

Design Note: Dynamic High-Order Filters

Rev: 1.0.0
Date: 12th March 2004

1 Purpose

This document describes how to dynamically program high-order filters using AnadigmDesigner®2 using algorithmic dynamic reconfiguration.

AnadigmDesigner®2 supports two powerful features that greatly extend the application of Anadigm's field programmable analog arrays (FPAAs) to filter applications.

- Firstly, it is tightly coupled to a the **AnadigmFilter** filter synthesis tool. This allows for the rapid and easy creation of high order filters.
- Secondly, for any user application constructed using configurable analog modules (CAMs), it will automatically generate the necessary application programming interfaces (APIs) to perform **real-time dynamic programming** under software control by a microprocessor. This can be done using
 - o *algorithmic* method (where the processor calculates new circuit programming data) or
 - o *state-driven* method (where programming data segments are pre-calculated by AnadigmDesigner®2).



** The state-driven method can be applied to any circuit variations, where the designer sets up different circuit 'states' using AnadigmDesigner®2, which generates the necessary information to transition real-time from one state to another using minimal data sets under the control of a microprocessor. This is fully described in the help information for AnadigmDesigner®2.*

Here we focus on the *algorithmic* method, where the designer wants the target system to re-tune its filters in response to real-time events, and where the filter settings are unpredictable or need to be arbitrary.

2 Constructing Dynamic Filters

2.1 Building the Filter

Build the desired filter using AnadigmFilter. To do this, start AnadigmDesigner@2, and select *Tools->AnadigmFilter*.

Select a filter characteristic that satisfies a typical frequency response that you will require. This frequency characteristic will be varied under software control when complete.

Having chosen a response, note the resulting [f_0 , *gain* and *Q*] settings for each biquad and [f_0 and *gain*] for each bilinear stage. To do this, either

- select “*List CAMs*” in AnadigmFilter, which gives you a list of all biquad and bilinear parameter settings for that filter, or
- select “*Build circuit*” in AnadigmFilter and for each stage (CAM) note the resulting settings in the CAM parameter setting dialog.

Figure 1 shows an example of the former - the bilinear and biquad parameters for a 1kHz 5th order lowpass Chebyshev filter.

Instance	Clock	Module	Type	Phase	Fo-Khz	DC Gn	HF Gn	Q
CAM: Stage0	ClockA 0	FILTERBILINR	LP	1	0.178	1		
CAM: Stage1	ClockA 0	FILTERBIQUAD	LP	1	0.967	1		8.82
CAM: Stage2	ClockA 0	FILTERBIQUAD	LP	1	0.614	1		2.14

Figure 1

2.2 Building a dynamic interface

The CAMs that are used to build this circuit have the facility for being dynamically updated under software control using a C-code API. The functions take parameters f_0 & *Gain* (and *Q* in the case of biquads).

These need to be re-assigned new values as we change the filter dynamically (see Section 2.3).



*Details on how to extract C-code APIs and include them into system software is covered in other documents. The reader is referred to AnadigmDesigner@2 help information, and also to the reference kit entitled “**Dynamic Programming Starter Guide – Subwoofer Filter**” (document number SK01SUBW-U001) on the Anadigm Web Site (www.anadigm.com).*

2.3 Applying The New Filter Settings

The new CAM parameter settings that must be applied by your system software are determined as follows.

1. Note the nominal settings as shown in Section 2.1
2. Note the respective parameter range limits as recommended by AnadigmDesigner®2.
3. Scale all frequency settings by the same factor
4. Scale gain using the gain parameter
 - a. On the first filter stage if increasing gain*
 - b. On the last filter stage if reducing gain*
5. Keep all Q settings the same

So in example in Section 2.1, the new settings for a 3kHz 5th order Chebyshev filter with a passband gain of 0.7 would be:

Stage 0:	F_0	0.534
	Gain	1.0
Stage 1:	F_0	2.901
	Gain	1.0
	Q	8.82
Stage 2:	F_0	1.842
	Gain	0.7
	Q	2.14



* The scaling of gain should be done with care to avoid signal clipping, which may be averted by performing some of the gain scaling elsewhere in the biquad/bilinear chain. The recommendation above is made purely for noise considerations. It is recommended that maximum gain levels be checked using the AnadigmDesigner®2 simulator first to ensure that clipping does not occur.

3 Some More Insight

All filters can be mathematically described using high-order polynomial expressions.

$$H(s) = \frac{1}{s^6 + 3.864s^5 + 7.464s^4 + 9.142s^3 + 7.464s^2 + 3.864s + 1}$$

Classical filter approximations, such as the normalized Butterworth 6th order filter above, deliver frequency responses that best approximate a 'brick wall' response.

All such expressions can be re-written as products of simpler ones, where the numerator and denominator are up to first order expressions ("bilinear") or second order ("biquadratic").

The normalized Chebyshev 4th filter below shows this.

$$H(s) = \frac{1}{(s^2 + 0.279s + 0.9865)} \cdot \frac{1}{(s^2 + 0.6737s + 0.2794)}$$

These map one-for-one onto the AnadigmDesigner®2 CAM building blocks **FilterBilinear** and **FilterBiquad** (see Section 2.1).

In these expressions $H(s)$ describes the frequency response of the filter, where s is a complex variable. $H(s)$ reaches infinity whenever the denominator of the expression equates to zero. These values of s are called *poles*. Similarly, the *zeros* of the filter are the values of s for which numerators of $H(s)=0$.

Figures Figure 2 and Figure 3 plot the real and imaginary components of s for the poles of a Butterworth and Chebyshev low-pass filters respectively.

As can be seen, Butterworth poles, when plotted in this manner, lie in a semi-circle and Chebyshev poles lie in a semi-ellipse. It is this positioning of the poles that give rise to the maximally flat passband for the Butterworth, and the rippled passband & more rapid initial roll-off of the Chebyshev characteristic.

This characteristic shape formed by the locations of the poles in the pole/zero diagram is termed the *root locus*.

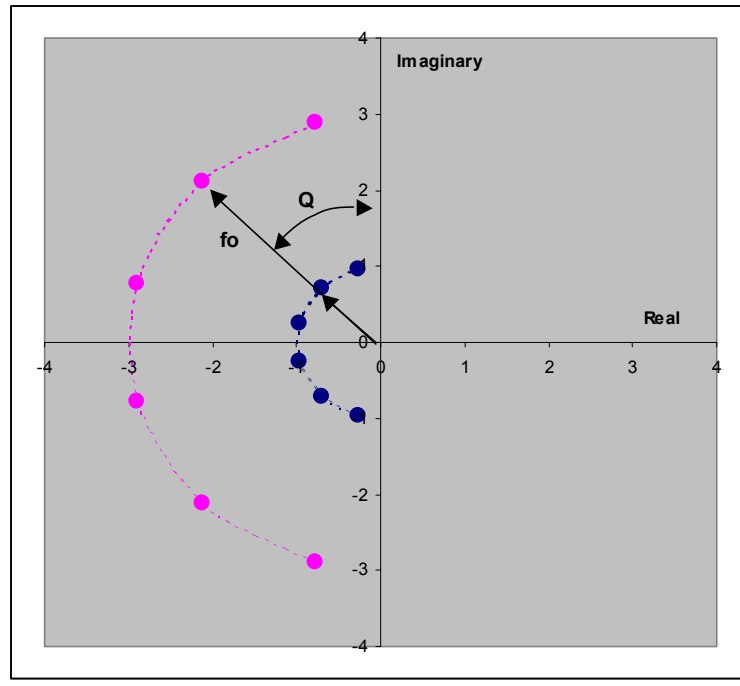


Figure 2 – Butterworth poles

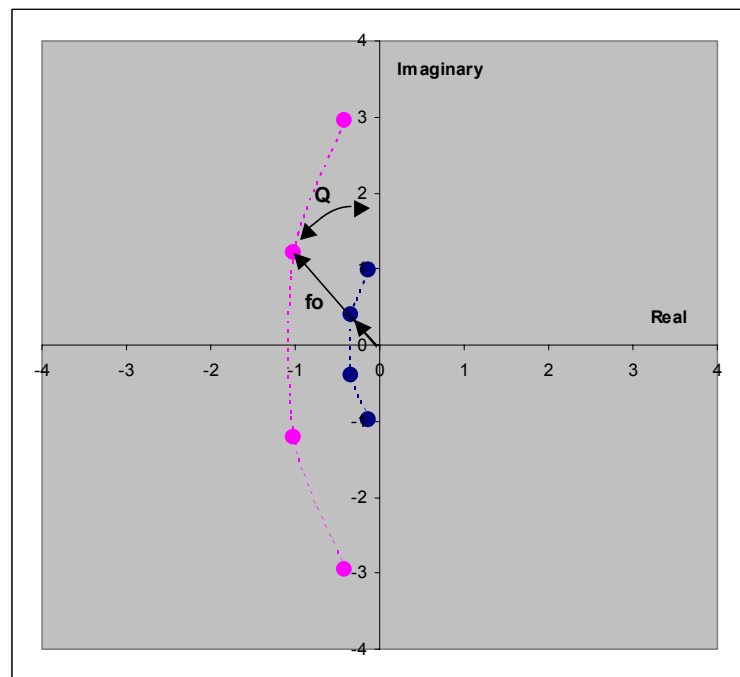


Figure 3 – Chebyshev poles

These poles lie in conjugate pairs for each biquad and as a single real pole for each bilinear. For both, the length of the line from the origin to the pole, f_0 , is the natural frequency of the pole. For biquads, the Q factor is reflected by the angle subtended with the imaginary axis as shown (the smaller the angle, the higher the Q).

In the case of Butterworth filters, all values of f_0 in the biquads are the same, and the Q factors are different

In the case of Chebyshev filter both f_0 and Q vary between the biquads.

So to scale frequency of the overall filter without affecting its characteristic response, all that needs to happen is to maintain the Q factors of the poles, and scale all values of f_0 by the same amount.

This gives a new frequency setting, whilst maintaining the characteristic shape of the filter, because the root locus forms a semi-circle for a Butterworth filter and a semi-ellipse for a Chebyshev – the root locus simply expands or contracts.