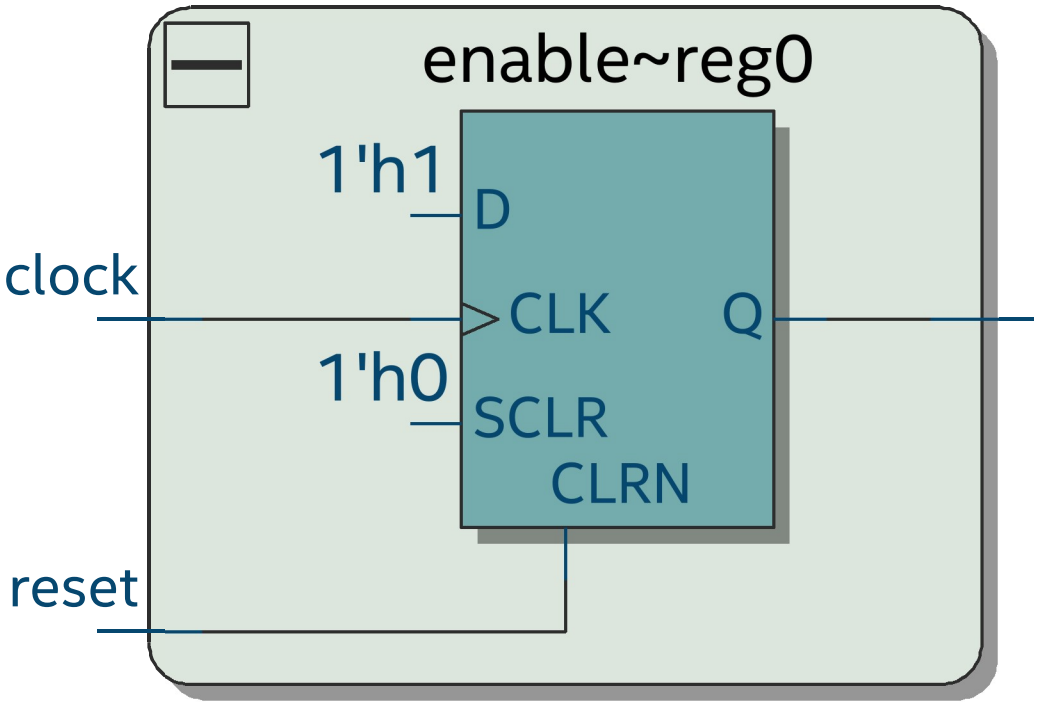
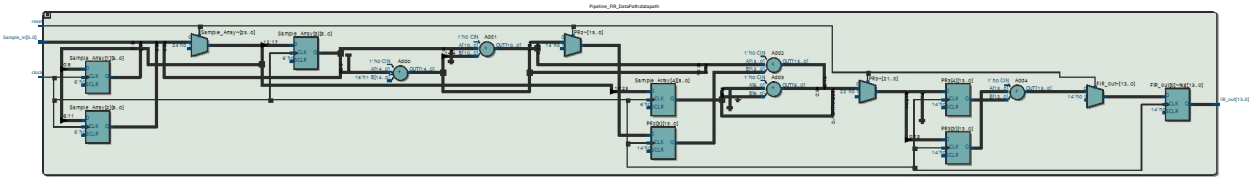
```
1  /*-------------------------------------------------------------------------------------------
   ---------
2  Name Lamin Jammeh
3  CLass: EE417 Summer 2024
4  FINAL PROJECT: Controller
5  Group: Ron Kalin/ Lamin Jammeh
6  Project Description: The Controller module drives the enable signal. This signal will
   enable the
7  computation of the FIR_MAC
8  -------------------------------------------------------------------------------------------
   --------*/
9
10 module Pipeline_FIR_Controller (
11                                  input clock,
12                                  input reset,
13                                  output reg enable
14                                 );
15
16 // Control logic for the FIR filter
17 always @ (posedge clock or posedge reset)
18 begin
19    if (reset)
20       begin
21         enable <= 0;
22       end
23       else
24       begin
25         enable <= 1;
26       end
27 end
28
29 endmodule
30
```

Pipeline_FIR_Controller:controller

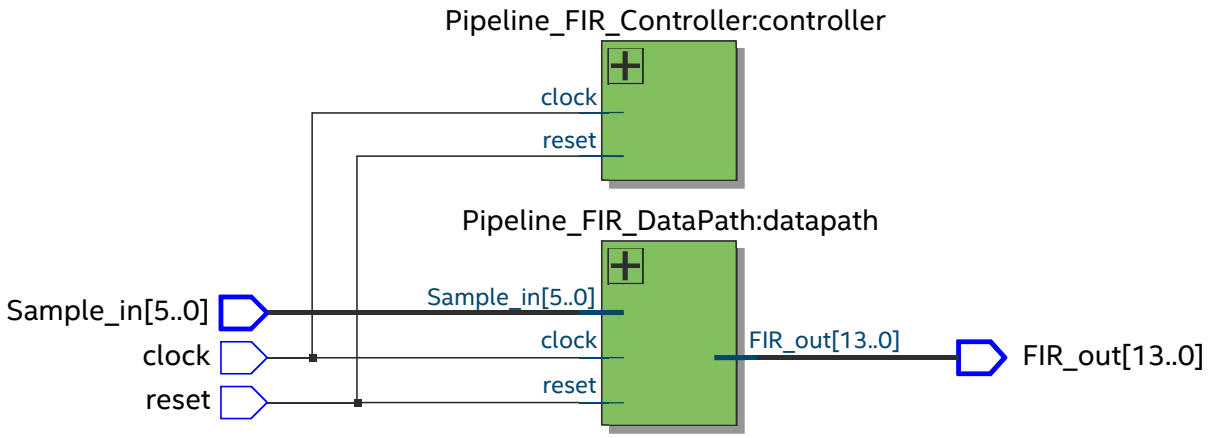


```
1  /*-----------------------------------------------------------------------------------------------
   --------------
2  Name Lamin Jammeh
3  CLass: EE417 Summer 2024
4  FINAL PROJECT: Datapath
5  Group: Ron Kalin/ Lamin Jammeh
6  Project Description: The Datapath module computes Filters the input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

```
66            for (k = 3; k <= FIR_order; k = k + 1)
67               PR3[k] <= 0;
68
69         //The outpput register
70                FIR_out <= 0;
71         end
72      else
73      /*---------------------------------------
74      if reset is low do the following
75      *************1 => move the Sample in into a cutset (Input register) to reduce idle time
    of the input
76      *************2 => insert the PR at the input of the add and perform x[n] * b(n) and save
    in Pipeline register (PRn[n-1])
77      *************3 => add all the PR registers at the input of the output register and save
    in the Output register
78      ---------------------------------------*/
79         begin
80         //The input shift register
81            Sample_Array[1] <= Sample_in;
82            for (k = 2; k <= FIR_order; k = k + 1)
83               Sample_Array[k] <= Sample_Array[k-1];
84
85         //The pipeline register at PR0
86             PR0[0] <= b0 * Sample_in;
87             PR0[1] <= b1 * Sample_Array[1];
88             PR0[2] <= b2 * Sample_Array[2];
89             PR0[3] <= b3 * Sample_Array[3];
90             PR0[4] <= b4 * Sample_Array[4];
91
92         //The pipeline register at PR1
93             PR1[1] <= b1 * Sample_Array[1] + PR0[1];
94             PR1[2] <= b2 * Sample_Array[2];
95             PR1[3] <= b3 * Sample_Array[3];
96             PR1[4] <= b4 * Sample_Array[4];
97
98         //The pipeline register at PR2
99             PR2[2] <= b2 * Sample_Array[2] + PR1[2];
100            PR2[3] <= b3 * Sample_Array[3];
101            PR2[4] <= b4 * Sample_Array[4];
102
103        //The pipeline register at PR3
104            PR3[3] <= b3 * Sample_Array[3] + PR2[3];
105            PR3[4] <= b4 * Sample_Array[4];
106
107        //The outpput register
108               FIR_out <= PR3[3] + PR3[4];
109        end
110
111  endmodule
112
```



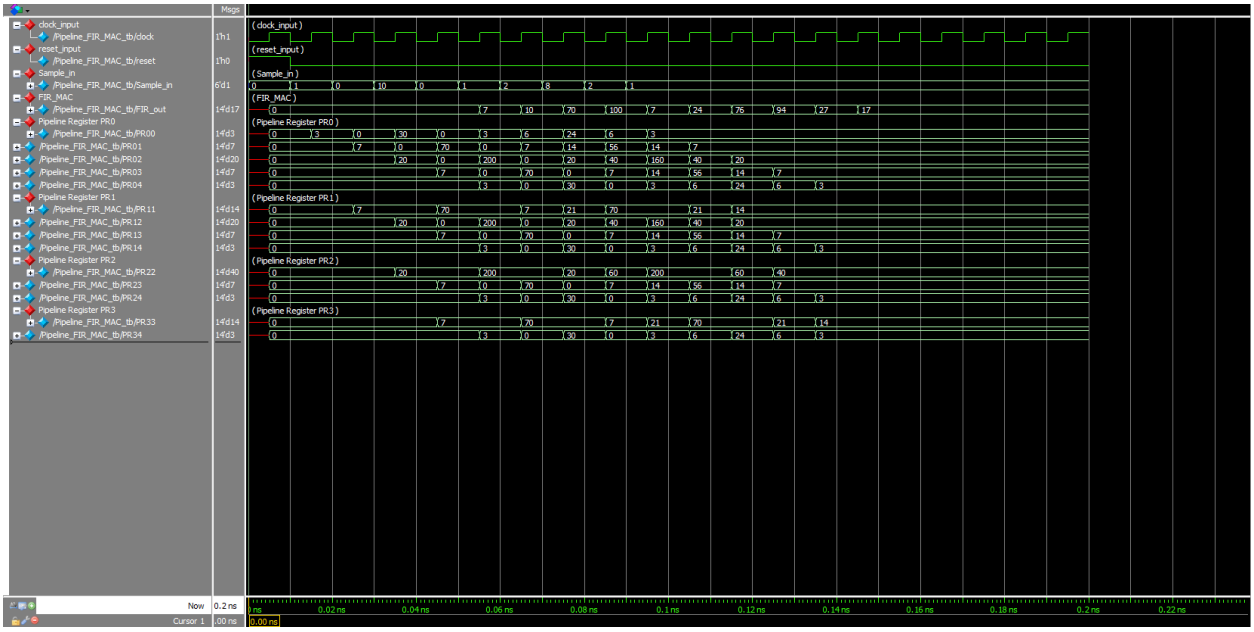
```
/*-------------------------------------------------------------------------------------------
---------
Name Lamin Jammeh
CLass: EE417 Summer 2024
FINAL PROJECT: FIR MAC module
Group: Ron Kalin/ Lamin Jammeh
Project Description: This Module combines the Datapath with the controller to form a
FIR_MAC.
-------------------------------------------------------------------------------------------
--------*/

module Pipeline_FIR_MAC (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 the outputs
output [word_size_out - 1:0] FIR_out;

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

// Internal signals
wire enable;

// Instantiate the DataPath module
Pipeline_FIR_DataPath #(FIR_order, Sample_size, weight_size, word_size_out) datapath (
    .FIR_out(FIR_out),
    .Sample_in(Sample_in),
    .clock(clock),
    .reset(reset)
);

// Instantiate the Controller module
Pipeline_FIR_Controller controller (
    .clock(clock),
    .reset(reset),
    .enable(enable)
);

endmodule

```


Pipeline_FIR_Controller:controller
clock
reset
Pipeline_FIR_DataPath:datapath
Sample_in[5..0]
clock
reset
Sample_in[5..0]
clock
reset
FIR_out[13..0]
FIR_out[13..0]

```
/*-------------------------------------------------------------------------------------------------
---------
Name Lamin Jammeh
CLass: EE417 Summer 2024
FINAL PROJECT: FIR MAC module
Group: Ron Kalin/ Lamin Jammeh
Project Description: testbench for the FIR_MAC with Pipelining
-------------------------------------------------------------------------------------------------
--------*/

module Pipeline_FIR_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)

// Define the wires and registers for the test bench
wire    [word_size_out -1:0]    FIR_out;
reg     [Sample_size -1:0]      Sample_in;
reg                             clock, reset;

// Define the unit under test UUT
Pipeline_FIR_MAC UUT (FIR_out, Sample_in, clock, reset);

// Define Probes to observe the Pipeline register PR0
wire    [word_size_out -1:0]    PR00;   assign PR00 = UUT.datapath.PR0[0];
wire    [word_size_out -1:0]    PR01;   assign PR01 = UUT.datapath.PR0[1];
wire    [word_size_out -1:0]    PR02;   assign PR02 = UUT.datapath.PR0[2];
wire    [word_size_out -1:0]    PR03;   assign PR03 = UUT.datapath.PR0[3];
wire    [word_size_out -1:0]    PR04;   assign PR04 = UUT.datapath.PR0[4];

// Define Probes to observe the Pipeline register PR1
wire    [word_size_out -1:0]    PR11;   assign PR11 = UUT.datapath.PR1[1];
wire    [word_size_out -1:0]    PR12;   assign PR12 = UUT.datapath.PR1[2];
wire    [word_size_out -1:0]    PR13;   assign PR13 = UUT.datapath.PR1[3];
wire    [word_size_out -1:0]    PR14;   assign PR14 = UUT.datapath.PR1[4];

// Define Probes to observe the Pipeline register PR2
wire    [word_size_out -1:0]    PR22;   assign PR22 = UUT.datapath.PR2[2];
wire    [word_size_out -1:0]    PR23;   assign PR23 = UUT.datapath.PR2[3];
wire    [word_size_out -1:0]    PR24;   assign PR24 = UUT.datapath.PR2[4];

// Define Probes to observe the Pipeline register PR3
wire    [word_size_out -1:0]    PR33;   assign PR33 = UUT.datapath.PR3[3];
wire    [word_size_out -1:0]    PR34;   assign PR34 = UUT.datapath.PR3[4];

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

// Instantiate and toggle the reset signal
initial
   begin
      reset = 1;
      #10 reset = 0;
   end

// Integer for file handle
integer f;
integer i;

// Apply different input samples and observe the outputs
initial
   begin
```

```
68          f = $fopen("output.txt", "w");
69          $fwrite(f, "\t\tTime\tSample_in\tFIR_out\n");
70
71          // Apply the input samples and log the output
72          for (i = 0; i < 10; i = i + 1)
73             begin
74                case (i)
75                    0:  Sample_in = 0;
76                    1:  Sample_in = 1;
77                    2:  Sample_in = 0;
78                    3:  Sample_in = 10;
79                    4:  Sample_in = 0;
80                    5:  Sample_in = 1;
81                    6:  Sample_in = 2;
82                    7:  Sample_in = 8;
83                    8:  Sample_in = 2;
84                    9:  Sample_in = 1;
85                    10: Sample_in = 0;
86                    11: Sample_in = 63;
87                    12: Sample_in = 0;
88                    default: Sample_in = 0;
89                endcase
90                #10; // wait for the output to settle
91                $fwrite(f, "%d\t   %d\t      %d\n", $time, Sample_in, FIR_out);
92             end
93
94             $fclose(f);
95             #100 $stop;
96          end
97
98    endmodule
99
```

Bitwave

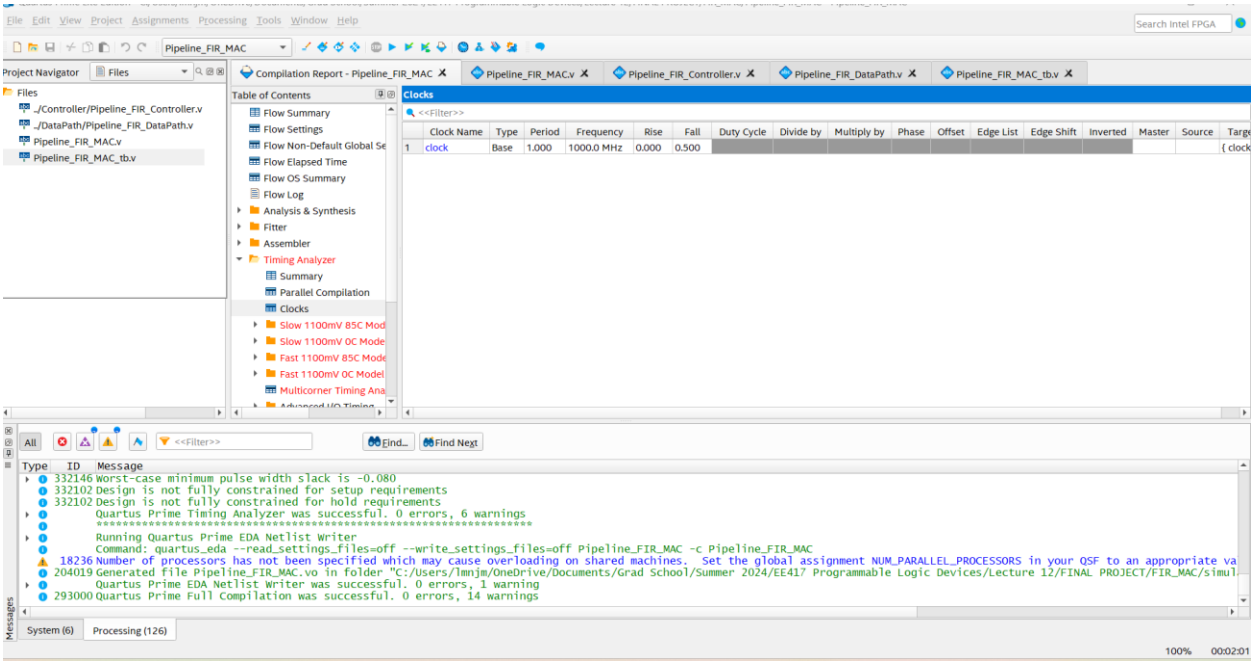


Testbench results from the text file

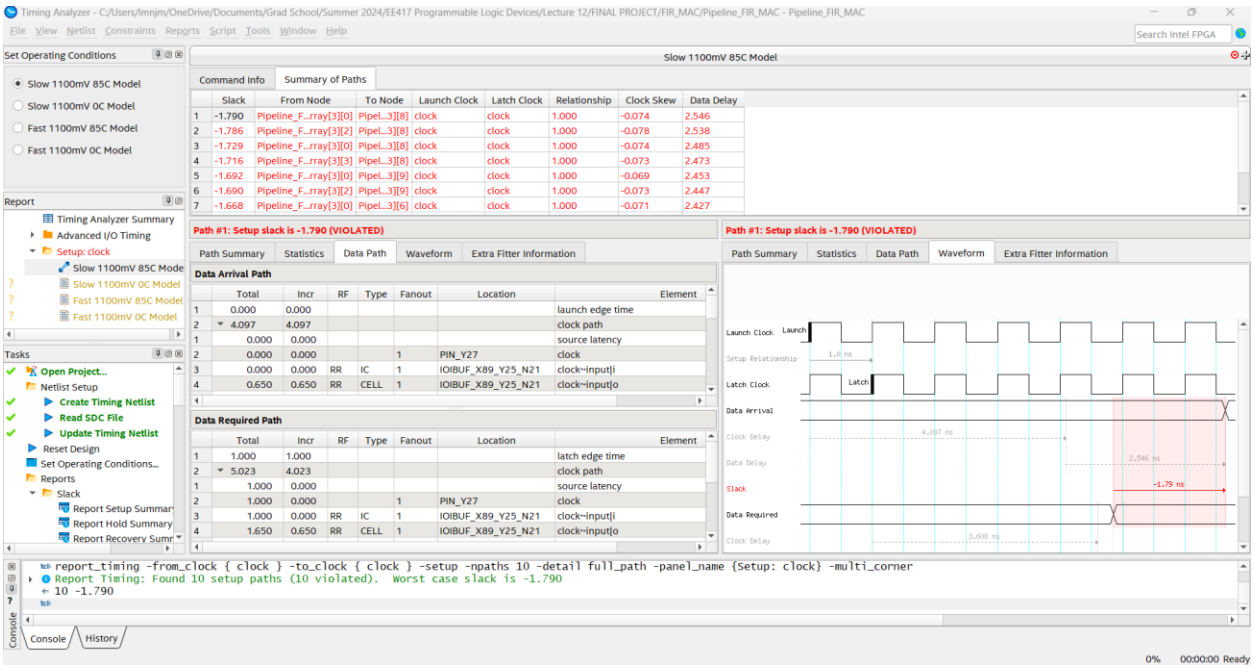
Time	Sample_in	FIR_out
10	0	0
20	1	0
30	0	0
40	10	0
50	0	0
60	1	7
70	2	10
80	8	70
90	2	100
100	1	7

Timing Analysis for the clock signal

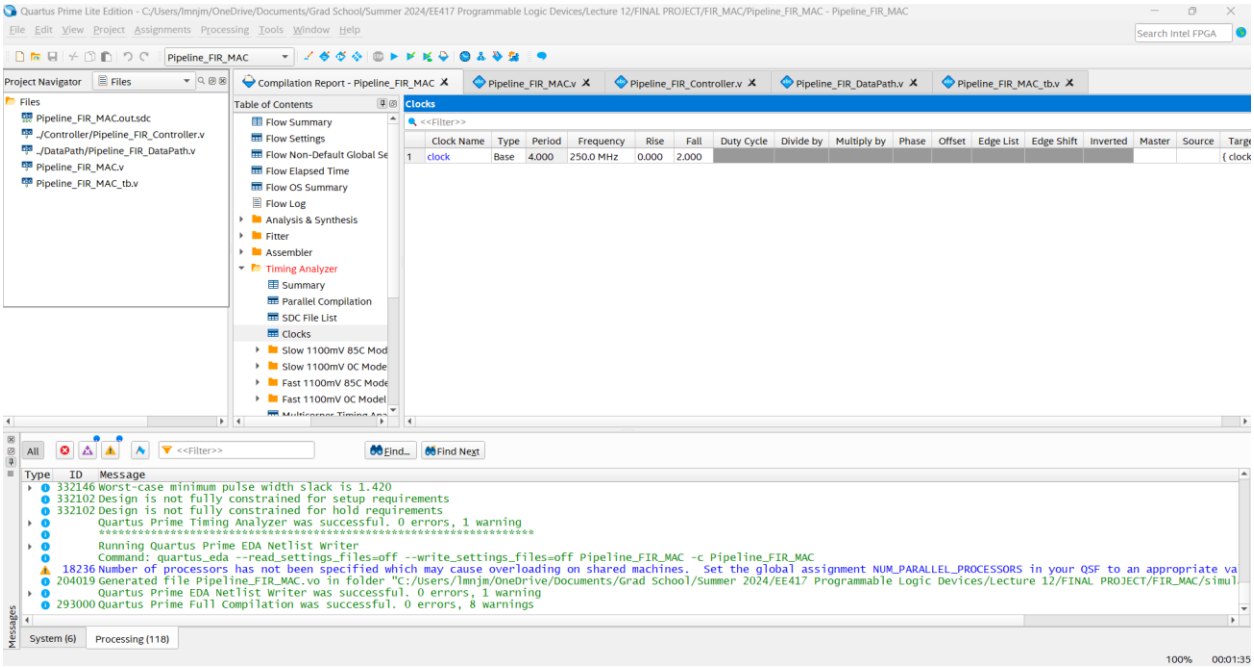
@ default time when clock period is 1ns



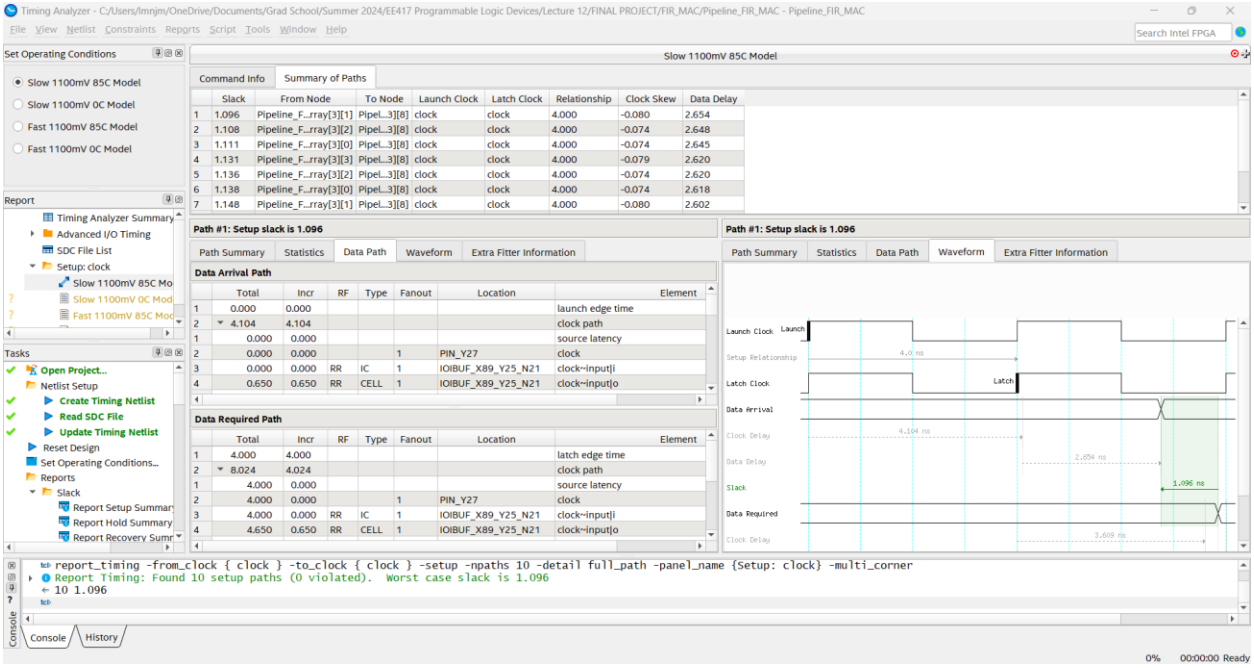
Time report at a slack of -1.790



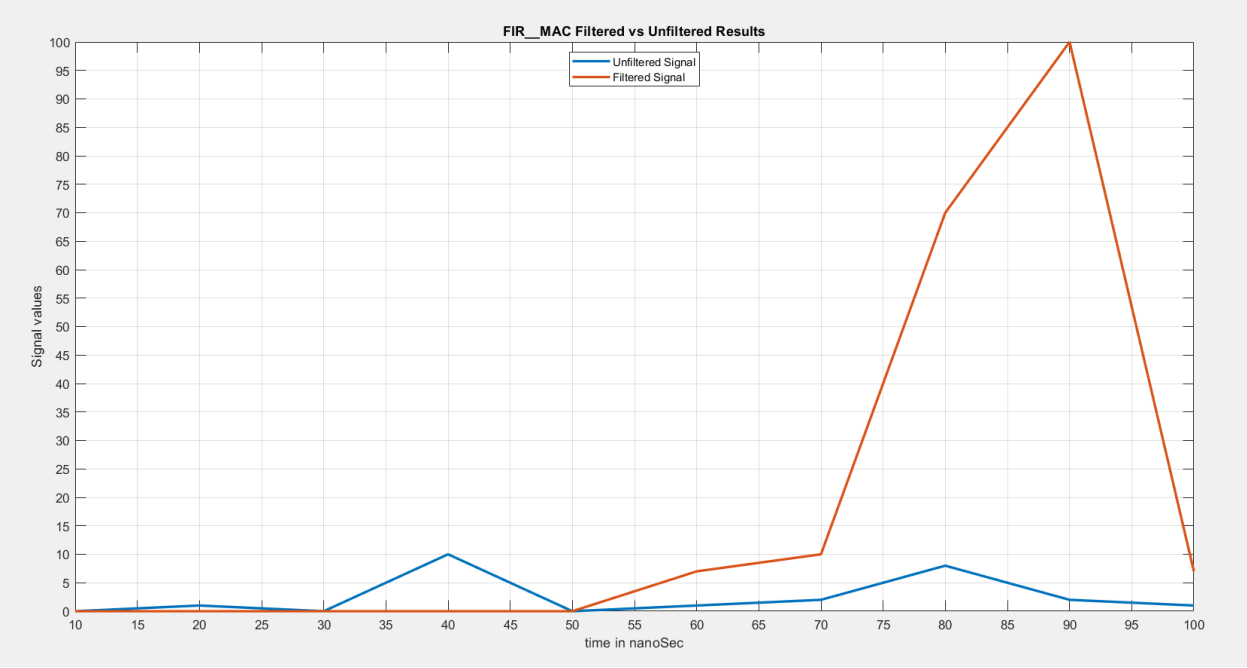
@ default time when clock period is 4ns



Time report at a slack of 1.096



```
%↙
------------------------------------------------------------------------------- ↙
--------------
% Name Lamin Jammeh
% CLass: EE417 Summer 2024
% Lesson 10 HW Question 3
% Group: Ron Kalin/ Lamin Jammeh
% Project Description: The Testbench results are ploted to show the filtered and ↙
unfiltered signals
%↙
------------------------------------------------------------------------------- ↙
--------------

%Step1 define the TestBench result
time = 10:10:100;
Sample_in = [0, 1, 0, 10, 0, 1, 2, 8, 2, 1];
FIR_out = [0, 0, 0, 0, 0, 7, 10, 70, 100, 7];

%Step2 plot the TestBench results
plot(time, Sample_in, 'LineWidth',2);
hold on;
plot(time, FIR_out, 'LineWidth',2);
grid on;
legend('Unfiltered Signal','Filtered Signal', 'Location','north');

title('FIR__MAC Filtered vs Unfiltered Results');
xticks(0:5:100);
yticks(0:5:100);
xlabel('time in nanoSec');
ylabel('Signal values');
hold off;
```


FIR__MAC Filtered vs Unfiltered Results
Unfiltered Signal
Filtered Signal
Signal values
time in nanoSec
100
95
90
85
80
75
70
65
60
55
50
45
40
35
30
25
20
15
10
5
0
10
15
20
25
30
35
40
45
50
55
60
65
70
75
80
85
90
95
100