ECE580 Project 1 – Implementation of EMC Filter for Robust FlexRay Operation

Mark Keranen

University of Michigan – Dearborn
College of Engineering and Computer Science

Outline

Introduction

An emerging alternative to standard gasoline powered vehicles is the electric or hybrid-electric vehicle (HEV). HEVs produce fewer emissions than their gasoline powered counterparts. They do however come with their own unique challenges. One of these challenges is the electromagnetic interference inherent to their design and power source. Electromagnetic compatibility (EMC) is a critical factor in ensuring robust vehicle operation and occupant safety. This report proposes several solutions to an identified electromagnetic interference (EMI) occurrence in a HEV which will allow proper operation of the FlexRay communications module.

Background / Problem Statement

The FlexRay communications system is a time-deterministic communications protocol for in-vehicle control applications. It is designed to provide high speed distributed control for advanced automotive applications. It is known, that for FlexRay to operate properly, the noise on the power line must be <40dBμA between frequencies of 2MHz to 7MHz.

Noise on the power line exceeding this requirement has been identified and measured to be above the allowable limit (PENG, et al., 2016). Figure 1 displays interference currents in excess of 60dB near 4MHz during an electric assist braking event.

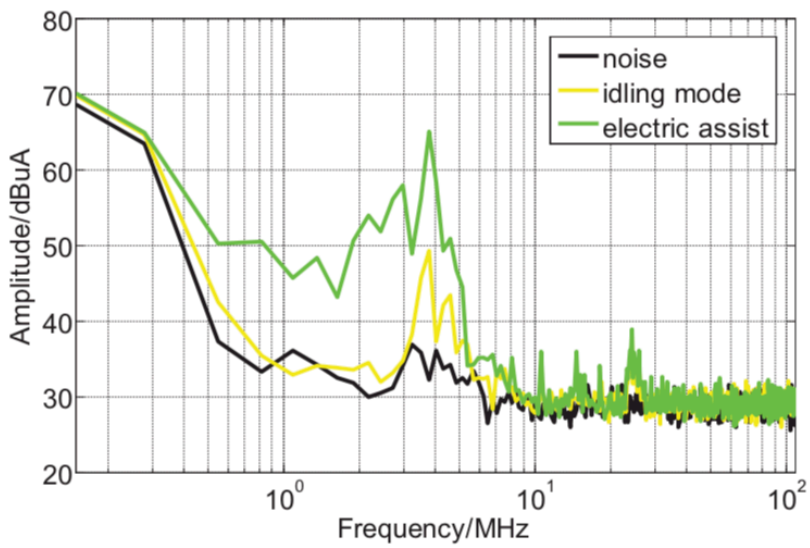

Figure 1. Interference current frequency spectrum during idle and electric assist modes (PENG, et al., 2016)

Proposed Design Solution

To allow proper operation of the FlexRay module, interference current must be reduced by at least 25dB at frequencies greater than 2MHz.

Three design solutions will be implemented:

1. Analog Implementation
2. IIR Implementation
3. FIR Implementation

Design Specification

Figure 2 displays the filter design specifications that will allow proper FlexRay module operation. The specification is as follows:

- 0 to -3dB in the pass band
- Attenuation of >25dB at frequencies >2MHz
- Cutoff frequency of 1MHz
- Stop frequency of 2MHz

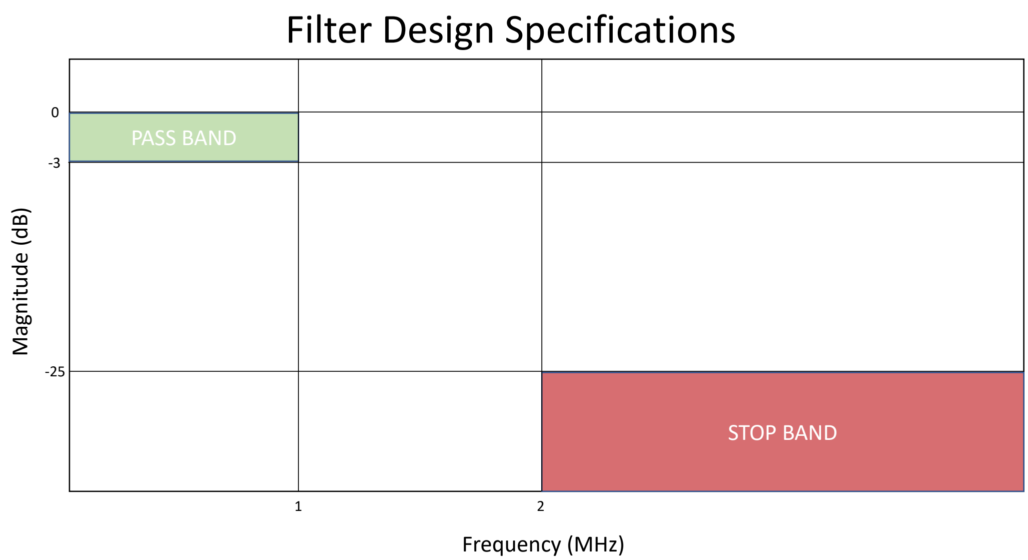

Figure 2. Filter Design Specification

Analog Implementation

Filter type and order selection is dictated by filter design specifications, included in Figure 2. With 25dB attenuation required in the stop band, Figure 3 demonstrates the need for a 5th order filter. A maximally flat filter type was selected to minimize magnitude impact in the pass band.

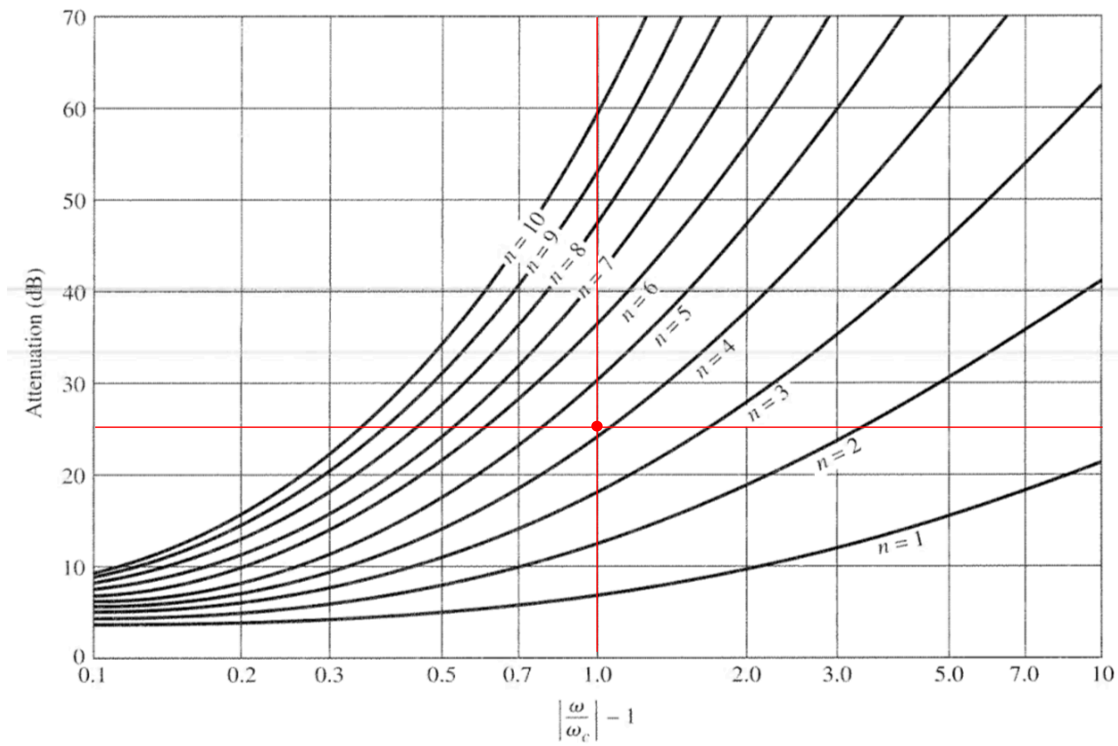

Figure 3. Attenuation vs normalized frequency for maximally flat filter prototypes (Pozar, 1998)

Given by filter type and order selection, the normalized polynomial for a 5th order low-pass Butterworth filter is as follows:

$$T(s) = (s + 1)(s^2 + 0.6180s + 1)(s^2 + 1.6180s + 1)$$

Because this polynomial is provided for a normalized cutoff frequency ($\Omega_c = \frac{1\,rad}{sec}$) it is required to scale for the desired cutoff frequency, in this case, 1MHz.

Substituting $\frac{s}{2\pi \times 10^6}$ for *s,* gives:

$$T(s) = \left(\frac{s}{2\pi \times 10^6} + 1\right)\left(\frac{s}{2\pi \times 10^6}^2 + 0.6180\frac{s}{2\pi \times 10^6} + 1\right)\left(\frac{s}{2\pi \times 10^6}^2 + 1.6180\frac{s}{2\pi \times 10^6} + 1\right)$$

In this form, *T(s)* represents the analog implementation of a 5th order, low-pass Butterworth filter with cutoff frequency at 1MHz. Figure 4 displays the frequency response with overlaid design requirements. Note that all design specifications are met in this implementation.

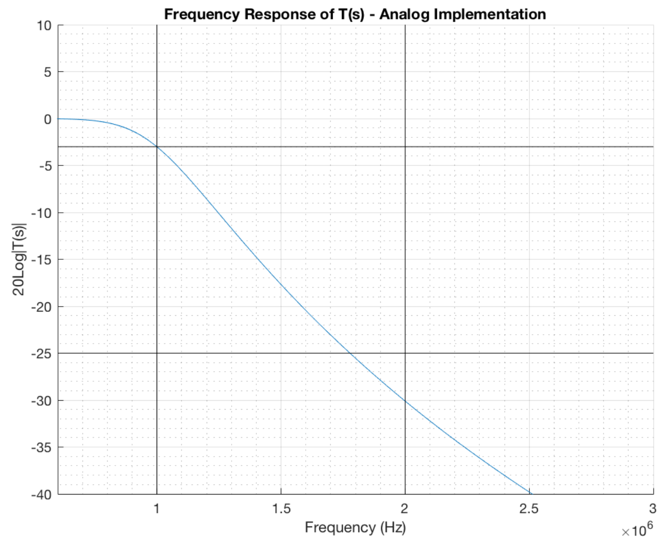

Figure 4. Frequency Response of T(s) - Analog Implementation of Maximally Flat Low Pass Filter

Circuit implementation of a 5th order, low-pass Butterworth filter for a circuit beginning with a shunt element is shown below in Figure 5, where g_n values represent frequency scaled circuit element values gathered from Table 8.3 in David Pozar's '*Microwave Engineering*' (Pozar, 1998). Source impedance of 1 Ω was assumed, as such, impedance scaling of circuit elements is not required.

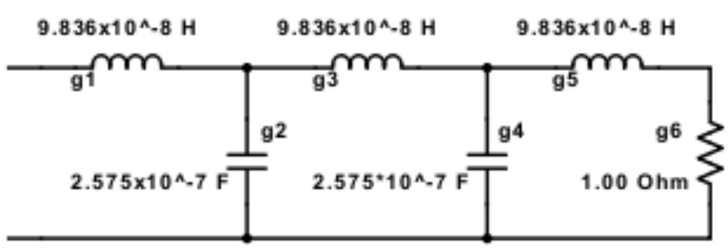

Figure 5. Circuit diagram of analog implementation

Capacitor elements were scaled to the desired cutoff frequency using:

$$L' = \frac{L}{\omega_c}$$

Inductive elements were scaled to the desired cutoff frequency using:

$$C' = \frac{C}{\omega_c}$$

IIR Implementation

Implementation of an infinite impulse response (IIR) filter follows the analog implementation, with the same cutoff frequency and filter order. The filter was generated in the analog (*s*) domain and then mapped to the frequency (*z*) domain using the bilinear transform.

Figure 6 shows the frequency response of H(s) for a 5th order low pass Butterworth filter.

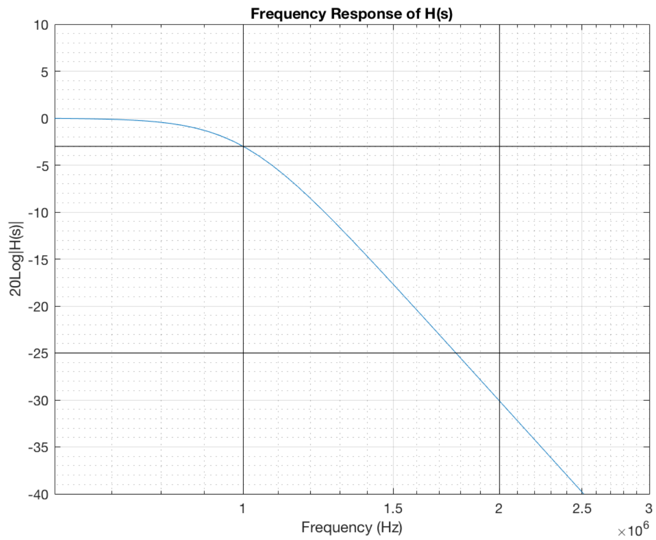

Figure 6. Frequency response of low pass Butterworth filter in s-domain

H(s) was subsequently pre-warped and mapped to the z-domain using the bilinear transform with a sampling frequency of 15MHz. 15MHz sampling frequency was selected to capture all frequencies in the frequency band of interest (2MHz – 7MHz). Figure 7 displays the frequency response of the resulting filter in the z-domain. Note that at 2MHz, the frequency response of this filter exceeds the required 25dB attenuation.

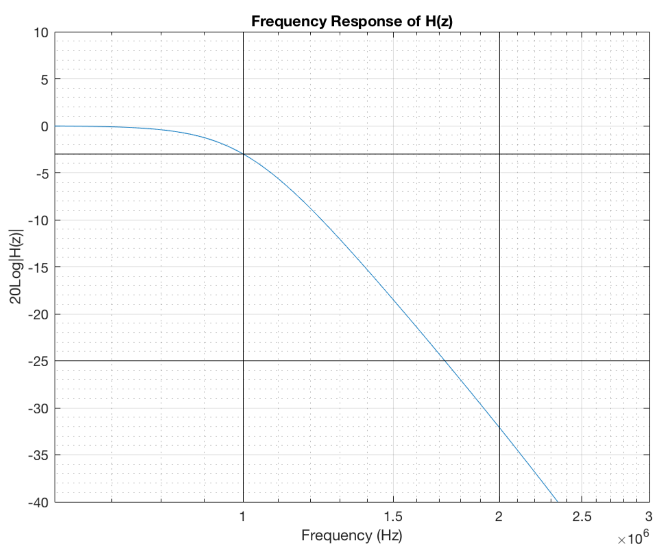

Figure 7. Frequency response of low pass Butterworth filter mapped to z-domain

Figure 8 & Figure 9 below display a block diagram depiction of the IIR filter implementations in the z-domain and the corresponding implementation in the time domain. Note that b_n and a_n depict coefficients in the numerator and denominator of H(z). The diagrams depict the expanded form of the filtering scheme as follows:

$$X(z) \;\rightarrow\; [H(z)] \;\rightarrow\; Y(z)$$

Where

$$H(z) = \frac{b_0 + b_1 z^{-1} + b_2 z^{-1} + b_3 z^{-1} + b_4 z^{-1} + b_5 z^{-1}}{1 + \ a_1 z^{-1} + a_2 z^{-1} + a_3 z^{-1} + a_4 z^{-1} + a_5 z^{-1}}$$

Splitting H(z) into two components $H_1(z)$ and $H_2(z)$ gives:

$$X(z) \;\rightarrow\; [H_1(z)] \rightarrow W(z) \rightarrow [H_2(z)] \;\rightarrow\; Y(z)$$

Where

$$H_1(z) = \ b_0 + b_1 z^{-1} + b_2 z^{-1} + b_3 z^{-1} + b_4 z^{-1} + b_5 z^{-1}$$

$$H_2(z) = \frac{1}{1 - \ a_1 z^{-1} + a_2 z^{-1} + a_3 z^{-1} + a_4 z^{-1} + a_5 z^{-1}}$$

With H(z) coefficient values specified in Table 1.

Table 1. H(z) coefficient values

n	0	1	2	3	4	5
b_n	0.000218	0.001095	0.002189	0.002189	0.001095	0.000219
a_n	1	-3.647222	5.459623	-4.168367	1.617601	-0.254629

Combining the elements of the previous equations into a block diagram format gives the following.

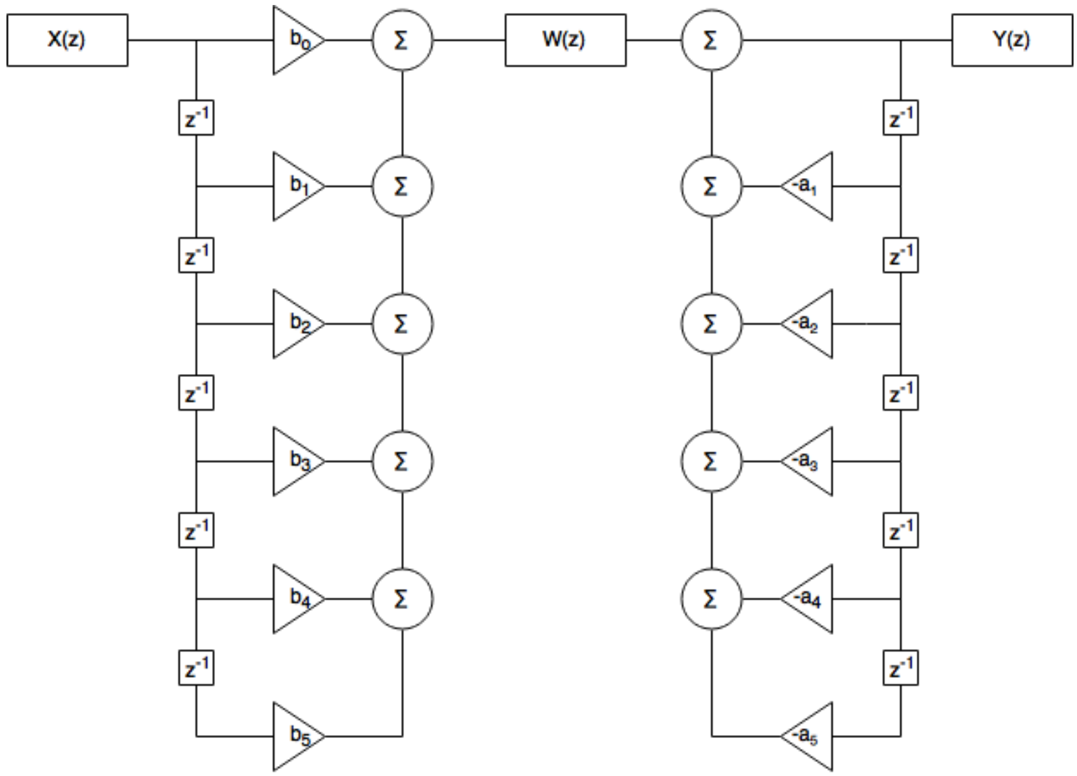

Figure 8. Block diagram representing filter implementation in z-domain

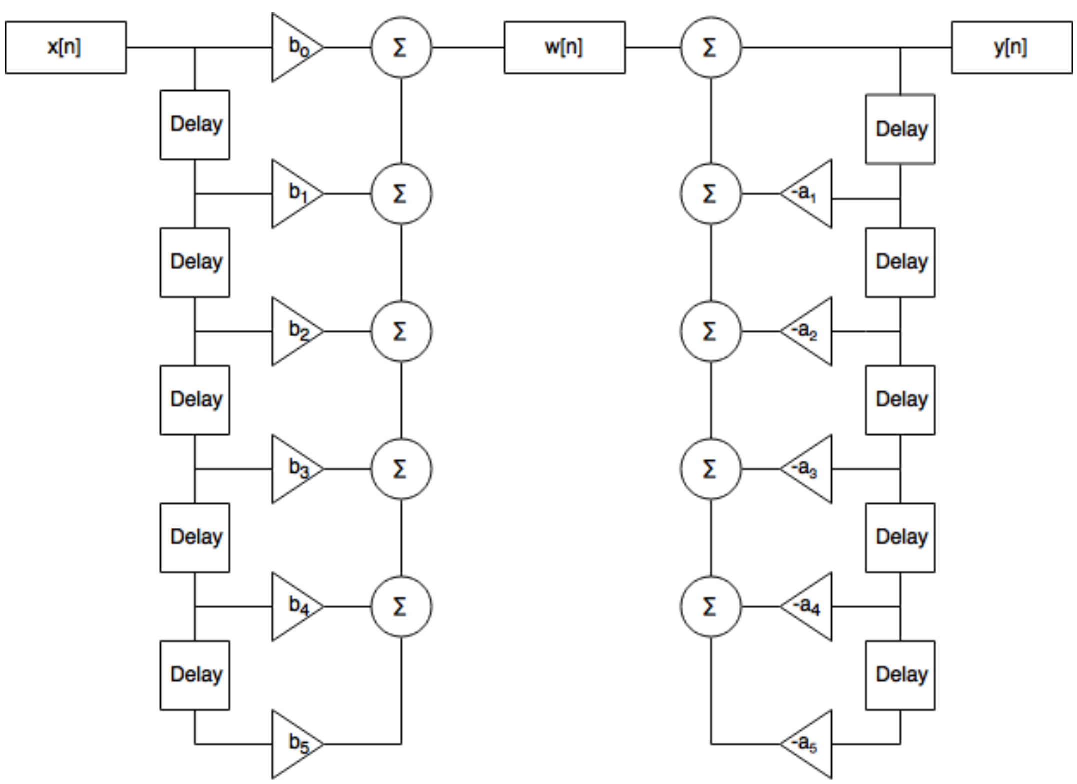

Figure 9. Block diagram representing filter implementation in the discrete time domain

FIR Implementation

A finite impulse response (FIR) filter implementation was performed with Kaiser window parameters calculated using the general form Kaiser equations. Figure 10 displays normalized FIR filter response between 0 and π.

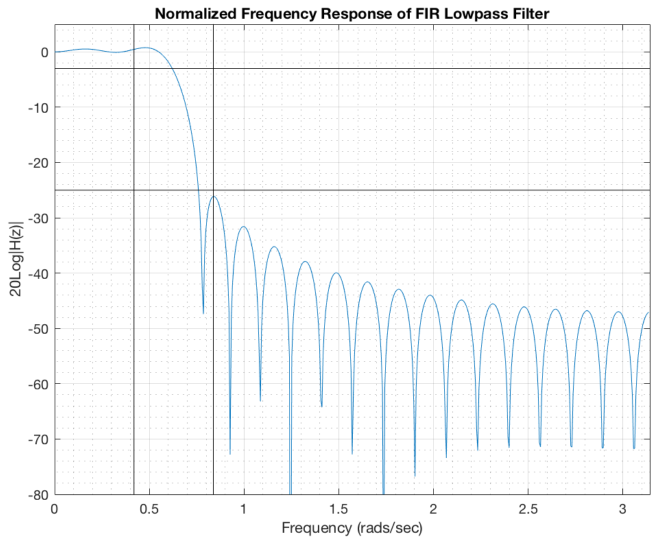

Figure 10. Normalized frequency response of FIR filter

Figure 11 displays frequency response of the same FIR filter, un-normalized and shown across frequency in Hz. Note that at frequencies >2MHz, this filter implementation provides >25dB attenuation.

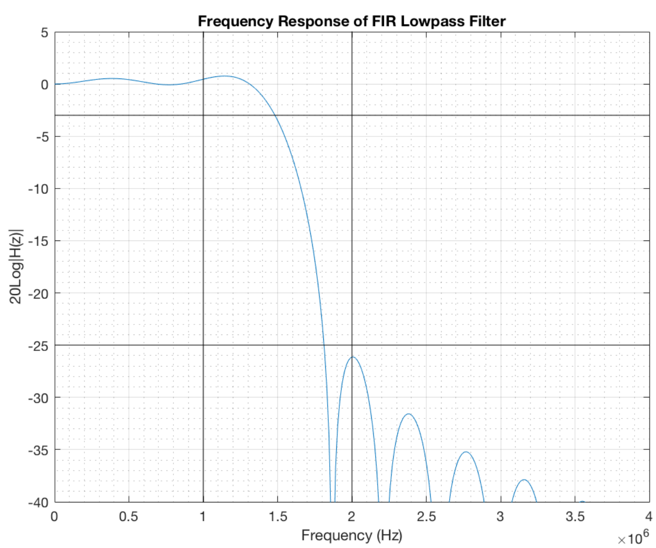

Figure 11. Frequency response of FIR filter

Summary of MATLAB Implementation

The three filters described in this report were implemented with the use of MATLAB. A full copy of the source code is available in the appendix. Several functions were made use of throughout, including freqs, freqz, butter, and bilinear. freqs and freqz were used to calculate the frequency responses of analog and digital filters, respectively. These responses were then plotted for each filter type. The butter function was paired with the bilinear function to generate an analog filter in the s domain, which was then mapped to the z domain.

Conclusion

Analog, IIR, and FIR filter types were implemented, all of which met the stated design requirement of 25dB attenuation at >2MHz. Reduction of interference current by this magnitude will ensure proper operation of the FlexRay module and ensure vehicle occupant safety.

Future Work

To meet end product requirements, several additional items are required. Analog filter implementation assumes 1 Ω source impedance. For proper frequency response and interference current attenuation, impedance scaling of the circuit is required. Actual source impedance should be investigated and identified to allow confirmation of the analog filter implementation.

Additionally, a cost of implementation study should be performed to select the ideal filter type for this application. Several factors such as DSP processing capabilities, PCB space, and added cost of implementation to the project must be explored for each solution option.

Appendices

Appendix1 – Full MATLAB Code

```
%ECE580 Project 1 - Winter 2018
%By: Mark Keranen
%Implementation of EMC filter to ensure proper operation of FlexRay Module

%Requirement: FlexRay module requires <40dB noise between 2-7MHz on power
%line for proper operation

%Currently as Measured: Noise between 2-7MHz approaches 65dB.

%Solution: Provide >25dB attenuation at >2MHz using:
%   1. Analog Implementation
%   2. IIR Implementation of Analog Solution
%   3. FIR Implementation


%Analog Implementation*****************************************************

%Fifth Order Maximally Flat Low Pass Filter, frequency shifted for wc=1MHz
%T(s) = 1*10^-30 s^5 + 3.236◊10^-24 s^4 + 5.23592◊10^-18 s^3 +
%       5.23592◊10^-12 s^2 + 3.236◊10^-6 s + 1
numerator = [0 0 0 0 1];
denominator = [1*10^-30 3.236*10^-24 5.23592*10^-18 5.23592*10^-12 3.236*10^-
6 1];
w = logspace(-3, 9, 1000); %freq range

T=freqs(numerator, denominator, w); %Calculate frequency response of T(s)

hold on
semilogx(w, 20*log10(abs(T))) %Plot magnitude response in dB
hold off

axis([6*10^5 3*10^6 -40 10])
grid on
grid minor

%Add Design Specs to Chart to confirm specifications are met
passBandLowerLimit = refline(0,-3);
stopBandUpperLimit = refline(0,-25);
passBandLowerLimit.Color='k';
stopBandUpperLimit.Color='k';

cutOffFrequency = line([1*10^6 1*10^6], [-80 10]);
stopBandStartFrequency = line([2*10^6 2*10^6], [-80 10]);
cutOffFrequency.Color='k';
stopBandStartFrequency.Color='k';

title('Frequency Response of T(s) - Analog Implementation');
xlabel('Frequency (Hz)');
ylabel('20Log|T(s)|');

%IIR Implementation of Analog Design***************************************
```

```
figure()

fc = 1*10^6;  %Cutoff frequency = 1 MHz
fs = 15*10^6; %Sampling Frequency = 15 MHz; 7MHz * 2 = 14MHz +1 for SF
fp = fc;  %Match frequency for prewarping
filterOrder = 5; %Order calculated based on design specifications

%Calculate Transfer function coefficients and H(s)
[b,a] = butter(filterOrder,2*pi*fc,'s');
[h,w] = freqs(b,a);

%Convert frequency to MHz
w = (w/(2*pi));

%Plot frequency response of filter
semilogx(w, 20*log10(abs(h)));
axis([6*10^5 3*10^6 -40 10])
grid on
grid minor

%Plot design requirements
passBandLowerLimit = refline(0,-3);
stopBandUpperLimit = refline(0,-25);
passBandLowerLimit.Color='k';
stopBandUpperLimit.Color='k';

cutOffFrequency = line([1*10^6 1*10^6], [-80 10]);
stopBandStartFrequency = line([2*10^6 2*10^6], [-80 10]);
cutOffFrequency.Color='k';
stopBandStartFrequency.Color='k';

title('Frequency Response of H(s)');
xlabel('Frequency (Hz)');
ylabel('20Log|H(s)|');

%Use bilinear transform to find map s-domain filter to z-domain
figure()

%Perform bilinear transform using coefficients generated for s-domain,
%passing sampling frequency and match frequency

[numd,dend] = bilinear(b,a,fs,fp); %H(z)

%Calculate frequency response of H(z)
[hd,wd] = freqz(numd,dend);

%Un-normalize frequency
wd = ((wd/(2*pi))*fs);

%Plot H(z) frequency response
semilogx(wd, 20*log10(abs(hd)));
axis([6*10^5 3*10^6 -40 10])
grid on
grid minor
```

```
%Plot design specs
passBandLowerLimit = refline(0,-3);
stopBandUpperLimit = refline(0,-25);
passBandLowerLimit.Color='k';
stopBandUpperLimit.Color='k';

cutOffFrequency = line([1*10^6 1*10^6], [-80 10]);
stopBandStartFrequency = line([2*10^6 2*10^6], [-80 10]);
cutOffFrequency.Color='k';
stopBandStartFrequency.Color='k';

title('Frequency Response of H(z)');
xlabel('Frequency (Hz)');
ylabel('20Log|H(z)|');

%FIR Implementation*******************************************************

%Setting up kaiser window parameters

n=36;           %Determined using Kaiser window equations
beta=1.339;     %Determined using Kaiser window equations
firFilter = fir1(n,pi/15, kaiser(n+1,beta)); %Create FIR filter with kaiser
window

figure()
[d,wd]=freqz(firFilter,1); %calculating H(z)
plot(wd,20*log10(abs(d)))
title('Normalized Frequency Response of FIR Lowpass Filter')
xlabel('Frequency (rads/sec)')
ylabel('20Log|H(z)|')
axis([0 pi -80 5])
grid on
grid minor

%Plot normalized design specifications
passBandLowerLimit = refline(0,-3);
stopBandUpperLimit = refline(0,-25);
passBandLowerLimit.Color='k';
stopBandUpperLimit.Color='k';

cutOffFrequency = line([2*pi/15 2*pi/15], [-80 10]);
stopBandStartFrequency = line([4*pi/15 4*pi/15], [-80 10]);
cutOffFrequency.Color='k';
stopBandStartFrequency.Color='k';

%Un-normalize frequency and replot to display frequency response in MHz
figure()
plot(fs/2*wd/pi,20*log10(abs(d)))
axis([0 4*10^6 -40 5])
grid on
grid minor
xlabel('Frequency (Hz)')
ylabel('20Log|H(z)|')
title('Frequency Response of FIR Lowpass Filter')

%Plot design requirements
```

```
passBandLowerLimit = refline(0,-3);
stopBandUpperLimit = refline(0,-25);
passBandLowerLimit.Color='k';
stopBandUpperLimit.Color='k';

cutOffFrequency = line([1*10^6 1*10^6], [-80 10]);
stopBandStartFrequency = line([2*10^6 2*10^6], [-80 10]);
cutOffFrequency.Color='k';
stopBandStartFrequency.Color='k';
```

Bibliography

PENG, H., HU, J., JIANG, C., LIU, Q., XU, H., & HE, Z. (2016). Analysis for the EMI measurement and propagation path in Hybrid Electric Vehicle . *IEEE*, 1087-1090.

Pozar, D. (1998). *Microwave Engineering.*