Experiment 5

EE24mtech12008
Mannava Venkata saj

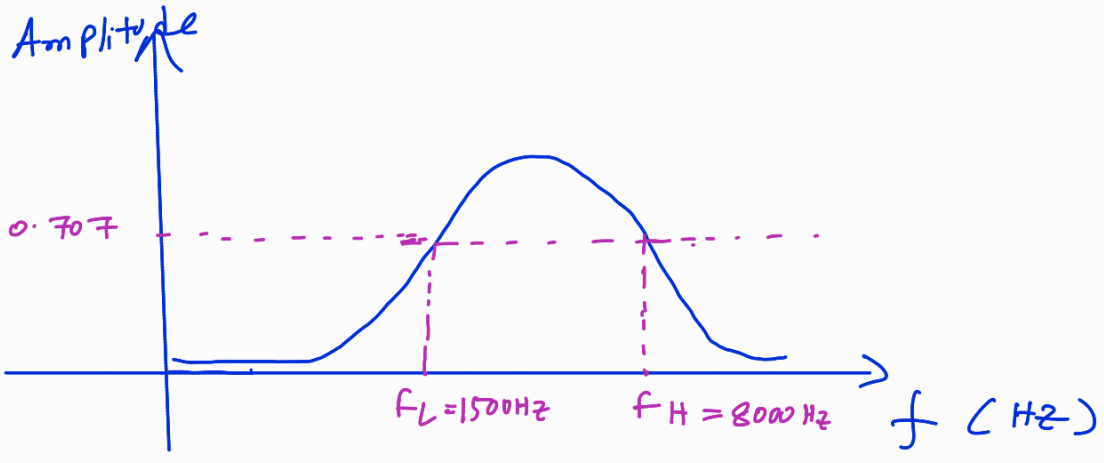
Design a band pass filter with the following specifications

Sampling frequency = 48000 Hz
$f_{stop\,1}$ = 500Hz
$f_{pass\,1}$ = 1500Hz
$f_{pass\,2}$ = 8000 Hz
$f_{stop\,2}$ = 9000 Hz

With the above specifications

→ The filter will allow frequencies between 1500 Hz and 8000 Hz

→ The filter will reject the frequencies below 500 Hz and above 9000 Hz

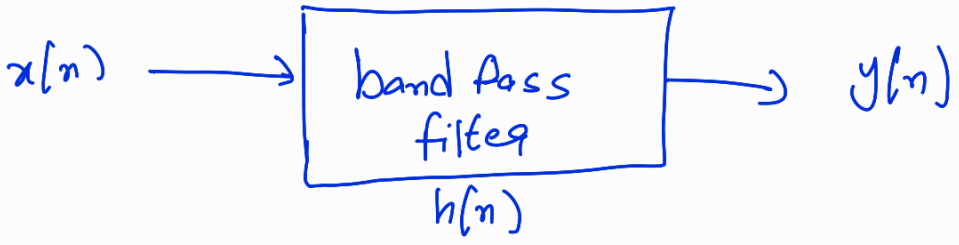


f_L = lower cutoff frequency = 1500Hz
f_H = upper Cutoff frequency = 8000 Hz

Generate the filter coefficients of the band pass filter from matlab with the above specifications

let $h(n)$ be the system response

let $x(n)$ be the input signal of length L and $h(n)$ be the system response of length M



$$y(n) = x(n) * h(n) = \sum_{k=0}^{L-1} x(k)\, h(n-k)$$

$$n = 0 \text{ to } L+M-1$$

This in matlab can be implement using "conv" command which will do the convolution of two signals.

→ we apply a filter to the signal of length L and response length M. the output signal we should take from $\frac{M}{2}$ to $\frac{M}{2}+L$

i.e $y(n) = x(n) * h(n)$

$y'(n) = y\left[\frac{M}{2} : \frac{M}{2}+L\right]$

→ Inorder to have the proper output of the filter we should make sure that $\boxed{L \geq M}$

Implementation FIR filter in Verilog in non Pipelining

→ initialize the shift register to zero

Shift_reg[N] = 8'b0;

→ All the Filter coefficients are predifined

→ Counter = 0

→ for every rising edge of clock

→ for(i = N-1; i > 0; i = i-1)

Shift_reg(i) = shift_reg(i-1);

if (counter < N)

Shift_reg(0) = input signal

counter = counter + 1

else

Shift_reg(0) = 0

```
acc = 0
for(i=0; i<N; i++)
    temp = shift_reg[i] * h[i]
    acc = acc + temp

output_signal = acc;
```

The above algorithm is ran in the intel quartus.

The resources utilized and fmax summary is given below.

Flow Summary

<<Filter>>

Flow Status	Successful - Thu Mar 27 23:11:24 2025
Quartus Prime Version	23.1std.1 Build 993 05/14/2024 SC Lite Edition
Revision Name	fir2
Top-level Entity Name	fir_filter_ven
Family	Cyclone V
Device	5CGXFC9A6U19C7
Timing Models	Final
Logic utilization (in ALMs)	859 / 113,560 (< 1 %)
Total registers	2001
Total pins	35 / 264 (13 %)
Total virtual pins	0
Total block memory bits	0 / 12,492,800 (0 %)
Total DSP Blocks	123 / 342 (36 %)
Total HSSI RX PCSs	0 / 5 (0 %)
Total HSSI PMA RX Deserializers	0 / 5 (0 %)
Total HSSI TX PCSs	0 / 5 (0 %)
Total HSSI PMA TX Serializers	0 / 5 (0 %)
Total PLLs	0 / 13 (0 %)
Total DLLs	0 / 4 (0 %)

Fmax Summary

Slow 1100mV 85C Model Fmax Summary

<<Filter>>

	Fmax	Restricted Fmax	Clock Name	Note
1	61.3 MHz	61.3 MHz	clk	

with non pipelining approach the tolerable clock frequency for the circuit is 63.1MHz

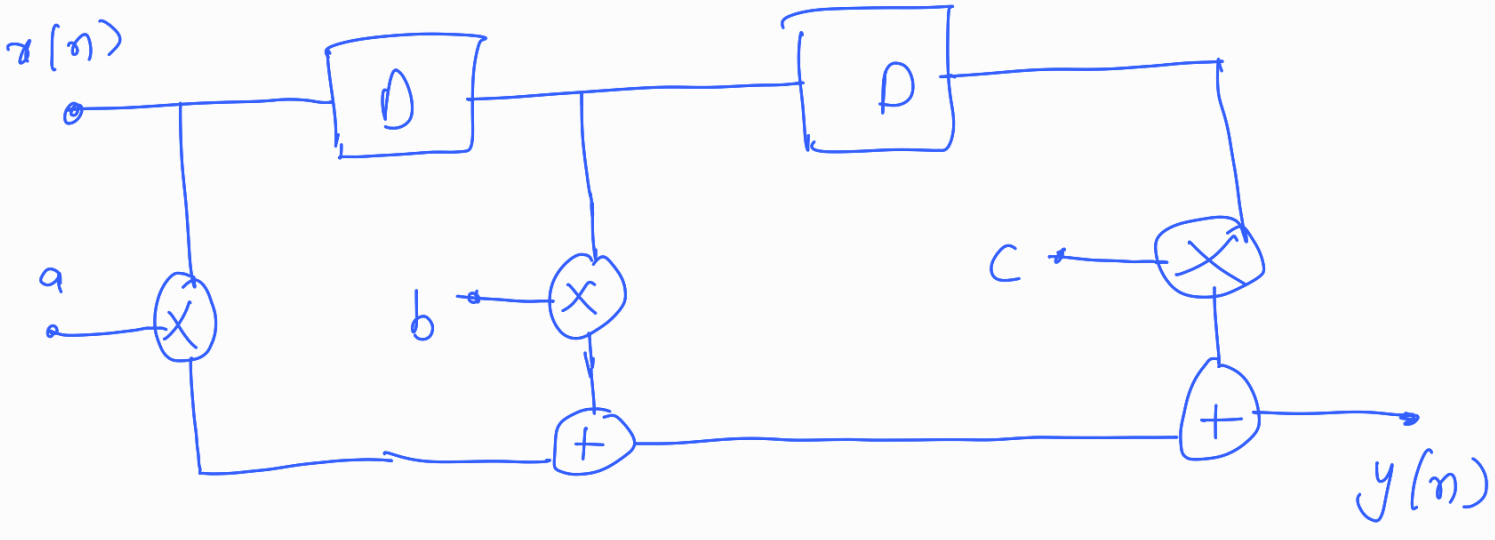
Implementation of FIR filter Using Pipelining

Pipelining :-

Pipelining transformation leads to a reduction in the critical path, which can be exploited to either increase in the Clock speed or Sample speed to reduce the power Consumption at same speed.

Consider an fir filter with three taps

$$y(n) = a x(n) + b x(n-1) + c x(n-2)$$



$$y[n] = \sum_{k=0}^{N-1} h(k)\, x[n-k]$$

if $N = 3$

$y(0) = h(0)\,x(0) + (\quad) + (\quad)$

$y(1) = x(1)\,h(0) + x(0)\,h(1) + (\quad)$

$y(2) = x(2)\,h(0) + x(1)\,h(1) + x(0)\,h(2)$

$y(3) = x(2)\,h(1) + x(1)\,h(2) + (\quad)$

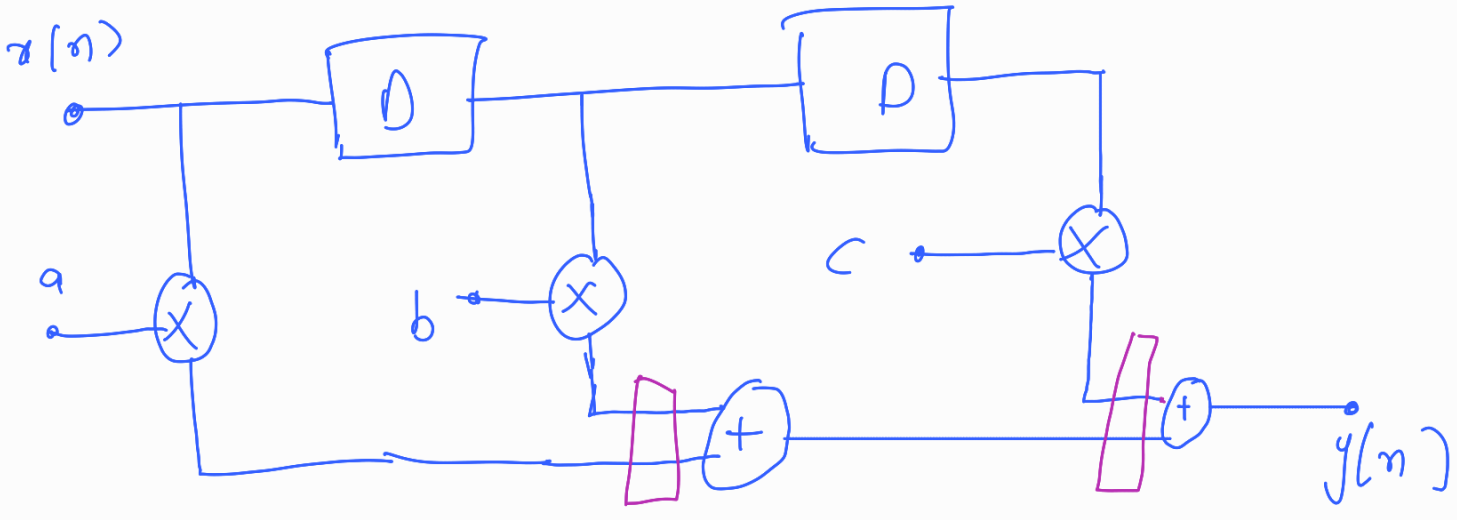
$y(4) = x(2)\,h(2) + (\quad) + (\quad)$

In the non Pipelining approach all the multiplications and additions are happening in only one clock cycle.

Now we apply Pipelining to increase the Fmax

→ Instead of Computing additions and multiplications in same clock cycle Compute all the multiplications in One clock cycle and compute all additions in another clock cycle.

Apply pipe lining



Algorithm for implementing fir filter in Pipelining in Verilog

→ initialize the shift register to zero

Shift_reg[N] = 8'b0;

→ All the Filter coefficients are predifined

→ Counter = 0

→ for every rising edge of clock

→ for(i = N-1; i > 0; i = i-1)

Shift_reg(i) = shift_reg(i-1);

if (counter < N)

Shift_reg(0) = input signal

Counter = counter + 1

else

Shift_reg(0) = 0

```
acc = 0
for (i=0; i< N; i = i+1)
    mul_reg[i] = shift_reg(i) * h(i)

for (i=0; i< N; i = i+1)
    acc = acc + mul_reg(i)

output_signal = acc;
```

Fmax summary for FIR filter Using Pipelining.

Slow 1100mV 85C Model Fmax Summary

C:/Users/kanek/OneDrive/Desktop/quartus_codes/FIR2/src/fir_filter_pp.v

	Fmax	Restricted Fmax	Clock Name	Note
1	72.99 MHz	72.99 MHz	clk	

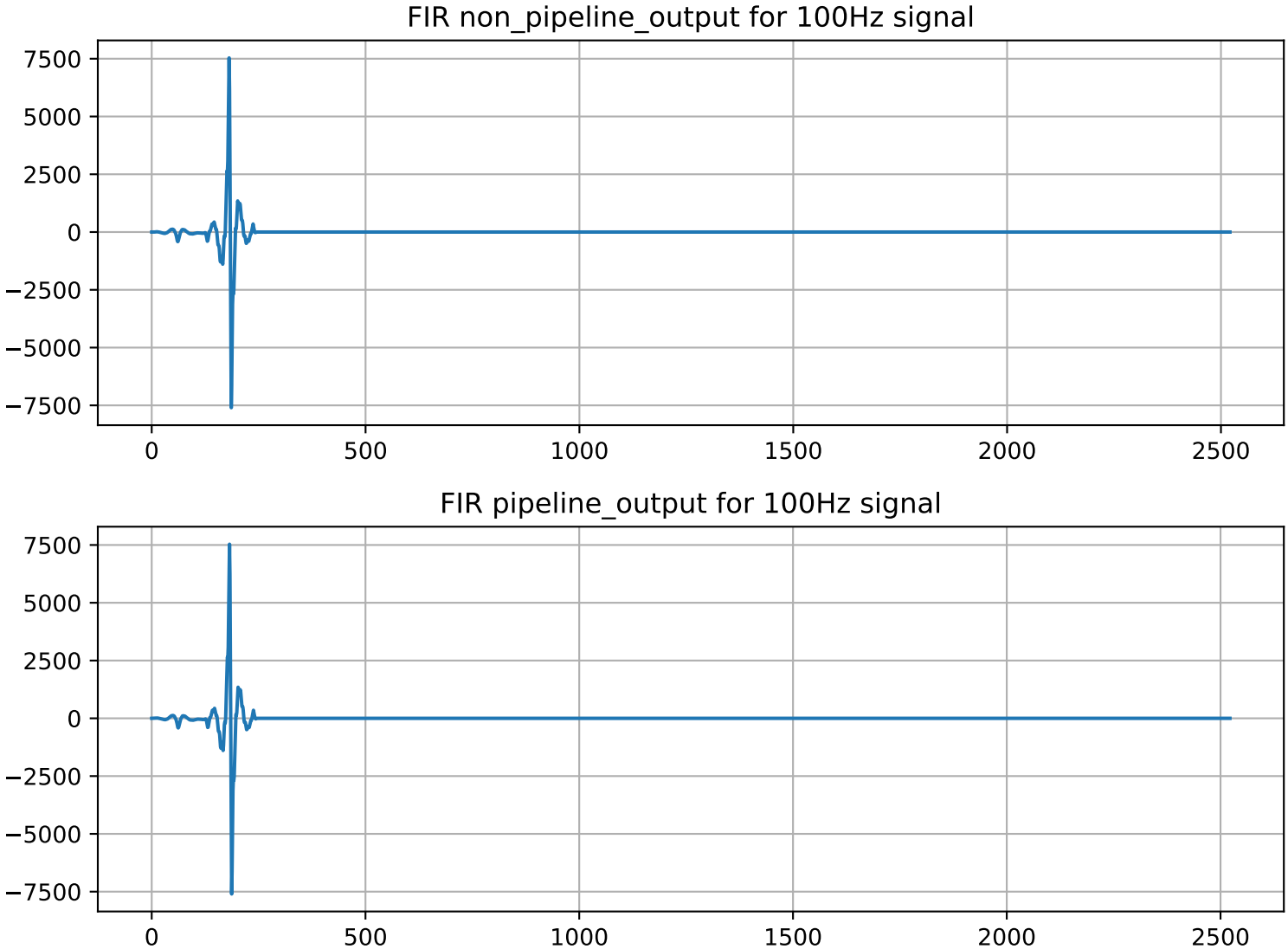
Using the Pipelining approach the fmax is increased from 61.3MHz to 72.99MHz.

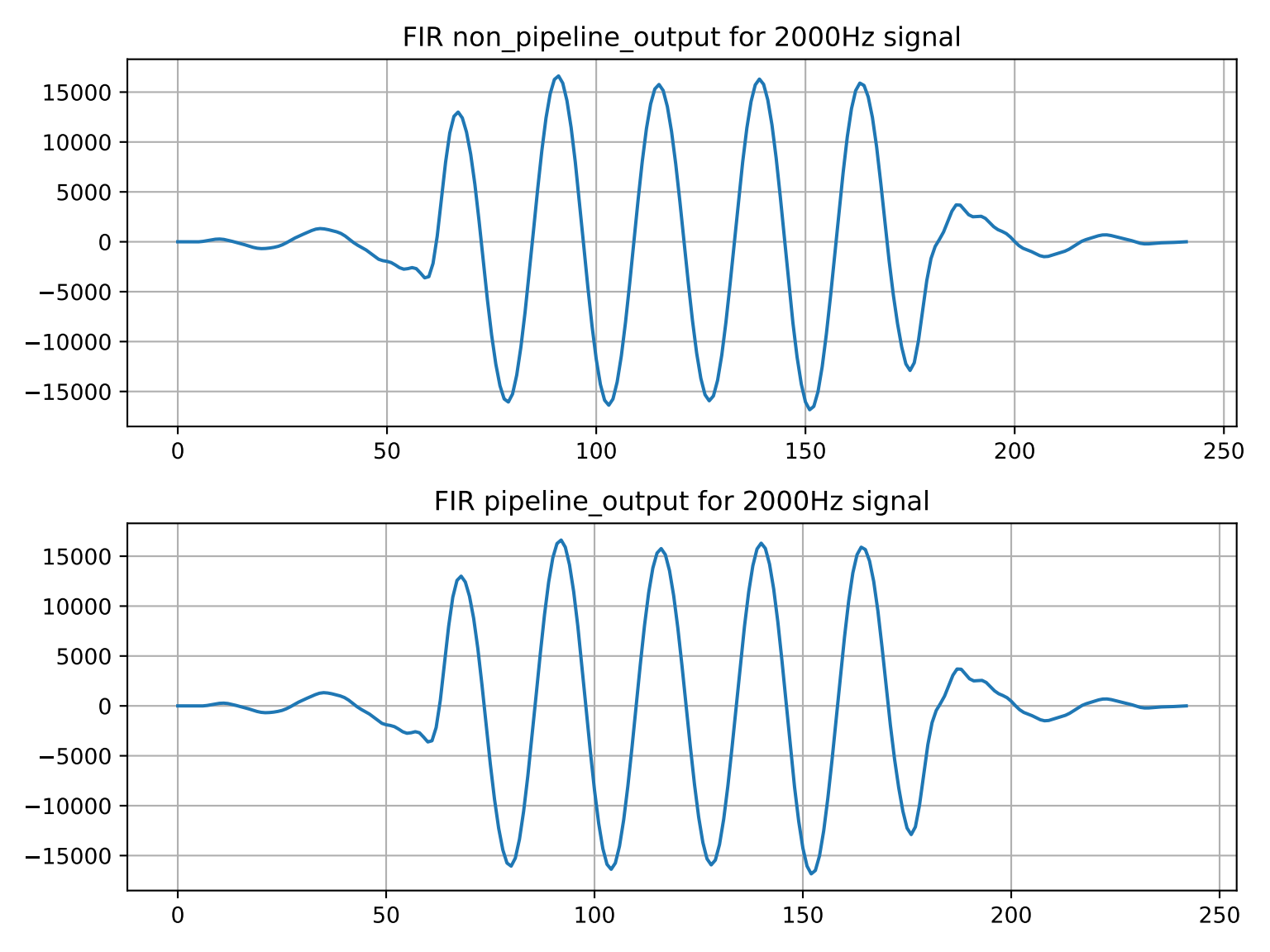
Resource Utilization for FIR filter Using Pipelining

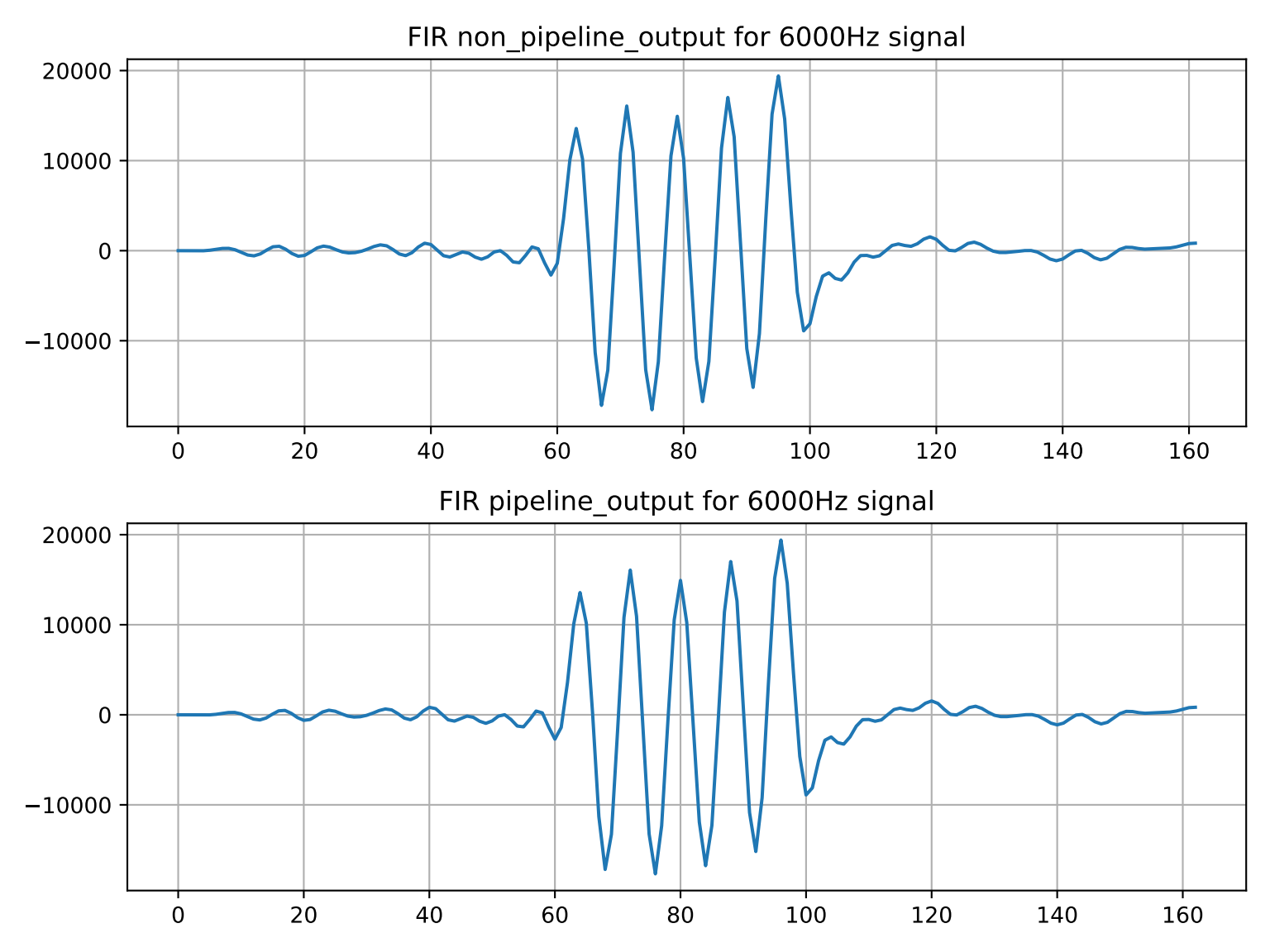
Flow Summary

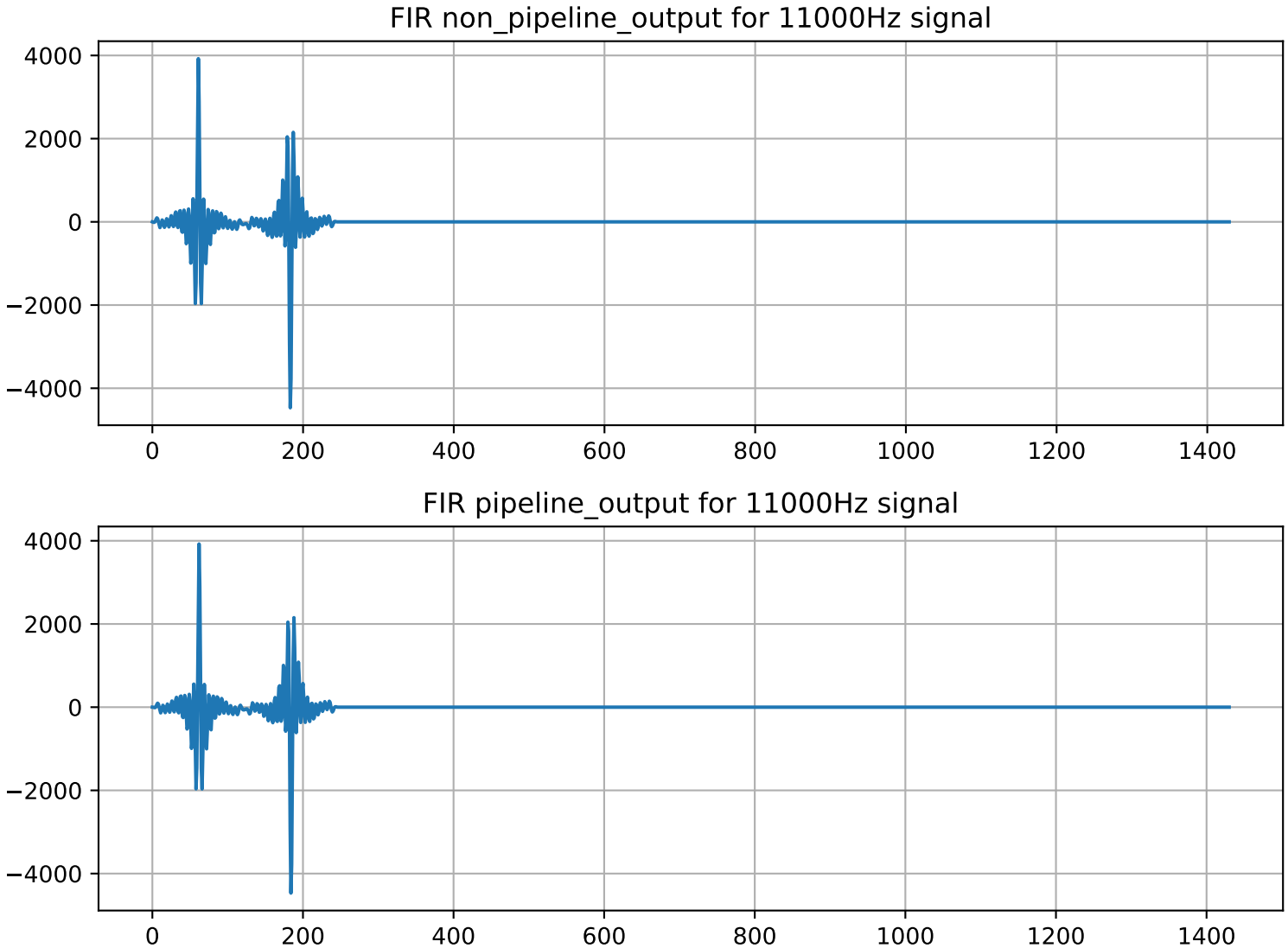
<<Filter>>

Flow Status	Successful - Thu Mar 27 23:11:24 2025
Quartus Prime Version	23.1std.1 Build 993 05/14/2024 SC Lite Edition
Revision Name	fir2
Top-level Entity Name	fir_filter_ven
Family	Cyclone V
Device	5CGXFC9A6U19C7
Timing Models	Final
Logic utilization (in ALMs)	859 / 113,560 (< 1 %)
Total registers	2001
Total pins	35 / 264 (13 %)
Total virtual pins	0
Total block memory bits	0 / 12,492,800 (0 %)
Total DSP Blocks	123 / 342 (36 %)
Total HSSI RX PCSs	0 / 5 (0 %)
Total HSSI PMA RX Deserializers	0 / 5 (0 %)
Total HSSI TX PCSs	0 / 5 (0 %)
Total HSSI PMA TX Serializers	0 / 5 (0 %)
Total PLLs	0 / 13 (0 %)
Total DLLs	0 / 4 (0 %)


FIR non_pipeline_output for 100Hz signal
7500
5000
2500
0
−2500
−5000
−7500
0
500
1000
1500
2000
2500
FIR pipeline_output for 100Hz signal
7500
5000
2500
0
−2500
−5000
−7500
0
500
1000
1500
2000
2500


FIR non_pipeline_output for 2000Hz signal
15000
10000
5000
0
−5000
−10000
−15000
0
50
100
150
200
250
FIR pipeline_output for 2000Hz signal
15000
10000
5000
0
−5000
−10000
−15000
0
50
100
150
200
250


FIR non_pipeline_output for 6000Hz signal
20000
10000
0
−10000
0
20
40
60
80
100
120
140
160
FIR pipeline_output for 6000Hz signal
20000
10000
0
−10000
0
20
40
60
80
100
120
140
160


FIR non_pipeline_output for 11000Hz signal
4000
2000
0
−2000
−4000
0
200
400
600
800
1000
1200
1400
FIR pipeline_output for 11000Hz signal
4000
2000
0
−2000
−4000
0
200
400
600
800
1000
1200
1400