
    parameter b3 = 8'd12;
    parameter b4 = 8'd4;

    parameter b5 = 8'd7;
    parameter b6 = 8'd8;
    parameter b7 = 8'd9;
    parameter b8 = 8'd12;
    parameter b9 = 8'd4;

    parameter b10 = 8'd7;
    parameter b11 = 8'd8;
    parameter b12 = 8'd9;
    parameter b13 = 8'd12;
    parameter b14 = 8'd4;

    always @(posedge clock) begin
        if (reset == 1) begin
            // Reset all samples, accumulation, and pipeline registers
            for (k = 0; k < order; k = k + 1) begin
                Samples[k] <= 0;
                PR_mul[k] <= 0;
                PR_add[k] <= 0;
            end
            acc <= 0;
            Data_out <= 0;
        end else begin
            // Shift samples
            Samples[0] <= Data_in;
            for (k = 1; k < order; k = k + 1) begin
                Samples[k] <= Samples[k - 1];
            end
```

```
67
68              // Compute multiplication results (pipeline stage 1)
69              PR_mul[0] <= b0 * Data_in;
70              PR_mul[1] <= b1 * Samples[0];
71              PR_mul[2] <= b2 * Samples[1];
72              PR_mul[3] <= b3 * Samples[2];
73              PR_mul[4] <= b4 * Samples[3];
74              PR_mul[5] <= b5 * Samples[4];
75              PR_mul[6] <= b6 * Samples[5];
76              PR_mul[7] <= b7 * Samples[6];
77              PR_mul[8] <= b8 * Samples[7];
78              PR_mul[9] <= b9 * Samples[8];
79
80              PR_mul[10] <= b10 * Samples[9];
81              PR_mul[11] <= b11 * Samples[10];
82              PR_mul[12] <= b12 * Samples[11];
83              PR_mul[13] <= b13 * Samples[12];
84              PR_mul[14] <= b14 * Samples[13];
85
86              // Pipeline stage for addition
87              PR_add[0] <= PR_mul[0] + PR_mul[1];
88              PR_add[1] <= PR_add[0] + PR_mul[2];
89              PR_add[2] <= PR_add[1] + PR_mul[3];
90              PR_add[3] <= PR_add[2] + PR_mul[4];
91              PR_add[4] <= PR_add[3] + PR_mul[5];
92              PR_add[5] <= PR_add[4] + PR_mul[6];
93              PR_add[6] <= PR_add[5] + PR_mul[7];
94              PR_add[7] <= PR_add[6] + PR_mul[8];
95              PR_add[8] <= PR_add[7] + PR_mul[9];
96
97              PR_add[9]  <= PR_add[8] + PR_mul[10];
98              PR_add[10] <= PR_add[9] + PR_mul[11];
99              PR_add[11] <= PR_add[10] + PR_mul[12];
100             PR_add[12] <= PR_add[11] + PR_mul[13];
101             PR_add[13] <= PR_add[12] + PR_mul[14];
102             // Final result after the last addition
103             acc <= PR_add[13];
104             Data_out <= acc;
105         end
106     end
107 endmodule
108
```

```
/*-------------------------------------------------------------------------
Name:       Lamin Jammeh
CLass:  CE6325 Fall_2024
Project:    1,2
Project Description: Teestbench for Project1
**creeate a clock signal
**Initialize all the input signals
**apply some Sample Data_in as [6325] in 10 time unit intervals
**apply reset to test the reset signal
**apply anothe set of sample Data_in as [2024] in 10 time unit intervals
**monitor the signals and end the test

-------------------------------------------------------------------------*/

module FIR_Filter_Project1_tb;

parameter order = 15;
parameter word_size_in = 8;
parameter word_size_out = 2 * word_size_in + 4;

//declare ports for the design
wire    [word_size_out -1:0]    Data_out;
reg     [word_size_in -1:0]      Data_in;
reg                             clock, reset;

//declare the unit under test UUT
FIR_Filter_Project1 UUT(.Data_out(Data_out),
                        .Data_in(Data_in),
                        .clock(clock),
                        .reset(reset)
                       );

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

//Instantiate the diiferent test scenarios to validate the design
//*******Scenario 1 initialize all input signals
initial
    begin
        reset = 1;
        Data_in =0;

        #10 reset = 0;  //wait for 10 timing units for the inital signals to go through

//Scenario 2 apply some Data_in samples and observe the outputs use [6  3   2   5] @ 100 time unit intervals
        #150    Data_in = 8'd6;         //make sure time is long enough for Data_in to mkae it to the last filter COefficient
        #150    Data_in = 8'd3;
        #150    Data_in = 8'd2;
        #150    Data_in = 8'd5;

//Scenario 3 test the reset signal to validate the behavior
        #10 reset = 1;          //Sample registers should be cleared
        #10 reset = 0;          //Sample register will accept Data_in

//Scenario 4 apple more samples to make sure the design works after reset
        #150    Data_in = 8'd2;
        #150    Data_in = 8'd0;
        #150    Data_in = 8'd2;
        #150    Data_in = 8'd4;

//stop the test
        #150    $stop;
    end
```

```
68
69  // Monitor output
70  initial
71      begin
72          $monitor("Time: %4t\t | Data_in: %0d\t | Data_out: %3d\t | reset: %b", $time,
            Data_in, Data_out, reset);    //use %10 as padding for time and data_out
73      end
74
75  endmodule
76
77
78
```