PLL Design Using the PLL Design Assistant Program

Michael H. Perrott
http://www.cppsim.com
July 2008

Table of Contents

Setup (Only Windows 2000/Xp/Vista Is Supported)

The PLL Design Assistant package is provided as part of the CppSim package available at http://www.cppsim.com. Simply download the file **setup_cppsim3.exe**, run it in Windows (i.e., double click on it in Windows Explorer), and then follow the setup instructions.

To run the program:
Click on the PllDesign icon created during the installation process.

Tool Basics

The PLL Design Assistant provides a graphical user interface methodology to the design of phase locked loops. Instructions for using the tool are provided in the pages that follow. It is worthwhile, however, to point out a few of the documentation features of the program before we begin.

The program allows you to save and later reload its user specified settings using the **Open** and **Save** commands under the **File** menu item. In addition, plots made by the tool can be saved in either color or black-and-white postscript format using the **Print Plot** command under the **File** menu item. Finally, a very handy feature is the ability to selectively save the parameter window or plot window to the Windows clipboard by using the **Copy Parameters to Clipboard** or **Copy Plot to Clipboard** commands, respectively, under the **Edit** menu item. This last feature allows one to easily include results from the tool into PowerPoint or Word documents.
Error messages generated during program operation will appear in the command window that appears when the tool is run.

Finally, the preferred way of closing the tool is to click **Close** under the **File** menu item.

Known Bugs

1) Designing G(f) according to an elliptic filter shape is currently not supported due to a Matlab compiler problem.

Introduction

The PLL Design Assistant program allows fast and straightforward design of phase locked loops at the transfer function level. In particular, the program takes as input a desired closed loop transfer function description and then automatically calculates the open loop parameters that must be chosen to achieve the design. The resulting closed loop pole/zero locations, transfer function, and step response are plotted with a simple touch of a button. The impact of non-idealities such as open loop gain and open loop pole variations on the closed loop response can be explored by entering in the variation values into the tool and observing the resulting shifts in the closed loop pole/zero locations, transfer function, and step response. In addition, the impact of parasitic poles and zeros can be accounted for by entering them into the tool and observing the resulting closed loop response. Finally, an estimate of the PLL noise performance can be viewed by entering in noise parameters such as the magnitude of detector noise and VCO noise and observing the resulting phase noise and rms jitter at the PLL output.

To illustrate the design methodology that should be applied when using the PLL Design Assistant program, we will focus on a particular class of phase locked loops known as Σ–Δ Fractional-N frequency synthesizers [1]. Figure 1 illustrates this PLL architecture, which consists of a phase-frequency detector (PFD), charge pump, loop filter, voltage controlled oscillator (VCO), and a frequency divider that is dithered between integer values to achieve fractional divide ratios. The realization of fractional divide ratios allows the synthesizer to achieve very high frequency resolution.

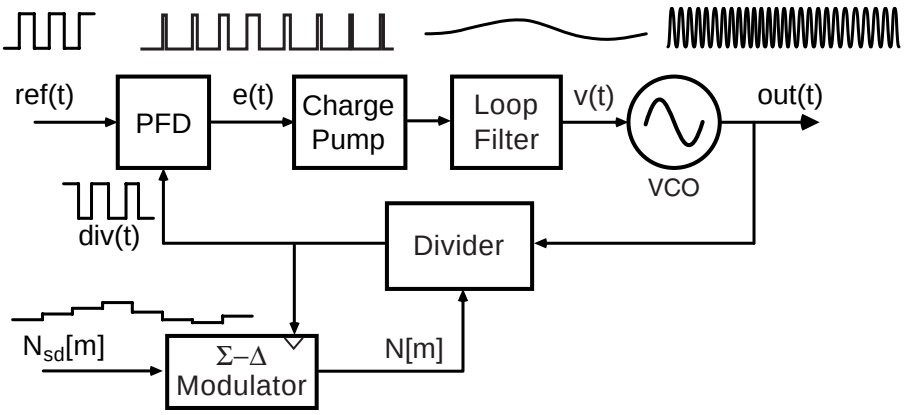

Figure 1: A Σ–Δ Fractional-N Frequency Synthesizer.

Dithering of the divide value by the Σ–Δ modulator allows high frequency resolution to be achieved, but also has the negative side effect of introducing quantization noise that degrades the overall PLL noise performance [1]. It is highly desirable to be able to calculate the impact of this quantization noise, along with other noise sources in the PLL shown in Figure 2, on the overall PLL performance. It is also desirable to calculate the dynamic response of the synthesizer in response to variations of the Σ–Δ input in order to evaluate stability and characterize the performance of the system when it is used as a transmitter [2]. The PLL Design Assistant program allows us to achieve both of these objectives by leveraging the modeling approach described in [3].

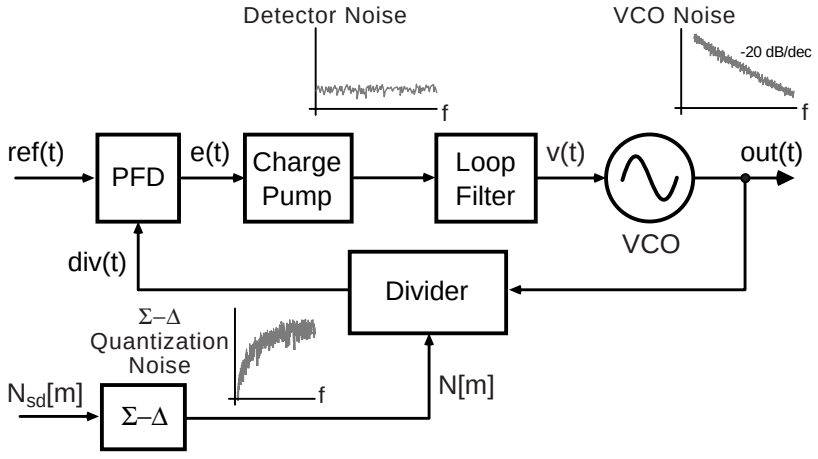

Figure 2: Noise Sources in a Σ–Δ Fractional-N Frequency Synthesizer.

Before continuing, it is worthwhile to mention the PFD topologies that will be assumed in this document. Figure 3 illustrates two popular structures that will be considered here, namely the XOR-

based and Tristate approaches. The XOR-based approach [8] has the advantage of achieving high linearity in its phase error characteristic and completely avoids dead zone issues, but has the disadvantage of producing higher detector noise from the charge pump as well as higher reference spurs in the PLL output phase noise than the Tristate design. The Tristate design [4] produces narrow pulses such that the charge pump need only be on during a small portion of the reference cycle, thereby reducing charge pump noise and reference spur feedthrough compared to its XOR-based counterpart. However, any mismatch between the up and down paths of its output degrades the linearity of its phase error characteristic, and care must be taken to insert a delay in the reset path of its registers to avoid dead zone problems. In general, it is our experience that XOR-based designs are the preferred choice for Σ–Δ fractional-N frequency synthesizers since high linearity is required to avoid folding the Σ–Δ quantization noise, and because charge pump noise and reference feedthrough are reduced in impact by the ability to have a high reference frequency. For classical integer-N synthesizers (i.e., for which the divide value is not dithered), high linearity is not required of its phase detector characteristics, so that the Tristate design is superior due to the reduced charge pump noise and reference feedthrough that it produces.

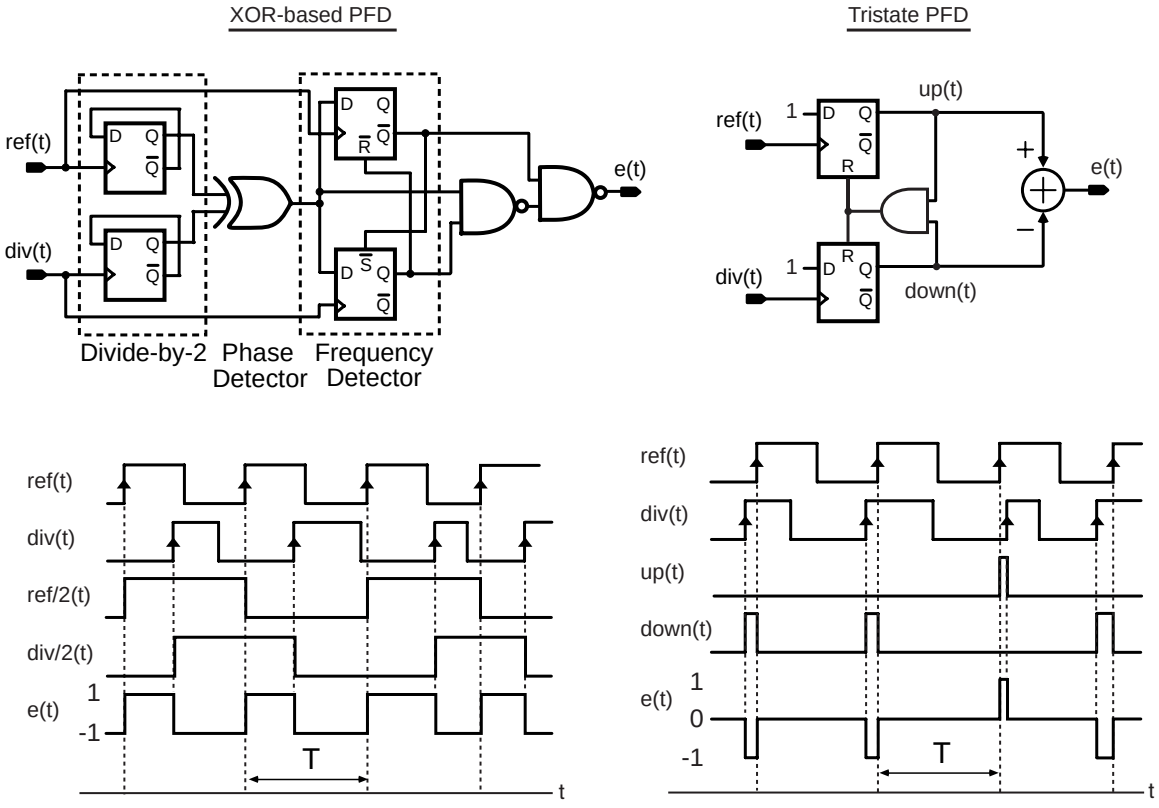

Figure 3. Phase-Frequency detector (PFD) topologies considered here.

It will be assumed in our modeling effort that the charge pump scales the PFD output by the value I_{cp} before the signal is fed into the loop filter. In other words, the input to the loop filter is a current that alternates between I_{cp} and $-I_{cp}$ if an XOR-based PFD is used, or a current that alternates between I_{cp}, 0, and $-I_{cp}$ if a Tristate PFD is used.

Given the above assumptions, a linearized model of the Σ–Δ fractional-N frequency synthesizer presented in [3] is shown in Figure 4. As discussed in [3], this linearized model can be used to analyze

the “small signal” dynamic properties of the PLL as well as its noise performance. By “small signal”, we consider only variations in the PLL frequency caused by small changes in the divide value. If the divide value is stepped by a large value, the PLL will cycle slip many times before re-acquiring frequency lock, and thereby invalidate our modeling assumptions. Fortunately, a well designed Σ–Δ fractional-N frequency synthesizer system can always be ramped in frequency slowly enough to avoid losing frequency lock, so that our modeling assumptions will be accurate for most practical cases. Otherwise, if an estimate of the nonlinear behavior exhibited during cycle slipping is desired, simulation tools such as described in [5] can be used.

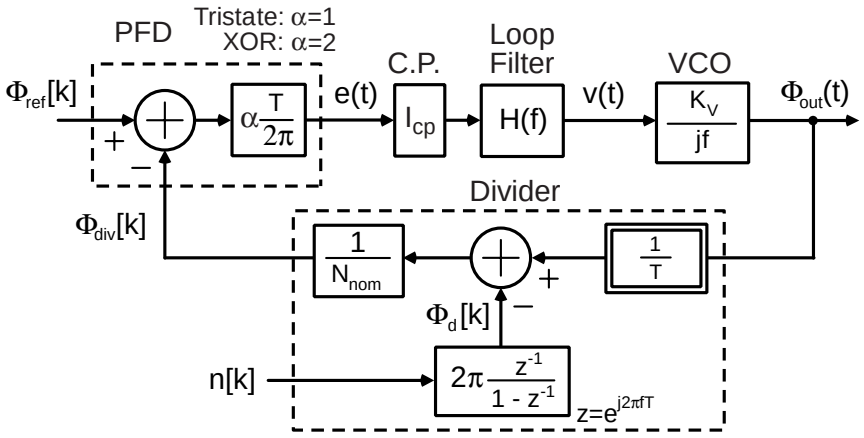

Figure 4. Linearized model of the Σ–Δ fractional-N frequency synthesizer.

The model shown in Figure 4 can be used to design the PLL dynamics to be stable using classical techniques based on examining the phase margin of its open loop response. However, the PLL Design Assistant software allows a much more direct means of design by enabling the user to directly set the characteristics of the closed loop dynamics rather than inferring the closed loop behavior from open loop analysis. The key to the approach is to first design the closed loop dynamics according to desired specifications, and then calculate the open loop response required to realize the desired closed loop behavior.

As discussed in [3], the closed loop response of the system can be parameterized in terms of a single transfer function we label as G(f). Table 1 describes the relationship between G(f) and its open loop counterpart, A(f). The open loop transfer function is, in turn, calculated from Figure 4 as specified in the table. Note that, throughout this document, we will switch between f, w, and s as the frequency variables of interest. In every case, we will assume that $w=2\pi f$ and $s=jw$.

Relationship between Closed Loop and Open Loop Transfer Functions

	f-domain	w-domain ($w=2\pi f$)	s-domain ($s=jw$)
Closed Loop	$G(f) = \frac{A(f)}{1 + A(f)}$	$G(w) = \frac{A(w)}{1 + A(w)}$	$G(s) = \frac{A(s)}{1 + A(s)}$
Open Loop	$A(f) = \frac{K_v I_{cp} \alpha H(f)}{N_{nom} 2\pi jf}$	$A(w) = \frac{K_v I_{cp} \alpha H(w)}{N_{nom} jw}$	$A(s) = \frac{K_v I_{cp} \alpha H(s)}{N_{nom} s}$

Table 1. Relationship between G(f) and A(f).

Once the open loop transfer function is calculated, the last step in the design process is to realize an appropriate loop filter implementation. The methodology of doing so will be explained later in this document.

Definitions

Given the above background, we can summarize the design process as

1) Designing G(f) to achieve desired closed loop characteristics,
2) Designing A(f) to realize the desired G(f).

In this section, we will outline notation and background to provide understanding of how to accomplish step 1. The following section will provide details describing how step 2 is to be achieved.

A. Bandwidth, Order, and Shape

Figure 5 illustrates the definitions of bandwidth and order for G(f) that will be assumed in this document.

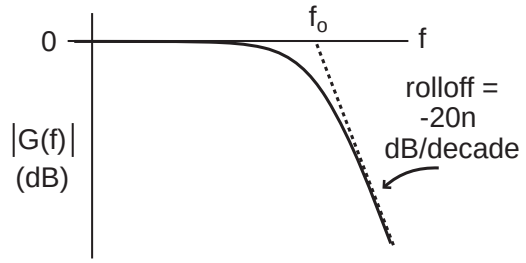

Figure 5. Definition of order and bandwidth for G(f).

Bandwidth is denoted as f_o, and is defined in an asymptotic manner as depicted in the figure above. This definition of bandwidth turns out to be more straightforward for calculations than the traditional 3 dB bandwidth specification. As an example, the asymptotic bandwidth of

$$G(f) = \frac{1}{(1+s/(2\pi f_p))^2}$$

is simply f_p, while the 3 dB bandwidth is $.6423*f_p$.

Order is denoted as n, and is defined according to the rolloff characteristic of the magnitude of G(f) rather than the number of independent state variables in the system. The reason for this definition is that the rolloff of G(f) is more directly of interest to the designer than the number of independent state variables.

The shape of G(f) is defined according to standard filter design methodologies. In particular, it describes the shape of the transfer function and corresponding step response for a given specification of order and bandwidth. Typical examples of shape specifications include Butterworth, Bessel, and Chebyshev.

It is important to note that the above definition of shape considers only the dominant poles in the PLL system. Loosely speaking, the dominant poles are the ones that most strongly influence the behavior

of G(f) and are directly specified to achieve the desired G(f) characteristic. In practice, any practical system also includes parasitic poles and zeros that have second order effects on G(f). We will consider the impact of such parasitics at a later point in this document.

B. Type

The type of a PLL is defined as the number of integrators in its open loop transfer function. In practical PLL implementations, the type is either I or II (note that the VCO contributes one integrator, so the type must be at least I). A type I PLL does not contain an integrator in its loop filter transfer function, whereas a type II PLL does contain an integrator in its loop filter transfer function. Most PLL's are implemented as type II systems for reasons to be described. However, type I systems have an advantage of having faster settling times [6], as will be discussed.

The primary advantage of including an integrator in the loop filter transfer function, as encountered in type II systems, is that it allows the output of the loop filter to achieve an arbitrary DC value while simultaneously forcing the phase error (i.e., the input of the loop filter) to have a steady-state value of zero. Forcing the phase error to zero is quite advantageous in many applications – the phase detector maintains a consistent DC operating point, and the value of the phase error is well controlled.

The DC level shifting property of an integrator is contrasted to that of a gain block in Figure 6. We see that the output DC level of a gain block directly depends on the DC level of its input, whereas the DC output value of an integrator can be arbitrarily set while maintaining a steady-state DC input value of zero.

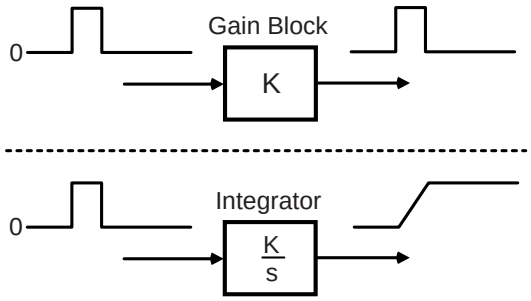

Figure 6. Illustration of integrator's ability to achieve arbitrary output DC levels.

Aside from the benefit of maintaining a consistent steady-state phase error value, type II PLL implementations also solve a DC range issue associated with the VCO control voltage. As illustrated in Figure 7, it turns out that most type I PLL implementations have insufficient gain in their loop filter to span the entire range of the VCO input. To understand this issue, first consider the fact that practical phase detector characteristics have a finite output range which, in turn, sets a limit on the range of the type I loop filter output. The actual range of the loop filter output is then set by its gain, which, in turn, must be set according to the desired bandwidth of the PLL and the value of the VCO gain, K_v (i.e., lowering the bandwidth of the PLL requires lower loop filter gain, and increasing K_v also requires lower loop filter gain). For most practical PLL systems, a reasonably high K_v value is required to meet frequency tuning range requirements and a relatively low PLL bandwidth for noise reasons, so that the loop filter output lacks sufficient range to span the entire voltage range input of the VCO.

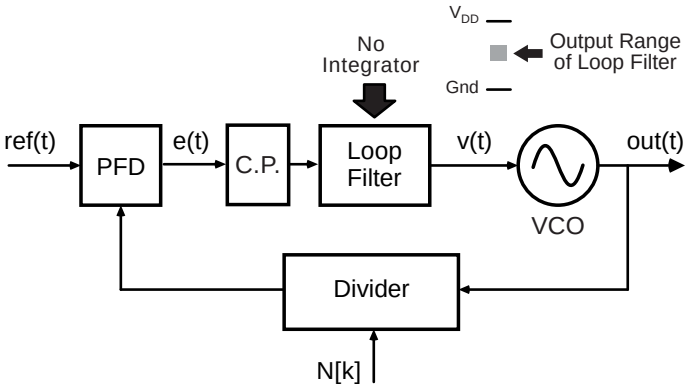

Figure 7. The problem of limited VCO range when using type I PLL's.

Figure 8 illustrates how the entire VCO frequency range is achieved with type I and type II PLL implementations. For type I systems, a D/A converter is added to the system in order to provide coarse tuning of the VCO control voltage, and the loop filter provides fine control of the VCO control voltage through the feedback action of the PLL. For type II systems, the entire VCO range is automatically achieved due to the fact that the integrator within the loop filter can arbitrarily set the DC level of the loop filter output. The simplified implementation of the type II approach has made it the popular choice among PLL designers.

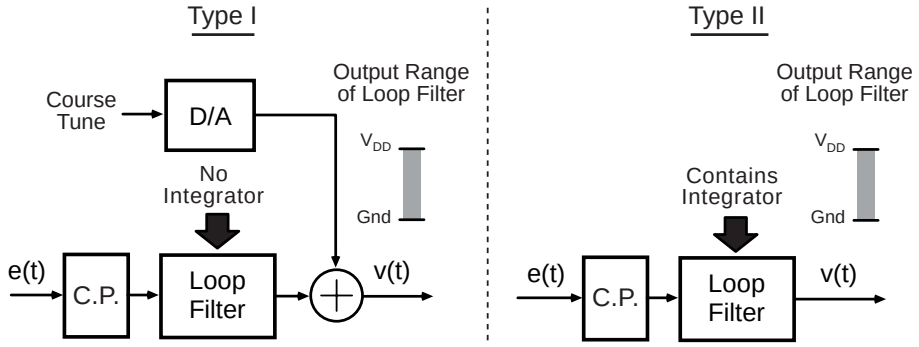

Figure 8. Achievement of the entire VCO range with type I and type II PLL's.

Given the simplicity of type II systems in achieving a consistent steady-state phase detector error and a full span of the VCO range, one might wonder why type I systems are ever employed. The answer is provided in Figure 9 – type II implementations yield undesired peaking behavior in the closed loop transfer function G(f), and also increase the closed loop settling time of the PLL. The key parameter that dictates both the peaking and settling time values is the ratio of the open loop zero, f_z, to the closed loop bandwidth, f_o, where f_z is included in the open loop transfer function in order to achieve stability of the system. Note that the transfer function peaking is parameterized in terms of the ratio f_{cp}/f_z (defined in [7]) – this ratio depends on the value of f_z/f_o, as explained in [7]. In any case, as f_z/f_o is increased, the magnitude of the frequency peaking and the amount of overshoot in the step response is increased, but the settling time is decreased. Typically, the value of f_z/f_o is set in the range of 1/10 to 1/6 (it must be much less than one to achieve stable PLL dynamics).

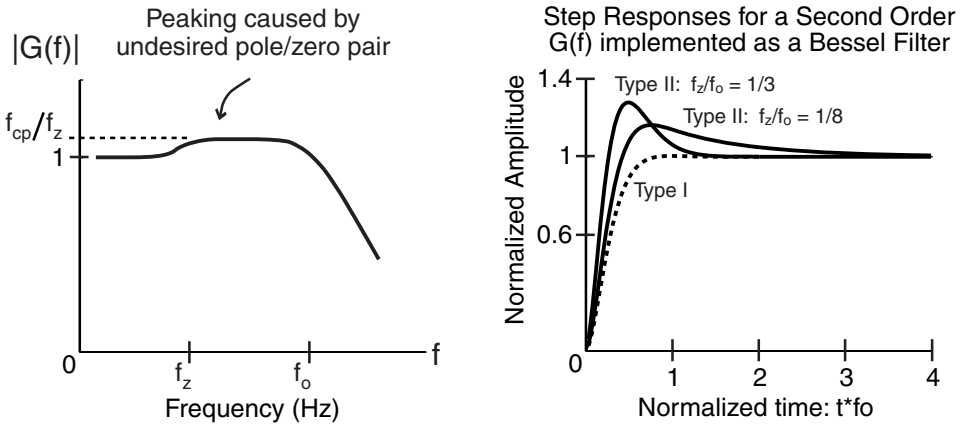

Figure 9. The negative consequences of using type II PLL implementations.

Computation of G(f)

Given the definitions in the previous section, the transfer function G(f) is designed automatically by the PLL Design Assistant software package. Specifically, the user simply enters in desired values for bandwidth, order, shape, type, and f_z/f_o, and the corresponding G(f) transfer function is automatically computed using classical filter design routines. The next section describes how to design the open loop PLL transfer function to achieve the desired G(f) response.

Loop Filter Design

The key PLL component used to achieve a desired PLL transfer function is the loop filter. In particular, the loop filter implements desired poles and zeros for realizing a given open loop transfer function and, in combination with the charge pump, also sets the overall open loop gain of the system. Three steps must be undertaken in designing the loop filter – selection of its transfer function, selection of its topology, and selection of its transfer function values (and, thereby, selection of the component values in its chosen topology).

A. Transfer Function Selection

To design the loop filter to achieve a desired G(f) specification, both its transfer function and the value of parameters associated with that transfer function must be properly chosen. The appropriate loop filter transfer function required to realize a given G(f) function is found using the lookup table shown in Table 2. The relevant specifications for this task are the desired order and type of G(f) – note that the bandwidth and shape of G(f) are controlled by the parameter values of the given loop filter transfer function. It should be noted that the table only includes a limited set of orders (1, 2, and 3) and types (I and II), but these are sufficient for most PLL applications due to the prohibitively high analog complexity required to achieve higher order or higher type values.

H(s) Topology For Different Type and Orders of G(f)

	Type I	Type II
Order 1	K_{LP}	$K_{LP}\frac{1+s/w_z}{s}$
Order 2	$\frac{K_{LP}}{1+s/w_p}$	$K_{LP}\frac{1+s/w_z}{s(1+s/w_p)}$
Order 3	$\frac{K_{LP}}{1+s/(w_pQ_p)+(s/w_p)^2}$	$\frac{K_{LP}(1+s/w_z)}{s(1+s/(w_pQ_p)+(s/w_p)^2)}$

where $K_{LP} = K\frac{N_{nom}}{K_v I_{cp}\alpha}$

Calculated from software

Table 2. Loop filter transfer function for G(f) of different order and types.

Examination of Table 2, as well as Figure 4, reveals that the desired DC gain of the loop filter, K_{LP}, is a function of a gain term, K, and the PLL parameters:

1) Nominal divide value, N_{nom},
2) VCO gain, K_v, (units are assumed to be Hz/V),
3) PFD topology, α,
4) Charge pump current, I_{cp}. (units assumed to be Amps)

The gain term, K, will be determined automatically for a given G(f) specification by the PLL Design Assistant software. As for the other parameters, the nominal divide value is set as the ratio of the output frequency to the choice of reference frequency, the VCO gain is typically set by the desired frequency range of the PLL, and the PFD topology is set according to the benefits of an XOR-based versus Tristate design. However, the charge pump current is a free parameter, and can be adjusted to either increase or decrease the required value of K_{LP}. As will be discussed in the noise analysis section of this document, the appropriate setting of the charge pump current is determined by the required noise performance of the PLL – a higher charge pump current value yields lower detector noise in general. The maximum value of the charge pump current is typically constrained by limits on the loop filter component values or power dissipation.

B. Circuit Topology Selection

Given that the loop filter transfer function has been chosen according to Table 2, an appropriate loop filter topology is then selected to realize that transfer function. Popular choices for loop filter topologies can be divided into two categories: passive and active.

Figure 10 illustrates possible loop filter implementations using passive components. The advantage of the passive approach is that good noise performance can be achieved with minimal power consumption, and that the overall PLL implementation is simplified by minimizing the amount of active circuitry. These advantages have led to widespread use of the RC network shown for the Type II, Order 2 implementation in Figure 10. Passive approaches are rarely used for higher order loop filter implementations due to the awkwardness of using inductors.

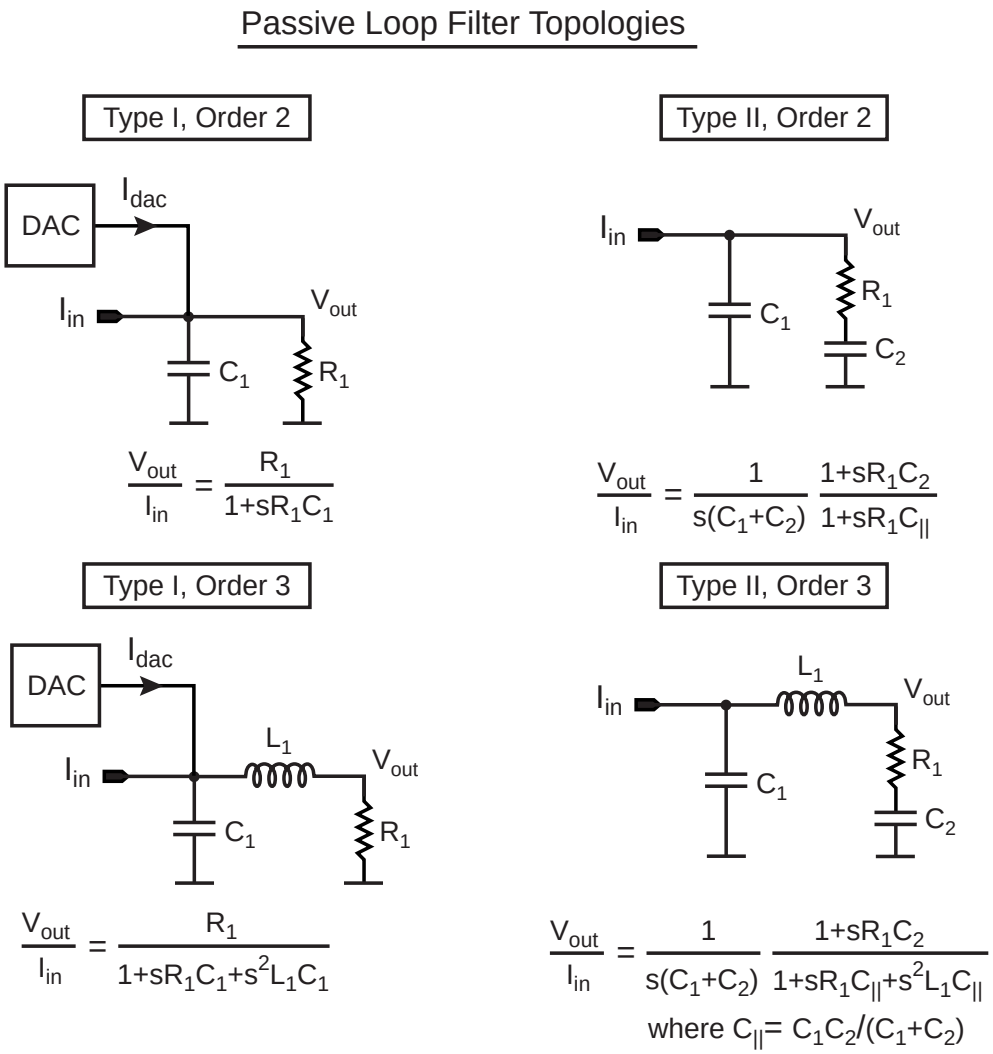

Figure 10. Passive loop filter topologies.

Figure 11 illustrates active counterparts to the passive topologies shown in Figure 10. There are two advantages to using an active, rather than passive, approach for the loop filter implementation: 1) the charge pump need not support a wide voltage range at its output since the active approach consistently sets its value to that of a reference voltage, 2) third order PLL dynamics can be achieved without requiring inductors.

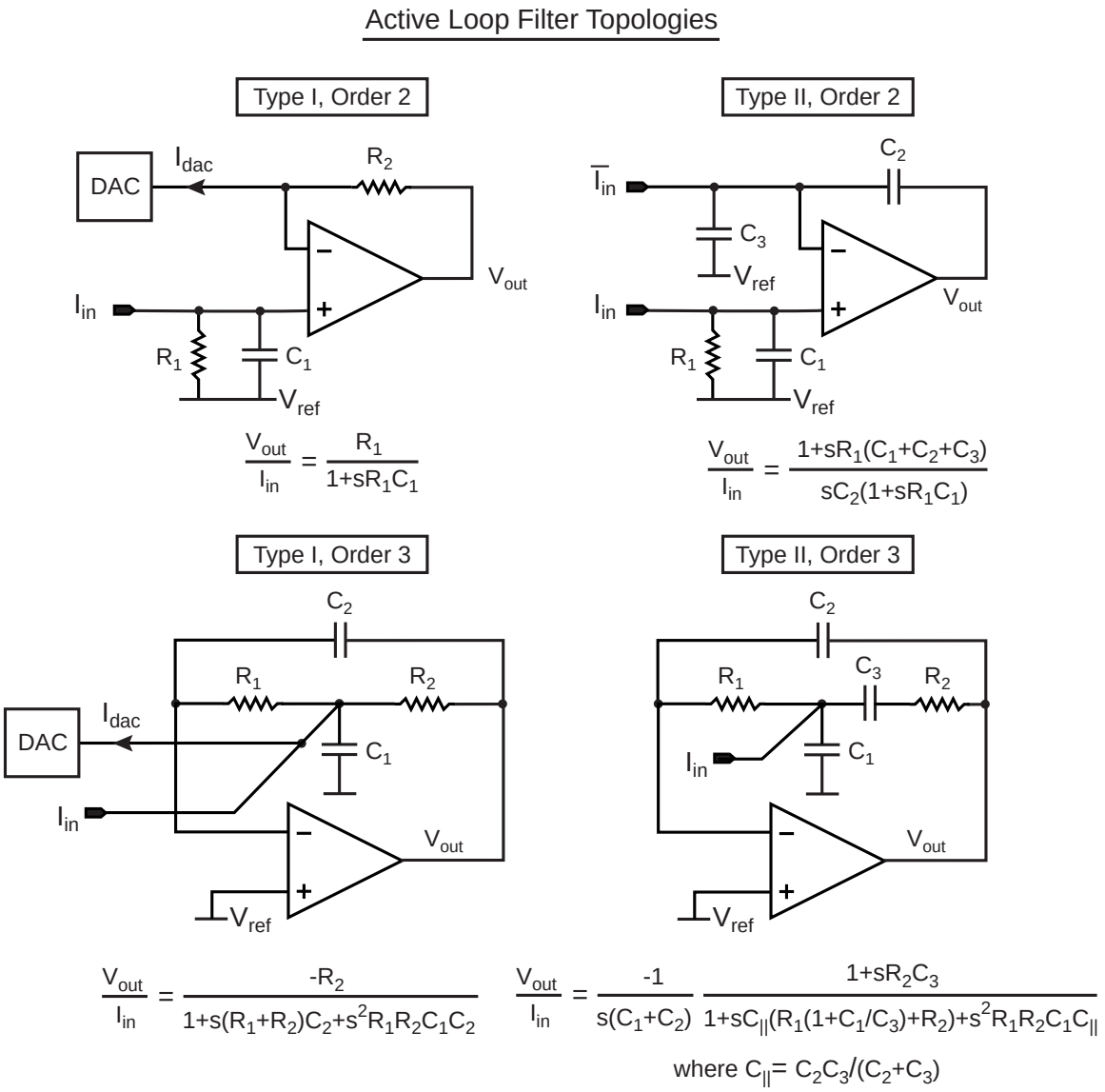

Figure 11. Active loop filter topologies.

In general, there are two design rules that should be followed when using an active loop filter implementation. First, the output of the charge pump should always feed directly into a high-Q capacitor (i.e. a capacitor with minimal series resistance) in order to attenuate its high frequency content before it feeds into the loop filter opamp. The reason for doing so is that the opamp has limited bandwidth and can exhibit nonlinear behavior if it is directly driven with the high frequencies that are present in the charge pump output. The second rule is that the feedback of the opamp should be configured to achieve unity gain from the opamp terminals to its output. By doing so, the input referred noise of the opamp is not amplified in its influence on the loop filter output. The active topologies shown in Figure 11 achieve both of these desired characteristics.

C. Computation of Parameters

Given that the loop filter transfer function and its corresponding topology have been selected, the next step is to calculate the parameters of the transfer function and then the component values of its

corresponding topology. It is at this point that we now turn to the PLL Design Assistant software as it will perform this task for us. We will use an example to illustrate the relevant principles.

D. Example Design

Design a third order, Type II PLL with a Butterworth filter response and bandwidth of f_o=300 kHz. Choose the ratio of f_z/f_o to be 1/8.

Step 1: Select the loop filter transfer function based on Table 2:

$$H(s) = \frac{K_{LP}(1+s/w_z)}{s(1+s/(w_pQ_p)+(s/w_p)^2)} \qquad \text{where } K_{LP} = K\frac{N_{nom}}{K_v I_{cp}\alpha}$$

Step 2: Select the corresponding loop filter topology.
Step 3: Solve for K, w_p, Q_p, and w_z in H(s) by using the software tool. Once these values are known, they are used to solve for the component values in the chosen loop filter topology. Figure 12 illustrates the entry of the desired G(f) parameters into the tool along with the resulting loop filter parameter values that it calculates (note that $w_p = 2\pi f_p$ and $w_z = 2\pi f_z$). Figure 13 displays the resulting closed loop step response and pole/zero locations of the PLL. Note that one can also plot the corresponding closed loop transfer function if desired. In addition, the user can set the axis limits for all plots in the designated boxes, and can zoom into a portion of the plot figure by simply dragging a box across the desired plot area with the mouse.

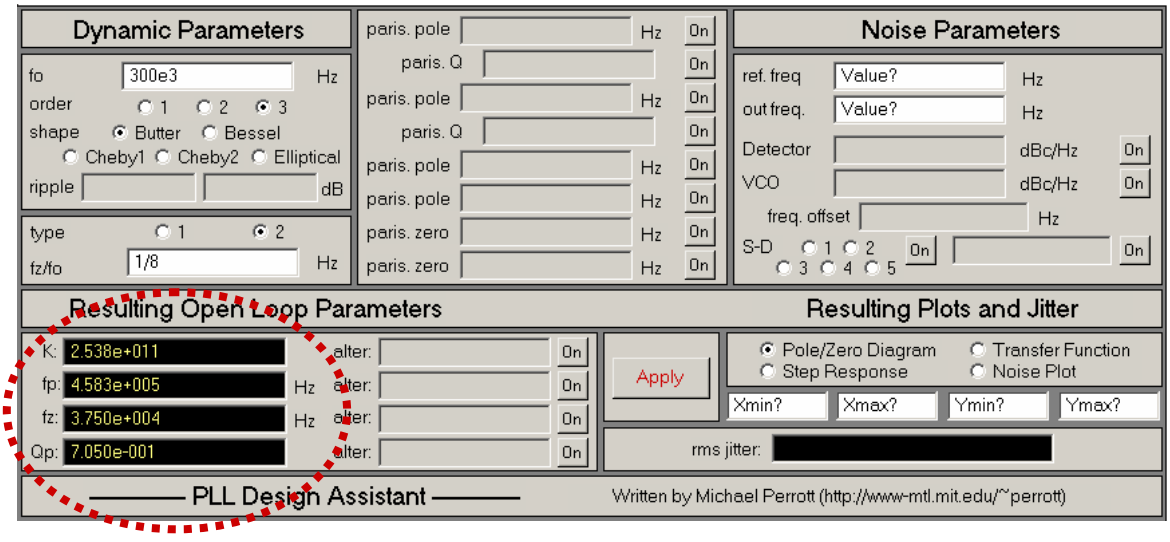

Figure 12: Calculate loop filter parameter values using the PLL Design Assistant.

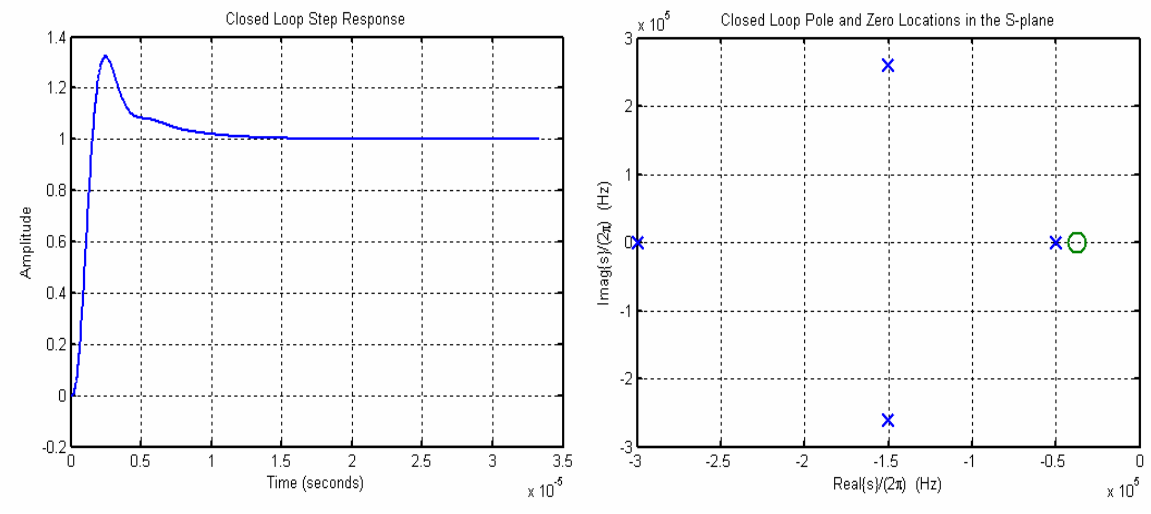

Figure 13: Plot of closed loop step response and pole/zero locations using the tool.

Impact of Open Loop Parameter Variations

In a practical PLL implementation, the value of the open loop gain and loop filter pole and zero locations will vary from chip to chip and across temperature. Classical PLL design methodologies address this issue by examining the phase margin of the open loop transfer function across all such variations to infer whether the system remains stable. The problem with this approach is that it provides little information on other important issues such as the change in settling time or closed loop transfer function that will occur due to such changes.

Using the PLL Design Assistant, one can directly examine the changes in the closed loop step response, frequency response, and pole/zero locations by simply entering in the variations of each open loop parameter into the tool.

Figure 14 displays the inclusion of open loop parameter variations in the tool. These variations are entered using Matlab notation, so that, for instance, values ranging from -20% to 20% in increments of 10% would be specified as

-0.2:0.1:0.2

In this case, we vary the open loop gain and loop filter pole values by percentage changes of {-20%, 0, 20%} as specified by the notation -0.2:0.2:0.2

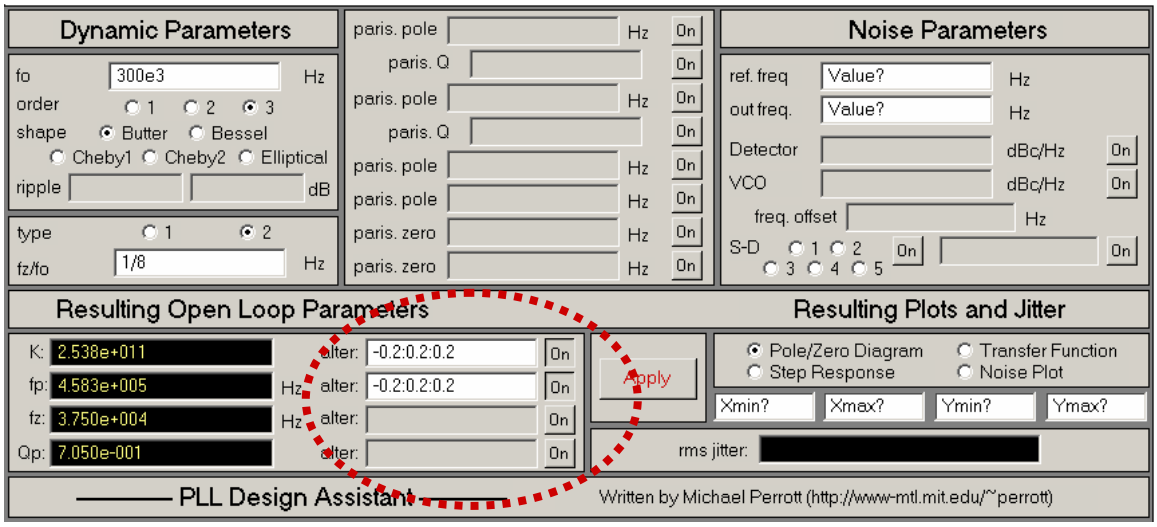

Figure 14: Examine the impact of changes in the open loop parameters.

Figure 15 displays the resulting closed loop step response and pole/zero locations due to the above open loop parameter variations. The step response plots allow us to quickly assess that the system has no stability issues over the range of parameter variations entered. The pole/zero locations also confirm that the system remains stable as evidenced by the fact that the poles remain in the left half of the S-plane. Note that the user can effectively do root locus plots as a function of open loop gain or other open loop parameters with this feature.

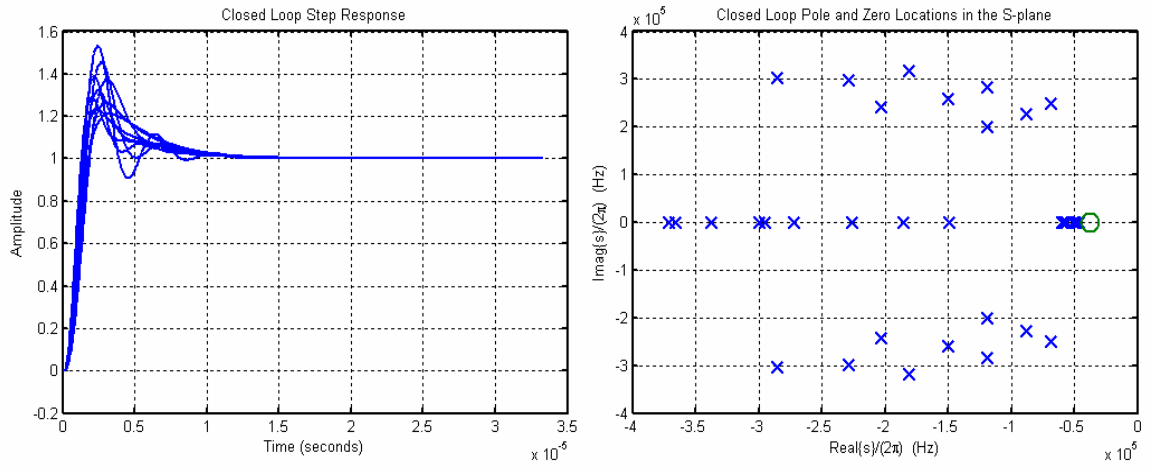

Figure 15. Resulting variations in the step response and pole/zero locations.

Impact of Open Loop Parasitic Poles and Zeros

In addition to open loop parameter variations, a practical PLL implementation also includes parasitic poles and zeros in its open loop transfer function that shift the ideal closed loop function G(f) away from its desired shape. More specifically, the dominant closed loop poles, which were designed under ideal assumptions, shift away from their desired positions when parasitic poles/zeros are included. The resulting system is typically inferior with respect to its robustness against instability, and does not retain the desired shape (i.e., Butterworth, etc.) for G(f).

The PLL Design Assistant allows the designer to directly accommodate parasitic poles and zeros into the design process. When a nominal value for the parasitic pole/zero locations is provided (as obtained by Spice simulations or measurements), the tool automatically adjusts the open loop parameter values so that the closed loop dominant pole locations are placed at the same locations as calculated for the ideal case of having no parasitic poles/zeros [7].

As an example, suppose that an active loop filter is used, and the limited bandwidth of the loop filter opamp adds in a complex parasitic pole pair, a real parasitic pole, and a real parasitic zero to the ideal response assumed in Table 2. We can view the addition of the parasitic poles and zeros as a modification of the ideal loop filter transfer function, H(w), as follows:

$$H(w) \Rightarrow H(w)\,\frac{(1+s/w_{paris\_zero1})}{(1+s/(Q_{paris}w_{paris1})+(s/w_{paris1})^2)(1+s/w_{paris2})}$$

Let us further assume that Spice simulations have been performed that provide nominal values of

1) $w_{paris\_zero} = 2\pi(10MHz)$
2) $w_{paris1} = 2\pi(3MHz)$, $Q_{paris} = 4$
3) $w_{paris2} = 2\pi(2MHz)$

Figure 16 displays the entry of these parasitic values into the tool, and shows the updated open loop parameter values that should be used to design the loop filter. Figure 17 displays the updated closed loop step response and frequency response of the system. Examination of the step response reveals that it is relatively unchanged from the ideal case shown in Figure 13. The influence of the parasitic poles and zeros can be better seen in the closed loop frequency response. In particular, the complex parasitic pole pair with Q of 4 is seen to cause a small amount of peaking around the frequency of 3 MHz, as expected.

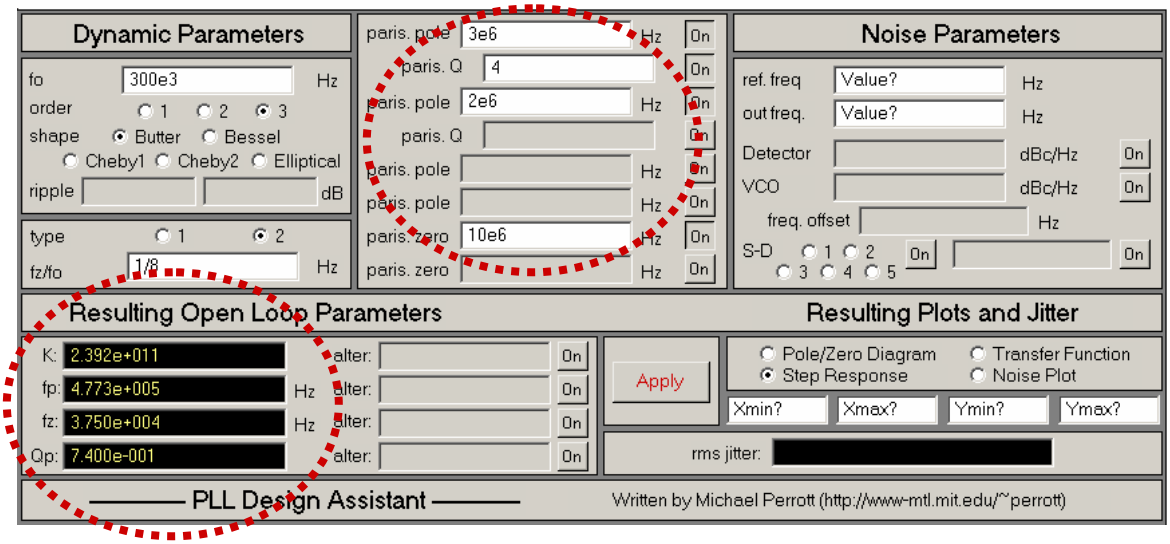

Figure 16. Updated open loop parameter values based on nominal parasitic poles/zeros.

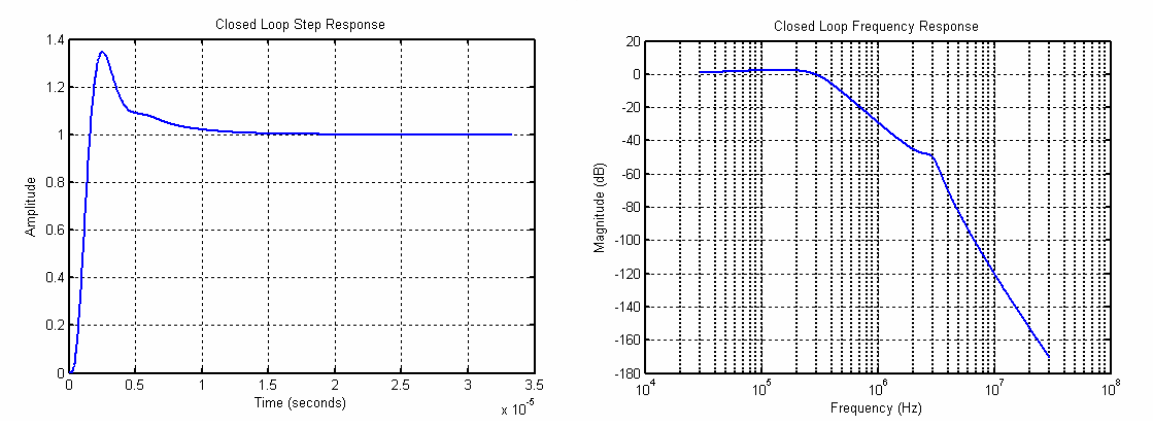

Figure 17. Updated step and frequency response based on nominal parasitic poles/zeros.

Of course, in a practical system, the parasitic poles and zeros will vary from chip to chip and across temperature. The PLL Design Assistant allows one to assess the impact of such variations through direct entry of the variations into the tool. The notation for doing so is slightly different from that used to specify open loop variations as discussed above. In particular, variations for a given parasitic parameter value are specified in the tool as

[value_nom value_1 value_2 … value_k]

where value_nom denotes the nominal parasitic parameter value that the tool will assume when calculating the open loop parameters to achieve the desired closed loop dominant pole locations. The entries of value_1 through value_k designate values over which the tool will calculate variations in the closed loop response.

Figure 18 illustrates an example in which we have varied the real parasitic pole across the range of 1.2 MHz to 2.8 MHz, with a nominal setting of 2 MHz. One should observe that the open loop parameter values are unchanged from those in Figure 16 since the nominal parasitic values are unchanged. However, as shown in Figure 19, the step response and frequency response curves produced by the tool now have three curves associated with them corresponding to the nominal case and the two variation values specified.

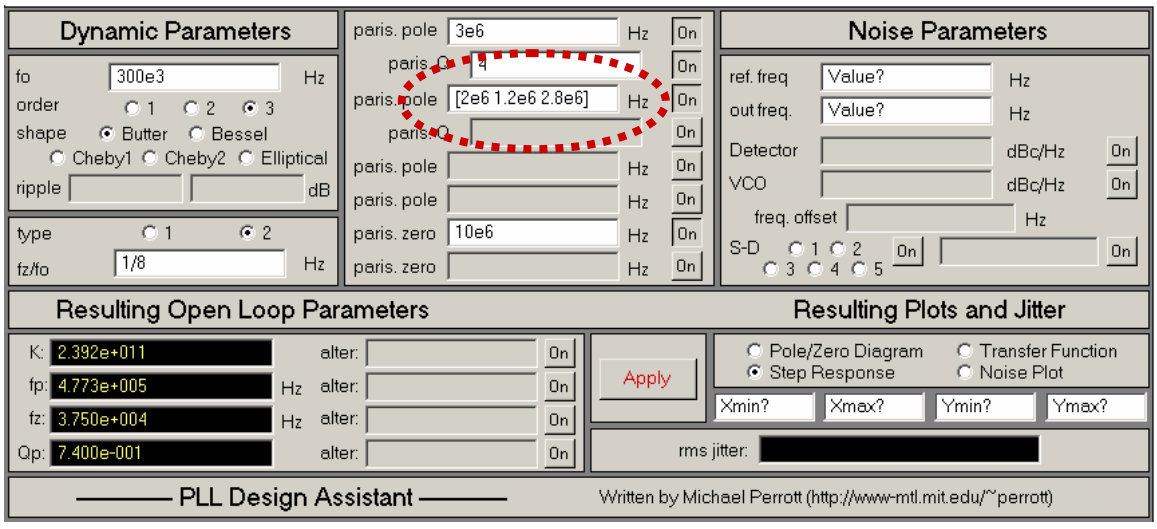

Figure 18. Evaluation of variations in parasitic parameter values.

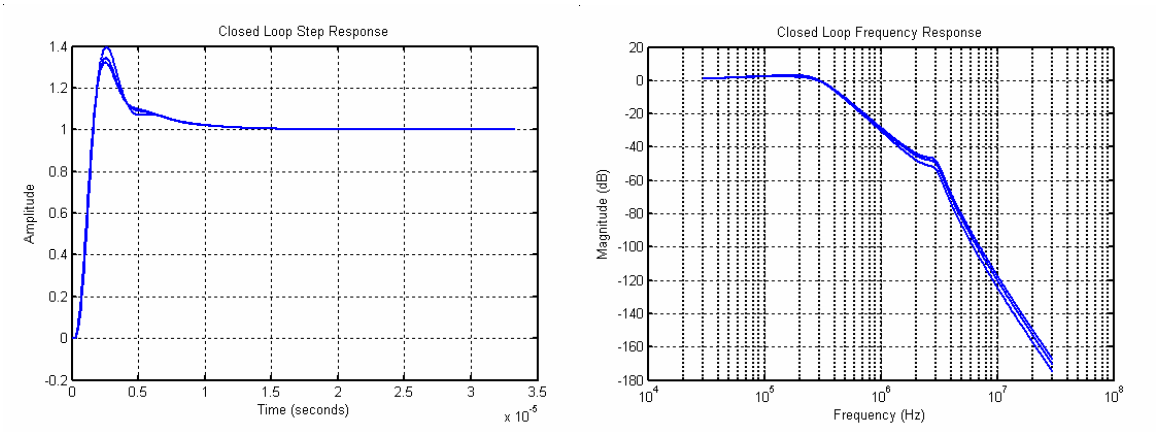

Figure 19. Updated step and frequency response curves based on parasitic parameter variations.

It should be noted that the user can include open loop parameter variations in combination with parasitic parameter variations to examine the overall variations in the closed loop step response, frequency response, and pole/zero values. To do so, the user simply enters in the variation values in the appropriate boxes.

Note:
An important issue that the user must be aware of is that the parasitic pole/zero values cannot come arbitrarily close to the closed loop bandwidth of the system without numerical problems occurring in the tool. If such problems occur, the tool will either warn the user through a dialog box, or simply refuse to display updated plots and values. The user can check error messages generated by the tool by looking at the command prompt that is created when the tool is run.

PLL Noise Performance

The PLL Design Assistant tool calculates the theoretical phase noise and jitter performance of a given phase-locked loop design. This analysis is based on the modeling approach described in [3], which is summarized in pictorial form by Figure 20. As illustrated by the figure, phase noise is assumed to be influenced by two primary sources in the PLL – detector noise and VCO noise. Detector noise is considered to be the addition of white (i.e., has a flat spectrum) and spurious noise, and is composed of noise due to reference and divider jitter, charge pump noise, and spurious noise from the reference frequency (we will ignore spurious noise). VCO noise is assumed to roll off at -20 dB/decade, and is caused primarily by thermal noise sources in the VCO structure. Note that, in practice, VCO noise actually rolls off at a higher rate than -20 dB/decade at low frequencies due to the influence of 1/f noise – we will ignore this fact in our analysis based on the assumption that such low frequency noise is filtered out by the highpass action of the PLL dynamics, as shown in Figure 20, before influencing the output noise of the synthesizer.

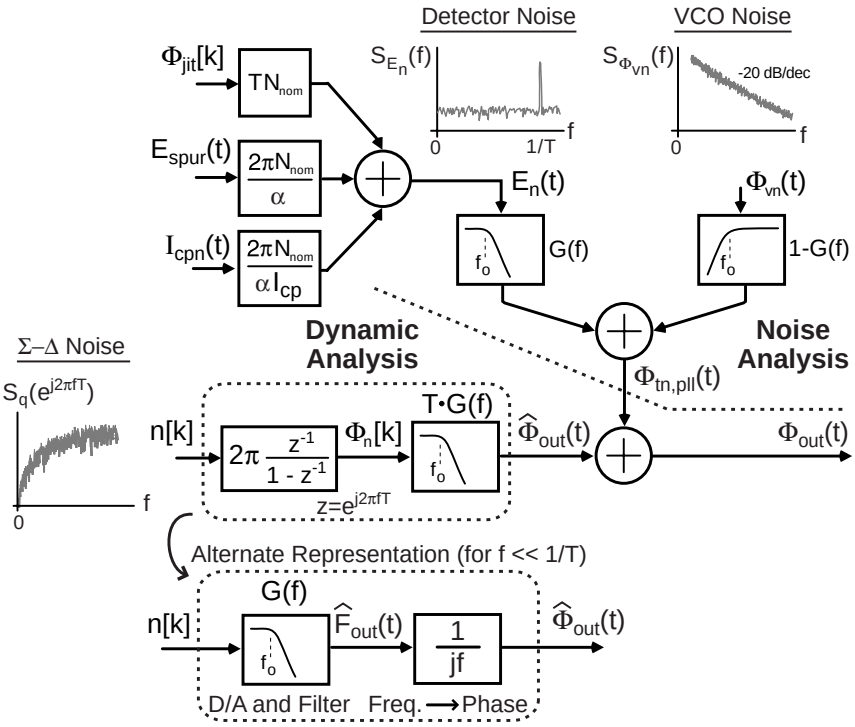

Figure 20. Modeling approach for PLL noise analysis.

To estimate the output phase noise and jitter of a PLL, the user simply enters in the dynamic parameters of the system, as discussed in previous sections, and then enters in the parameters requested in the Noise Analysis section of the tool. These parameters include the reference frequency, output frequency, and the spectral density parameters of the detector noise and VCO noise.

Examination of Figure 20 reveals that Σ–Δ fractional-N frequency synthesizers add an additional noise source – Σ–Δ quantization noise. The spectrum of this noise is shaped according to the order and architecture of the Σ–Δ topology employed. The PLL Design Assistant program will assume that a MASH structure [9], which is sampled at a rate equal to the **ref freq.** parameter in the tool, is used by default, and will automatically calculate the noise spectral density given a specified Σ–Δ order. The default value of the Σ–Δ quantization noise step size is one VCO period, whose value is set as the inverse of the **out freq.** parameter in the tool. The user also has the option of directly entering in the Noise Transfer Function (NTF) of the Σ–Δ modulator [9] if a topology other than the MASH structure is used. The NTF option can be used to alter the value of the Σ–Δ quantization step size, as explained later in this document.

A. Basic Parameter Entry

Figure 21 displays noise parameters entered into the tool given the same dynamic parameters as previous examples. The key parameters for the analysis are the values of the reference frequency, output frequency, detector and VCO noise spectral densities, and the order of the Σ–Δ modulator. Note that each of the noise sources can be selectively turned on or off to allow easy assessment of their individual contributions. In addition to producing a phase noise plot, the tool also estimates the rms jitter at the PLL output. The jitter calculation is performed by integrating the phase noise spectral

density across the frequency range entered into the tool in the region highlighted by the lower circle in Figure 21.

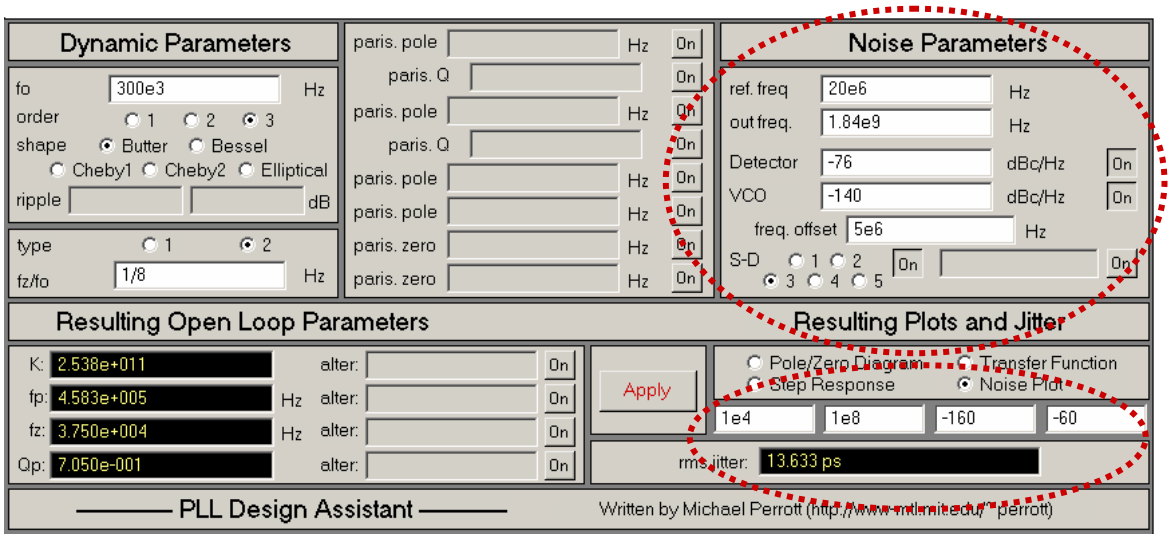

Figure 21. Entry of noise parameters to estimate phase noise and jitter.

Figure 22 displays the phase noise plot produced by the tool given the parameter entries shown in Figure 21. The relative contribution of each noise source is readily seen in the plot, so that we see for this case that Σ–Δ quantization noise dominates in the frequency range spanning 2 to 10 MHz. To confirm the accuracy of this phase noise plot, we also performed a simulation of the system using the CppSim tool available at http://www-mtl.mit.edu/~perrott, which uses simulation techniques described in [5] to achieve high accuracy. The simulated phase noise plot, which is also shown in Figure 22, reveals very close correspondence to its calculated counterpart. Note that the spikes in the simulated plot correspond to spurious noise caused by the reference frequency.

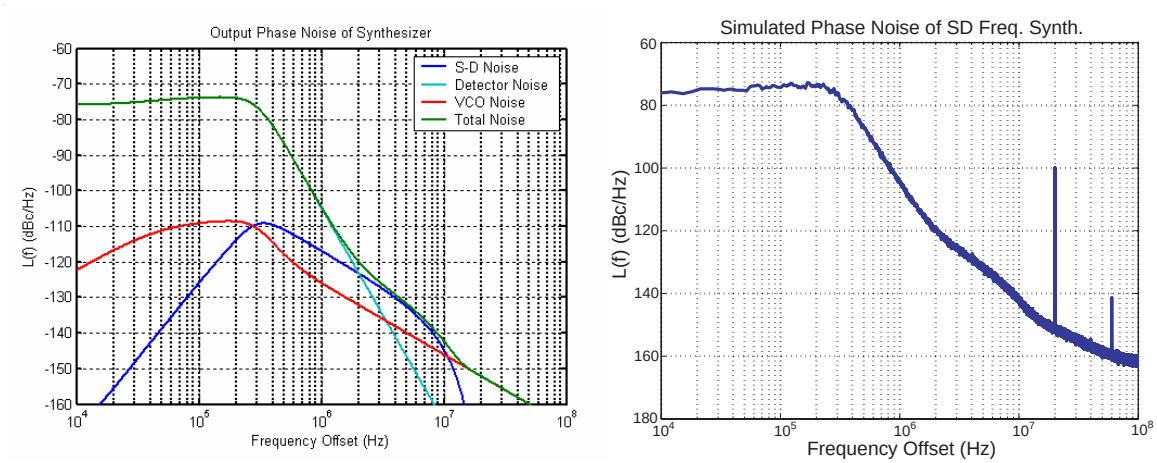

Figure 22. Calculated phase noise compared to simulated results.

B. Inclusion of 1/f Noise

The noise performance of modern integrated phase-locked loops is often degraded by 1/f noise produced in the charge pump and/or reference buffer. The most recent version of the PLL Design

Assistant allows the user to include such noise by specifying its corner frequency relative to the flat portion of the PFD-referred noise. The example below illustrates how to specify a 1/f noise corner of 1 kHz with the flat portion of the output-referred PFD-referred noise being specified at -90 dBc/Hz. Note that the pole/zero double caused by having a type II PLL produces peaking in the noise response close to the offset frequency of 100 kHz. Also note that the phase noise at 1 kHz is -87 dBc/Hz – at this point, the 1/f noise and flat portion of the noise have equal values of -90 dBc/Hz such that their addition leads to overall noise of -87 dBc/Hz at 1 kHz.

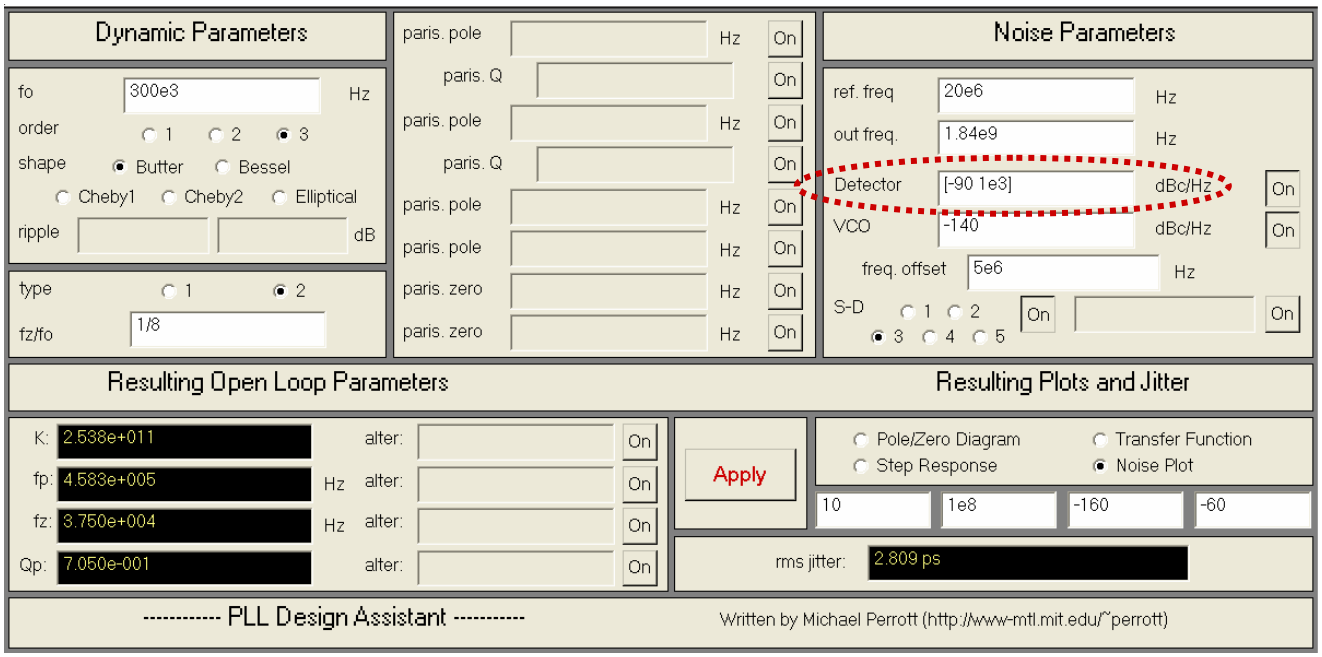

Figure 23. Inclusion of 1/f noise in detector.

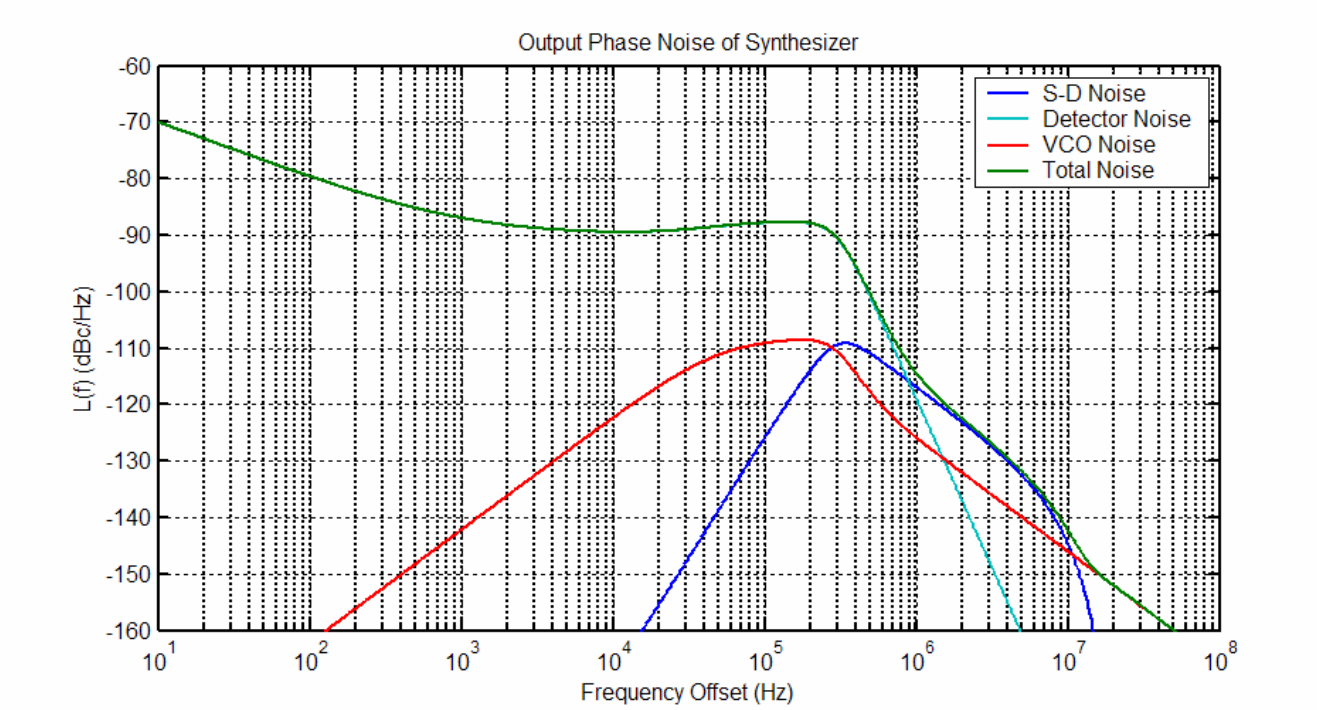

Figure 24. Calculated phase noise with 1/f detector noise included.

In some cases, the 1/f noise does not have a slope of -10 dB/decade as assumed above. The example below illustrates adding 1/f noise with a corner frequency of 1 kHz as above, but with a slope of -15 dB/decade.

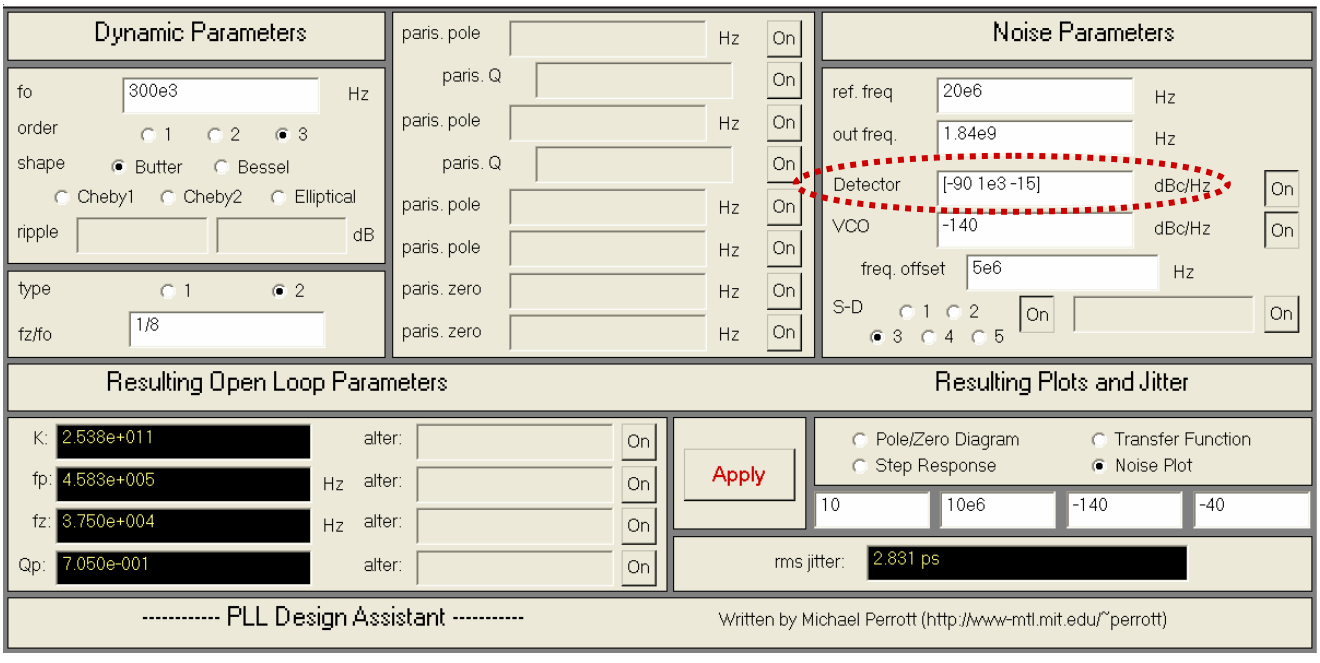

Figure 25. Specification of slope of -15 dB/decade and corner frequency of 1kHz for 1/f noise in detector.

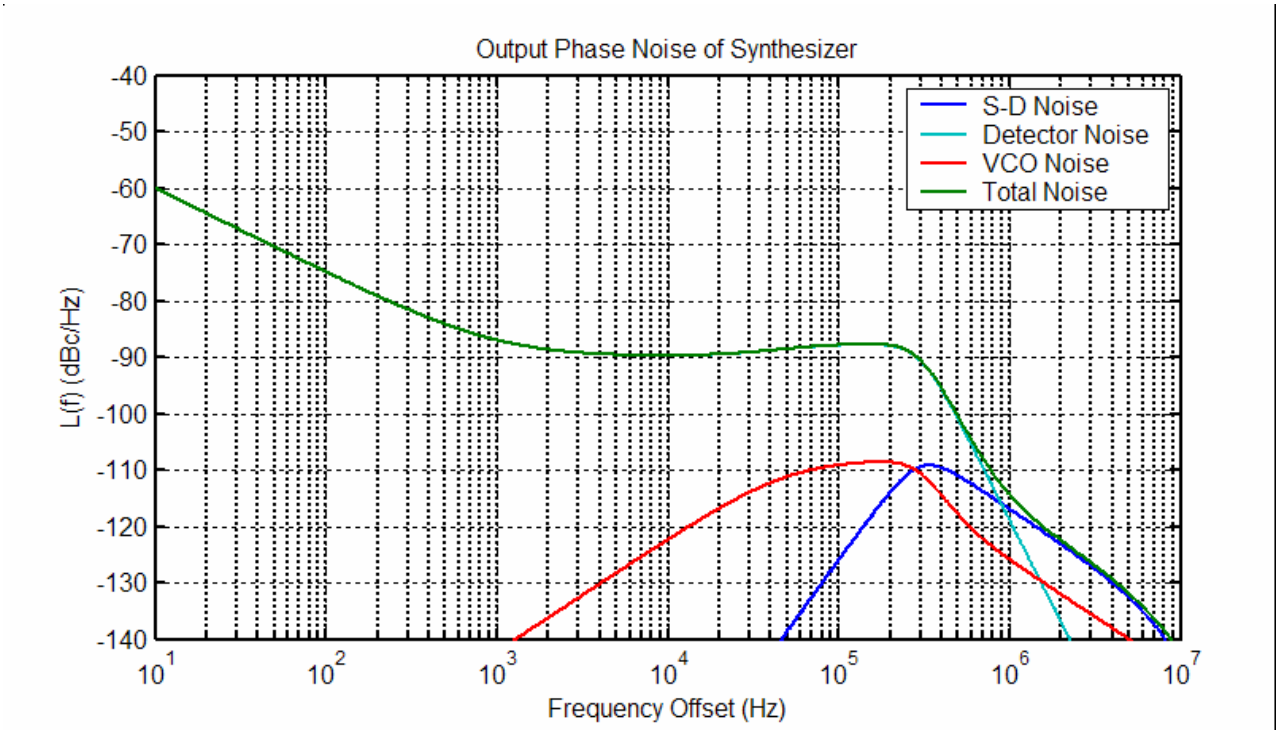

Figure 26. Calculated phase noise with 1/f detector noise of slope -15 dB/decade.

Finally, note that you can also add $1/f^3$ noise to the VCO in a similar manner. The example below illustrates the inclusion of such noise with a corner frequency of 1 kHz and slope of -35 dB/decade (the nominal slope is -30 dB/decade). In this example, all other noise sources were turned off and the PLL bandwidth set to a low value in order to show the VCO noise without influence from the PLL.

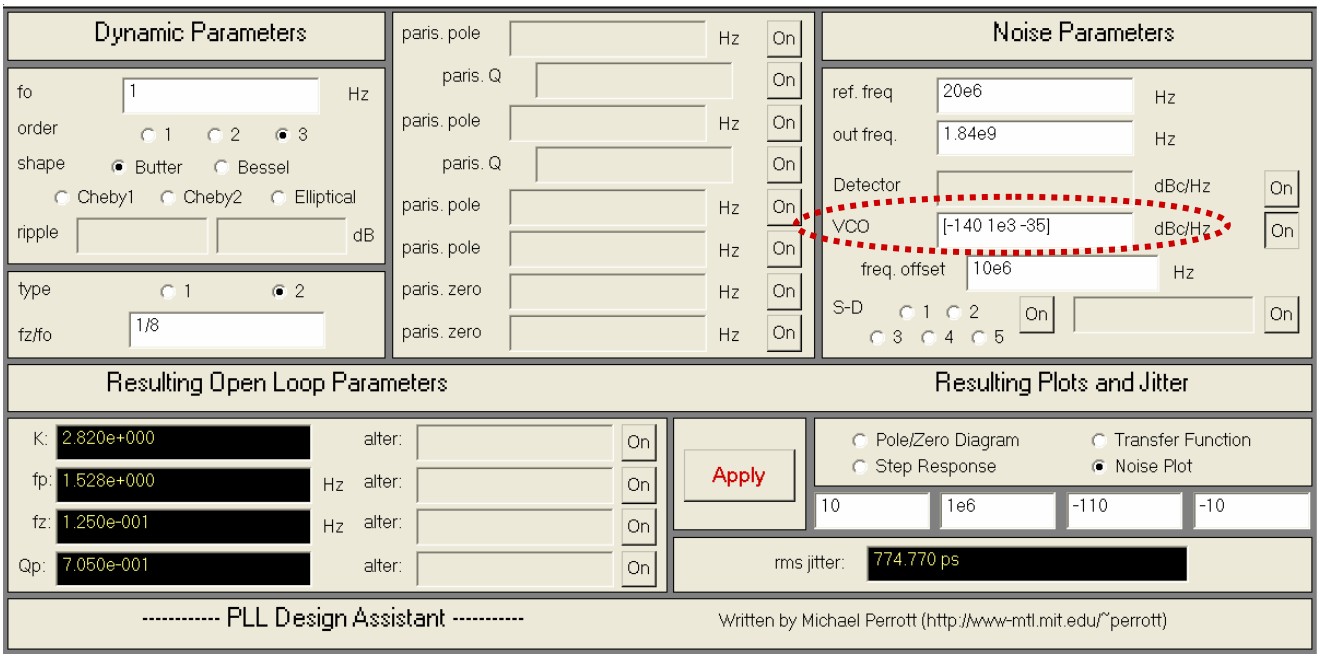

Figure 27. Specification of $1/f^3$ noise in VCO.

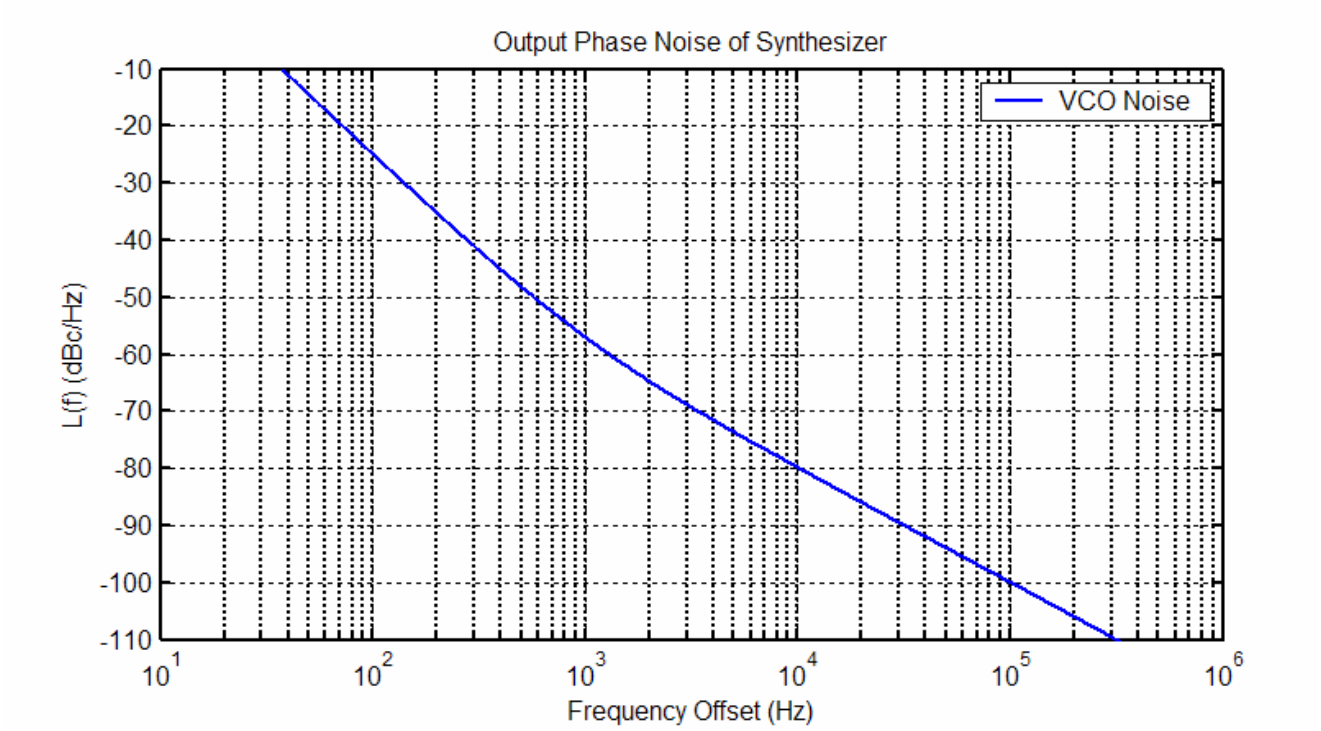

Figure 28. Calculated phase noise of VCO with $1/f^3$ behavior included.

C. Inclusion of Parasitic Poles

Figure 22 revealed that Σ–Δ quantization noise was the dominant influence of phase noise over the frequency range of 2 MHz to 10 MHz. Inspection of Figure 20 reveals that Σ–Δ noise is filtered by G(f), so that purposeful addition of parasitic poles in the loop filter should aid in suppressing this noise source. The PLL Design Assistant tool allows straightforward investigation of this idea – we can simply input an open loop parasitic pole value into the tool and then re-plot the phase noise.

The inclusion of an open loop parasitic pole at 1.2 MHz is shown in Figure 29, and the resulting phase noise plot shown in Figure 30. Note that the closed loop pole corresponding to this parasitic pole is quite close to its open loop value due to the fact that it is placed at a much higher frequency than the dominant poles. Examination of Figure 30 reveals that the Σ–Δ noise has been suppressed, so that it no longer dominates as a noise source. The simulated plot shown in Figure 30 further confirms this result.

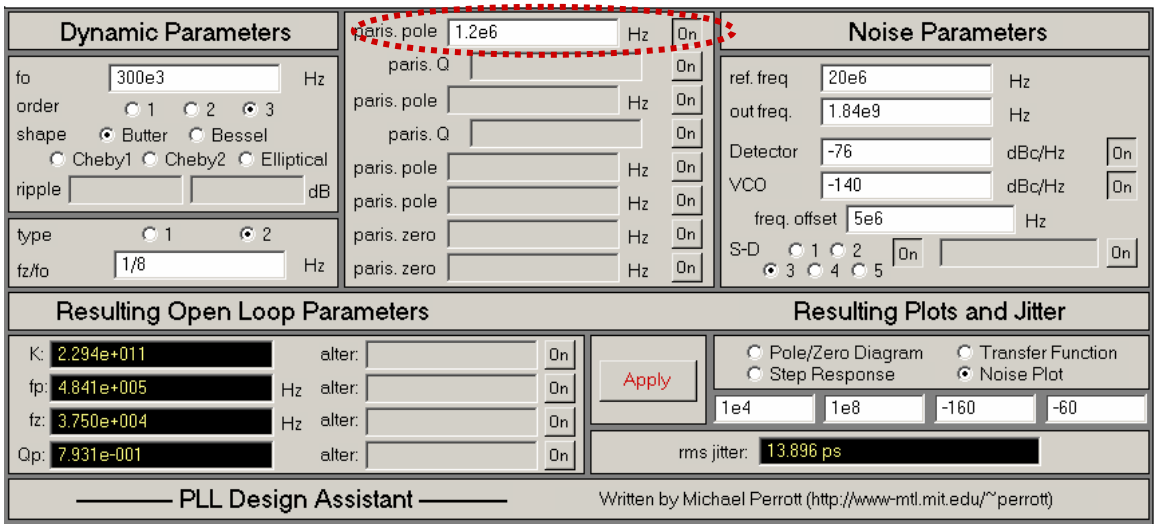

Figure 29. Inclusion of open loop parasitic pole in PLL.

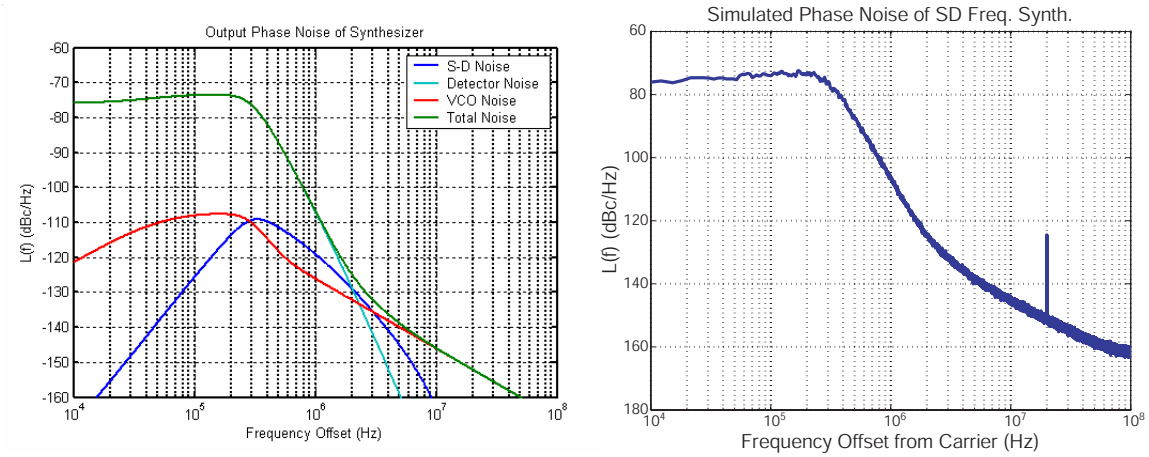

Figure 30. Calculated phase noise compared to simulated results.

D. Impact of Open Loop Parameter Variations

Just as it was important to investigate the impact of open loop parameter variations on the stability of the PLL dynamics, it is also important to assess noise performance over those same variations. To do so, one simply needs to enter in the open loop parameter variations into the tool and observe the resulting phase noise plot. Figure 31 demonstrates the inclusion of these variations, and Figure 32 shows the resulting phase noise plot. Another item of interest in this analysis is that the rms jitter estimate is now stated in terms of max and min values, as shown in Figure 31.

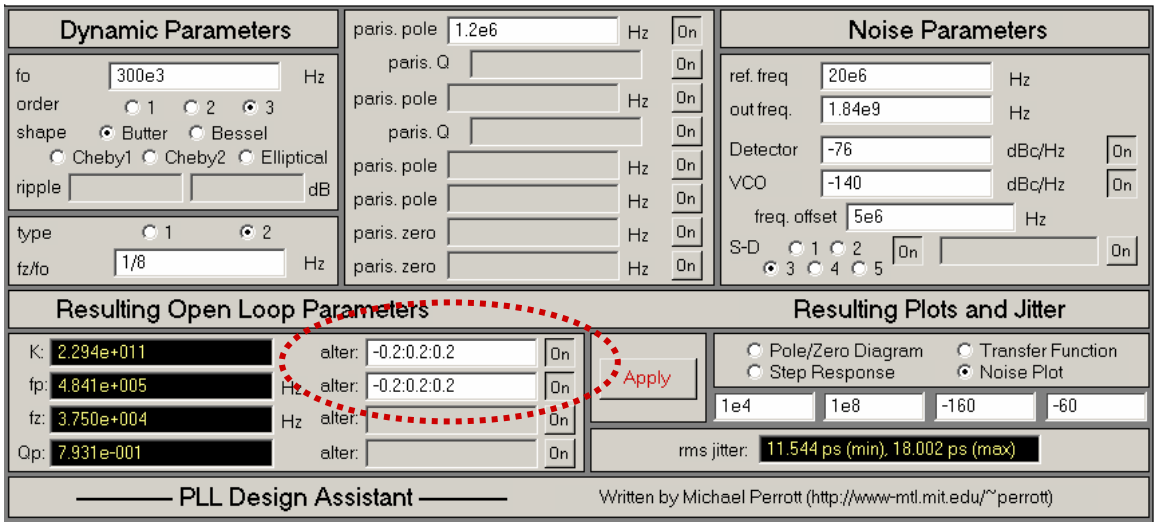

Figure 31. Inclusion of open loop parameter variations in noise analysis.

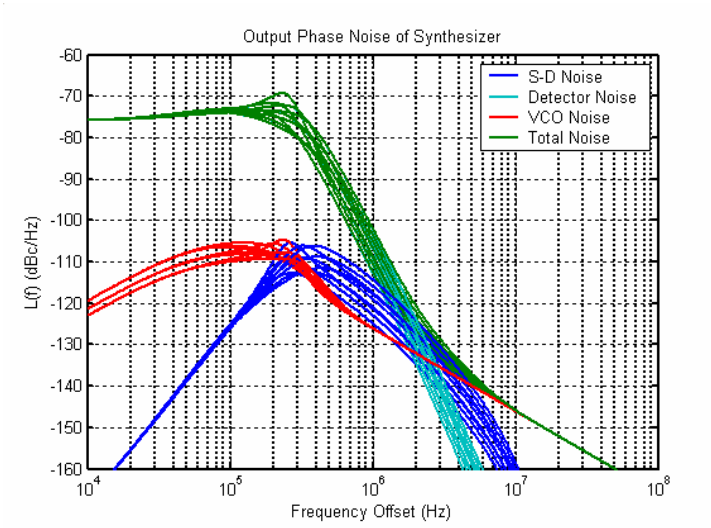

Figure 32. Calculated phase noise over different parameter variations.

E. Description of Σ–Δ Noise using a Noise Transfer Function (NTF)

Rather than simply choosing the order of a MASH Σ–Δ topology to estimate the impact of Σ–Δ quantization noise, we are also free to directly enter the NTF of the modulator. To do so, simply enter in the coefficients of the NTF in Matlab vector form. As an example, the NTF of

$$B(z) = 1 - 3z^{-1} + 3z^{-2} - 1z^{-3}$$

is entered as [1 -3 3 -1], as shown in Figure 33.

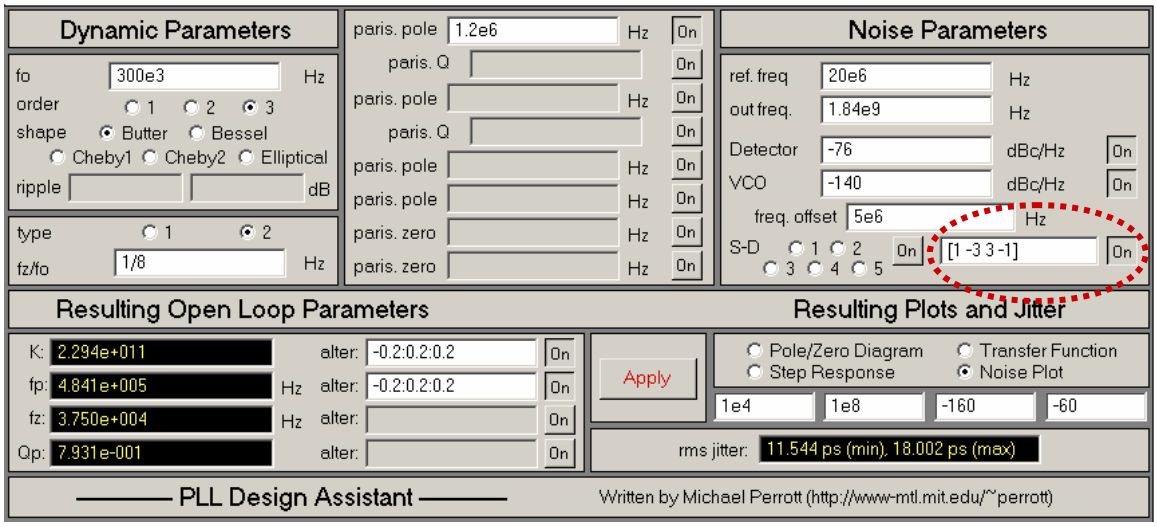

Figure 33. Specifying a Noise Transfer Function (NTF) for the Σ–Δ noise.

Note that the Σ–Δ quantization step size is assumed to be one VCO period, which is the inverse of the VCO frequency as defined according to the **out freq.** parameter in the tool (i.e., 1.84 GHz in Figure 33). One can accommodate different quantization step sizes through appropriate specification of the NTF. For instance, if the step size where chosen to be 2 VCO cycles (i.e. the VCO feeds into a divide-by-2 circuit before going into a divider with variable divide value control), then for the above example the user would enter 2*[1 -3 3 -1] for the NTF. If an architecture were chosen such that the step size was half a VCO period, then 0.5*[1 -3 3 -1] would be entered for the NTF.

To enter in a NTF that contains *both* a numerator and denominator, such as NTF = B(z)/A(z), then simply enter in the numerator in Matlab vector form followed by the denominator in Matlab vector form, and place a comma between the two vectors. As an example, if

$$B(z) = 1 - 3z^{-1} + 3z^{-2} - 1z^{-3}, \quad A(z) = 1 + 0.5z^{-1}$$

then [1 -3 3 -1],[1 0.5] would be entered into the NTF form of the PLL Design Assistant.

F. Supplementary Information: Impact of Charge Pump Current

As described in [3], the model shown in Figure 20 allows us to calculate the output phase noise spectral density as a function of the spectral densities of detector noise, VCO noise, and Σ–Δ quantization noise. In this section, let us focus on the influence of charge pump noise (a component of detector noise) – this exercise will provide intuition on the tradeoffs involved in setting the value of the charge pump current.

Based on Figure 20, it is straightforward to calculate the output phase noise spectral density as a function of the charge pump noise spectral density as:

$$S_{\Phi_{out}}(f) = \left(\frac{1}{I_{cp}}\right)^2\left(\frac{2\pi}{\alpha}N_{nom}\right)^2|G(f)|^2 S_{I_{cpn}}(f) + \text{other noise sources}$$

Given the above expression, one might conclude that the influence of charge pump noise on the output phase noise decreases according to the square of the charge pump current, I_{cp}. However, the spectral density of the charge pump current noise, $S_{Icpn}(f)$, is also influenced by the value of I_{cp}, and must be accounted for in our analysis.

To explore the impact of the charge pump current setting on the charge pump current noise, consider the current mirror circuit shown in Figure 34. As depicted in the figure, the bias and output devices both have associated noise sources that will influence the output noise (in practice, the bias noise can be lowered by placing a large capacitor at the gate of transistors M_1 and M_2). We will consider only thermal noise in the analysis that follows, and will assume that it is described in terms of a current source referenced to the output of its respective device as shown in the figure. Note that thermal noise has a flat spectral density with magnitude equal to its variance.

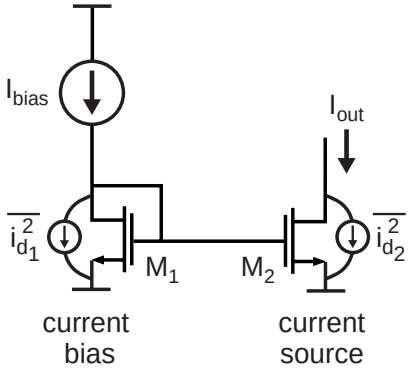

Figure 34. A simple model to gain intuition about charge pump noise.

The overall charge pump noise is composed of the sum of noise sources from individual transistors. The noise from a given MOSFET device has a thermal noise variance that that is proportional to the value of its transconductance, g_m. Combining these facts, the overall charge pump spectral density (assumed constant over frequency) is seen to be proportional to g_m of the current producing devices:

$$S_{I_{cpn}}(f) \propto \overline{i_d^2} = 4kT\gamma g_m$$

Assuming that the MOSFET follows a square law for the relationship between its gate voltage and its output current, it can be shown that

$$g_m \propto \sqrt{I_d W}$$

where I_d and W are the device current and width, respectively. Finally, under the same square law assumption, the minimum voltage from drain to source of the MOSFET device that is required to place the MOSFET device in the saturation region, called the saturation voltage, is related to the device current and width as

$$V_{ds_{sat}} = V_{gs} - V_T \propto \sqrt{I_d / W}$$

The last expression reveals that, in order to keep the saturation voltage at a fixed value, an increase in device current must be accompanied by a corresponding increase in its width.

When designing the charge pump, it is appropriate to examine the impact of different charge pump current values under the assumption that the saturation voltage remains fixed. The reason for doing so is that the saturation voltage is a key parameter in achieving high output impedance for the current carrying device over the desired output voltage range. Based on the above relationships, we therefore make the following argument:

For fixed V_{gs}-V_T

I_d/W constant ⇒ $g_m \propto I_d$

⇒ $S_{I_{cpn}}(f) \propto I_{cp}$

⇒ $S_{\Phi_{out}}(f)\Big|_{S_{I_{cpn}}(f) \text{ component}} \propto \frac{1}{I_{cp}}$

Therefore, we see that the charge pump noise influences the overall output phase noise by a factor that is inversely proportional to the charge pump current setting – higher values of charge pump current lead to lower output phase noise. When designing the PLL, the charge pump current should therefore be set to as large a value as possible while still achieving reasonable power dissipation and reasonable component values in the loop filter. This last point is the critical one for integrated circuit implementations --- typically, the charge pump current is limited by how large of capacitor values can be realized in the loop filter implementation.

Application to Other PLL Circuits

The PLL Design Assistant tool can be applied to a wide variety of PLL systems beyond the Σ–Δ fractional-N synthesizer discussed in the previous sections. In this section we will examine its application to PLL topologies that use a mixer for the phase detector, and to clock and data recovery circuits that use a Hogge phase detector.

A. Mixer-based phase detector PLL

Figure 35 displays a phase-locked loop that uses a mixer as its phase detector, which is often used as an FM demodulator in radio applications. We will assume in our analysis that the input is always a sine wave, but that the output of the PLL can be either a sine or a square wave.

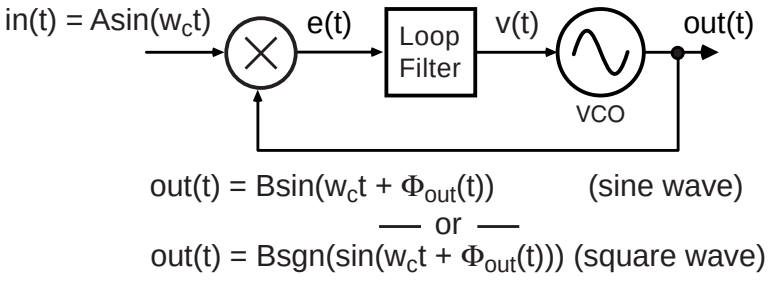

Figure 35. A phase locked loop using a mixer as its phase detector.

Figure 36 shows a linearized model of the system, and highlights the fact that the phase detector gain is a function of the input and output signal shapes and amplitudes. In practice, it is assumed that the amplitudes are held to a controlled value so that a consistent detector gain can be achieved.

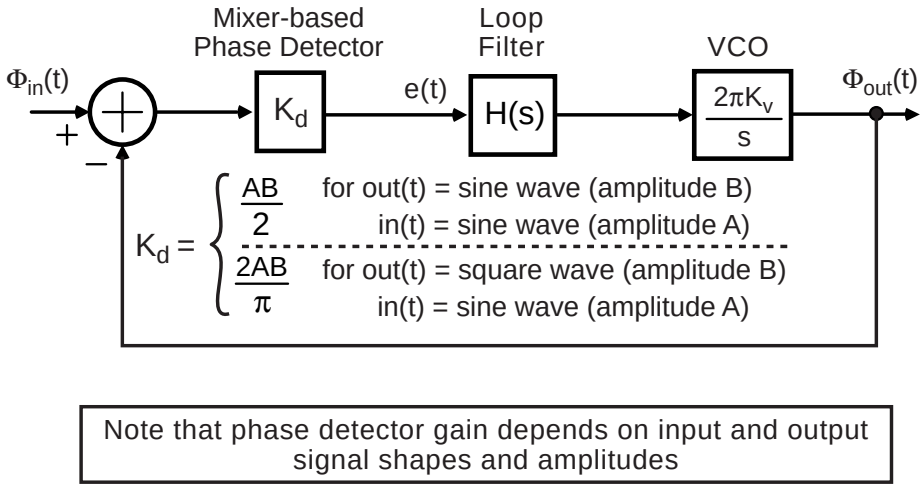

Figure 36. Linearized model of mixer-based PLL.

Table 3 describes the relationships between the open loop and closed loop transfer functions of the mixed-based PLL, which are similar in form to those shown in Table 2 for the Σ–Δ fractional-N synthesizer. Given the similarity, it should not be surprising that the design of the mixer-based PLL follows the same methodology discussed in the previous sections, namely

$$\text{H(s) from Table 2, but now } K_{LP} = K\frac{1}{2\pi K_v K_d}$$

Calculated from software

Thus, the only difference from the previous approach is that the loop filter gain must be set according to a different formula to accommodate the different phase detector and the fact that a charge pump is not present in the system.

Relationship between Closed Loop and Open Loop Transfer Functions

	f-domain	w-domain (w=2πf)	s-domain (s=jw)
Closed Loop	$G(f) = \frac{A(f)}{1 + A(f)}$	$G(w) = \frac{A(w)}{1 + A(w)}$	$G(s) = \frac{A(s)}{1 + A(s)}$
Open Loop	$A(f) = \frac{2\pi K_v K_d H(f)}{2\pi jf}$	$A(w) = \frac{2\pi K_v K_d H(w)}{jw}$	$A(s) = \frac{2\pi K_v K_d H(s)}{s}$

$$K_d = \begin{cases} \frac{AB}{2} & \text{for out(t) = sine wave (amplitude B)} \\ & \text{in(t) = sine wave (amplitude A)} \\ \hline \frac{2AB}{\pi} & \text{for out(t) = square wave (amplitude B)} \\ & \text{in(t) = sine wave (amplitude A)} \end{cases}$$

Table 3. Relationship between G(f) and A(f) for mixer-based PLL.

B. Clock and Data Recovery Circuits

Figure 37 shows a clock and data recovery circuit whose function is to create a properly aligned clock to an incoming data pattern, and retime the input data according to that clock. It accomplishes this task by using phase locked loop principles – a phase error signal, e(t), is produced and then used to properly set the VCO phase and frequency through closed loop feedback.

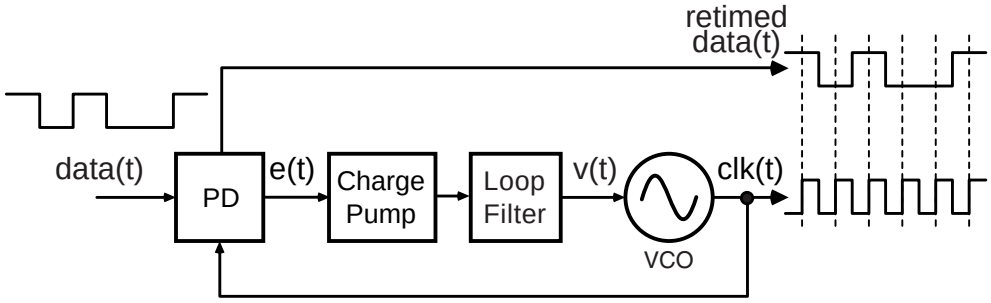

Figure 37. A clock and data recovery (CDR) circuit.

The generation of the phase error signal is generally performed by one of the phase detector topologies shown in Figure 38, which are classified as either linear or bang-bang approaches. Linear detectors, for which the Hogge detector is a common implementation, create a continuous error signal that leads to linear behavior in the tracking characteristics of the PLL. Bang-bang detectors quantize the phase error signal, which leads to nonlinear tracking characteristics.

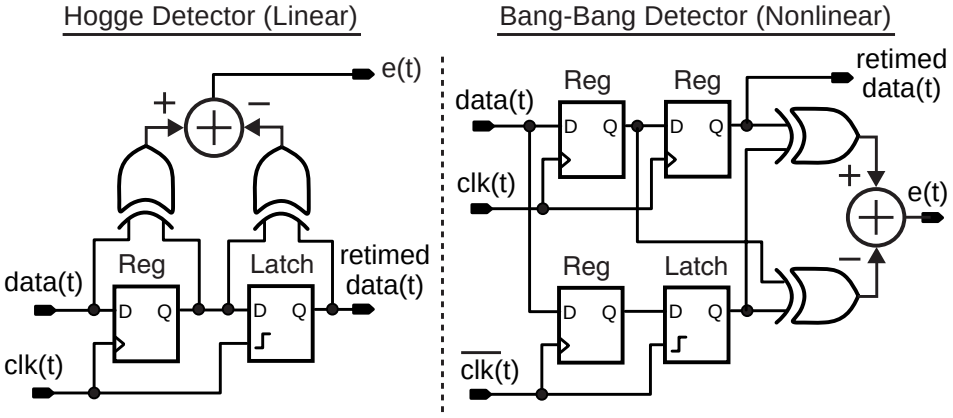

Figure 38. Hogge and bang-bang phase detectors.

The PLL Design Assistant program assumes linear PLL dynamics, and therefore cannot be applied to CDR circuits that use a bang-bang detector. The tool can be applied to systems using a linear detector, however, as will now be discussed. In our analysis, it will be assumed that a Hogge detector is used whose associated signals are shown in Figure 39.

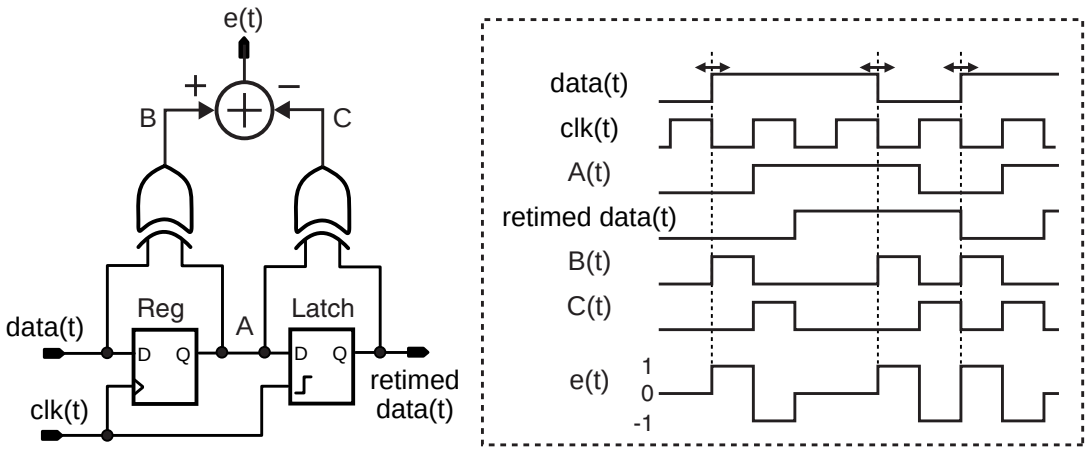

Figure 39. Signals associated with the Hogge detector.

Figure 40 displays a linearized model of a CDR employing a Hogge detector, which reveals that the gain of the phase detector depends on the transition density of the input. Typically, the input follows a PRBS pattern whose average transition density is 1/2.

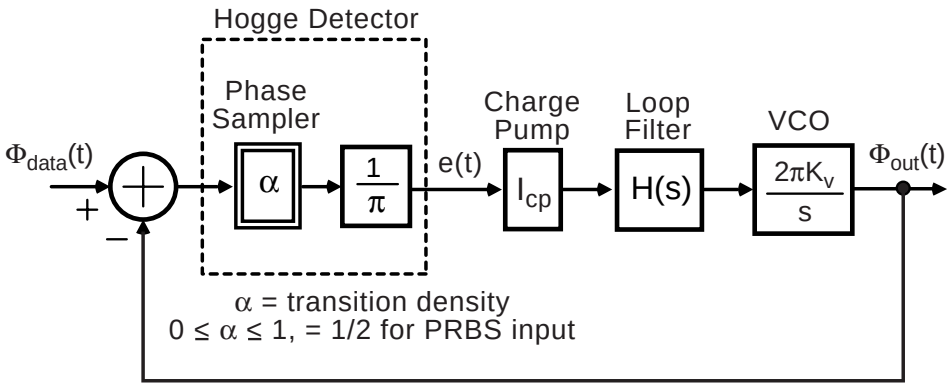

Figure 40. Linearized model of a CDR using a Hogge detector.

Table 4 displays the relationship between G(f) and A(f) given the linearized model in Figure 40. As with the previous PLL circuits, the form of the equations are quite similar, so that the design procedure follows the same methodology as before except that a different loop filter gain is selected. In this case, we have

$$\text{H(s) from Table 2, but now } K_{LP} = K\,\frac{1}{2K_v I_{cp}\alpha}$$

Calculated from software

Relationship between Closed Loop and Open Loop Transfer Functions

	f-domain	w-domain (w=2πf)	s-domain (s=jw)
Closed Loop	$G(f) = \frac{A(f)}{1 + A(f)}$	$G(w) = \frac{A(w)}{1 + A(w)}$	$G(s) = \frac{A(s)}{1 + A(s)}$
Open Loop	$A(f) = \frac{2K_v I_{cp} \alpha H(f)}{2\pi jf}$	$A(w) = \frac{2K_v I_{cp} \alpha H(w)}{jw}$	$A(s) = \frac{2K_v I_{cp} \alpha H(s)}{s}$

$\alpha = 1/2$ for PRBS input

Table 4. Relationship between G(f) and A(f) for CDR using a Hogge detector.

References

[1] T.A. Riley, M.A. Copeland, and T.A. Kwasniewski, “Delta-Sigma Modulation in Fractional-N Frequency Synthesis”, Journal of Solid-State Circuits (JSSC), vol 28, no 5, pp 553-559, May 1993.

[2] M.H. Perrott, T. Tewksbury, and C. Sodini, “A 27 mW CMOS Fractional-N Synthesizer using Digital Compensation for 2.5 Mb/s GFSK Modulation”, JSSC, vol 32, no 12, pp 2048-2060, Dec 1997.

[3] M.H. Perrott, M.D. Trott, C.G. Sodini, “A Modeling Approach for Sigma-Delta Fractional-N Frequency Synthesizers Allowing Straightforward Noise Analysis”, JSSC, vol 38, no 8, pp 1028-1038, Aug 2002.

[4] Thomas H. Lee, “The Design of CMOS Radio-Frequency Integrated Circuits”, Cambridge University Press, 1998

[5] M.H. Perrott, “Fast and Accurate Behavioral Simulation of Fractional-N Synthesizers and other PLL/DLL Circuits”, Design Automation Conference (DAC), 2002, pp 498-503.

[6] S. Willingham, M.H. Perrott, B. Setterberg, A. Grzegorek, W. McFarland, “An Integrated 2.5 GHz Sigma-Delta Frequency Synthesizer with 5 microseconds Settling and 2 Mb/s Closed Loop Modulation”, International Solid-State Circuits Conference (ISSCC), 2000, pp 200-201.

[7] C.Y. Lau, M.H. Perrott, “Phase Locked Loop Design at the Transfer Function Level Based on a Direct Closed Loop Realization Algorithm”, submitted to DAC, 2003.

[8] A. Hill, A. Surber, “The PLL Dead Zone and How to Avoid It”, RF Design, pages 131-134, Mar 1992.

[9] S. Norsworthy, R. Schreier, G. Temes, “Delta-Sigma Data Converters: Theory, Design, and Simulation”, IEEE Press, 1997.