

# Report

## 1. Theory of FIR, Convolutions, DTFT, etc.

\* FIR (Finite Impulse Response) Filter is a filter whose impulse Response is of finite duration, because it settles to zero in finite time.

\* They can be discrete-time or continuous-time & digital or analog.

\* For a Causal Discrete-time FIR filter of order  $N$ , each value of the output sequence is a weighted sum of the most recent input values.

$$y[n] = \sum_{i=0}^N b_i \cdot x[n-i]$$

\* The above computation is also known as discrete Convolution.

$$(f * g)[n] = \sum_{m=-\infty}^{\infty} f[m]g[n-m] \quad (\text{for infinite}) \quad \text{or} \quad = \sum_{m=-M}^M f(n-m)g(m) \quad (\text{for finite})$$

\* DTFT: Discrete-Time Fourier Transform: The term discrete-time refers to the fact that the transform operates on discrete data, often samples whose interval has units of time.

$$X_{2\pi}(\omega) = \sum_{n=-\infty}^{\infty} x[n] e^{-i\omega n}$$

\* The Fast Fourier Transform (FFT) is an algorithm for computing one cycle of the DFT, & its inverse produces one cycle of the inverse DFT.

## 2. Fixed Point Arithmetic:

To compute the FIR filter in Atmega8 microprocessor, we need to convert the floating point decimals to Fixed point. The Coefficients, inputs must be converted to Fixed point format. From the Hardware multiplier reading material, this can be understood. In our case of 8-bits for each coefficient & input, the first(NSB)bit acts as the sign bit & the rest 7 bits acts as the Fixed point approximation of the decimal number. Hence only values between  $(-1, 1)$  can be fed into the system. So our data must necessarily be Normalized.

## 3. DTFT:

- \* While the signals in Time Domain may be approximate, the DTFT computed by FFT (Fast Fourier Transform) gives correct output w.r.t the signals in Frequency Domain.
- \* DTFT of a DC signal is ideally an unit impulse function at 0 Hz. We can see that it is the case in our signal also.
- \* DTFT of a sine signal is a jump in the signal at the operating frequency ( $\approx 1800$  Hz). We get 2 such jumps for both positive & negative value of frequency.
- \* DTFT of a Whitenoise is ideally Flat, but is not the case in Real life signals. The signal is within the limit of Power spectrum.

## 4. Implementation Details

(a) Input: The input is provided as Fixed Point Integers in .db format at the end of the Code. As the size of input is large, it is not stored in SRAM, but stays in the Program Flash Memory.

(b) Output: The output is of 2 Bytes each. It is written into the SRAM from the address 0X0060. The output must be copied separately and ~~converting~~ <sup>Converted</sup> to Floating point numbers by dividing with  $2^7$ .

(c) Coefficients: The Coefficients are taken as input in .db format & stored in SRAM address starting from 0X043E

(d) Multiplication Details: For multiplication, the FMULS command is used to do Signed Multiplication on Fractions. R19 is the register which has the inputs & R18 has coefficients. Result is accumulated into R6:R5:R4 as 3 Bytes, but only R6:R5 is written into SRAM (2 Bytes).

NOTE: As the input signals are Causal, the First 'N' outputs do not have 'N' product terms, hence output for periodic signals will be non-periodic in the beginning.



## INPUTS Stored in Flash Memory



# Data Registers

The screenshot displays the Atmel Studio IDE in Advanced Mode. The main window shows the assembly file `main.asm` for the project `sakthi_partc_ee2016`. The code includes instructions for loading coefficients into registers and storing them in SRAM.

```
MOV R9,R17;
LDI XL,0X3E;      //SRAM address for storing the Coefficients (from 0x043E)
LDI XH,0X04;
LDI ZL, LOW(COEFF<<1);    //Loading address of Coefficients in the Flash memory to Pointer register
LDI ZH, HIGH(COEFF<<1);

LOOP1: LPM R23,Z;
ADIW ZH:ZL,1;      //loading the value of coefficients into registers
ST X+, R23;        //storing the coefficients into the SRAM
DEC R17;
CPI R17,0X00;
BRNE LOOP1;

LDI YL,0X60;      //SRAM location pointer for the OUTPUTS( starting from 0x0060 )
LDI YH,0X00;
```

The Solution Explorer on the right shows the project structure, including `main.asm`. The Memory window at the bottom displays the contents of the `data` registers, starting from address `0x0000`. The memory dump shows a series of hexadecimal values, with the first few rows being:

Address	Value
0x0000	2fa8 0000 6894 0000 0501 0000 0000 0000 86ce 0500 ce05 0202 0443 01fc
0x001E	015a f800 fffe 0000 0000 0000 2000 0000 0000 0000 0000 0000 0000 0000
0x003C	0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0001 0000 0000
0x005A	0000 0000 0200 0000 00c ffa3 001d ff97 0068 00a9 0000 ff56 ff97 0068 00a9
0x0078	0000 ff56 ff97 0068 00a9 0000 ff56 ff97 0068 00a9 0000 ff56 ff97 0068 00a9
0x0096	0000 ff56 ff97 0068 00a9 0000 ff56 ff97 0068 00a9 0000 ff56 ff97 0068 00a9
0x00B4	0000 ff56 ff97 0068 00a9 0000 ff56 ff97 0068 00a9 0000 ff56 ff97 0068 00a9
0x00D2	0000 ff56 ff97 0068 00a9 0000 ff56 ff97 0068 00a9 0000 ff56 ff97 0068 00a9
0x00F0	0000 ff56 ff97 0068 00a9 0000 ff56 ff97 0068 00a9 0000 ff56 ff97 0068 00a9
0x010E	0000 ff56 ff97 0068 00a9 0000 ff56 ff97 0068 00a9 0000 ff56 ff97 0068 00a9
0x012C	0000 ff56 ff97 0068 00a9 0000 ff56 ff97 0068 00a9 0000 ff56 ff97 0068 00a9
0x014A	0000 ff56 ff97 0068 00a9 0000 ff56 ff97 0068 00a9 0000 ff56 ff97 0068 00a9
0x0168	0000 ff56 ff97 0068 00a9 0000 ff56 ff97 0068 00a9 0000 ff56 ff97 0068 00a9
0x0186	0000 ff56 ff97 0068 00a9 0000 ff56 ff97 0068 00a9 0000 ff56 ff97 0068 00a9
0x01A4	0000 ff56 ff97 0068 00a9 0000 ff56 ff97 0068 00a9 0000 ff56 ff97 0068 00a9
0x01C2	0000 ff56 ff97 0068 00a9 0000 ff56 ff97 0068 00a9 0000 ff56 ff97 0068 00a9

The status bar at the bottom indicates the system is ready, with the time 01:54 PM on 10-11-2020.

## Coefficients Stored in SRAM(5-tap for example)

The screenshot displays the Atmel Studio IDE in 'Advanced Mode' for a project named 'sakthi\_partc\_ee2016'. The main window shows the assembly file 'main.asm' with the following code:

```
MOV R9,R17;
LDI XL,0X3E;      //SRAM address for storing the Coefficients (from 0x043E)
LDI XH,0X04;
LDI ZL, LOW(COEFF<<1);    //Loading address of Coefficients in the Flash memory to Pointer register
LDI ZH, HIGH(COEFF<<1);

LOOP1: LPM R23,Z;
ADIW ZH:ZL,1;      //loading the value of coefficients into registers
ST X+, R23;        //storing the coefficients into the SRAM
DEC R17;
CPI R17,0X00;
BRNE LOOP1;

LDI YL,0X60;      //SRAM location pointer for the OUTPUTS( starting from 0x0060 )
LDI YH,0X00;
```

The 'Memory' window (Memory 4) shows a dump of memory starting at address 0x03B1. The data is organized in columns of 15. The memory dump shows that the coefficients are stored in SRAM, with the first few bytes being 0x00 and the last few bytes being 0x00. The memory dump is as follows:

Address	0x03B1	0x03C0	0x03CF	0x03DE	0x03ED	0x03FC	0x040B	0x041A	0x0429	0x0438	0x0447	0x0456	0x0465	0x0474	0x0483	0x0492
data	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	?? ?? ?? ?? ?? ?? ?? ?? ?? ?? ?? ?? ?? ?? ??	?? ?? ?? ?? ?? ?? ?? ?? ?? ?? ?? ?? ?? ?? ??	?? ?? ?? ?? ?? ?? ?? ?? ?? ?? ?? ?? ?? ?? ??	?? ?? ?? ?? ?? ?? ?? ?? ?? ?? ?? ?? ?? ?? ??

The status bar at the bottom indicates the system is 'Ready' and the time is 01:54 PM on 10-11-2020.

## Outputs stored in SRAM

The screenshot displays the Atmel Studio IDE in Advanced Mode, showing the assembly code for a program named 'sakthi\_partc\_ee2016'. The code is written in assembly language and includes comments explaining its functionality. The program is configured for an ATmega8 microcontroller.

**Assembly Code:**

```
MOV R9,R17;
LDI XL,0X3E;      //SRAM address for storing the Coefficients (from 0x043E)
LDI XH,0X04;
LDI ZL, LOW(COEFF<<1);    //Loading address of Coefficients in the Flash memory to Pointer register
LDI ZH, HIGH(COEFF<<1);

LOOP1: LPM R23,Z;
ADIW ZH:ZL,1;      //loading the value of coefficients into registers
ST X+, R23;        //storing the coefficients into the SRAM
DEC R17;
CPI R17,0X00;
BRNE LOOP1;

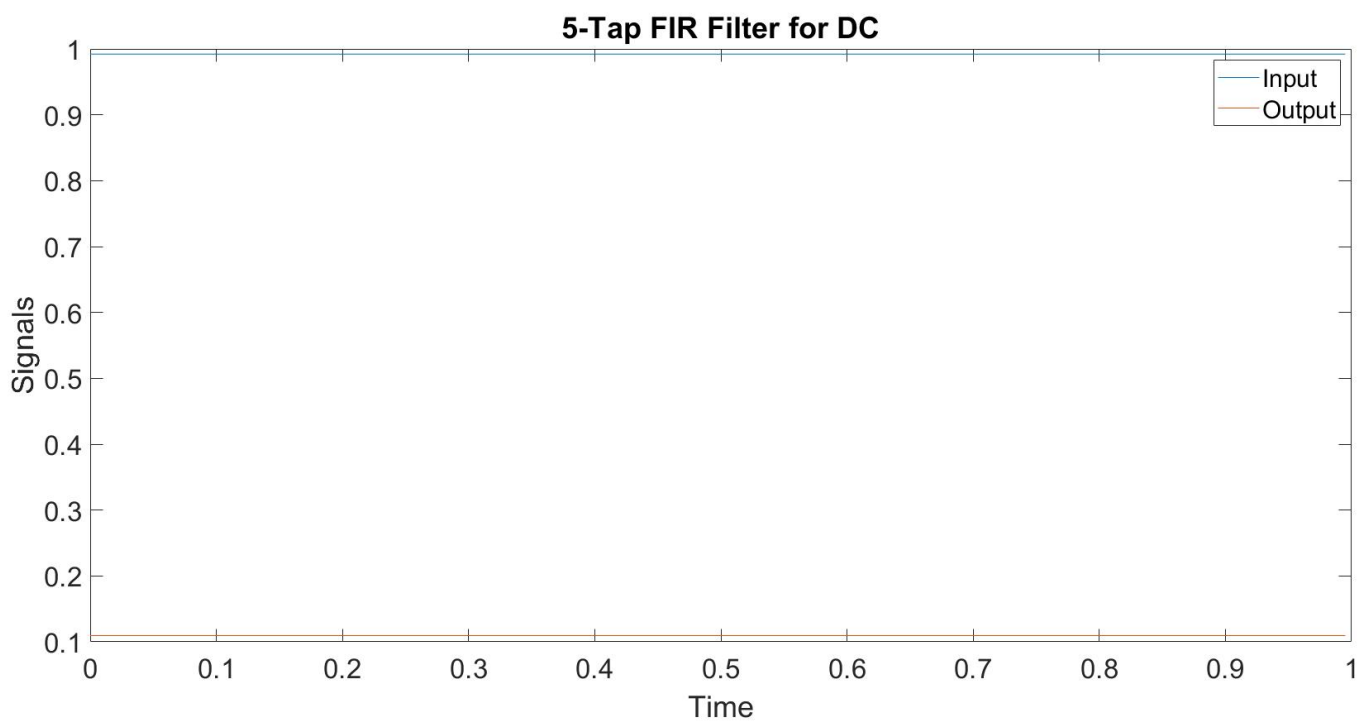
LDI YL,0X60;      //SRAM location pointer for the OUTPUTS( starting from 0x0060 )
LDI YH,0X00;
LDI ZL, LOW(INPUT<<1);    //loading the address of input location into the Pointer registers
LDI ZH, HIGH(INPUT<<1);
```

**Memory Dump:**

The Memory window shows the contents of SRAM starting at address 0x0060. The data is organized in columns of 15 bytes each. The memory dump shows a repeating pattern of values, likely representing the coefficients and input data stored in SRAM.

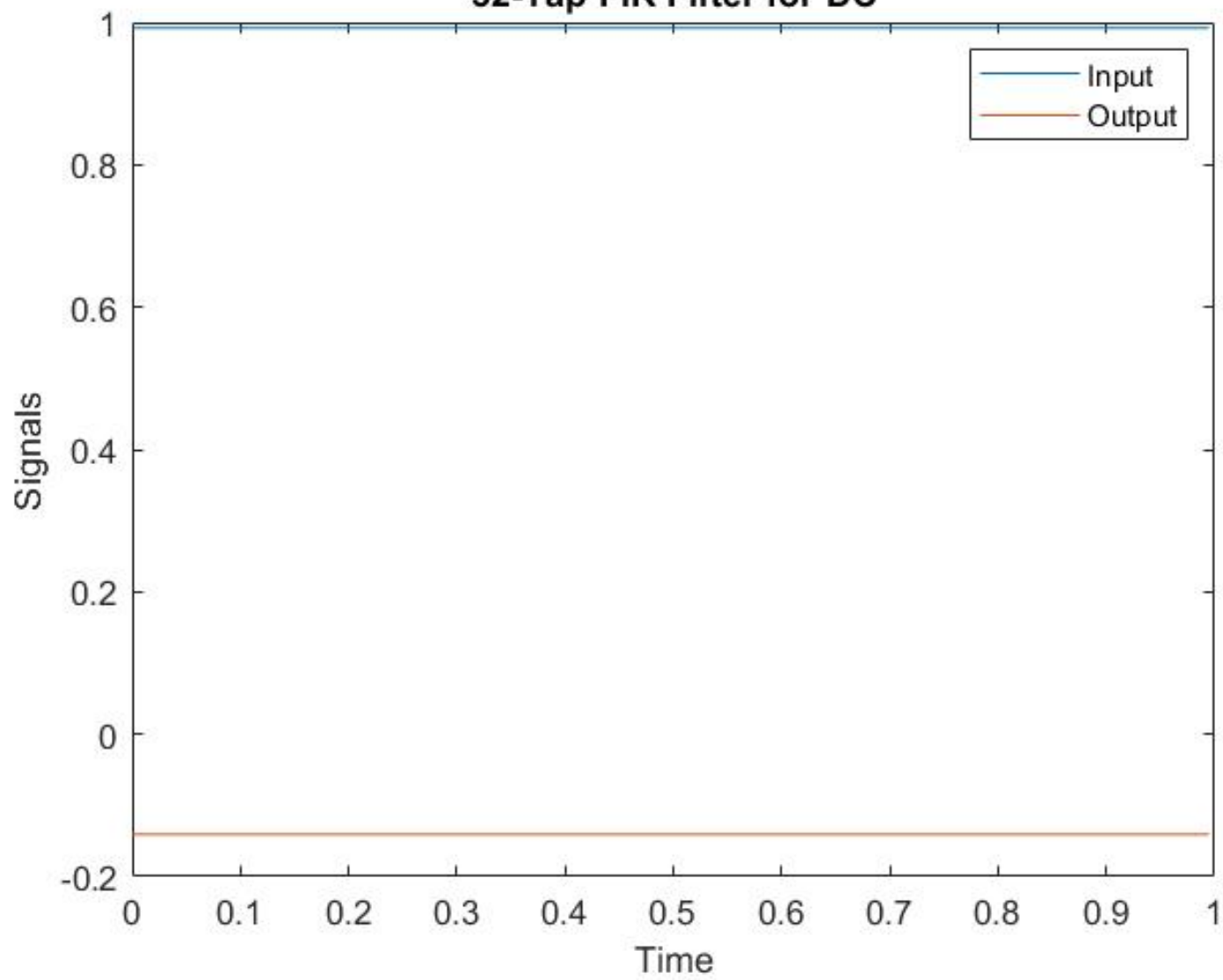
Address	0000	000c	ffa3	001d	ff97	0068	00a9	0000	ff56	ff97	0068	00a9	0000	ff56	ff97
data 0x0060	0000	000c	ffa3	001d	ff97	0068	00a9	0000	ff56	ff97	0068	00a9	0000	ff56	ff97
data 0x0070	0068	00a9	0000	ff56	ff97	0068	00a9	0000	ff56	ff97	0068	00a9	0000	ff56	ff97
data 0x0080	0068	00a9	0000	ff56	ff97	0068	00a9	0000	ff56	ff97	0068	00a9	0000	ff56	ff97
data 0x0090	0068	00a9	0000	ff56	ff97	0068	00a9	0000	ff56	ff97	0068	00a9	0000	ff56	ff97
data 0x00a0	0068	00a9	0000	ff56	ff97	0068	00a9	0000	ff56	ff97	0068	00a9	0000	ff56	ff97
data 0x00b0	0068	00a9	0000	ff56	ff97	0068	00a9	0000	ff56	ff97	0068	00a9	0000	ff56	ff97
data 0x00c0	0068	00a9	0000	ff56	ff97	0068	00a9	0000	ff56	ff97	0068	00a9	0000	ff56	ff97
data 0x00d0	0068	00a9	0000	ff56	ff97	0068	00a9	0000	ff56	ff97	0068	00a9	0000	ff56	ff97
data 0x00e0	0068	00a9	0000	ff56	ff97	0068	00a9	0000	ff56	ff97	0068	00a9	0000	ff56	ff97
data 0x00f0	0068	00a9	0000	ff56	ff97	0068	00a9	0000	ff56	ff97	0068	00a9	0000	ff56	ff97
data 0x0100	0068	00a9	0000	ff56	ff97	0068	00a9	0000	ff56	ff97	0068	00a9	0000	ff56	ff97
data 0x0110	0068	00a9	0000	ff56	ff97	0068	00a9	0000	ff56	ff97	0068	00a9	0000	ff56	ff97
data 0x0120	0068	00a9	0000	ff56	ff97	0068	00a9	0000	ff56	ff97	0068	00a9	0000	ff56	ff97
data 0x0130	0068	00a9	0000	ff56	ff97	0068	00a9	0000	ff56	ff97	0068	00a9	0000	ff56	ff97
data 0x0140	0068	00a9	0000	ff56	ff97	0068	00a9	0000	ff56	ff97	0068	00a9	0000	ff56	ff97
data 0x0150	0068	00a9	0000	ff56	ff97	0068	00a9	0000	ff56	ff97	0068	00a9	0000	ff56	ff97
data 0x0160	0068	00a9	0000	ff56	ff97	0068	00a9	0000	ff56	ff97	0068	00a9	0000	ff56	ff97
data 0x0170	0068	00a9	0000	ff56	ff97	0068	00a9	0000	ff56	ff97	0068	00a9	0000	ff56	ff97
data 0x0180	0068	00a9	0000	ff56	ff97	0068	00a9	0000	ff56	ff97	0068	00a9	0000	ff56	ff97
data 0x0190	0068	00a9	0000	ff56	ff97	0068	00a9	0000	ff56	ff97	0068	00a9	0000	ff56	ff97
data 0x01a0	0068	00a9	0000	ff56	ff97	0068	00a9	0000	ff56	ff97	0068	00a9	0000	ff56	ff97
data 0x01b0	0068	00a9	0000	ff56	ff97	0068	00a9	0000	ff56	ff97	0068	00a9	0000	ff56	ff97
data 0x01c0	0068	00a9	0000	ff56	ff97	0068	00a9	0000	ff56	ff97	0068	00a9	0000	ff56	ff97

## 5) RESULTS:

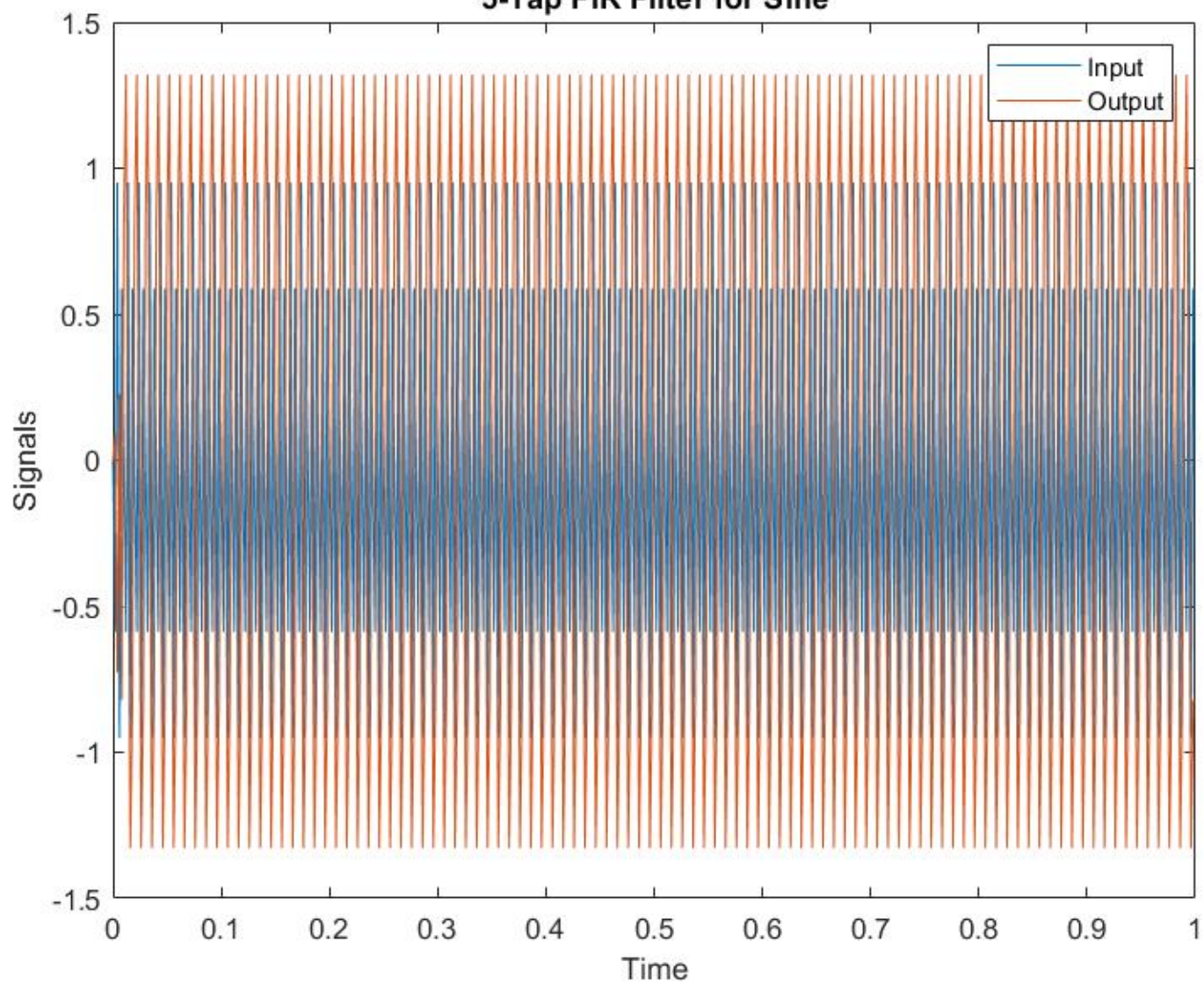


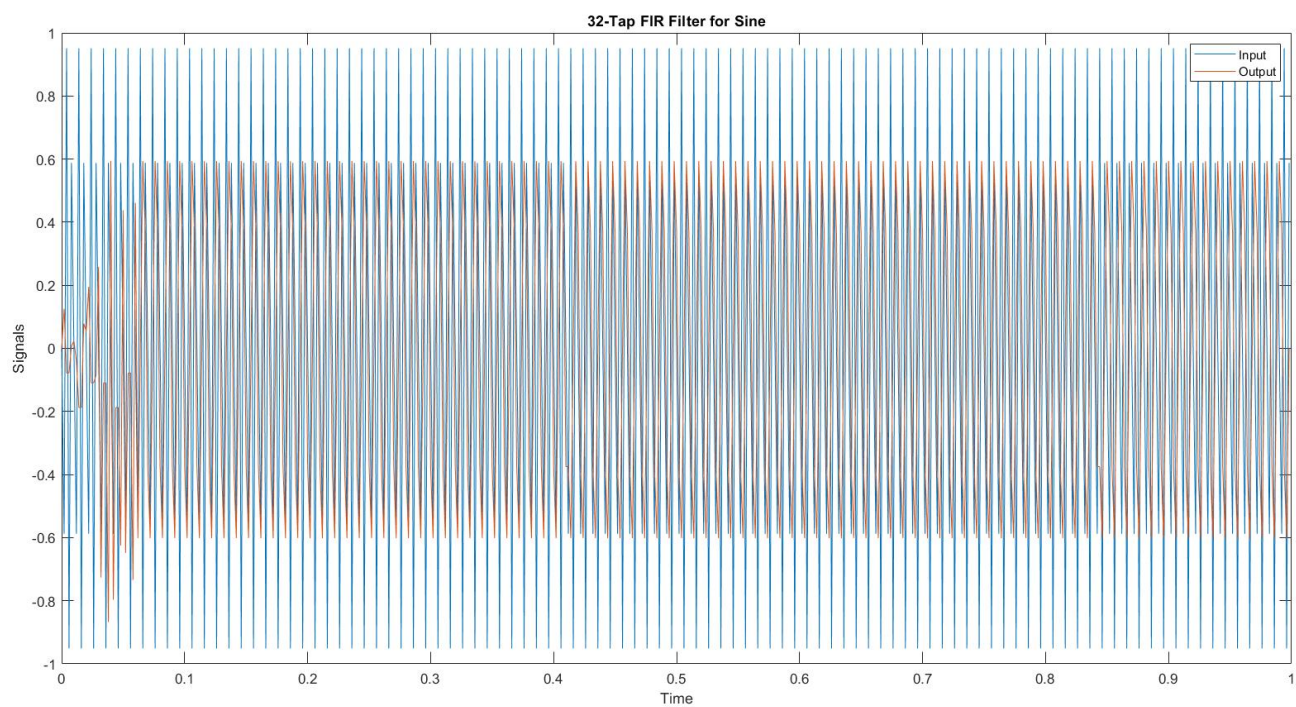


**32-Tap FIR Filter for DC**

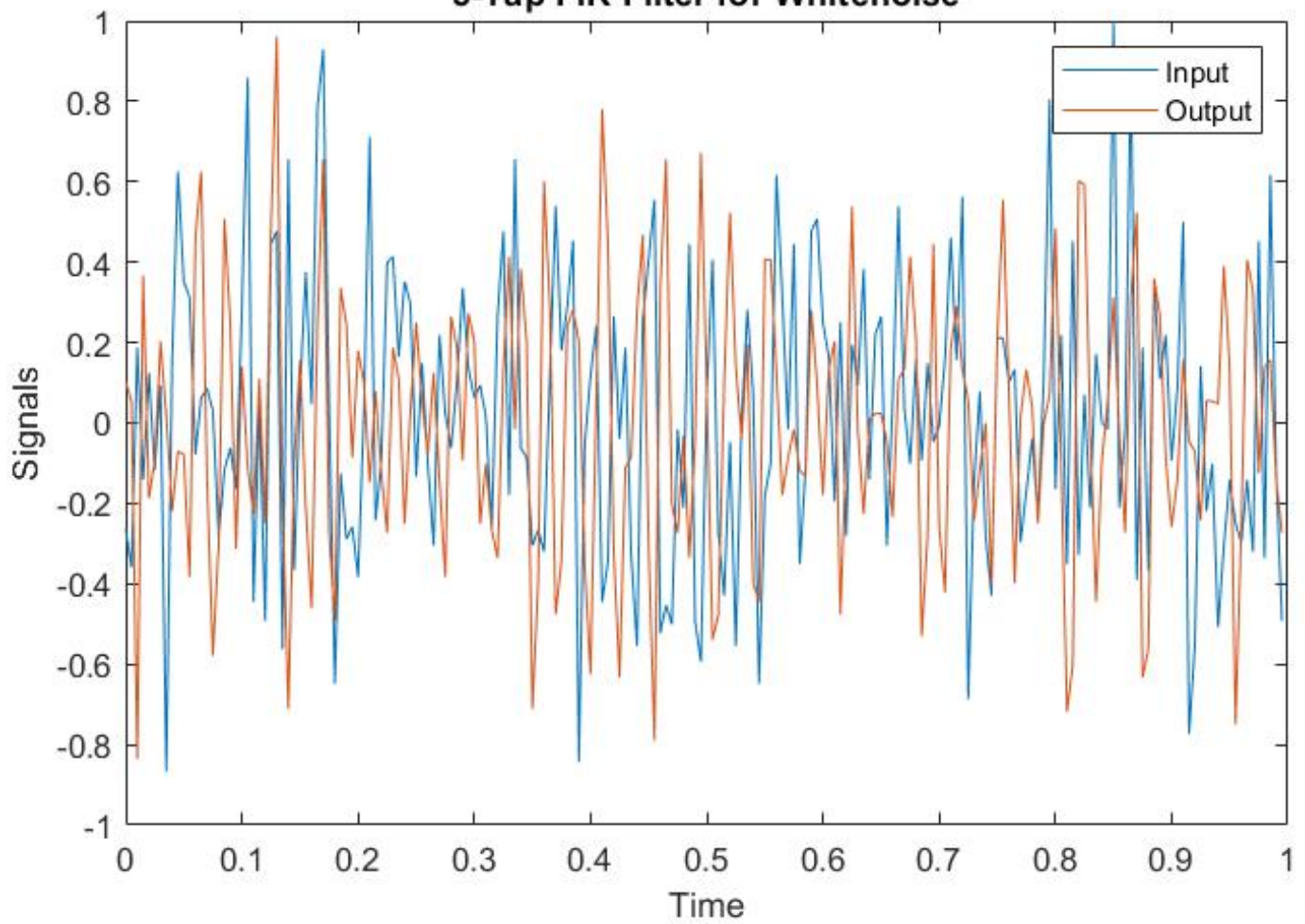


**5-Tap FIR Filter for Sine**



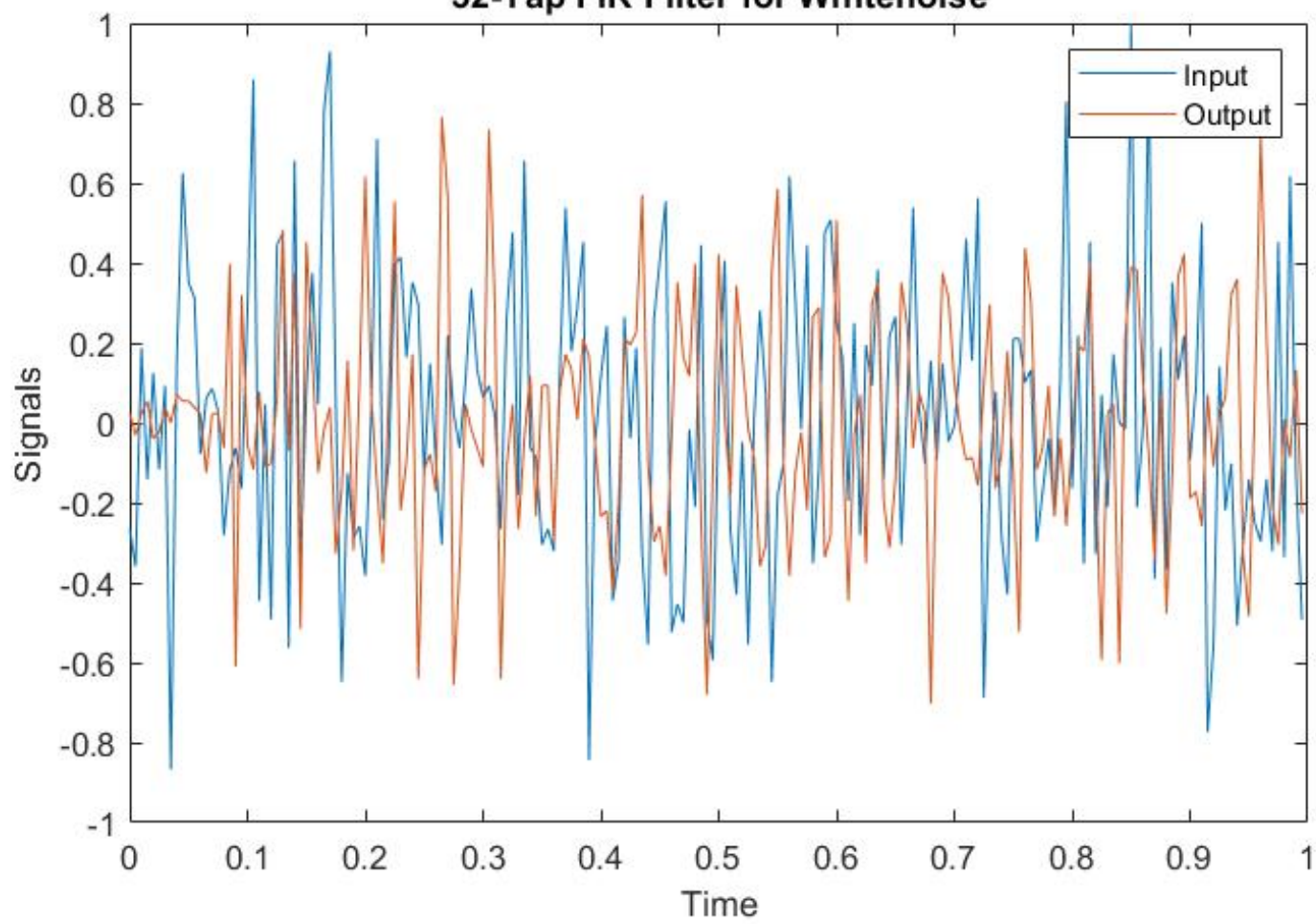


**5-Tap FIR Filter for Whitenoise**

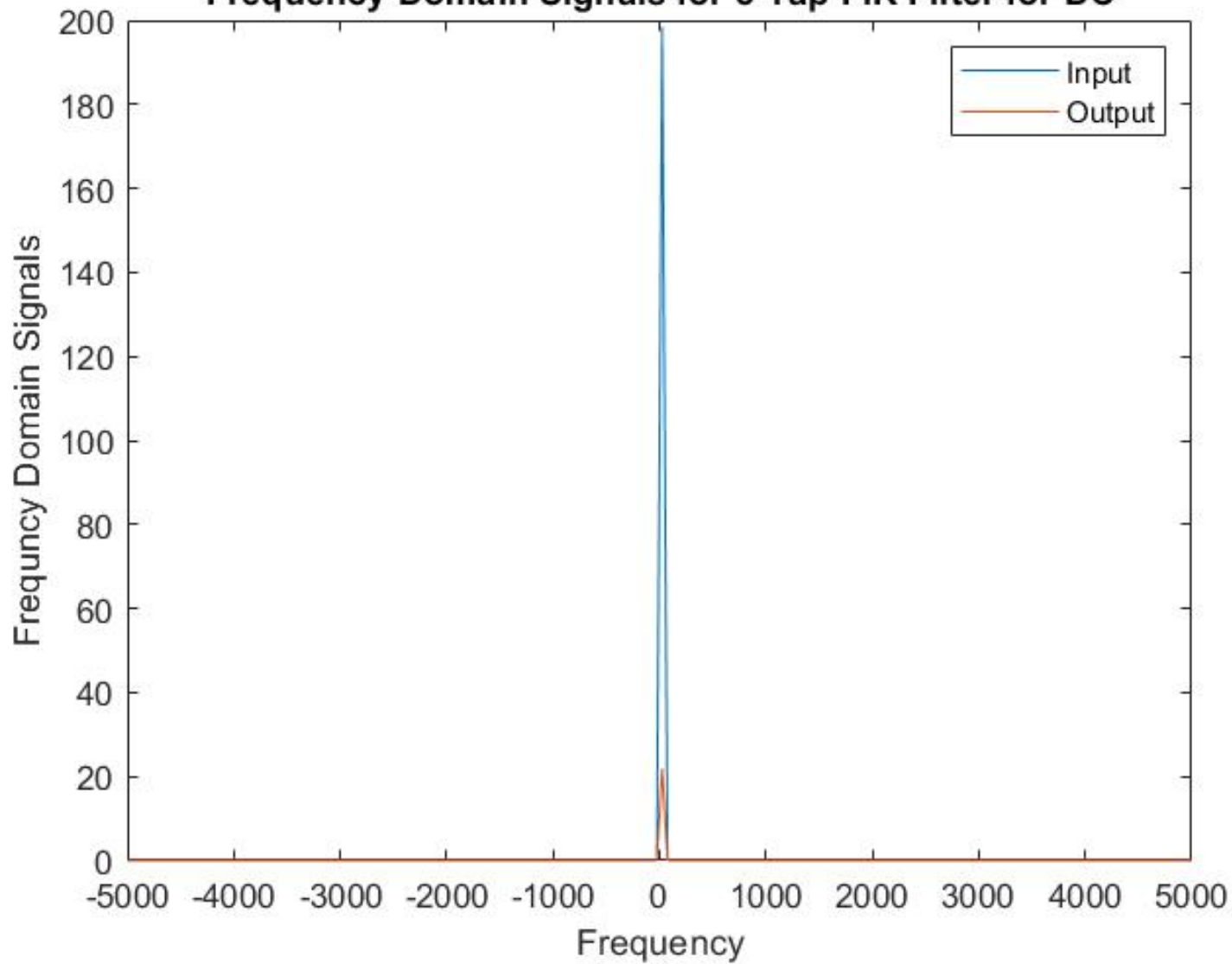




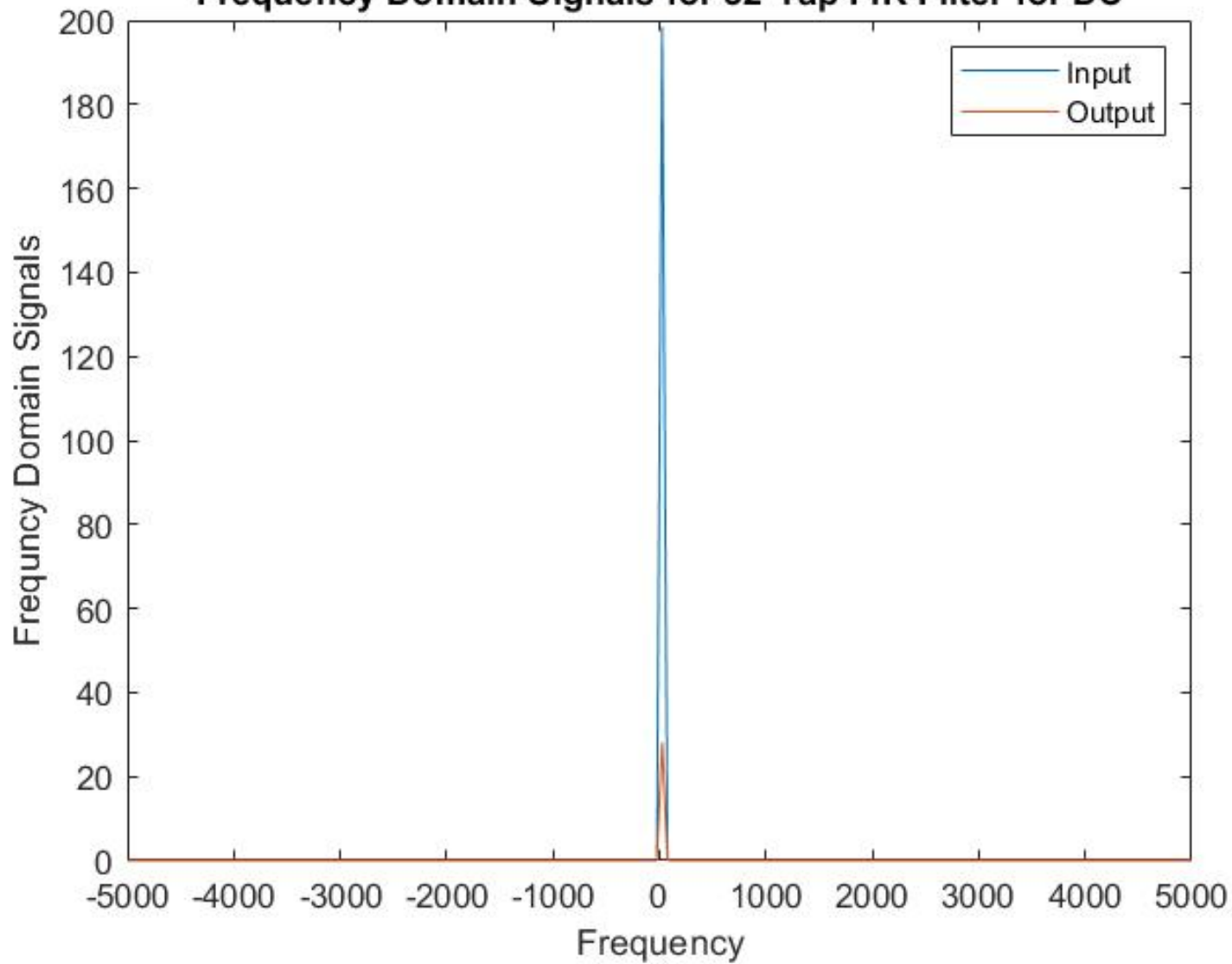
**32-Tap FIR Filter for Whitenoise**

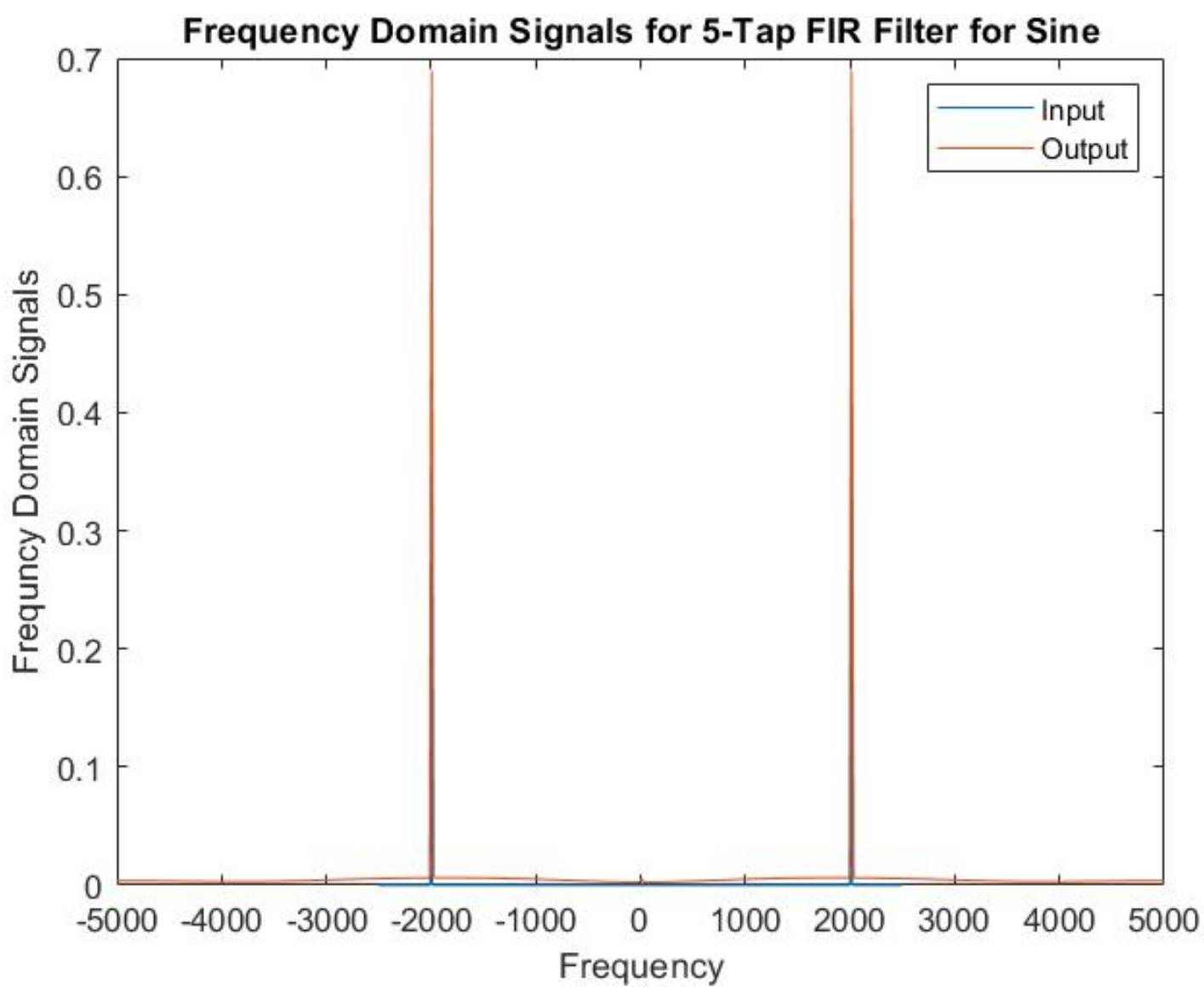


**Frequency Domain Signals for 5-Tap FIR Filter for DC**

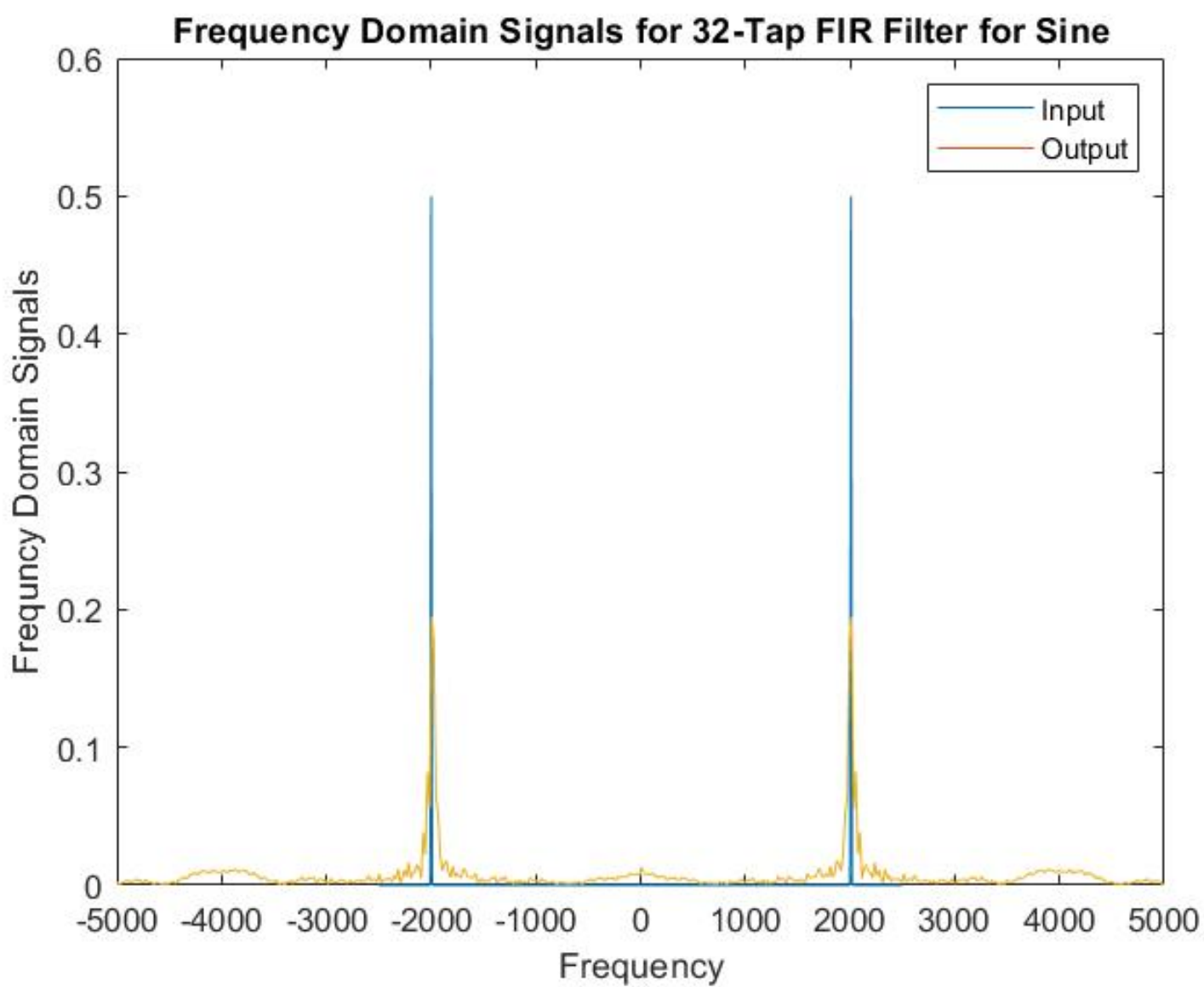


**Frequency Domain Signals for 32-Tap FIR Filter for DC**

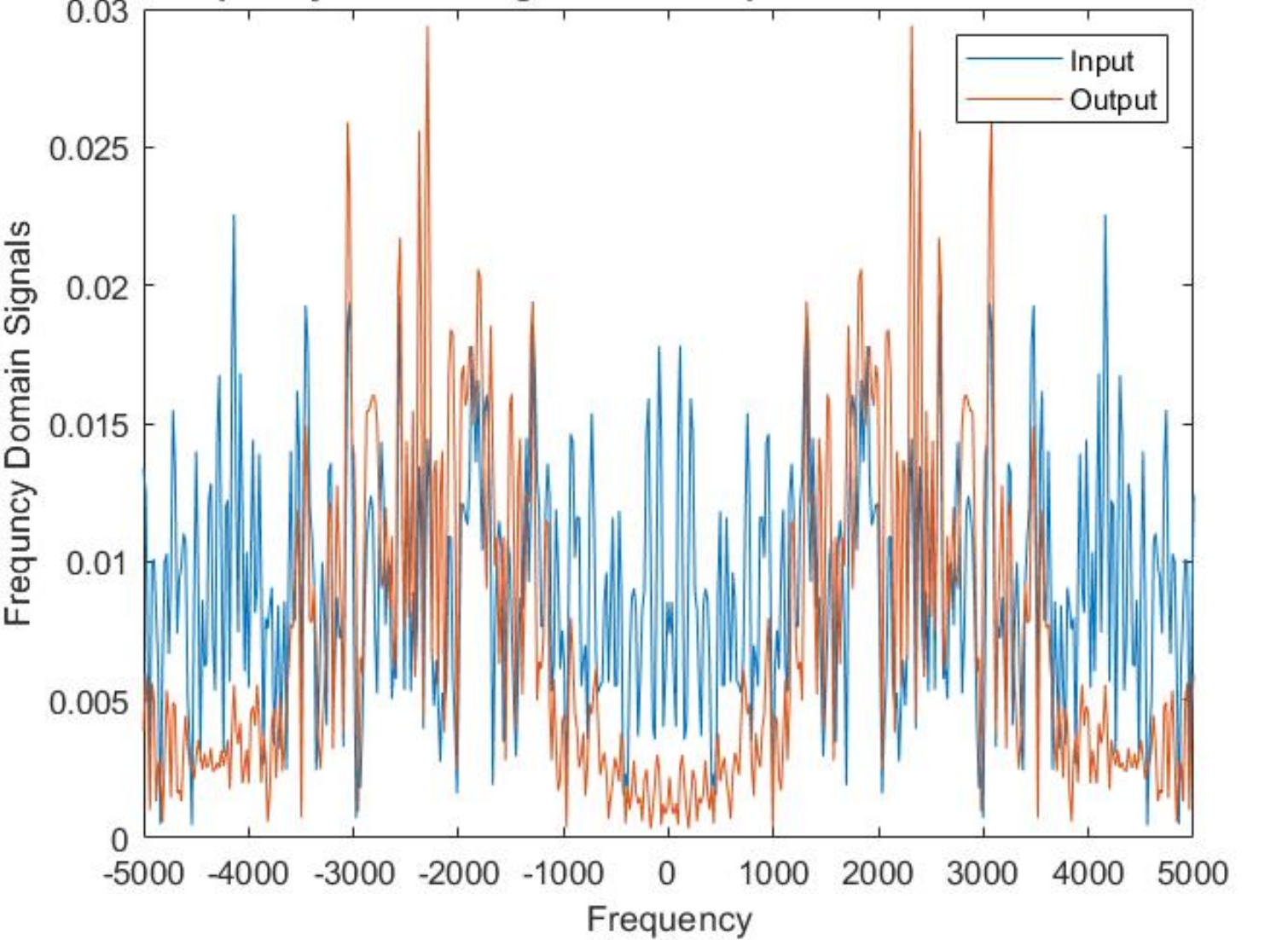








**Frequency Domain Signals for 5-Tap FIR Filter for Whitenoise**



**Frequency Domain Signals for 32-Tap FIR Filter for Whitenoise**

