

Lesson 12 Final Project

Multiply and Accumulate MAC operation

Objective

In computing, especially digital signal processing, the multiply–accumulate (MAC) or multiply–add (MAD) operation is a common step that computes the product of two numbers and adds that product to an accumulator. The hardware unit that performs the operation is known as a multiplier–accumulator (MAC unit); the operation itself is also often called a MAC or a MAD operation. The MAC operation modifies an accumulator a :

$$a \leftarrow a + (b \times c) [1]$$

The FIR MAC (Multiply and Accumulator) code was designed and tested the coherence of the data and find the latency (in terms of clock cycles) for each design. The actual propagation delay in ns for the design was found without pipelining and with different locations for the cut-set. Comment on the latency and alignment of data in your simulation results. Registers will be added at the outputs of all the multipliers. This way multiplication will occur in one clock cycle and then addition will occur in the following clock cycle. Report the latency and the Hardware usage in this case. [1]

This design will implement a FIR MAC 4th order Datapath-controller structure with the use of pipelining. The Verilog design includes a testbench to verify the results. The testbench input data and results were saved as a text file and plotted in MATLAB. Timing Analysis is included, and report information will show the maximum throughput or clock frequency that can be implemented on the design. A block diagram will be presented that clearly shows the Datapath-controller structure of the design with the datapath and control signals defined. The state graph of the controller will be shown along with the building blocks of the datapath. Finally Simulation results in Questa with internal probes for functionality verification will be depicted showing specific cases, such as reset, dataflow between registers with expected delays ... etc.

Application

Modern computers may contain a dedicated MAC, consisting of a multiplier implemented in combinational logic followed by an adder and an accumulator register that stores the result. The output of the register is fed back to one input of the adder, so that on each clock cycle, the output of the multiplier is added to the register. Combinational multipliers require a large amount of logic, but can compute a product much more quickly than the method of shifting and adding typical of earlier computers [1]

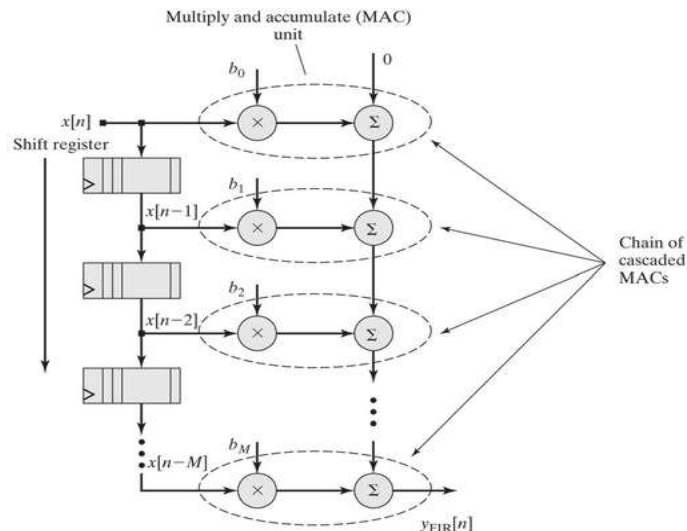


Figure 12.1 - FIR MAC diagram

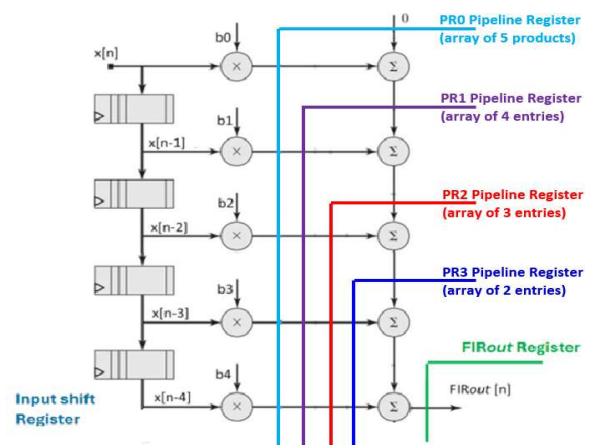


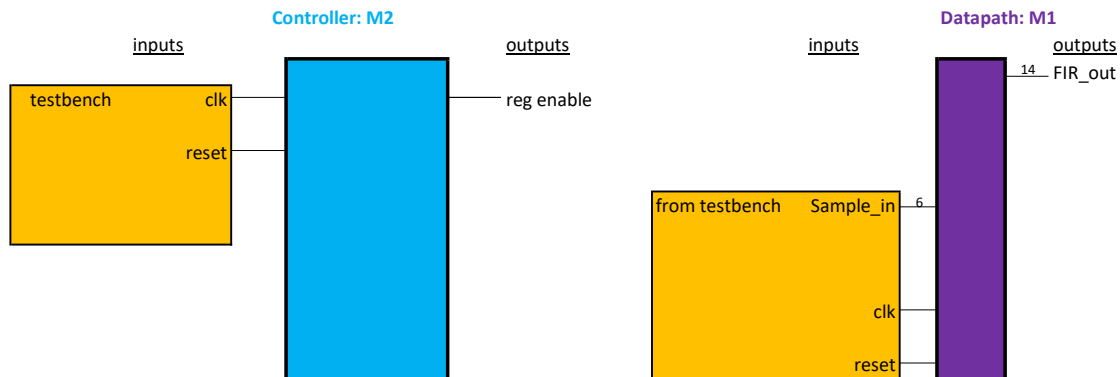
Figure 12.2 - FIR MAC diagram, 4th order, showing pipelining cutlines

Design Methodology

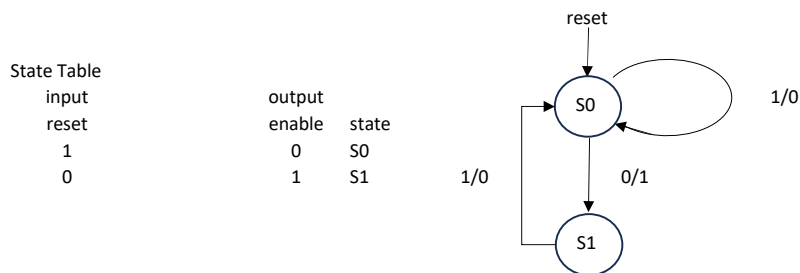
A controller - datapath structure will be utilized with a reduced number of states vs. a FSM design

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.

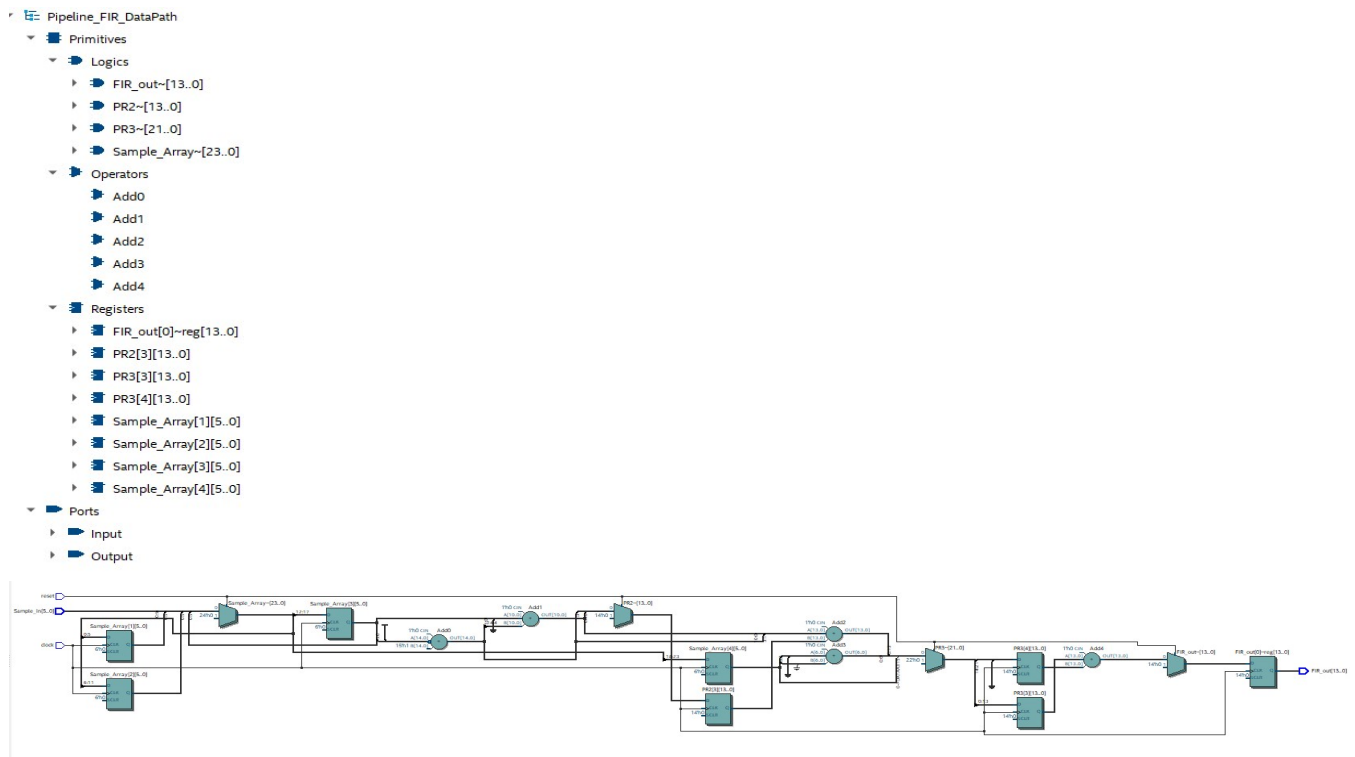
Block Diagram



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
10 module Pipeline_FIR_DataPath (FIR_out, Sample_in, clock, reset);
11
12 //define the parameter sets for the design
13 parameter FIR_order = 4;
14 parameter Sample_size = 6; //maximum sample value is 63
15 parameter weight_size = 5; //maximum value may be 31
16 parameter word_size_out = Sample_size + weight_size + 3; //Log2(2^6 * 2^5 * (order+1))
17
18 //define output
19 output reg [word_size_out -1:0] FIR_out;
20
21 //define inputs
22 input [Sample_size -1:0] Sample_in;
23 input clock, reset;
24
25 //define the filter coefficients
26 parameter b0 = 5'd3;
27 parameter b1 = 5'd7;
28 parameter b2 = 5'd20;
29 parameter b3 = 5'd7;
30 parameter b4 = 5'd3;
31
32 reg [Sample_size -1:0] Sample_Array[1:FIR_order]; //5th coefficient multiplied by
33 Data_in
34 integer k;
35
36 //define PR0 to PR3 as registers
37 reg [word_size_out -1:0] PR0 [0:FIR_order];
38 reg [word_size_out -1:0] PR1 [1:FIR_order];
39 reg [word_size_out -1:0] PR2 [2:FIR_order];
40 reg [word_size_out -1:0] PR3 [3:FIR_order];
41
42 //define the transition logic
43 always @ (posedge clock)
44 if (reset == 1)
45
46 /*-----
47 if reset is high do the following
48 ****set all Pipeline registers to zero
49 *****PR[0:order-1] = 0 Input register
50 *****PR[0:order-1] = 0 Pipeline register
51 *****PR[0:order-1] = 0 Output register
52 -----*/
53
54 begin
55 //The input shift register
56 for (k=1; k <= FIR_order; k = k + 1)
57 Sample_Array[k] <= 0;
58
59 //The pipeline register
60 for (k = 0; k <= FIR_order; k = k + 1)
61 PR0[k] <= 0;
62
63 //The pipeline register
64 for (k = 1; k <= FIR_order; k = k + 1)
65 PR1[k] <= 0;
66
67 //The pipeline register
68 for (k = 2; k <= FIR_order; k = k + 1)
69 PR2[k] <= 0;
70
71 //The pipeline register
72 for (k = 3; k <= FIR_order; k = k + 1)
73 PR3[k] <= 0;
74

```

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

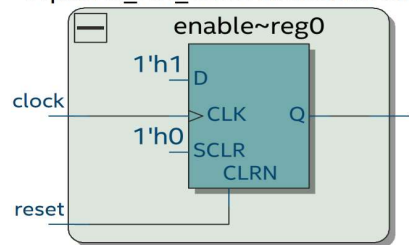
```

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(2^6 * 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

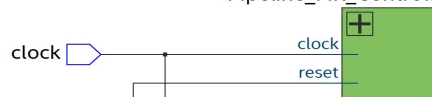
```

RTL Viewer

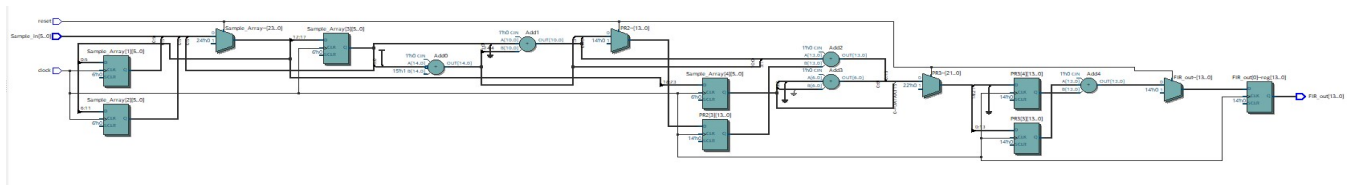
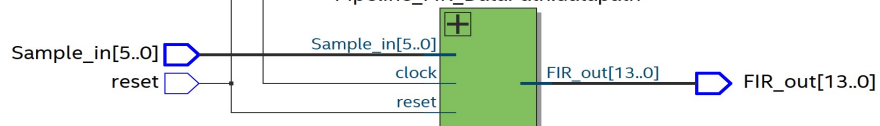
Pipeline_FIR_Controller:controller



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
```

RTL Simulation -from Questa



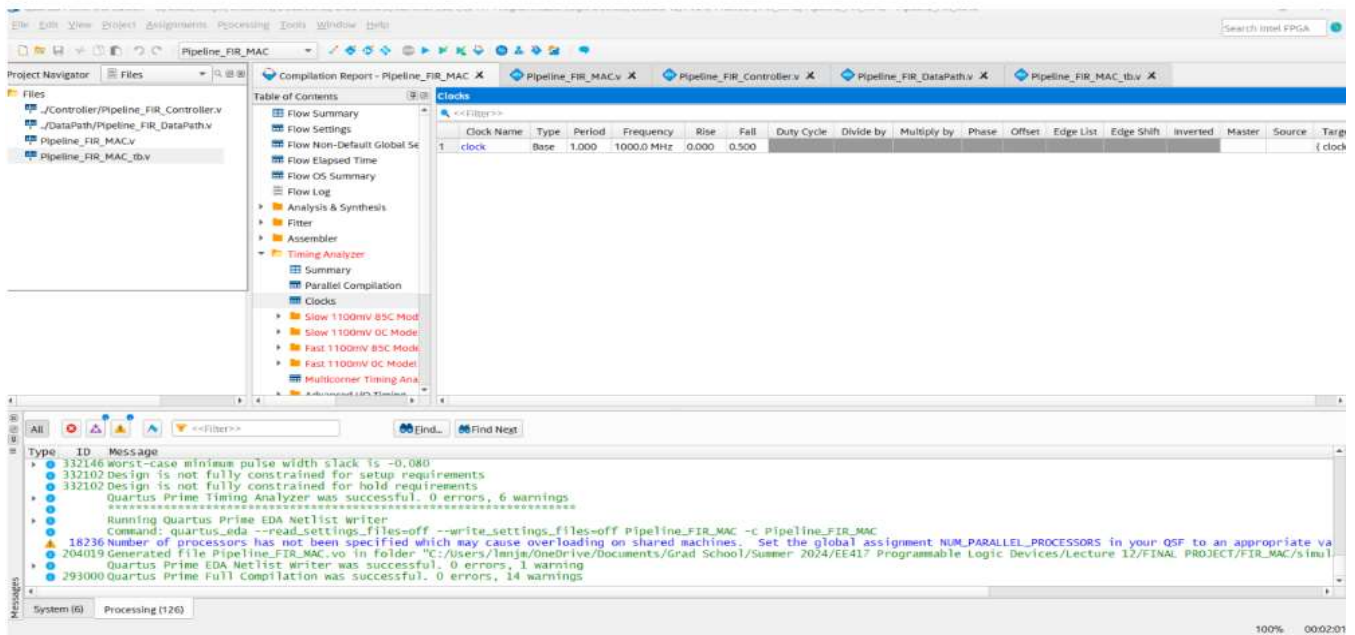
Testbench results from the text file

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

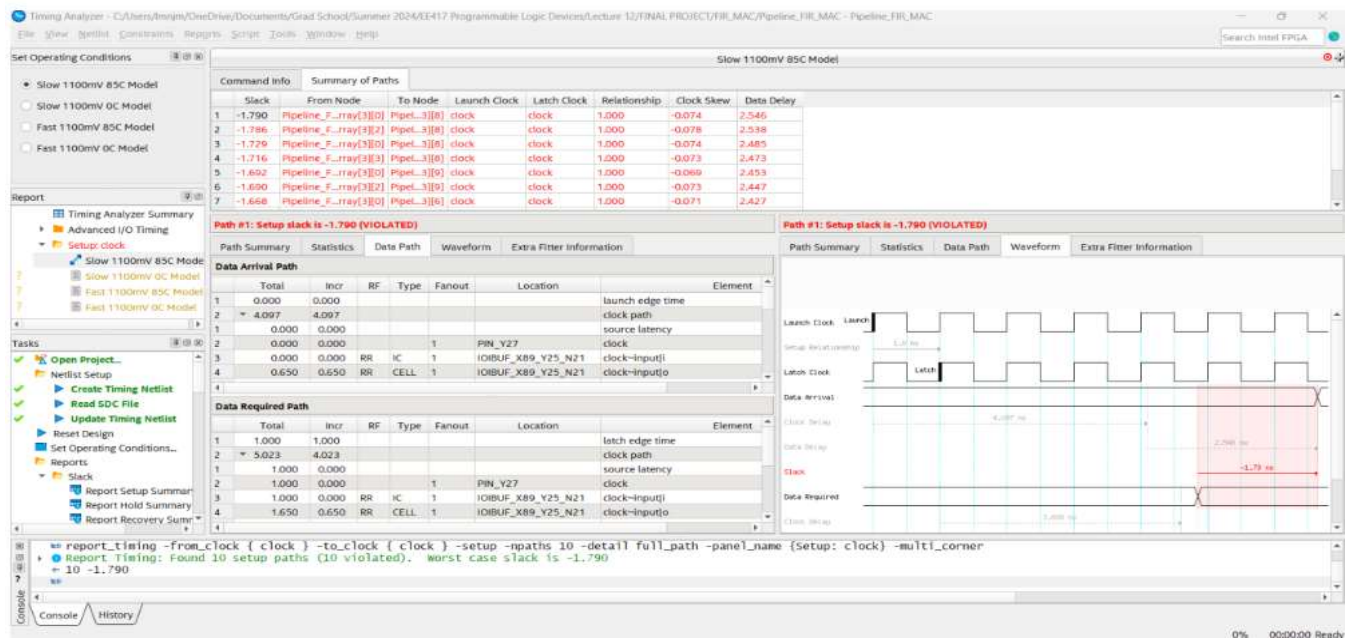
Timing Analysis Reports

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/lanjia/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

Files

- Pipeline_FIR_MAC.out.sdc
- J/Controller/Pipeline_FIR_Controller.v
- J/DataPath/Pipeline_FIR_DataPath.v
- Pipeline_FIR_MAC.v
- Pipeline_FIR_MAC.tb.v

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 OC Model
 - Fast 1100mV 85C Model
 - Fast 1100mV OC Model
 - Multiuser Timing Analysis

Table of Contents

Clock Name	Type	Period	Frequency	Rise	Fall	Duty Cycle	Divide by	Multiply by	Phase	Offset	Edge List	Edge Shift	Inverted	Master	Source	Target
1 clock	Base	4.000	250.0 MHz	0.000	2.000											(clock

Messages

Type: ID Message

- 332146 Worst-case minimum pulse width slack is 1.420
- 332102 Design is not fully constrained for setup requirements
- 332102 Design is not fully constrained for hold requirements
- Quartus Prime Fitting Analyzer was successful. 0 errors, 1 warning
- Running Quartus Prime EDA Netlist Writer
- Command: quartus.eda --read_settings_files=off --write_settings_files=off Pipeline_FIR_MAC -c Pipeline_FIR_MAC
- 18216 Number of processors has not been specified which may cause overloading on shared machines. Set the global assignment NUM_PARALLEL_PROCESSORS in your QSF to an appropriate value.
- 204019 Generated file Pipeline_FIR_MAC.vo in folder "C:/Users/lanjia/OneDrive/Documents/Grad School/Summer 2024/EE417 Programmable Logic Devices/Lecture 12/FINAL PROJECT/FIR_MAC/s18u1"
- Quartus Prime EDA Netlist Writer was successful. 0 errors, 1 warning
- 293000 Quartus Prime Full compilation was successful. 0 errors, 8 warnings

System (6) Processing (118)

100% 00:01:35

Time report at a slack of 1.096

Timing Analyzer - C:/Users/lanjia/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 OC Model
- Fast 1100mV 85C Model
- Fast 1100mV OC Model

Report

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

Tasks

- Open Project...
- 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

Slack	From Node	To Node	Launch Clock	Latch Clock	Relationship	Clock Skew	Data Delay
1 1.096	Pipeline_F_array[3][1] Pipel_3[R]	clock	clock	clock	4.000	-0.080	2.654
2 1.108	Pipeline_F_array[3][2] Pipel_3[R]	clock	clock	clock	4.000	-0.074	2.648
3 1.111	Pipeline_F_array[3][0] Pipel_3[R]	clock	clock	clock	4.000	-0.074	2.645
4 1.131	Pipeline_F_array[3][3] Pipel_3[R]	clock	clock	clock	4.000	-0.079	2.620
5 1.136	Pipeline_F_array[3][2] Pipel_3[R]	clock	clock	clock	4.000	-0.074	2.620
6 1.138	Pipeline_F_array[3][0] Pipel_3[R]	clock	clock	clock	4.000	-0.074	2.618
7 1.148	Pipeline_F_array[3][1] Pipel_3[R]	clock	clock	clock	4.000	-0.080	2.602

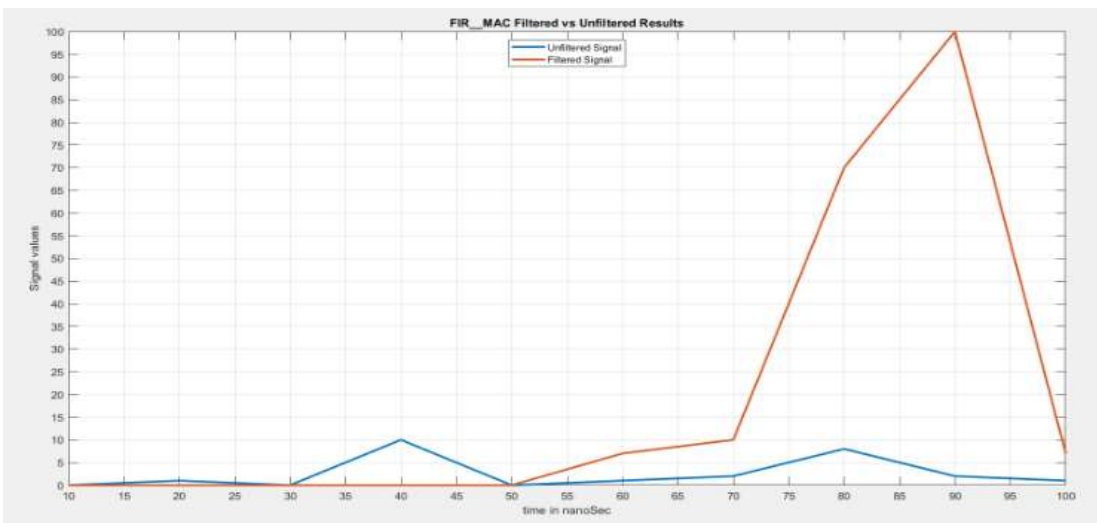
Path #1: Setup slack is 1.096

Path Summary	Statistics	Data Path	Waveform	Extra Filter Information
Data Arrival Path				
Total	Incr	RF	Type	Fanout
1	0.000	0.000		
2	4.104	4.104		
3	0.000	0.000		
4	0.000	0.000		
5	0.000	0.000		
6	0.000	0.000		
7	0.000	0.000		
8	0.000	0.000		
9	0.000	0.000		
10	0.000	0.000		
11	0.000	0.000		
12	0.000	0.000		
13	0.000	0.000		
14	0.000	0.000		
15	0.000	0.000		
16	0.000	0.000		
17	0.000	0.000		
18	0.000	0.000		
19	0.000	0.000		
20	0.000	0.000		
21	0.000	0.000		
22	0.000	0.000		
23	0.000	0.000		
24	0.000	0.000		
25	0.000	0.000		
26	0.000	0.000		
27	0.000	0.000		
28	0.000	0.000		
29	0.000	0.000		
30	0.000	0.000		
31	0.000	0.000		
32	0.000	0.000		
33	0.000	0.000		
34	0.000	0.000		
35	0.000	0.000		
36	0.000	0.000		
37	0.000	0.000		
38	0.000	0.000		
39	0.000	0.000		
40	0.000	0.000		
41	0.000	0.000		
42	0.000	0.000		
43	0.000	0.000		
44	0.000	0.000		
45	0.000	0.000		
46	0.000	0.000		
47	0.000	0.000		
48	0.000	0.000		
49	0.000	0.000		
50	0.000	0.000		
51	0.000	0.000		
52	0.000	0.000		
53	0.000	0.000		
54	0.000	0.000		
55	0.000	0.000		
56	0.000	0.000		
57	0.000	0.000		
58	0.000	0.000		
59	0.000	0.000		
60	0.000	0.000		
61	0.000	0.000		
62	0.000	0.000		
63	0.000	0.000		
64	0.000	0.000		
65	0.000	0.000		
66	0.000	0.000		
67	0.000	0.000		
68	0.000	0.000		
69	0.000	0.000		
70	0.000	0.000		
71	0.000	0.000		
72	0.000	0.000		
73	0.000	0.000		
74	0.000	0.000		
75	0.000	0.000		
76	0.000	0.000		
77	0.000	0.000		
78	0.000	0.000		
79	0.000	0.000		
80	0.000	0.000		
81	0.000	0.000		
82	0.000	0.000		
83	0.000	0.000		
84	0.000	0.000		
85	0.000	0.000		
86	0.000	0.000		
87	0.000	0.000		
88	0.000	0.000		
89	0.000	0.000		
90	0.000	0.000		
91	0.000	0.000		
92	0.000	0.000		
93	0.000	0.000		
94	0.000	0.000		
95	0.000	0.000		
96	0.000	0.000		
97	0.000	0.000		
98	0.000	0.000		
99	0.000	0.000		
100	0.000	0.000		
101	0.000	0.000		
102	0.000	0.000		
103	0.000	0.000		
104	0.000	0.000		
105	0.000	0.000		
106	0.000	0.000		
107	0.000	0.000		
108	0.000	0.000		
109	0.000	0.000		
110	0.000	0.000		
111	0.000	0.000		
112	0.000	0.000		
113	0.000	0.000		
114	0.000	0.000		
115	0.000	0.000		
116	0.000	0.000		
117	0.000	0.000		
118	0.000	0.000		
119	0.000	0.000		
120	0.000	0.000		
121	0.000	0.000		
122	0.000	0.000		
123	0.000	0.000		
124	0.000	0.000		
125	0.000	0.000		
126	0.000	0.000		
127	0.000	0.000		
128	0.000	0.000		
129	0.000	0.000		
130	0.000	0.000		
131	0.000	0.000		
132	0.000	0.000		
133	0.000	0.000		
134	0.000	0.000		
135	0.000	0.000		
136	0.000	0.000		
137	0.000	0.000		
138	0.000	0.000		
139	0.000	0.000		
140	0.000	0.000		
141	0.000	0.000		
142	0.000	0.000		
143	0.000	0.000		
144	0.000	0.000		
145	0.000	0.000		
146	0.000	0.000		
147	0.000	0.000		
148	0.000	0.000		
149	0.000	0.000		
150	0.000	0.000		
151	0.000	0.000		
152	0.000	0.000		
153	0.000	0.000		
154	0.000	0.000		
155	0.000	0.000		
156	0.000	0.000		
157	0.000	0.000		
158	0.000	0.000		
159	0.000	0.000		
160	0.000	0.000		
161	0.000	0.000		
162	0.000	0.000		
163	0.000	0.000		
164	0.000	0.000		
165	0.000	0.000		
166	0.000	0.000		
167	0.000	0.000		
168	0.000	0.000		
169	0.000	0.000		
170	0.000	0.000		
171	0.000	0.000		
172	0.000	0.000		
173	0.000	0.000		
174	0.000	0.000		
175	0.000	0.000		
176	0.000	0.000		
177	0.000	0.000		
178	0.000	0.000		
179	0.000	0.000		
180	0.000	0.000		
181	0.000	0.000		
182	0.000	0.000		
183	0.000	0.000		
184	0.000	0.000		
185	0.000	0.000		
186	0.000	0.000		
187	0.000	0.000		
188	0.000	0.000		
189	0.000	0.000		
190	0.000	0.000		
191	0.000	0.000		
192	0.000	0.000		
193	0.000	0.000		
194	0.000	0.000		
195	0.000	0.000		
196	0.000	0.000		
197	0.000	0.000		
198	0.000	0.000		
199	0.000	0.000		
200	0.000	0.000		
201	0.000	0.000		
202	0.000	0.000		

Matlab Testbench Data input and Results

```
-----  
% Name Lamin Jammeh  
% Class: EE417 Summer 2024  
% Lesson 10 HW Question 3  
% Group: Ron Kalin/ Lamin Jammeh  
% Project Description: The Testbench results are plotted 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, 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;
```



Conclusion

Reset high means Sample_input, internal registers, and output registers become zero.
When reset is low FIR filter is active. When filter is active, the actions happen on the positive clock edge.

FIR filter with pipeline registers were placed all adder inputs.
Simulation results show each pipelining register passes the values saved in its array to the next one.
The top two values are added together while the rest pass as is to the next register. The throughput (new output) is available at every clock cycle.
With the pipelining, the longest propagation delay for the combinational logic between the registers can be shortened, which will consequently allow reducing clock period and increasing clock frequency. With an output available at every clock cycle, the throughput of the module increases. The pipelining improves hardware utilization efficiency.

Sources

- [1] Wikipedia "Multiply-accumulate operation"
- [2] Ciletti, Michael D.. Advanced Digital Design with the Verilog HDL (p. 583). Pearson Education. Kindle Edition.