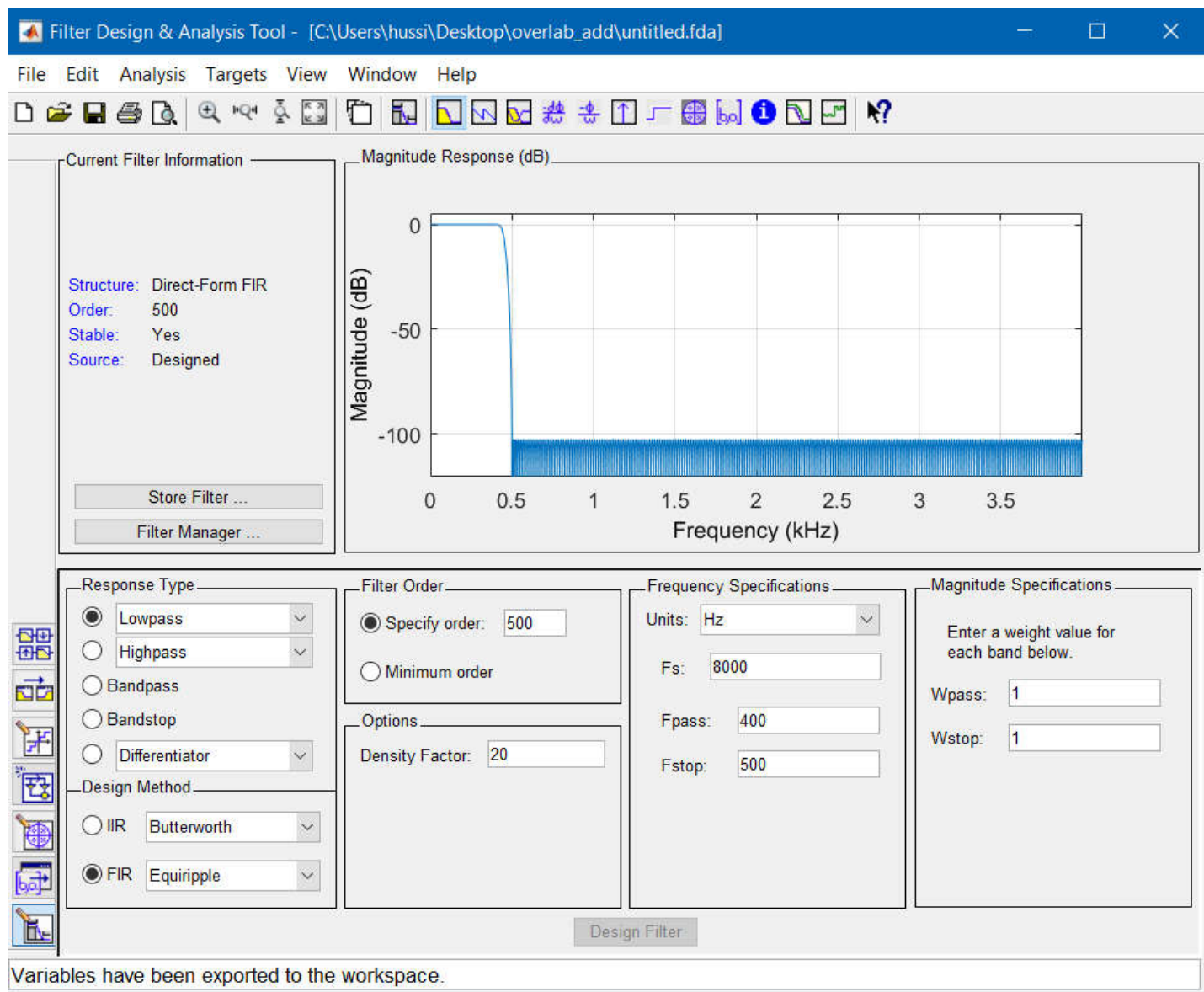
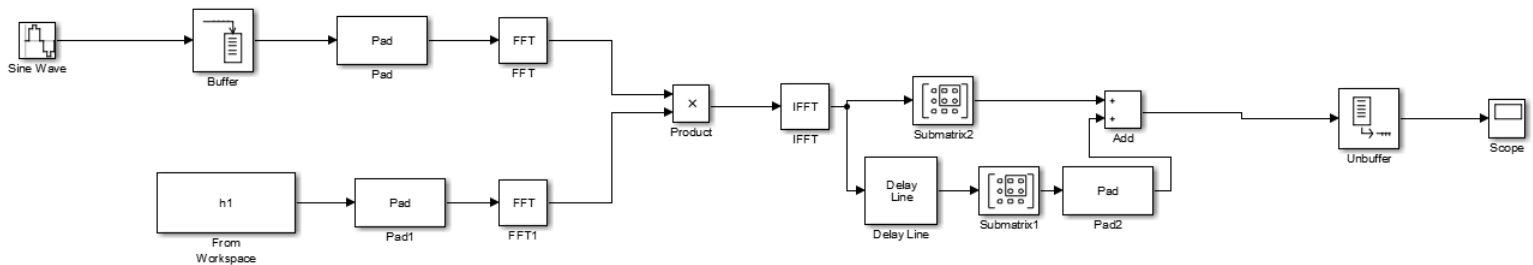


## Overlap-add:

Filter coefficients:



## Overlap-add implementation:



### Walkthrough:

- Converting the input signal into blocks with the “Buffer”, length of block = 1548 “L”
- Hence size of FFT =  $L+M-1$  which = 2048, we will add padding for both the block and the filter “h1”.

Pad value:  
0

Output column mode: User-specified

Column size:  
2048

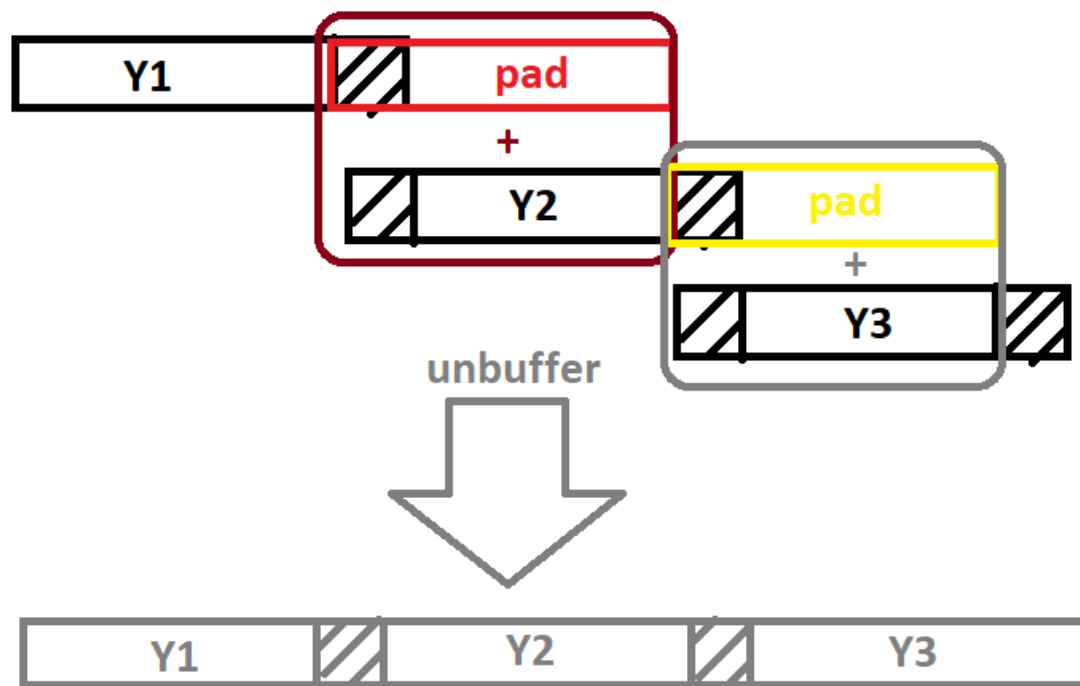
- Now we can make FFT for each “the signal and the filter” then we can multiply outputs with the product block.
- This is the output of a block of the signal convolved with the filter.
- To return the output to time domain we use “IFFT Block”.
- We want to add the blocks together to retrieve the desired output of the whole signal rather than blocks of the signal.
- Using the submatrix blocks with these parameters:

Starting row: Index	Starting row: First
Starting row index: 1549	Ending row: Index
Ending row: Last	Ending row index: 1548

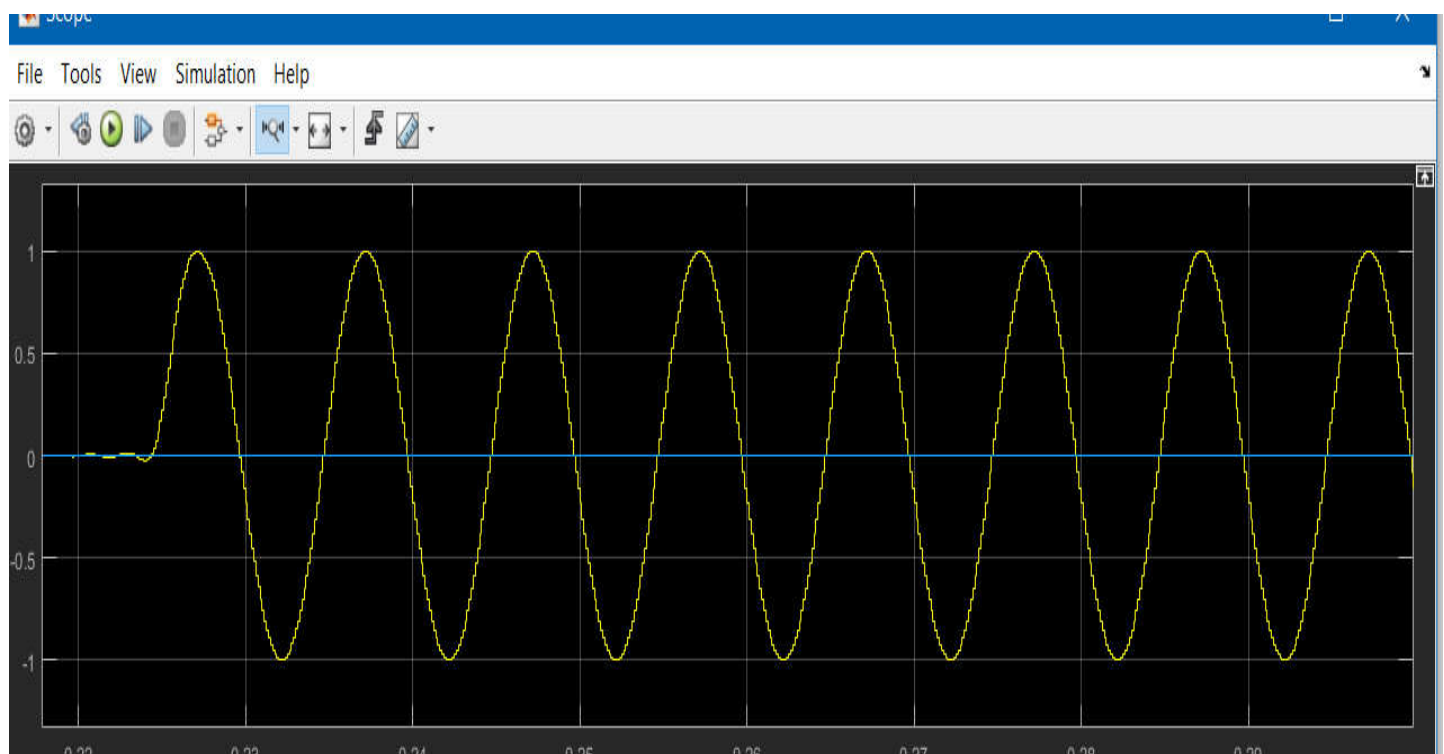
- Submatrix-1 size = 500, Submatrix-2 size = 1548.

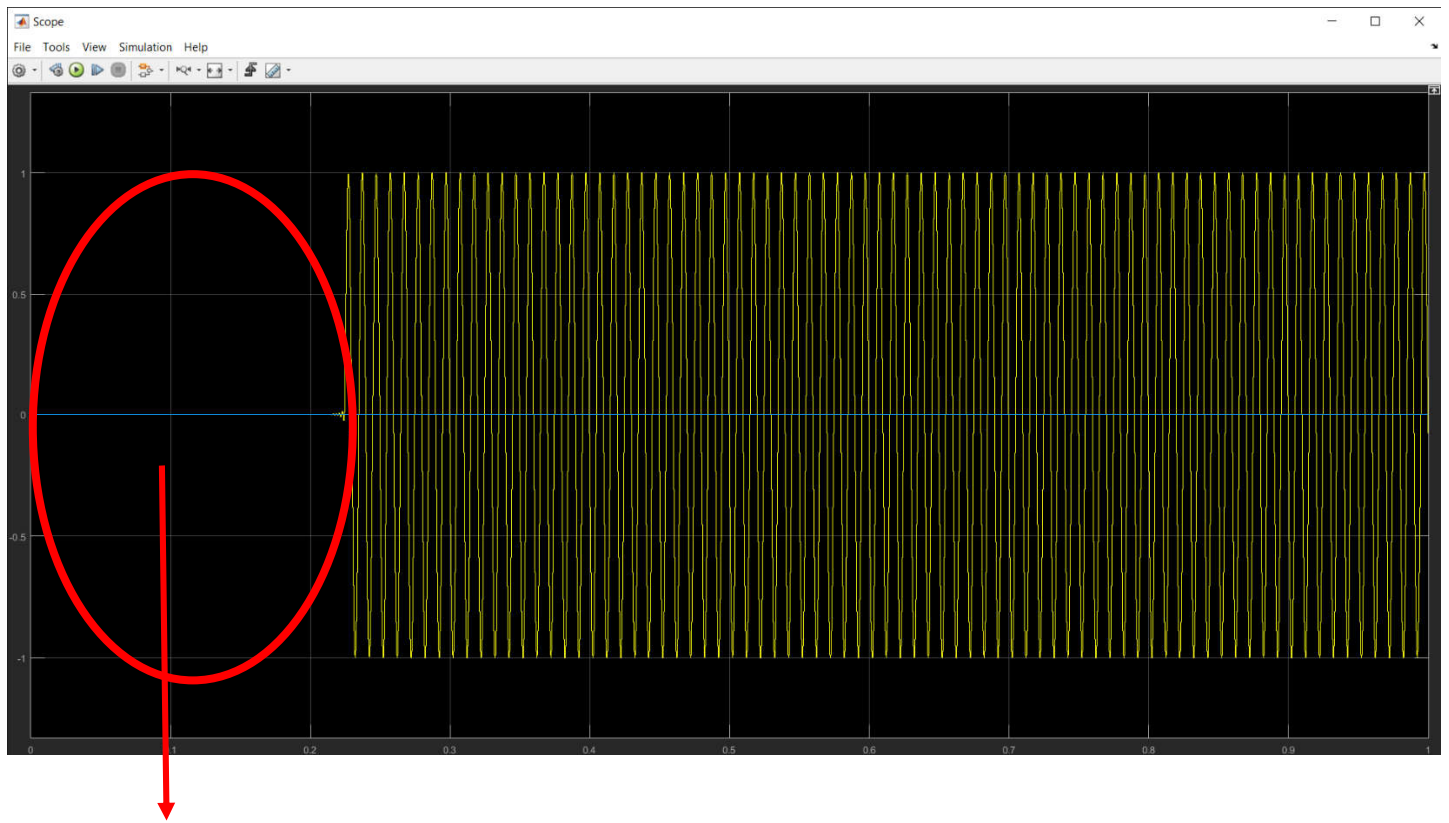
To achieve the overlap-add algorithm:

- we add delay block before “submatrix-1” and delay size = 2048
- padding submatrix-1 to have same length as submatrix-2
- Hence, they have same size now we can add them.
- “Unbuffer block “: converts it into a sample-based signal (the original format of the input)
- Then we can view output with “scope block”



Output:

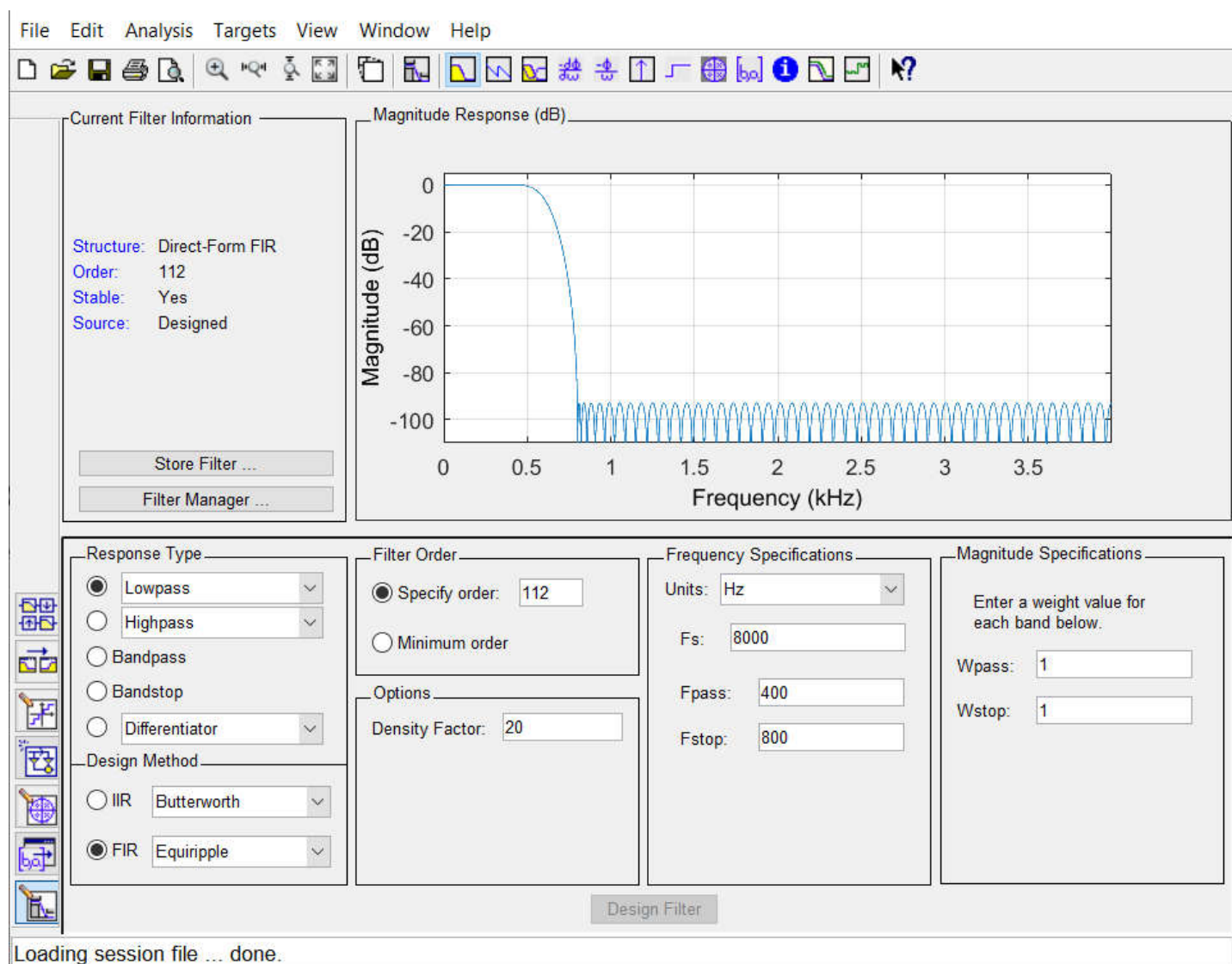




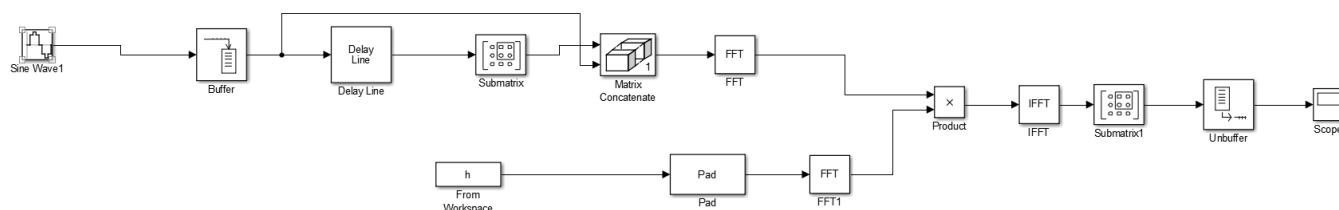
Delay due to the operations

## Overlap-save:

Filter coefficients:



## Overlap-save implementation:



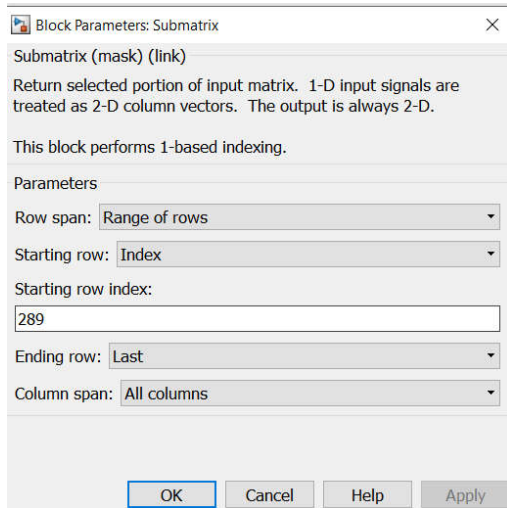
## Walkthrough:

- Converting the input signal into blocks with the “Buffer”, length of block =400 “L”

Output buffer size (per channel):

400

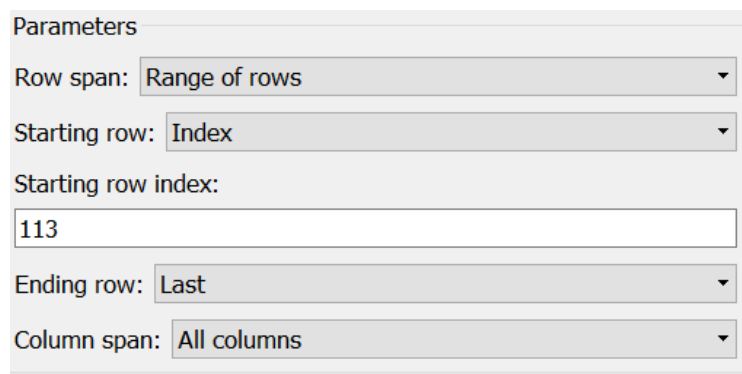
- We will delay every block of “L” size using “delay line block”
- We can choose last M-1 elements using submatrix block and append them to the first of next block.



- Then we concatenate padding output with buffer block to have length of 512
- On the other side we add padding to the filter coefficients matrix to be the same size of the concatenated matrix we had.
- We will make FFT to the two matrices then get the product of them.
- We will operate IFFT to the result to return it to time domain.

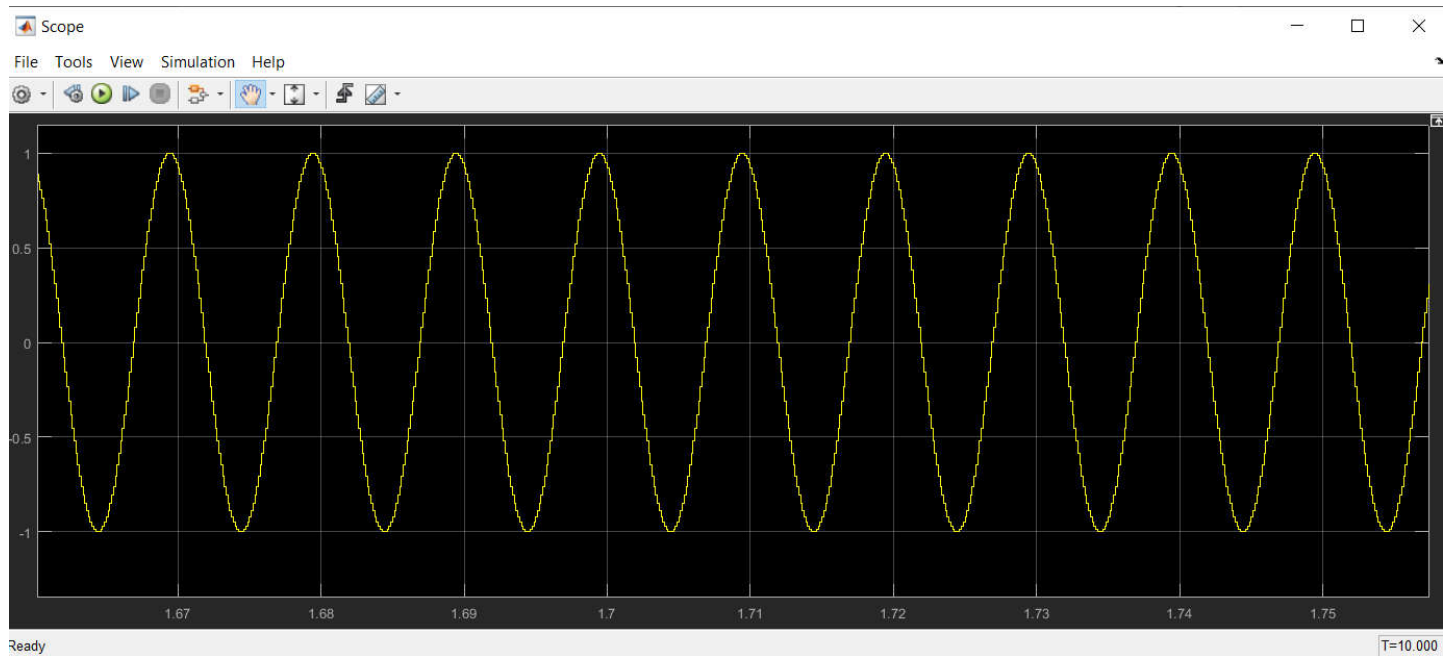
To achieve the overlap-save algorithm:

- We will remove M-1 elements from the first of every block using submatrix block.



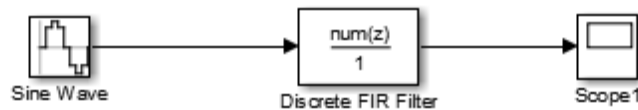
- Then we Unbuffer the result signal and display it on the scope.

Output:



## Linear Convolution: (We used 2 methods)

### \*First Method: Discrete FIR Filter



- This block will make linear Convolution Between the input signal and the filter coefficients (for this example we used coefficients of overlap-add).

Block Parameters: Discrete FIR Filter

Discrete FIR Filter

Independently filter each channel of the input over time using an FIR filter. You can specify filter coefficients using either tunable dialog parameters or separate input ports, which are useful for time-varying coefficients.

A DSP System Toolbox license is required to use a filter structure other than Direct Form.

Main Data Types

Coefficient source: Dialog parameters

Filter structure: Direct form

Coefficients: h1

Input processing: Elements as channels (sample based)

Initial states: 0

Control

☐ Show enable port

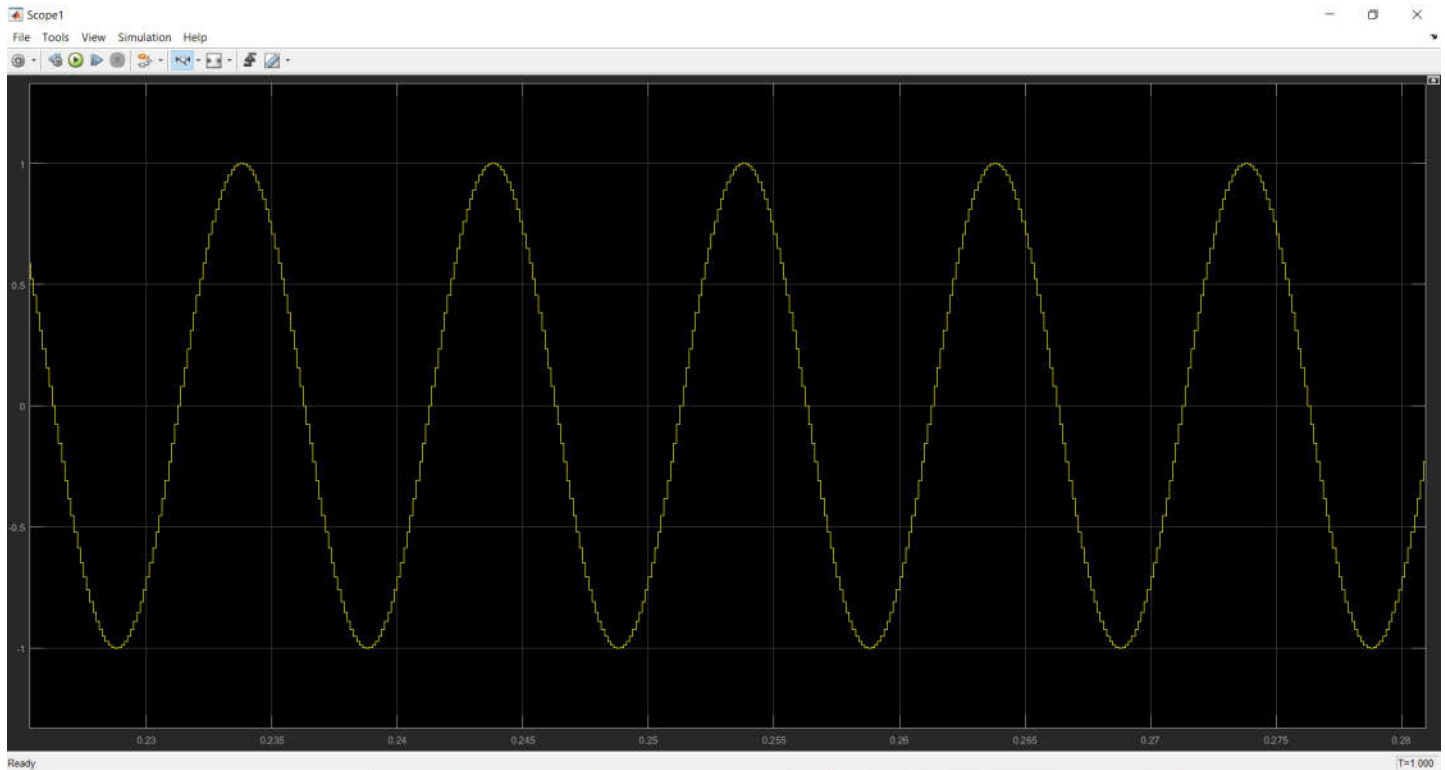
External reset: None

Sample time (-1 for inherited): -1

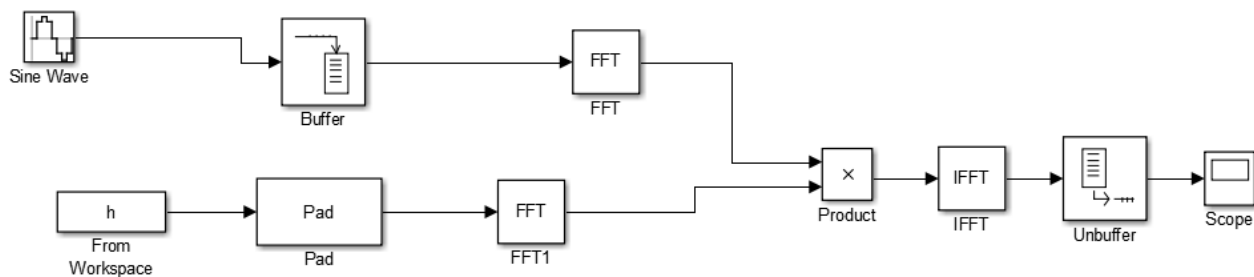
OK Cancel Help Apply



Output:

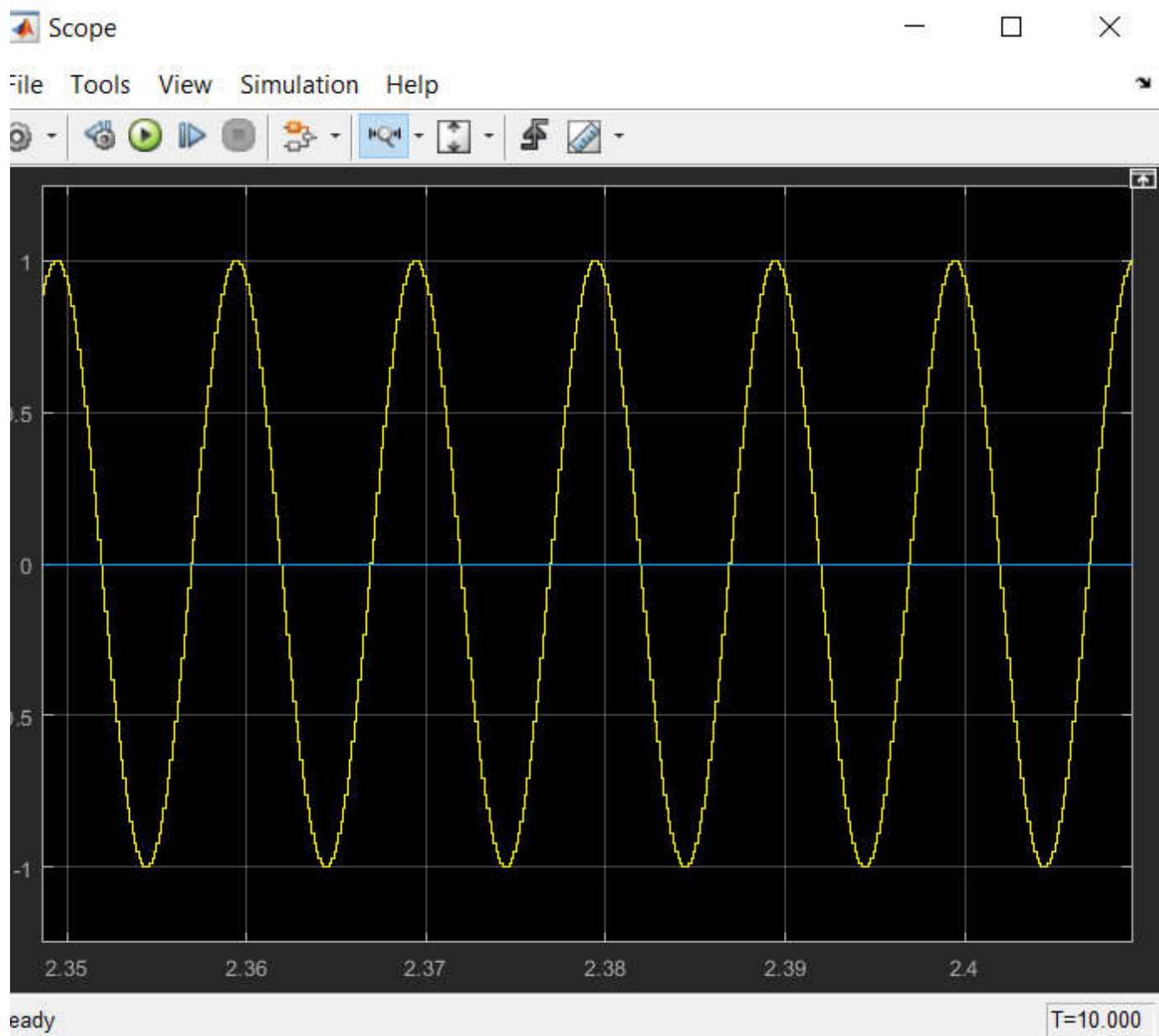


## \*Second Method:



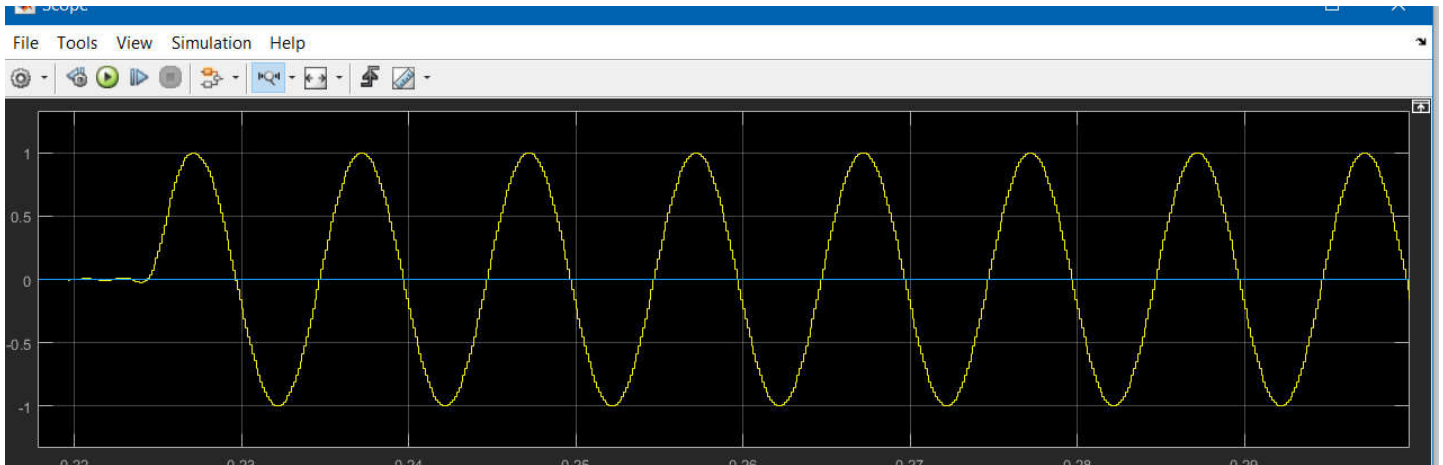
- We will convert the input signal into blocks of length 400 using buffer block then operate FFT on them.
- On the other side we will add padding filter coefficients (we used Overlap-save coefficients in this example) to make its size 400 then operate FFT on it too.
- We get the product of the 2 matrices then operate IFFT to return to time domain.
- Then we Unbuffer the result signal and display it on the scope block.

## Output:

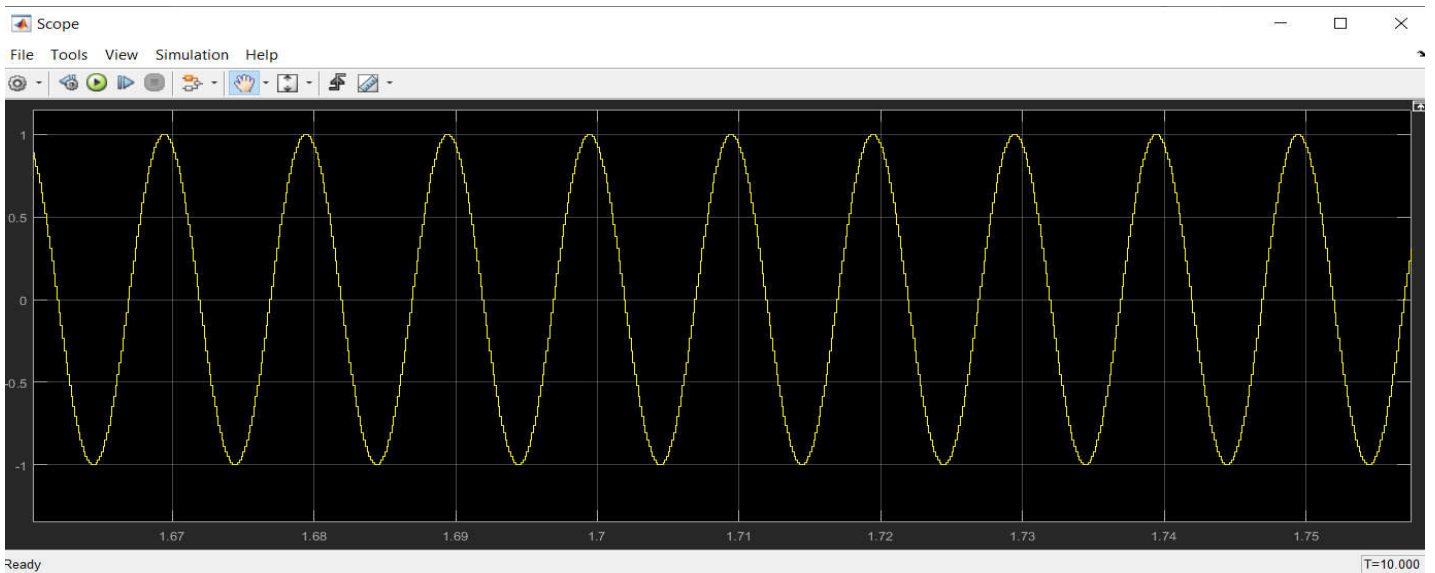


## Comparing Overlap-add, Overlap-save and Linear Conv. Outputs:

\*Overlap-add:



\*Overlap-save:



Linear Conv. (the two methods):

