

## Lesson 12 Final Project

## Finite Impulse Response Multiply-Accumulate FIR\_MAC Project

## Objective

In Digital Signal Processing (DSP), an input signal can be filtered using a Finite Impulse Response with Multiply-Accumulator (FIR\_MAC) filter. The filter coefficient can be calculated using MATLAB using the `fir1` function giving the design specifications (cut-off frequency, Max-frequency, filter order, and filter Type) are already determined. The filter function is used in MATLAB with a sample signal  $s$  to determine if the given coefficient can filter the sample signal  $s$ . The coefficients and the filter order are then used to implement the design of the FIR\_MAC in DSP.

The Finite Impulse Response with Multiply-Accumulator (FIR\_MAC) filter in this design will take in an input signal and multiply the signal with coefficient. The output of the multiply is temporarily stored in a pipeline register (PR) to reduce hardware idle time, thereby improving the latency of the clock signal. A similar PR is implemented at the input of the adder which performs the Accumulator section. The output of the Multiplier and the input of the adder and summed to delivered to input of the next adder on the next filter order. The number of adders and multiplexers is determined by the filter order of the design. To maintain the signal coherency a PR is added at the output of each Multiplier and input of each Accumulator [2].

The filter in this design has two submodules and a main or top module. The two submodules are the controller and the Datapath modules. The controller module enables the clock and reset signals. The Datapath looks for the posedge of the clock signal and the state of the reset signal. When reset is high; all registers and output of the FIR\_MAC goes to zero, and at reset low the Datapath module start processing input signals on the next posedge of the clock. To make sure the time requirement for the hardware is met the clock period was change from 1.0ns to 4.0ns, this changes the slack time from negative to positive to give the hardware some leeway in case of propagation delay[3].

## Application

Processing analog signals can come with lots of challenges since the analog signal are prone to noise which can be very difficult and expensive to clean. These can lead to wrong data especially from biomedical devices. The application of FIR\_MAC filters in digital signal processing suppresses the Powerline Interference (PLI)[1]. When FIR\_MAC filters are implemented in FPGA devices, the filter can be used for multiple DSP applications without the need for hardware change. The filter coefficients and filter order can be changed at the programming level. This is why FIR\_MAC filters are preferred in instrumentation and measurement devices.

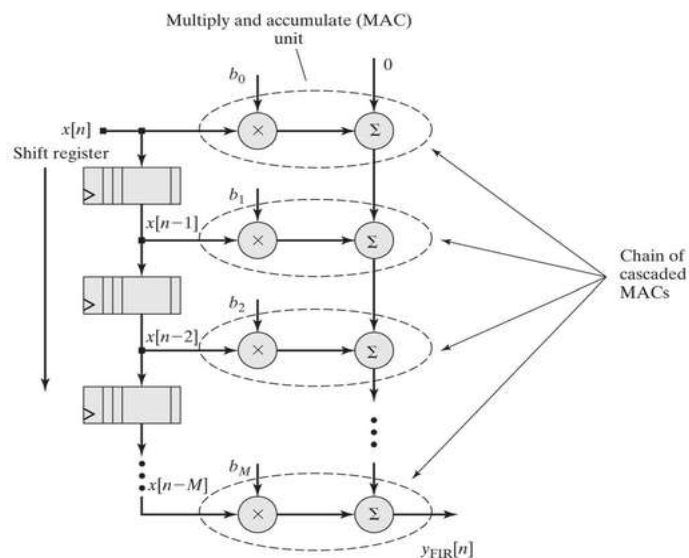
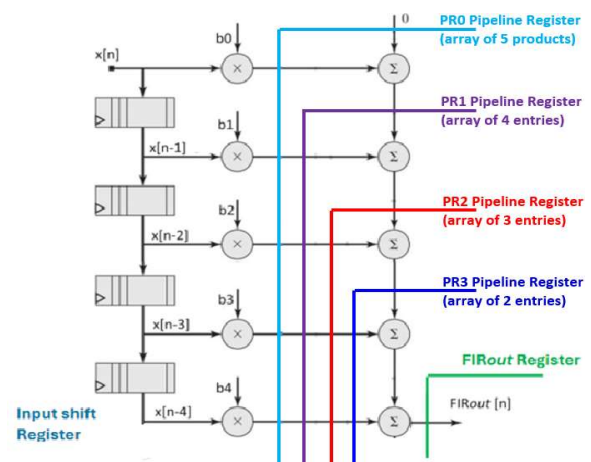


Figure 12.1 - FIR MAC diagram

Figure 12.2 -  
FIR MAC diagram, 4th order, showing pipelining cutlines

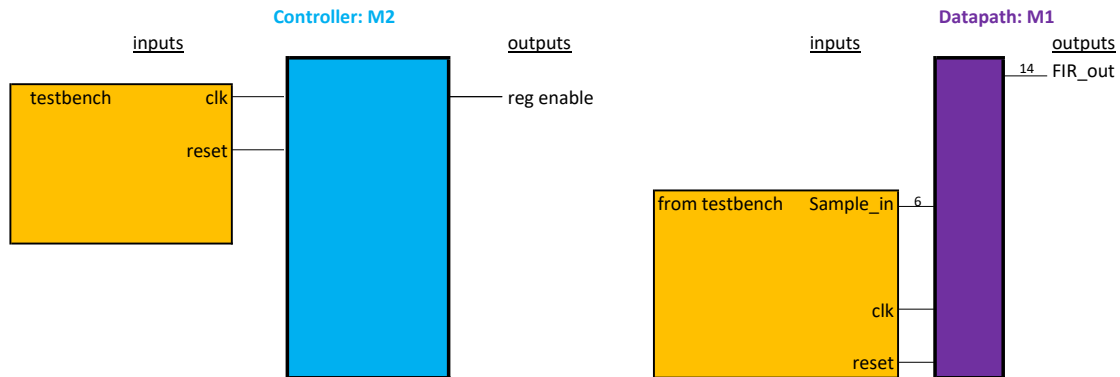
## Design Methodology

A controller - datapath structure will be utilized with a reduced number of states vs. a FSM design. The design also implements pipeline to reduce Hardware Idle time

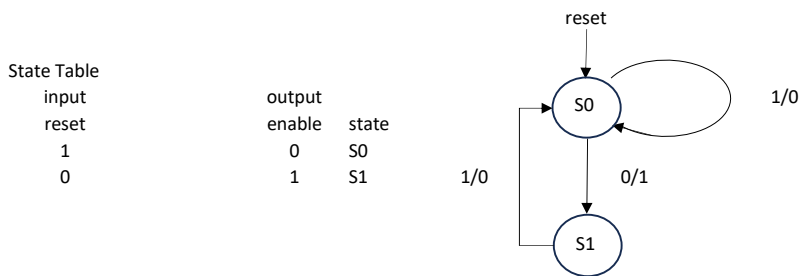
A block diagram and Finite State Machine (FSM) were developed from the Datapath in order to design the Controller code.

Ready was added as an output to the Datapath and as an input to the Controller.

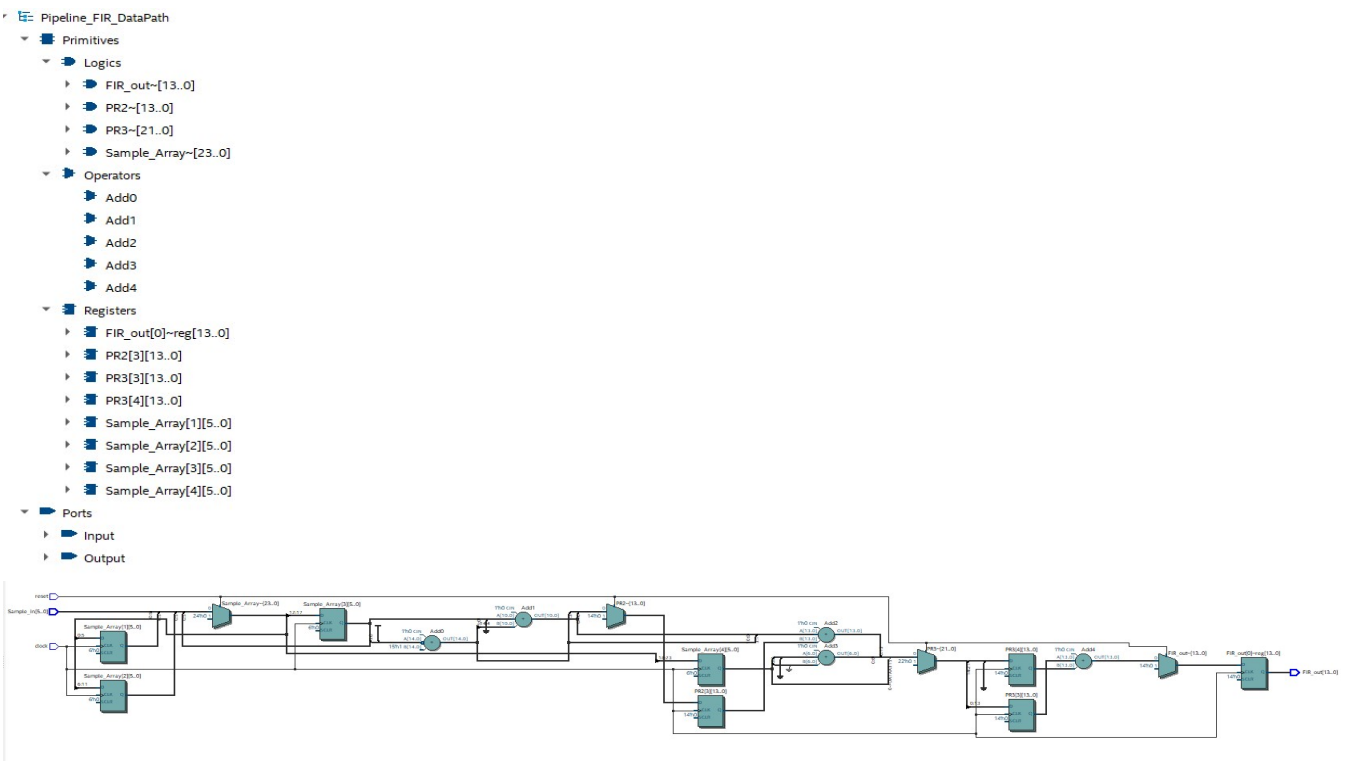
## Block Diagram



## FSM (State Graph) for Controller



## Building Blocks of the Datapath - RTL Viewer



## Design Verilog HDL Code -

Controller

ate: August 02, 2024

Pipeline\_FIR\_Controller.v

Project: Pipeline\_FIR\_DataPath

```

1 // ee417 Final Project L12A2
2 // Name: Ron Kalin / Lamin Jammeh, Date: 08-02-24 Group: Kalin/Jammeh
3 // Design: FIR filter chain of cascaded MACs, given 5-bit coefficients
4 // inputs are 6-bit positive values, output register is 14 parallel bits
5 // Project Description: Controller module drives the enable signal, which enables the
6 // computation of the FIR_MAC
7
8 module Pipeline_FIR_Controller (
9     input clock,
10    input reset,
11    output reg enable
12);
13
14 // Control logic for the FIR filter
15 always @ (posedge clock or posedge reset)
16 begin
17     if (reset)
18     begin
19         enable <= 0;
20     end
21     else
22     begin
23         enable <= 1;
24     end
25 end
26
27 endmodule
28

```

## Datapath

Date: August 02, 2024

Pipeline\_FIR\_DataPath.v

Project: Pipeline\_FIR\_DataPa

```

1 /*-----
2 Name Lamin Jammeh / Ron Kalin
3 Class: EE417 Summer 2024
4 FINAL PROJECT: Datapath
5 Group: Ron Kalin/ Lamin Jammeh
6 Project Description: Datapath module computes filters input Sample by Multiplying and
7 Accumulating
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(2^6 * 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[FIR_order]; //5th coefficient multiplied by
32 Data_in
33 integer k;
34
35 //define PR0 to PR3 as registers
36 reg [word_size_out -1:0] PR0 [0:FIR_order];
37 reg [word_size_out -1:0] PR1 [1:FIR_order];
38 reg [word_size_out -1:0] PR2 [2:FIR_order];
39 reg [word_size_out -1:0] PR3 [3:FIR_order];
40
41 //define the transition logic
42 always @ (posedge clock)
43 if (reset == 1)
44 /*-----
45 if reset is high do the following
46 *****set all Pipeline registers to zero
47 *****PR[0:order-1] = 0 Pipeline register
48 *****QR = 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 //The pipeline register
62 for (k = 2; k <= FIR_order; k = k + 1)
63 PR2[k] <= 0;
64 //The pipeline register
65 for (k = 3; k <= FIR_order; k = k + 1)
66 PR3[k] <= 0;

```

ate: August 02, 2024

Pipeline\_FIR\_DataPath.v

Project: Pipeline\_FIR\_DataPath

```

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

```

Top level module

Date: August 02, 2024

Pipeline\_FIR\_MAC.v

Project: Pipeline\_FIR\_DataPath

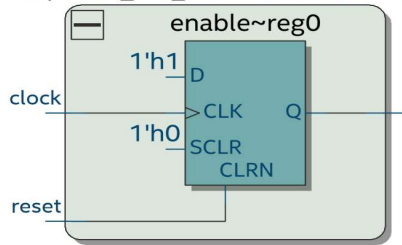
```

1 // Name Lamin Jammeh / Ron Kalin
2 // CLass: EE417 Summer 2024
3 // FINAL PROJECT: FIR MAC module
4 // Group: Ron Kalin/ Lamin Jammeh
5 // Summary: Module combines Datapath with controller to form a FIR_MAC
6
7 module Pipeline_FIR_MAC (FIR_out, Sample_in, clock, reset);
8
9 // Define the parameter sets for the design
10 parameter FIR_order = 4;
11 parameter Sample_size = 6; // Max sample value 2^6= 63
12 parameter weight_size = 5; // Max value may be 31
13 parameter word_size_out = Sample_size + weight_size + 3; // log2(2^2 * 2^5 * (order+1))
14
15 //define the outputs
16 output [word_size_out - 1:0] FIR_out;
17
18 //define the inputs
19 input [Sample_size - 1:0] Sample_in;
20 input clock, reset;
21
22 // Internal signals
23 wire enable;
24
25 // Instantiate the DataPath module
26 Pipeline_FIR_DataPath #(FIR_order, Sample_size, weight_size, word_size_out) datapath (
27 .FIR_out(FIR_out),
28 .Sample_in(Sample_in),
29 .clock(clock),
30 .reset(reset)
31 );
32
33 // Instantiate the Controller module
34 Pipeline_FIR_Controller controller (
35 .clock(clock),
36 .reset(reset),
37 .enable(enable)
38 );
39
40 endmodule
41

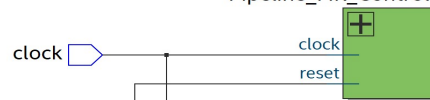
```

# RTL Viewer

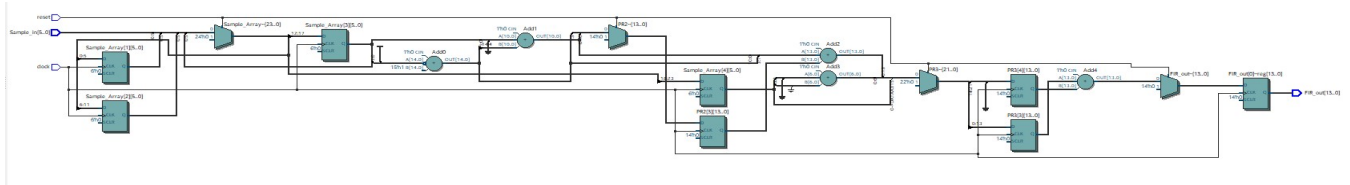
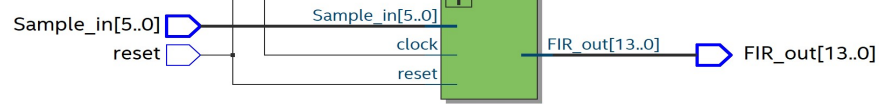
Pipeline\_FIR\_Controller:controller



Pipeline\_FIR\_Controller:controller



Pipeline\_FIR\_DataPath:datapath





```

1  /*-----
2  Name Ron Kalin / Lamin Jammeh
3  Class: EE417 Summer 2024
4  FINAL PROJECT: FIR MAC module
5  Group: Ron Kalin/ Lamin Jammeh
6  Project Description: testbench for the FIR_MAC with Pipelining
7  -----*/
8
9  module Pipeline_FIR_MAC_tb;
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  = 2 * Sample_size + 2; // maximum possible output 63*31*(4+1)
16
17  // Define the wires and registers for the test bench
18  wire [word_size_out -1:0] FIR_out;
19  reg [Sample_size -1:0] Sample_in;
20  reg clock, reset;
21
22  // Define the unit under test UUT
23  Pipeline_FIR_MAC UUT (FIR_out, Sample_in, clock, reset);
24
25  // Define Probes to observe the Pipeline register PR0
26  wire [word_size_out -1:0] PR00; assign PR00 = UUT.datapath.PR0[0];
27  wire [word_size_out -1:0] PR01; assign PR01 = UUT.datapath.PR0[1];
28  wire [word_size_out -1:0] PR02; assign PR02 = UUT.datapath.PR0[2];
29  wire [word_size_out -1:0] PR03; assign PR03 = UUT.datapath.PR0[3];
30  wire [word_size_out -1:0] PR04; assign PR04 = UUT.datapath.PR0[4];
31
32  // Define Probes to observe the Pipeline register PR1
33  wire [word_size_out -1:0] PR11; assign PR11 = UUT.datapath.PR1[1];
34  wire [word_size_out -1:0] PR12; assign PR12 = UUT.datapath.PR1[2];
35  wire [word_size_out -1:0] PR13; assign PR13 = UUT.datapath.PR1[3];
36  wire [word_size_out -1:0] PR14; assign PR14 = UUT.datapath.PR1[4];
37
38  // Define Probes to observe the Pipeline register PR2
39  wire [word_size_out -1:0] PR22; assign PR22 = UUT.datapath.PR2[2];
40  wire [word_size_out -1:0] PR23; assign PR23 = UUT.datapath.PR2[3];
41  wire [word_size_out -1:0] PR24; assign PR24 = UUT.datapath.PR2[4];
42
43  // Define Probes to observe the Pipeline register PR3
44  wire [word_size_out -1:0] PR33; assign PR33 = UUT.datapath.PR3[3];
45  wire [word_size_out -1:0] PR34; assign PR34 = UUT.datapath.PR3[4];
46
47  // Instantiate the clock signal
48  initial
49  begin
50      clock = 0;
51      forever #5 clock = ~clock;
52  end
53
54  // Instantiate and toggle the reset signal
55  initial
56  begin
57      reset = 1;
58      #10 reset = 0;
59  end
60
61  // Integer for file handle
62  integer f;
63  integer i;
64
65  // Apply different input samples and observe the outputs
66  initial
67  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
```



### Testbench results from the text file

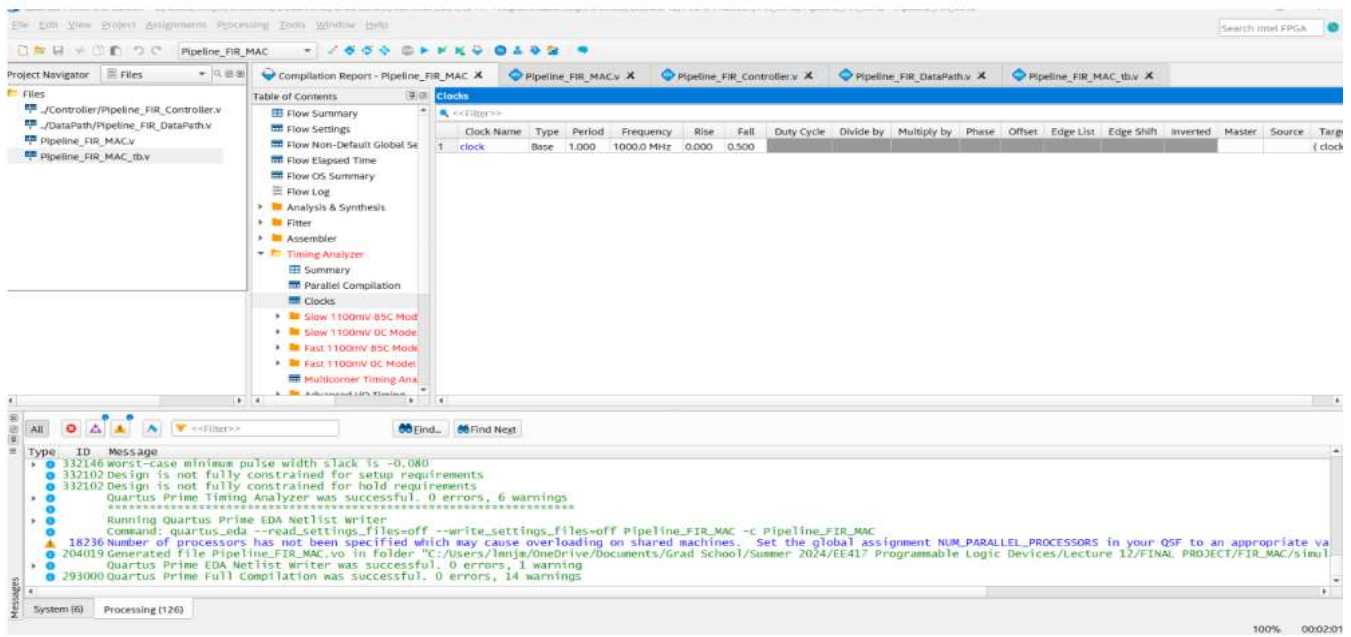
Time	Sample_in	FIR_out
10	0	0
20	1	0
30	0	0
40	10	0
50	0	0
60	1	7
70	2	10
80	8	70
90	2	100
100	1	7



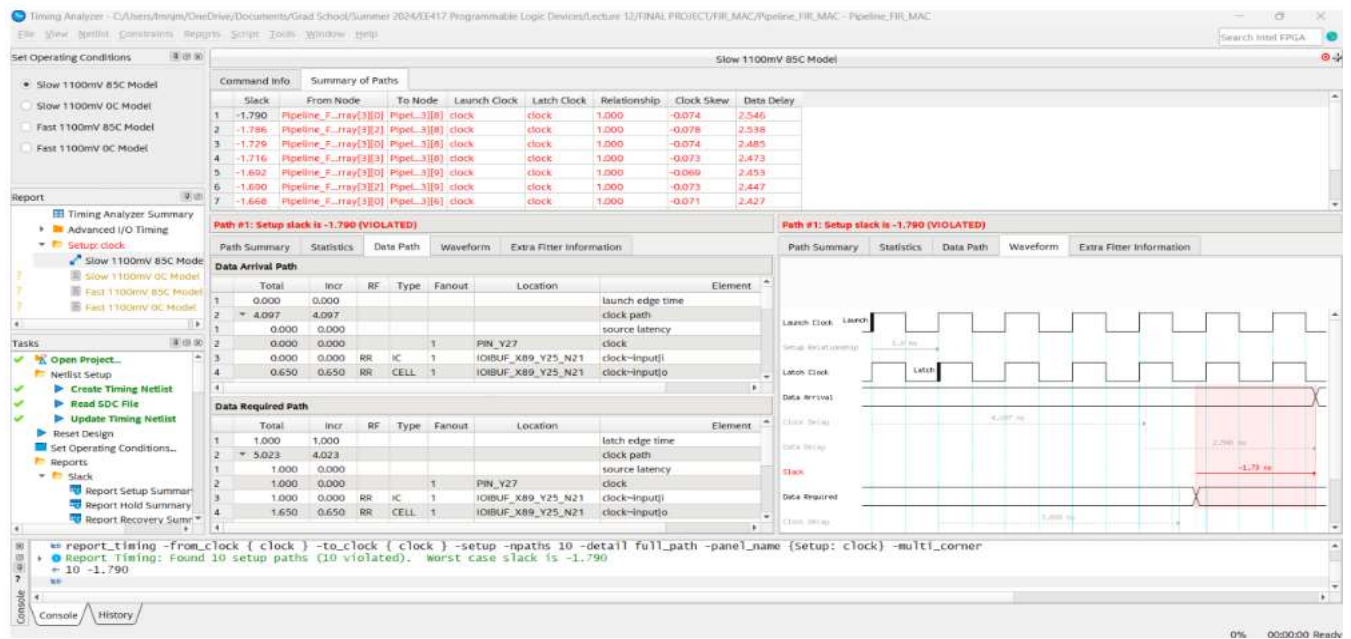
## Timing Analysis Reports

### Timing Analysis for the clock signal

@ default time when clock period is 1ns



Time report at a slack of -1.790



## Timing Analysis Reports (cont'd)

@ default time when clock period is 4ns

Quartus Prime Lite Edition - C:/Users/Imnjb/OneDrive/Documents/Grad School/Summer 2024/EE417 Programmable Logic Devices/Lecture 12/FINAL PROJECT/FIR\_MAC/Pipeline\_FIR\_MAC - Pipeline\_FIR\_MAC

File Edit View Project Assignments Processing Tools Window Help

Project Navigator Files

Table of Contents

- Flow Summary
- Flow Settings
- Flow Non-Default Global Settings
- Flow Elapsed Time
- Flow OS Summary
- Flow Log
- Analysis & Synthesis
- Filter
- Assembler
- Timing Analyzer
  - Summary
  - Parallel Compilation
  - SDC File List
  - Clocks
  - Slow 1100mV 85C Model
  - Slow 1100mV 0C Model
  - Fast 1100mV 85C Model
  - Fast 1100mV 0C Model
  - Multi-corner Timing Analysis

Messages

System (6) Processing (118)

100% 00:01:35

Time report at a slack of 1.096

Timing Analyzer - C:/Users/Imnjb/OneDrive/Documents/Grad School/Summer 2024/EE417 Programmable Logic Devices/Lecture 12/FINAL PROJECT/FIR\_MAC/Pipeline\_FIR\_MAC - Pipeline\_FIR\_MAC

File View Reports Constraints Reports Script Tools Window Help

Set Operating Conditions

- Slow 1100mV 85C Model
- Slow 1100mV 0C Model
- Fast 1100mV 85C Model
- Fast 1100mV 0C Model

Report

- Timing Analyzer Summary
- Advanced I/O Timing
- SDC File List
- Setup clock
- Slow 1100mV 85C Model
- Slow 1100mV 0C Model
- Fast 1100mV 85C Model
- Fast 1100mV 0C Model

Tasks

- Open Project...
- Netlist Setup
- Create Timing Netlist
- Read SDC File
- Update Timing Netlist
- Reset Design
- Set Operating Conditions...
- Reports
- Report Setup Summary
- Report Hold Summary
- Report Recovery Summary

Command Info

Summary of Paths

Slack	From Node	To Node	Launch Clock	Latch Clock	Relationship	Clock Skew	Data Delay
1.096	Pipeline_F_array[3][1] Pipel_3[R]	Pipel_3[R]	clock	clock	4.000	-0.080	2.654
1.108	Pipeline_F_array[3][2] Pipel_3[R]	Pipel_3[R]	clock	clock	4.000	-0.074	2.648
1.111	Pipeline_F_array[3][0] Pipel_3[R]	Pipel_3[R]	clock	clock	4.000	-0.074	2.645
1.131	Pipeline_F_array[3][3] Pipel_3[R]	Pipel_3[R]	clock	clock	4.000	-0.079	2.620
1.136	Pipeline_F_array[3][2] Pipel_3[R]	Pipel_3[R]	clock	clock	4.000	-0.074	2.620
1.138	Pipeline_F_array[3][0] Pipel_3[R]	Pipel_3[R]	clock	clock	4.000	-0.074	2.618
1.148	Pipeline_F_array[3][1] Pipel_3[R]	Pipel_3[R]	clock	clock	4.000	-0.080	2.602

Path #1: Setup slack is 1.096

Path Summary Statistics Data Path Waveform Extra Filter Information

Data Arrival Path

Total	Incr	RF	Type	Fanout	Location	Element
0.000	0.000					launch edge time
4.104	4.104					clock path
0.000	0.000					source latency
0.000	0.000	RR	IC	1	PIN_Y27	clock
0.650	0.650	RR	CELL	1	IOIBUF_XB9_Y25_N21	clock-input[0]

Data Required Path

Total	Incr	RF	Type	Fanout	Location	Element
4.000	4.000					latch edge time
8.024	4.024					clock path
4.000	0.000					source latency
4.000	0.000	RR	IC	1	PIN_Y27	clock
4.650	0.650	RR	CELL	1	IOIBUF_XB9_Y25_N21	clock-input[0]

Path #1: Setup slack is 1.096

Path Summary Statistics Data Path Waveform Extra Filter Information

Launch Clock Launch

Setup (Relax) constraint

Latch Clock Latch

Data Arrival

Clock Delay

Slack

Data Required

Clock Delay

report\_timing -from\_clock { clock } -to\_clock { clock } -setup -npaths 10 -detail full\_path -panel\_name {Setup: clock} -multi\_corner

Report Timing: Found 10 setup paths (0 violated). Worst case slack is 1.096

10 1.096

Console History

0% 00:00:00 Ready