

Modeling and Simulation of Jitter in PLL Frequency Synthesizers

Ken Kundert

Abstract — A methodology is presented for predicting the jitter performance of a Phase-Locked Loop (PLL) using simulation that is both accurate and efficient. The methodology begins by characterizing the noise behavior of the blocks that make up the PLL using transistor-level simulation. For each block, the jitter is extracted and provided as a parameter to behavioral models for inclusion in a high-level simulation of the entire PLL. This approach is efficient enough to be applied to PLLs acting as frequency synthesizers with large divide ratios.

I. INTRODUCTION

Phase-locked loops (PLLs) are used in wireless receivers to implement a variety of functions, such as frequency synthesis, clock recovery, and demodulation. One of the major concerns in the design of PLLs is noise or jitter performance. Jitter from the PLL directly acts to degrade the noise floor and selectivity of a transceiver.

Demir proposed an approach for simulating PLLs whereby a PLL is described using behavioral models simulated at a high level [1,2]. The models are written such that they include jitter in an efficient way. He also devised a powerful new simulation algorithm that is capable of characterizing the circuit-level noise behavior of blocks that make up a PLL that is based on solving a set of nonlinear stochastic differential equations [3, 4]. Finally, he gave formulas that can be used to convert the results of the noise simulations on the individual blocks into values for the jitter parameters for the corresponding behavioral models [5]. This approach provides accurate and efficient prediction of PLL jitter behavior once the noise behavior of the blocks has been characterized. However, it requires the use of an experimental simulator that is not readily available.

This paper presents the relevant ideas of Demir, but while he focussed on presenting the conceptual aspects of modeling and simulating jitter in PLLs, this paper concentrates more on the practical aspects. It presents all the information a designer would need to predict the noise and jitter of

a PLL synthesizer. The jitter extraction methodology is based on the commercially available SpectreRF simulator [17,18] and presents behavioral models for Verilog-A, a standard, non-proprietary analog behavioral modeling language [6, 14]. Both SpectreRF and Verilog-A are options to the Spectre circuit simulator [12], available from Cadence Design Systems.¹

II. FREQUENCY SYNTHESIS

The block diagram of a PLL operating as a frequency synthesizer is shown in Figure 1 [8].² It consists of a reference

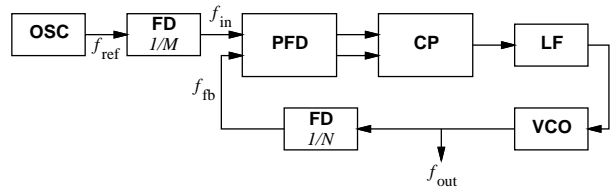


Fig. 1. The block diagram of a frequency synthesizer.

oscillator (OSC), a phase/frequency detector (PFD), a charge pump (CP), a loop filter (LF), a voltage-controlled oscillator (VCO), and two frequency dividers (FDs). The PLL is a feedback loop that, when in lock, forces f_{fb} to be equal to f_{in} . Given a reference frequency f_{ref} , the frequency at the output of the PLL is

$$f_{out} = \frac{N}{M} f_{ref} \cdot \quad (1)$$

By choosing the frequency divide ratios and the reference frequency appropriately, the synthesizer generates an out-

1. SpectreRF is currently the only commercial simulator that is suitable for characterizing the jitter of the blocks that make up a PLL. SPICE and its descendants are not suitable because they only perform noise analysis about a DC operating point and so do not take into account the time-varying nature of these circuits. Harmonic balance simulators do perform noise analysis about a periodic operating point, which is a critical prerequisite, but they have convergence, accuracy, and performance problems with blocks such as the PFD/CP, FD and VCO that are strongly nonlinear.

2. Frequency synthesis is used as an example, but the concepts presented are easily applied to other applications, such as clock recovery and FM demodulation. In addition, they are also applicable to other types of PLL-based synthesis, such as fractional- N synthesis.

put signal at the desired frequency that inherits much of the stability of the reference oscillator. In RF transceivers, this architecture is used to generate the local oscillator (LO) at a programmable frequency, which tunes the transceiver to the desired channel.

A. Phase-Domain Noise Model

If the signals around the loop are interpreted as phase, then the small-signal noise behavior of the loop can be explored by linearizing the components and evaluating the transfer functions. Figure 2 shows this *phase-domain* model.

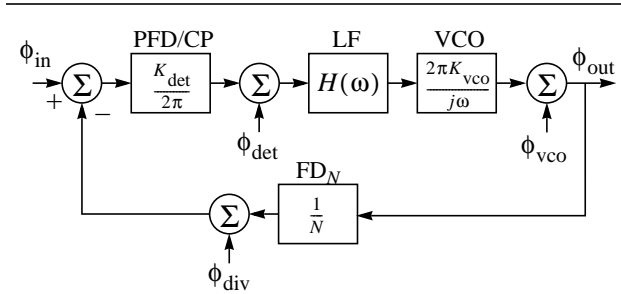


Fig. 2. Linear time-invariant phase-domain model of the synthesizer shown in Figure 1. The out-board frequency divider is removed for simplicity. The ϕ 's represent various sources of noise.

Define

$$T_{\text{fwd}} = \frac{K_{\text{det}}}{2\pi} H(\omega) \frac{2\pi K_{\text{vco}}}{j\omega} = \frac{K_{\text{det}} K_{\text{vco}} H(\omega)}{j\omega} \quad (2)$$

as the forward gain of the loop. Then the transfer function from the various noise sources to the output are

$$T_{\text{in}} = \frac{\phi_{\text{out}}}{\phi_{\text{in}}} = \frac{T_{\text{fwd}}}{1 + T_{\text{fwd}}/N} = \frac{NT_{\text{fwd}}}{N + T_{\text{fwd}}} \quad (3)$$

$$T_{\text{vco}} = \frac{\phi_{\text{out}}}{\phi_{\text{vco}}} = \frac{N}{N + T_{\text{fwd}}} \quad (4)$$

And by inspection,

$$T_{\text{div}} = \frac{\phi_{\text{out}}}{\phi_{\text{div}}} = -T_{\text{in}} \quad (5)$$

and

$$T_{\text{det}} = \frac{\phi_{\text{out}}}{\phi_{\text{det}}} = \frac{2\pi T_{\text{in}}}{K_{\text{det}}} \quad (6)$$

On this last transfer function, we have simply referred ϕ_{det} to the input by dividing through by the gain of the phase detector.

As $\omega \rightarrow \infty$, $T_{\text{fwd}} \rightarrow 0$ because of the VCO and the low-pass filter, and so $T_{\text{in}}, T_{\text{det}}, T_{\text{div}} \rightarrow 0$ and $T_{\text{vco}} \rightarrow 1$. At high frequencies, the noise of the PLL is that of the VCO.

Clearly this must be so because the low-pass LF blocks any feedback at high frequencies.

As $\omega \rightarrow 0$, $T_{\text{fwd}} \rightarrow \infty$ because of the $1/j\omega$ term from the VCO. So at DC, $T_{\text{in}}, T_{\text{div}} \rightarrow N$ and $T_{\text{vco}} \rightarrow 0$. At low frequencies, the noise of the PLL is contributed by the OSC, PFD/CP, FD_M and FD_N , and the noise from the VCO is diminished by the gain of the loop.

Consider further the asymptotic behavior of the loop and the VCO noise at low offset frequencies ($\omega \rightarrow 0$). Oscillator phase noise in the VCO results in the power spectral density $S_{\phi_{\text{vco}}}$ being proportional to $1/\omega^2$, or $S_{\phi_{\text{vco}}} \sim 1/\omega^2$ (neglecting flicker noise). If the LF is chosen such that $H(\omega) \sim 1$, then $T_{\text{fwd}} \sim 1/\omega$, and noise contribution from the VCO to the output, $T_{\text{vco}}^2 S_{\phi_{\text{vco}}}$, is finite and nonzero. If the LF is chosen such that $H(\omega) \sim 1/\omega$, as it typically is when a true charge pump is employed, then $T_{\text{fwd}} \sim 1/\omega^2$ and the noise contribution to the output from the VCO goes to zero at low frequencies.

III. JITTER DEFINITIONS

Jitter is an uncertainty or randomness in the timing of events. In the case of a synthesizer, the events of interest are the transitions in the output signal. One models jitter in a signal by starting with a noise-free signal $v(t)$ and displacing time with a stochastic process $j(t)$. The noisy signal becomes

$$v_n(t) = v(t + j(t)) \quad (7)$$

For simplicity, j is assumed to be a zero-mean Gaussian process, but it may be non-stationary. In addition, v will be assumed to be T -periodic.

There are two types of blocks that are used in the construction of a PLL, driven blocks and autonomous blocks. Each type exhibits a different type of jitter. Driven blocks, such as the PFD, CP, and FD exhibit phase modulation, or *PM jitter*. Autonomous blocks, such as the OSC and VCO, exhibit frequency modulation, or *FM jitter*. Table I previews the basic characteristics of PM and FM jitter, which are discussed more fully after cyclostationary noise is introduced.

TABLE I
CHARACTERISTICS OF PM AND FM JITTER.

Jitter	Type	Circuits	J
PM	synchronous	driven (PFD/CP, FD)	$\frac{\sqrt{\text{var}(n_v, t_c)}}{\dot{v}(t_c)}$
FM	accumulating	autonomous (OSC, VCO)	\sqrt{aT}

A. Cyclostationary Processes

A process is T -cyclostationary if its mean and autocorrelation function (and hence, its variance) is bounded and T -periodic. Cyclostationary processes result from circuits that are driven by a large periodic signal. Such a signal acts to modulate the noise generated by bias dependent noise sources, such as shot noise sources present in forward-biased semiconductor junctions or the thermal noise generated by nonlinear resistors. It also modulates the transfer characteristics of the circuit from the noise source to the output. In a PLL synthesizer in steady-state, the OSC, PFD, CP, VCO, and FD all have a large periodic signal present and so exhibit cyclostationary noise.

Define η_T to be a T -cyclostationary process. If it is sampled every T seconds, the resulting process, $\{\eta(kT)\}$ where $k = 0, 1, 2, 3, \dots$, is stationary.

The variance of flicker noise processes is unbounded and so they are neither stationary nor cyclostationary. Flicker noise is not explicitly considered in this paper. It is not conceptually difficult to include, but it serves to complicate the presentation and flicker noise models tend to be expensive to implement in the time domain [3].

B. PM Jitter

PM jitter is a synchronous jitter exhibited by driven systems. In the PLL, the PFD, CP, and FDs, all exhibit PM jitter. In these components, an output event occurs as a direct result of, and some time after, an input event. PM jitter is a random fluctuation in the delay between the input and the output events. PM jitter is so named because it is a modulation of the phase of the signal by a random process with zero mean and bounded variance. Thus, the frequency of the output signal exactly comensurate with that of the input, but the phase of the output signal fluctuating randomly with respect to that of the input.

PM jitter is modeled using (7), except j is replaced by j_{PM} ,

$$v_n(t) = v(t + j_{\text{PM}}(t)) \quad (8)$$

where $j_{\text{PM}} = \eta_T$ and η_T is T -cyclostationary. If j_{PM} is further restricted to be a white Gaussian T -cyclostationary process, then $v_n(t)$ exhibits *simple PM jitter*. The essential characteristic of simple PM jitter is that the jitter in each event is independent or uncorrelated. Driven circuits exhibit simple PM jitter if they are broadband and if the noise sources are Gaussian and small. The sources are considered small if the circuit responds linearly to the noise, though at the same time the circuit may be responding nonlinearly to the periodic drive signal.

As will be shown in Section IV, if j_{PM} is white and small, then from (29), the variation in v_n is also white. j_{PM} is

small if $|j_{\text{PM}}| \ll T/K$ where T is the period of v and K is the highest significant harmonic of v .

C. FM Jitter

FM jitter is exhibited by autonomous systems, such as oscillators, that generate a stream of spontaneous output transitions. In the PLL, the OSC and VCO exhibit FM jitter. FM jitter is characterized by a randomness in the time since the last output transition, thus the uncertainty of when a transition occurs accumulates with every transition. Thus, compared with a jitter free signal, the frequency of a signal exhibiting FM jitter fluctuates randomly, and the phase drifts without bound in the form of a random walk.

One can construct a signal that exhibits FM jitter from a T -cyclostationary process η_T using

$$j_{\text{FM}}(t) = \int_0^t \eta_T(\tau) d\tau, \quad (9)$$

$$v_n(t) = v(t + j_{\text{FM}}(t)). \quad (10)$$

While η_T is cyclostationary and so has bounded variance, from (9) it is clear that the variance of j_{FM} , and hence the phase difference between $v(t)$ and $v_n(t)$, can be unbounded.

If η_T is a white Gaussian T -cyclostationary process, then $v_n(t)$ exhibits *simple FM jitter*. In this case, the process $\{j_{\text{FM}}(kT)\}$ that results from sampling j_{FM} every T seconds is a discrete Wiener process and the phase difference between $v(kT)$ and $v_n(kT)$ is a random walk [9].

The essential characteristic of simple FM jitter is that the incremental jitter that accumulates over each cycle is independent or uncorrelated. Autonomous circuits exhibit simple FM jitter if they are broadband and if the noise sources are Gaussian and small. The sources are considered small if the circuit responds linearly to the noise, though at the same time the circuit may be responding nonlinearly to the oscillation signal. An autonomous circuit is considered broadband if there are no secondary resonant responses close in frequency to the primary resonance.

D. Metrics for Jitter

Define t_i be the sequence of times at which positive going zero crossings, henceforth referred to as *transitions*, occur in $v_n(t)$. Define $T_k = t_{k+1} - t_k$ and $J(k)$ to be the standard deviation of T_k ,

$$J(k) = \sigma(T_k) = \sqrt{\text{var}(T_k)}. \quad (11)$$

$J(k)$ is referred to as *period jitter*. It is a measure of jitter over a single cycle and has units of time. This definition is valid for both PM and FM jitter, but does not distinguish between them.

A metric that does distinguish between PM and FM jitter is the *long-term jitter* $J_i(k)$, the standard deviation of the length of i adjacent periods from reference point k ,

$$J_i(k) = \sigma(t_{i+k} - t_k) = \sqrt{\text{var}(t_{i+k} - t_k)}. \quad (12)$$

The period jitter $J(k)$ is related to the long-term jitter $J_i(k)$ with $J(k) = J_1(k)$. In steady state, the jitter is independent of k , and so the period and long-term jitter are denoted J and J_i .

To fully characterize an arbitrary jitter process, J_i must be known for all i . However, for simple PM and FM jitter one can determine J_i for all i simply by knowing J .

For simple PM jitter,

$$J_i(k) = \sqrt{\text{var}(t_{i+k} - t_k)}, \quad (13)$$

$$J_i(k) = \sqrt{\text{var}((i+k)T + j_{\text{PM}}(t_{i+k}) - (iT + j_{\text{PM}}(t_k)))} \quad (14)$$

$$J_i = \sqrt{2\text{var}(j_{\text{PM}})}, \quad (15)$$

where $j_{\text{PM}}(t)$ is T -cyclostationary and so $j_{\text{PM}} = j_{\text{PM}}(t_k)$ is independent of k . The factor of 2 stems from the length of an interval including the independent variation from two transitions. From (15), J_i is independent of i , and so

$$J_i = J \text{ for } i = 1, 2, \dots \quad (16)$$

For simple FM jitter, each transition is relative to the previous transition, and the variation in the length of each period is independent, so the variance in the time of each transition accumulates,

$$J_i = \sqrt{i}J \text{ for } i = 0, 1, 2, \dots, \quad (17)$$

$$J = \sigma(j_{\text{FM}}(t_k)) = \sqrt{\text{var}(j_{\text{FM}}(t_k))}. \quad (18)$$

The jitter behavior for PLLs is a combination of simple PM and simple FM jitter [13]. If the PLL has a closed-loop bandwidth of f_L , and $\tau_L = 1/2\pi f_L$, then for i such that $iT \ll \tau_L$, jitter from the VCO dominates and the PLL exhibits simple FM jitter. Similarly, if the reference oscillator exhibits simple FM jitter, then for large i (low frequencies), FM jitter again dominates. Between these two extremes, the PLL exhibits PM jitter. The amount of which depends on the characteristics of the loop and the level of PM jitter exhibited by the FDs and the PFD/CP. The behavior of a typical PLL is shown in Figure 3.

IV. PHASE NOISE

Recall that v is a T -periodic function and reformulate j as phase, then (7) is rewritten as

$$v_n(t) = v\left(t + \frac{\phi(t)}{2\pi f_c}\right) \quad (19)$$

where $\phi(t) = 2\pi f_c j(t)$ is in radians and $f_c = 1/T$. There are several common ways of describing the noise in this type

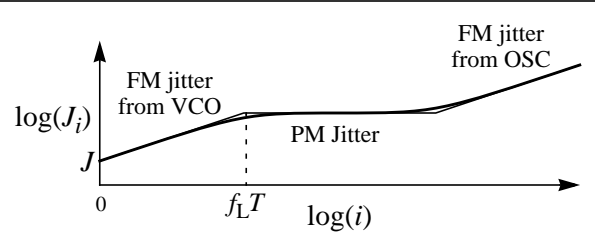


Fig. 3. Long-term jitter (J_i) for a PLL as a function of the number of cycles.

of signal. One common way is $S_\phi(f)$, the power spectral density of ϕ . Related, but much less common, is

$$S_\omega(f) = (2\pi f)^2 S_\phi(f), \quad (20)$$

the power spectral density of $\omega = d\phi/dt$. $S_\phi(f)$ and $S_\omega(f)$ are conceptually useful, but are not directly measurable because both ϕ and ω are not observable quantities. The other common measure is $S_v(f)$, the power spectral density of v , which is measurable with a spectrum analyzer. Often, especially when measuring oscillator phase noise, $S_v(f)$ close to f_c is given as a function of offset frequency and normalized relative to the power in the first harmonic of v , V_1 (defined below in (25)),

$$L(f_m) = \frac{S_v(f_c + f_m)}{2|V_1|^2}, \quad (21)$$

where $f_m = f - f_c$.

Relating $S_\phi(f)$ and $L(f_m)$: Assume that $\phi(t)$ is small and expand (19) using the Taylor series,

$$v_n(t) = v(t) + \frac{dv(t)}{dt} \frac{\phi(t)}{2\pi f_c}. \quad (22)$$

Model the noise due to jitter using additive noise,

$$v_n(t) = v(t) + n_v(t) \quad (23)$$

$$n_v(t) = \frac{dv(t)}{dt} \frac{\phi(t)}{2\pi f_c}. \quad (24)$$

Since $v(t)$ is periodic, it can be written as a Fourier series,

$$v(t) = \sum_{k=-\infty}^{\infty} V_k e^{j2\pi k f_c t} \quad (25)$$

where $j = \sqrt{-1}$. Differentiating (25) and substituting into (24),

$$n_v(t) = \frac{\phi(t)}{2\pi f_c} \sum_{k=-\infty}^{\infty} j2\pi k f_c V_k e^{j2\pi k f_c t} \quad (26)$$

$$n_v(t) = \phi(t) \sum_{k=-\infty}^{\infty} jkV_k e^{2\pi k f_c t}. \quad (27)$$

Computing the power spectral density of n_v gives

$$S_{n_v}(f) = \sum_{k=-\infty}^{\infty} |kV_k|^2 S_{\phi}(f - kf_c) \quad (28)$$

$$S_{n_v}(f_m) = \frac{S_{n_v}(f_c + f_m)}{2|V_1|^2} = \frac{1}{2} \sum_{k=-\infty}^{\infty} \left| \frac{kV_k}{V_1} \right|^2 S_{\phi}(f_m - (k-1)f_c) \quad (29)$$

Assume that $S_{\phi}(f)$ drops rapidly with increasing frequency, then for $f_m \ll f_c$

$$L(f_m) \cong \frac{1}{2} S_{\phi}(f_m). \quad (30)$$

Recall that several assumptions were made in developing this approximation. In particular, that $\phi(t)$ is small and that S_{ϕ} at low frequencies is much larger than at frequencies near f_c and higher. The first approximation is not valid for phase noise of a free running oscillator for small f_m because $\phi(t)$ is undergoing a random walk, and so is unbounded.

V. SOURCES OF JITTER

A. Thresholds

In systems where signals are continuous valued, an event is usually defined as a signal crossing a threshold in a particular direction. The threshold crossings of a noiseless periodic signal, $v(t)$, are precisely evenly spaced. However, when noise is added to the signal, $v_n(t) = v(t) + n_v(t)$, each threshold crossing is displaced slightly. Thus, a threshold converts additive noise to PM jitter.

The amount of displacement in time is determined by the amplitude of the noise signal, $n_v(t)$, and the slew rate of the periodic signal, $dv(t_c)/dt$, as the threshold is crossed. If the noise n_v is stationary, then

$$\text{var}(j_{\text{PM}}(t_c)) \cong \frac{\text{var}(n_v)}{\dot{v}(t_c)^2} \quad (31)$$

where t_c is the time of a threshold crossing.

Generally n_v is not stationary, but cyclostationary. It is only important to know when the noisy periodic signal $v_n(t)$ crosses the threshold, so the statistics of n_v are only significant at the time when $v_n(t)$ crosses the threshold,

$$\text{var}(j_{\text{PM}}(t_c)) = \frac{\text{var}(n_v, t_c)}{\dot{v}(t_c)^2}. \quad (32)$$

The jitter is computed from (31) or (32) using (16),

$$J = \sqrt{2\text{var}(j_{\text{PM}}(t_c))}. \quad (33)$$

B. Oscillator Phase Noise

FM jitter is strongly related to oscillator phase noise. Both are different ways of describing the same underlying phenomenon. In other words, all free running oscillators exhibit a behavior that is generically referred to as oscillator phase noise. Any noise in an autonomous system will cause the phase to drift freely because there is no reference signal with which to lock. When the phase fluctuations are measured in terms of time deviation over a period, it is referred to as FM jitter. If it is measured in terms of noise signal amplitude as a function of frequency, it is referred to as oscillator phase noise.

Relating J and S_{ϕ} : In order to determine the period jitter J of $v_n(t)$ for a noisy oscillator, assume that it exhibits simple FM jitter so that η in (9) is a white Gaussian noise process (this excludes flicker noise) with a PSD of

$$S_{\eta}(f) = a, \quad (34)$$

and an autocorrelation function of

$$R_{\eta}(t_1, t_2) = a\delta(t_1 - t_2), \quad (35)$$

where δ is a Dirac delta function. Then

$$j_{\text{FM}}(t) = \int_0^t \eta_T(\tau) d\tau \quad (36)$$

is a Wiener process [9], which has an autocorrelation function of

$$R_{j_{\text{FM}}}(t_1, t_2) = a \min(t_1, t_2). \quad (37)$$

The variance of j_{FM} over one period T is

$$\begin{aligned} \text{var}(j_{\text{FM}}(t+T) - j_{\text{FM}}(t)) &= \text{E}[(j_{\text{FM}}(t+T) - j_{\text{FM}}(t))^2] \\ &= \text{E}[j_{\text{FM}}(t+T)^2 - 2j_{\text{FM}}(t+T)j_{\text{FM}}(t) + j_{\text{FM}}(t)^2] \\ &= \text{E}[j_{\text{FM}}(t+T)^2] - 2\text{E}[j_{\text{FM}}(t+T)j_{\text{FM}}(t)] + \text{E}[j_{\text{FM}}(t)^2] \\ &= R_{j_{\text{FM}}}(t+T, t+T) - 2R_{j_{\text{FM}}}(t+T, t) + R_{j_{\text{FM}}}(t, t) \\ &= a(t+T) - 2at + at \\ &= aT \end{aligned} \quad (38)$$

Finally, jitter is the standard deviation of the variation in the period, and so

$$J = \sqrt{aT}. \quad (39)$$

We now have a way of relating the jitter of the oscillator to the PSD of η . However, what is needed is to relate the jitter to either S_{ϕ} or L . To do so, consider simple FM jitter written in terms of phase,

$$\phi_{\text{FM}}(t) = 2\pi f_c j_{\text{FM}}(t) = 2\pi f_c \int_0^t \eta_T(\tau) d\tau \quad (40)$$

and so from (34) and (40) the PSD of $\phi_{\text{FM}}(t)$ is

$$S_{\phi_{\text{FM}}}(f) = a \frac{(2\pi f_c)^2}{(2\pi f)^2} = \frac{a f_c^2}{f^2}. \quad (41)$$

Thus, a is the noise power in ϕ_{FM} where $f = f_c$, or

$$a = S_{\phi_{\text{FM}}}(f_c). \quad (42)$$

Given S_{ϕ} and f_c , a is found with (41)³ and then J is found with (39).

Relating $S_{\phi}(f)$ and $L(f_m)$: The phase variation in a simple FM jitter process is not bounded, and so (29) cannot be used to predict $L(f_m)$ of an oscillator. Demir shows that for a free-running oscillator exhibiting simple FM jitter

$$L(f_m) = \frac{1}{2} \frac{a f_c^2}{a^2 \pi^2 f_c^4 + f_m^2}, \quad (43)$$

which is a Lorentzian with corner frequency of

$$f_{\text{corner}} = \pi \frac{J^2}{T^2} f_c \ll f_c, \quad (44)$$

At frequencies above the corner,

$$L(f_m) = \frac{a f_c^2}{2 f_m^2} = \frac{1}{2} S_{\phi_{\text{FM}}}(f_m), \quad (45)$$

which agrees with (30) and Vendelin [19]. Thus, in free-running oscillators at frequencies above the corner, $L(f_m)$ and $S_{\phi}(f_m)$ are easily related.⁴

VI. MODEL OF PLL

The basic behavioral models for the blocks that make up a PLL are well known and so will not be discussed here in any depth [1,2]. Instead, only the techniques for adding jitter to the models are discussed.

Jitter is modeled in an AHDL by dithering the time at which events occur. This is efficient because it does not create any additional activity, rather it simply changes the time when existing activity occurs. Thus, models with jitter can run as efficiently as those without.

3. In general, (42) is not used when computing a because S_{ϕ} is not accurately known at f_c . Instead, f is chosen where S_{ϕ} is accurately known, and (41) is used.

4. When using (45) and (41) to determine a , choose f well above the corner frequency of (44) to avoid ambiguity and well below f_c to avoid the noise from other sources that occur at these frequencies.

A. Modeling PM Jitter

A feature of Verilog-A allows especially simple modeling of PM jitter. The *transition()* function, which is used to model signal transitions between discrete levels, provides a delay argument that can be dithered on every transition. The delay argument must not be negative, so a fixed delay that is greater than the maximum expected deviation of the jitter must be included. This approach is suitable for any model that exhibits PM jitter and generates discrete-valued outputs. It is used in the Verilog-A divider module shown in Listing I, which models PM jitter with (8) where j_{PM} is

LISTING I

FREQUENCY DIVIDER THAT MODELS PM JITTER.

```
// Frequency Divider with Jitter
#include "discipline.h"
#include "constants.h"
module divider (out, in);
input in; output out; electrical in, out;
parameter real Vlo=-1, Vhi=1;
parameter integer ratio=2 from [2:inf);
parameter integer dir=1 from [-1:1] exclude 0;
// dir=1 for positive edge trigger
// dir=-1 for negative edge trigger
parameter real tt=1n from (0:inf);
parameter real td=0 from (0:inf);
parameter real jitter=0 from [0:td/5);
parameter real ttol=1p from (0:td/5);
// recommend ttol << jitter
integer count, n, seed;
real dt;
analog begin
  @(initial_step) seed = -311;
  @(cross(V(in) - (Vhi + Vlo)/2, dir, ttol)) begin
    // count input transitions
    count = count + 1;
    if (count >= ratio)
      count = 0;
    n = (2*count >= ratio);
    // add jitter
    dt = 0.707*jitter*$dist_normal(seed,0,1);
  end
  V(out) <+ transition(n ? Vhi : Vlo, td+dt,tt);
end
endmodule
```

a stationary white discrete-time Gaussian random process. It is also used in Listing II, which models a simple PFD/CP.

LISTING II
PFD/CP MODEL WITH PM JITTER.

```
// Phase-Frequency Detector & Charge Pump
`include "discipline.h"
`include "constants.h"

module pfd_cp (out, ref, vco);
input ref, vco; output out; electrical ref, vco, out;

parameter real lout=100u;
parameter integer dir=1 from [-1:1] exclude 0;
    // dir=1 for positive edge trigger
    // dir=-1 for negative edge trigger
parameter real tt=1n from (0:inf);
parameter real td=0 from (0:inf);
parameter real jitter=0 from [0:td/5];
parameter real ttol=1p from (0:td/5);
    // recommend ttol << jitter

integer state, seed;
real dt;

analog begin
    @(initial_step) seed = 716;

    @(cross(V(ref), dir, ttol)) begin
        if (state > -1) state = state - 1;
        dt = 0.707*jitter*$dist_normal(seed,0,1);
    end

    @(cross(V(vco), dir, ttol)) begin
        if (state < 1) state = state + 1;
        dt = 0.707*jitter*$dist_normal(seed,0,1);
    end

    l(out) <+ transition(lout*state, td + dt, tt);
end
endmodule
```

Frequency Divider Model: The model, given in Listing I, operates by counting input transitions. This is done in the @cross block. The cross function triggers the @ block at the precise moment when its first argument crosses zero in the direction specified by the second argument. Thus, the @ block is triggered when the input crosses the threshold in the user specified direction. The body of the @ block increments the count, resets it to zero when it reaches ratio, then determines if count is above or below its midpoint (n is zero if the count is below the midpoint). It also generates a new random dither dT that is used later. Outside the @ block is code that executes continuously. It processes n to create the output. The value of the ?: operator is Vhi if n is 1 and Vlo if n is 0. Finally, the transition function adds a finite transition time of tt and a delay of td + dt. The finite transition time removes the discontinuities from the signal that could cause problems for the simulator. The jitter is embodied in dt, which varies randomly from transition to

transition. The jitter J represents the variation in a period, and dt the variation of a single transition, so from (16)

$$dt = \sigma(j_{PM}(t_c)) = \sqrt{2}J \cong 0.707J, \quad (46)$$

which compensates for the fact that both ends of each interval are varying. To avoid nonnegative delays, td must always be larger than dt.

PFD/CP Model: The model for a phase/frequency detector combined with a charge pump is given in Listing II. It implements a finite-state machine with a three-level output, -1, 0 and +1. On every transition of the VCO input in direction dir, the output is incremented. On every transition of the reference input in the direction dir, the output is decremented. If both the VCO and reference inputs are at the same frequency, then the average value of the output is proportional to the phase difference between the two, with the average being negative if the reference transition leads the VCO transition and positive otherwise [8]. As before, the time of the output transitions are randomly dithered by dt to model jitter. The output is modeled as an ideal current source and a finite transition time models the dead band in the CP.

B. Modeling FM jitter

OSC Model: The delay argument of the *transition()* function cannot be used to model FM jitter because of the cumulative nature of this type of jitter. When modeling a fixed frequency oscillator, the *timer()* function is used as shown in Listing III. At every output transition, the next transition is scheduled using the *timer()* function to be $T/K + J\delta/\sqrt{K}$ in the future, where δ is a unit-variance zero-mean random process and K is the number of output transitions per period. Typically, $K = 2$.

VCO Model: A VCO generates a sine or square wave whose frequency is proportional to the input signal level. VCO models, given in Listings IV and V, are constructed using three serial operations, as shown in Figure 4. First, the input signal is scaled to compute the desired output frequency. Then, the frequency is integrated to compute the output phase. Finally, the phase is used to generate the desired output signal. The phase is computed with *idmod*, a function that provides integration followed by a modulus operation. This serves to keep the phase bounded, which prevents a loss of numerical precision that would otherwise occur when the phase became large after a long period of time. Output transitions are generated when the phase passes $-\pi/2$ and $\pi/2$.

The jitter is modeled as a random variation in the frequency of the VCO. However, the jitter is specified as a variation in the period, thus it is necessary to relate the

LISTING III
FIXED FREQUENCY OSCILLATOR WITH FM JITTER.

```

// Fixed-Frequency Oscillator with Jitter
#include "discipline.h"
#include "constants.h"

module osc (out);
output out; electrical out;

parameter real freq=1 from (0:inf);
parameter real Vlo=-1, Vhi=1;
parameter real tt=0.01/freq from (0:inf);
parameter real jitter=0 from [0:0.1/freq];

integer n, seed;
real next, dT;

analog begin
  @(initial_step) begin
    seed = 286;
    next = 0.5/freq + $realtime;
  end

  @(timer(next)) begin
    n = !n;
    dT = jitter*$dist_normal(seed,0,1);
    next = next + 0.5/freq + 0.707*dT;
  end

  V(out) <+ transition(n ? Vhi : Vlo, 0, tt);
end
endmodule

```

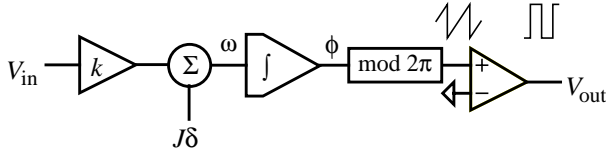


Fig. 4. Block diagram of VCO behavioral model that includes jitter.

variation in the period to the variation in the frequency. Assume that without jitter, the period is divided into K equal intervals of duration $\tau = T/K = 1/Kf_c$. The frequency deviation will be updated every interval and held constant during the intervals. With jitter, the duration of an interval is

$$\tau_i = \tau + \Delta\tau_i. \quad (47)$$

$\Delta\tau$ is a random variable with variance

$$\text{var}(\Delta\tau) = \frac{\text{var}(T)}{K} = \frac{J^2}{K}. \quad (48)$$

Therefore,

$$\Delta\tau_i = \frac{J\delta_i}{\sqrt{K}} \quad (49)$$

where δ is a zero-mean unit-variance Gaussian random process. The dithered frequency is

$$f_i = \frac{1}{K} \left(\frac{1}{\tau + \Delta\tau_i} \right) = \frac{\frac{1}{K\tau}}{1 + \frac{\Delta\tau_i}{\tau}} = \frac{f_c}{1 + K\Delta\tau_i f_c} \quad (50)$$

Let $\Delta T_i = K\Delta\tau_i$, then

$$f_i = \frac{f_c}{1 + \Delta T_i f_c}. \quad (51)$$

Finally, $\text{var}(\tau_i) = J^2/K$, and so $\Delta\tau_i = J\delta_i/\sqrt{K}$ and $\Delta T_i = \sqrt{K}J\delta_i$.

The @cross statement is used to determine the exact time when the phase crosses the thresholds, indicating the beginning of a new interval. At this point, a new random trial δ_i is generated.

The final model given in Listing IV. This model can be easily modified to fit other needs. Converting it to a model that generates sine waves rather than square waves simply requires replacing the last two lines with one that computes and outputs the sine of the phase. When doing so, consider reducing the number of jitter updates to one per period, in which case the factor of 1.414 should be changed to 1.

Listing V is a Verilog-A model for a quadrature VCO that exhibits FM jitter. It is an example of how to model an oscillator with multiple outputs so that the jitter on the outputs is properly correlated.

These models do not include the finite response time of a real VCO. Those dynamics would be separated out and included as part of the model for the LF.

C. Efficiency of the Models

Conceptually, a model that includes jitter should be just as efficient as one that does not because jitter does not increase the activity of the models, it only affects the timing of particular events. However, if jitter causes two events that would normally occur at the same time to be displaced so that they are no longer coincident, then a circuit simulator will have to use more time points to resolve the distinct events and so will run more slowly. For this reason, it is desirable to combine jitter sources to the degree possible.

To make the HDL models even faster, rewrite them in either Verilog-HDL or VHDL. Be sure to set the time resolution to be sufficiently small to prevent the discrete nature of time in these simulators from adding an appreciable amount of jitter.

LISTING IV
VCO MODEL THAT INCLUDES FM JITTER.

```
// Voltage Controlled Oscillator with Jitter
#include "discipline.h"
#include "constants.h"
module vco (out, in);
input in; output out; electrical out, in;
parameter real Vmin=0;
parameter real Vmax=Vmin+1 from (Vmin:inf);
parameter real Fmin=1 from (0:inf);
parameter real Fmax=2*Fmin from (Fmin:inf);
parameter real Vlo=-1, Vhi=1;
parameter real tt=0.01/Fmax from (0:inf);
parameter real jitter=0 from [0:0.25/Fmax];
parameter real ttol=1u/Fmax from (0:1/Fmax);
real freq, phase, dT;
integer n, seed;
analog begin
    @(initial_step) seed = -561;
    // compute the freq from the input voltage
    freq = (V(in) - Vmin)*(Fmax - Fmin)
        / (Vmax - Vmin) + Fmin;
    // bound the frequency (this is optional)
    if (freq > Fmax) freq = Fmax;
    if (freq < Fmin) freq = Fmin;
    // add the phase noise
    freq = freq/(1 + dT*freq);
    // phase is the integral of the freq modulo 2π
    phase = 2*M_PI*idtmmod(freq, 0.0, 1.0, -0.5);
    // update jitter twice per period
    // 1.414=sqrt(K), K=2 jitter updates/period
    @(cross(phase + 'M_PI/2, +1, ttol)) begin
        cross(phase - 'M_PI/2, +1, ttol) begin
            dT = 1.414*jitter*$dist_normal(seed,0, 1);
            n = (phase >= -'M_PI/2) && (phase < 'M_PI/2);
        end
    end
    // generate the output
    V(out) <+ transition(n ? Vhi : Vlo, 0, tt);
end
endmodule
```

Including PM Jitter into OSC: From the discussion of the phase-domain model of the synthesizer, it is clear that one can easily combine the output-referred noise of FD_M and FD_N and the input-referred noise of the PFD/CP with the output noise of OSC. A modified fixed-frequency oscillator model that supports two jitter parameters and the divide ratio M is given in Listing VI (more on the effect of the divide ratio on jitter in the next section). The `fmJitter` parameter is used to model the FM jitter of the reference

LISTING V
QUADRATURE DIFFERENTIAL VCO MODEL THAT INCLUDES FM JITTER.

```
// Quadrature Differential VCO with Jitter
#include "discipline.h"
#include "constants.h"
module quadVco (Plout,Nlout, PQout,NQout, Pin,Nin);
electrical Plout, Nlout, PQout, NQout, Pin, Nin;
output Plout, Nlout, PQout, NQout;
input Pin, Nin;
parameter real Vmin=0;
parameter real Vmax=Vmin+1 from (Vmin:inf);
parameter real Fmin=1 from (0:inf);
parameter real Fmax=2*Fmin from (Fmin:inf);
parameter real Vlo=-1, Vhi=1;
parameter real jitter=0 from [0:0.25/Fmax];
parameter real ttol=1u/Fmax from (0:1/Fmax);
parameter real tt=0.01/Fmax;
real freq, phase, dT;
integer i, q, seed;
analog begin
    @(initial_step) seed = 133;
    // compute the freq from the input voltage
    freq = (V(Pin,Nin) - Vmin) * (Fmax - Fmin)
        / (Vmax - Vmin) + Fmin;
    // bound the frequency (this is optional)
    if (freq > Fmax) freq = Fmax;
    if (freq < Fmin) freq = Fmin;
    // add the phase noise
    freq = freq/(1 + dT*freq);
    // phase is the integral of the freq modulo 2π
    phase = 2*M_PI*idtmmod(freq, 0.0, 1.0, -0.5);
    // update jitter where phase crosses π/2
    // 2=sqrt(K), K=4 jitter updates per period
    @(cross(phase - 3*M_PI/4, +1, ttol) or
        cross(phase - 'M_PI/4, +1, ttol) or
        cross(phase + 'M_PI/4, +1, ttol) or
        cross(phase + 3*M_PI/4, +1, ttol)) begin
        dT = 2*jitter*$dist_normal(seed,0,1);
        i = (phase >= -3*M_PI/4) && (phase < 'M_PI/4);
        q = (phase >= -'M_PI/4) && (phase < 3*M_PI/4);
    end
    // generate the I and Q outputs
    V(Plout) <+ transition(i ? Vhi : Vlo, 0, tt);
    V(Nlout) <+ transition(i ? Vlo : Vhi, 0, tt);
    V(PQout) <+ transition(q ? Vhi : Vlo, 0, tt);
    V(NQout) <+ transition(q ? Vlo : Vhi, 0, tt);
end
endmodule
```

oscillator, and the `pmJitter` parameter is used to model the PM jitter of FD_M , FD_N and PFD/CP. PM jitter is modeled

LISTING VI
FIXED-FREQUENCY OSCILLATOR WITH FM AND PM JITTER.

```
// Fixed-Frequency Oscillator with Jitter
`include "discipline.h"
`include "constants.h"

module osc (out);
output out; electrical out;

parameter real freq=1 from (0:inf);
parameter real ratio=1 from (0:inf);
parameter real Vlo=-1, Vhi=1;
parameter real tt=0.01*ratio/freq from (0:inf);
parameter real fmJitter=0 from [0:0.1/freq];
parameter real pmJitter=0 from [0:0.1*ratio/freq];

integer n, fmSeed, pmSeed;
real next, dT, dt, fmSD, pmSD;

analog begin
    @(initial_step) begin
        fmSeed = 286;
        pmSeed = -459;
        fmSD = fmJitter*sqrt(ratio/2);
        pmSD = pmJitter*sqrt(0.5);
        next = 0.5/freq + $realtime;
    end

    @(timer(next + dt)) begin
        n = !n;
        dT = fmSD*$dist_normal(fmSeed,0,1);
        dt = pmSD*$dist_normal(pmSeed,0,1);
        next = next + 0.5*ratio/freq + dT;
    end

    V(out) <+ transition(n ? Vhi : Vlo, 0, tt);
end
endmodule
```

in the oscillator without using a nonzero delay in the transition function. This is a more efficient approach because it avoids generating two unnecessary events per period. To get full benefit from this optimization, a modified PFD/CP given in Listing VII is used. This model runs more efficiently by removing support for jitter and the td parameter.

Merging the VCO and FD_N : If the output of the VCO is not used to drive circuitry external to the synthesizer, if the divider exhibits simple PM jitter, and if the VCO exhibits simple FM jitter, then it is possible to include the frequency division aspect of the FD_N as part of the VCO by simply adjusting the VCO gain and jitter. If the divide ratio of FD_N is large, the simulation runs much faster because the high VCO output frequency is never generated. The Verilog-A model for the merged VCO and FD_N is given in Listing VIII. It also includes code for generating a logfile containing the length of each period. The log-

LISTING VII
PFD/CP WITHOUT JITTER.

```
// Phase-Frequency Detector & Charge Pump
`include "discipline.h"
`include "constants.h"

module pfd_cp (out, ref, vco);
input ref, vco; output out; electrical ref, vco, out;

parameter real lout=100u;
parameter integer dir=1 from [-1:1] exclude 0;
    // dir = 1 for positive edge trigger
    // dir = -1 for negative edge trigger
parameter real tt=1n from (0:inf);
parameter real ttol=1p from (0:inf);

integer state;

analog begin
    @(cross(V(ref), dir, ttol)) begin
        if (state > -1) state = state - 1;
    end
    @(cross(V(vco), dir, ttol)) begin
        if (state < 1) state = state + 1;
    end

    I(out) <+ transition(lout * state, 0, tt);
end
endmodule
```

file is used in Section VIII when determining S_{VCO} , the power spectral density of the phase of the VCO output.

Recall that the PM jitter of FD_M and FD_N has already been included as part of OSC, so the divider model incorporated into the VCO is noiseless and the jitter at the output of the noiseless divider results only from the VCO jitter. Since the divider outputs one pulse for every N at its input, the variance in the output period is the sum of the variance in N input periods. Thus, the jitter at the output is \sqrt{N} times larger than the jitter at the input, or

$$J_{FD} = \sqrt{N} J_{VCO}. \quad (52)$$

Thus, to merge the divider into the VCO, the VCO gain must be reduced by a factor of N , the jitter increased by a factor of \sqrt{N} , and the divider model removed.

After simulation, it is necessary to refer the computed results, which are from the output of the divider, to the output of VCO, which is the true output of the PLL. Clearly the jitter can be computed from (52).

To determine the affect of the divider on $S_{\phi}(\omega)$, square both sides of (52) and apply (39)

$$a_{VCO} T_{VCO} = \frac{a_{FD} T_{FD}}{N}. \quad (53)$$

$T_{VCO} = T_{FD} / N$, and so

LISTING VIII
VCO WITH FD_N .

```

// Voltage Controlled Oscillator with Jitter
#include "discipline.h"
#include "constants.h"
module vco (out, in);
input in; output out; electrical out, in;
parameter real Vmin=0;
parameter real Vmax=Vmin+1 from (Vmin:inf);
parameter real Fmin=1 from (0:inf);
parameter real Fmax=2*Fmin from (Fmin:inf);
parameter real ratio=1 from (0:inf);
parameter real Vlo=-1, Vhi=1;
parameter real tt=0.01*ratio/Fmax from (0:inf);
parameter real jitter=0 from [0:0.25*ratio/Fmax];
parameter real ttol=1u*ratio/Fmax from (0:ratio/Fmax);
parameter real outStart=inf from (1/Fmin:inf);
real freq, phase, dT, delta, prev, Vout;
integer n, seed, fp;
analog begin
  @(initial_step) begin
    seed = -561;
    delta = jitter * sqrt(2*ratio);
    fp = $fopen("periods.m");
    Vout = Vlo;
  end
  // compute the freq from the input voltage
  freq = (V(in) - Vmin)*(Fmax - Fmin)
    / (Vmax - Vmin) + Fmin;
  // bound the frequency (this is optional)
  if (freq > Fmax) freq = Fmax;
  if (freq < Fmin) freq = Fmin;
  // apply the frequency divider, add the phase noise
  freq = freq / ratio;
  freq = freq/(1 + dT*freq);
  // phase is the integral of the freq modulo 1
  phase = idtmod(freq, 0.0, 1.0, -0.5);
  // update jitter twice per period
  @(cross(phase - 0.25, +1, ttol)) begin
    dT = delta * $dist_normal(seed, 0, 1);
    Vout = Vhi;
  end
  @(cross(phase + 0.25, +1, ttol)) begin
    dT = delta * $dist_normal(seed, 0, 1);
    Vout = Vlo;
    if ($realtime >= outStart)
      $fstrobe( fp, "%0.10e", $realtime - prev);
      prev = $realtime;
    end
  V(out) <+ transition(Vout, 0, tt);
end
endmodule

```

$$a_{VCO} = a_{FD} \quad (54)$$

From (41),

$$S_{VCO} \frac{f^2}{f_{VCO}^2} = S_{FD} \frac{f^2}{f_{FD}^2} \quad (55)$$

Finally, $f_{VCO} = N f_{FD}$, and so

$$S_{VCO} = N^2 S_{FD}. \quad (56)$$

Once FD_N is incorporated into the VCO, the VCO output signal is no longer observable, however the characteristics of the VCO output are easily derived from (52) and (56), which are summarized in Table II.

TABLE II
CHARACTERISTICS OF VCO OUTPUT RELATIVE TO THE OUTPUT OF FD_N ASSUMING THE VCO EXHIBITS SIMPLE FM JITTER AND THE FD_N IS NOISEFREE.

Frequency	Jitter	Phase Noise
$f_{VCO} = N f_{FD}$	$J_{VCO} = \frac{J_{FD}}{\sqrt{N}}$	$S_{\phi_{VCO}} = N^2 S_{\phi_{FD}}$

It is interesting to note that while the frequency at the output of FD_N is N times smaller than at the output of the VCO, except for scaling in the amplitude, the spectrum of the noise close to the fundamental is to a first degree unaffected by the presence of FD_N . In particular, the width of the noise spectrum is unaffected by FD_N . This is extremely fortuitous, because it means that the number of cycles we need to simulate is independent of the divide ratio N . Thus, large divide ratios do not affect the total simulation time.

To understand why FD_N does not affect the width of the noise spectrum, recall that while we started with a jitter that varied continuously with time, $j(t)$ in (7), for either efficiency or modeling reasons we eventually sampled it to end up with a discrete-time version. The act of sampling the jitter causes the spectrum of the jitter to be replicated at the multiples of the sampling frequency, which adds aliasing. This aliasing is visible, but not obvious, at high frequencies in Figure 8. However, especially with FM jitter, the phase noise amplitude at low frequencies is much larger than the aliased noise, and so the close-in noise spectrum is largely unaffected by the sampling. The affect of FD_N is to decimate the sampled jitter by a factor of N , which is equivalent to sampling the jitter signal, $j(t)$, at the original sample frequency divided by N . Thus, the replication is at a lower frequency, the amplitude is lower, and the aliasing is greater, but the spectrum is otherwise unaffected.

VII. CHARACTERIZING JITTER

The switching nature of the blocks in a PLL prevents use of the conventional noise analysis available from SPICE to characterize the noise of any of the blocks in a PLL with the possible exception of the LF. The SPICE noise analysis operates by linearizing the block about a DC operating point, which is not sufficient when the block exhibits switching behavior. SpectreRF and the simulator developed by Demir both linearize the circuit about a time-varying operating point and compute the noise at the output of the block while taking into account both the effect of the time-varying operating point on the bias-dependent noise sources and the time-varying nature of the transfer function from the noise source to the output. They differ in that SpectreRF is constrained to operate on periodic circuits. In addition, SpectreRF outputs noise as a function of frequency averaged over a period, while Demir's simulator computes the output noise as a function of time and integrated over all frequencies.

Both simulators linearize about the operating point and compute the noise as a post processing step. Thus, the noise does not affect the operating point calculation and so the simulation will not be accurate if the noise is large enough to affect the large-signal behavior of the circuit. Generally, the amplitude of the noise sources is quite small and so this is not a concern. However, in thresholding circuits, the noise present when the signal crosses the threshold gets amplified tremendously. When cascading several thresholding stages, the noise can be amplified to such a degree that it does change the large signal behavior, making the simulation inaccurate. This occurs in a FD implemented as a ripple counter with a large number of stages. In such cases it is necessary to break the circuit down and only characterize the jitter of one or two stages at a time. The maximum number of stages that can be characterized together is greater if the jitter is small relative to the transition time of the circuit, as shown in Figure 5.

A. Characterizing PM Jitter

SpectreRF's PNoise analysis computes the time-average power spectral density of the noise at the output of the block. If this noise is stationary (as opposed to cyclostationary), it is a simple matter to apply (31) to calculate the jitter. Simply choose a representative set of periodic inputs to the block and use SpectreRF's Periodic Steady State (PSS) analysis to compute the steady-state response. This computes the periodic operating point about which the noise analysis is performed. It also gives $dv(t_c)/dt$, the slew rate of the output at threshold crossing. Apply SpectreRF's PNoise analysis to compute the noise power at the output as a function of frequency. Choose the frequency range of the analysis so that the total noise at frequencies outside

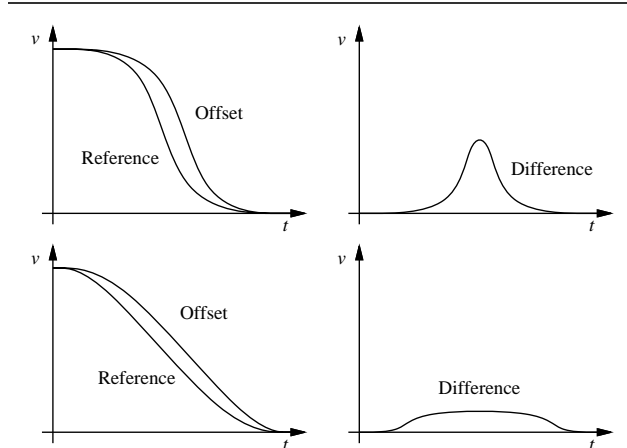


Fig. 5. The figures on the left show a reference signal, and one offset due to jitter. In both the top and bottom figures, the offset is the same (they have the same amount of jitter). The top signals have a smaller transition time than those on the bottom. The figures on the right show the difference between the reference signal and the offset signal. For the same amount of jitter, the maximum difference is larger when the transition time is smaller. If the difference is large enough to cause a nonlinear response, the noise simulations will not be accurate. This problem becomes significant when the jitter is about the same size of the transition time or larger. It limits the number of cascaded thresholding stages that can be characterized at one time.

the range is negligible. Thus, the noise should be at least 40 dB down and dropping at the highest frequency simulated. Finally, integrate the noise power over frequency and apply Wiener-Khinchin Theorem [16] to determine

$$\text{var}(n_v) = \int_{-\infty}^{\infty} S_{n_v}(f) df, \quad (57)$$

the total noise power [9], and apply (31).

In general, the noise is strongly cyclostationary and so the above procedure is insufficient. When the noise is cyclostationary the same procedure is used, except a gating function is applied to the output so that only the noise that occurs near the threshold crossing is considered in the jitter calculation. SpectreRF's PNoise analysis computes the time-average of the noise at the output and it is not possible in general to post-process the PNoise results to determine the noise at the time of the threshold crossing. Rather, a limiter is added to the output of the block and SpectreRF computes the noise at the output of the limiter. The limiter, given in Listing IX, is designed to saturate when the output of the block is outside a certain range to prevent any noise at the output from being considered except the noise present near when the signal crosses the threshold. The limiter is shown in Figure 6. The range of the limiter, V_L and V_H , is chosen such that the noise and

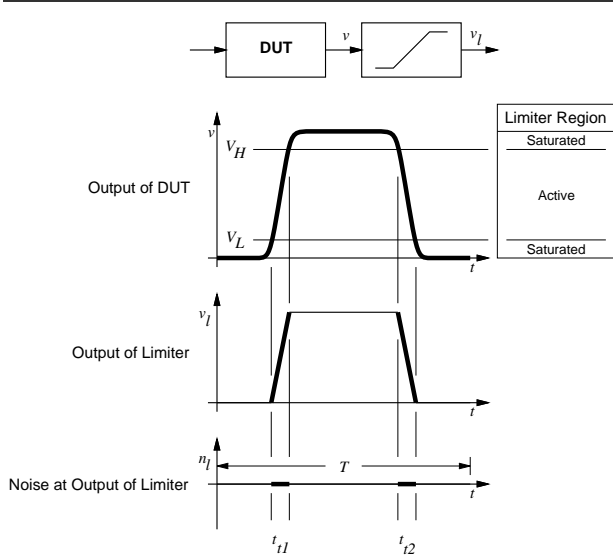


Fig. 6. When the device-under-test (DUT) exhibits cyclostationary noise, a limiter is applied to output of driven block to suppress noise present outside of the active transitions. $v(t)$ is the output of the block, $v_l(t)$ is the output of the limiter, and $n_l(t)$ is the noise at the output of the limiter. Noise on a waveform is denoted by using a thick trace. The output of the limiter is only noisy when it is inside its active region (when it is not limiting).

LISTING IX

LIMITER USED TO CHARACTERIZE PM JITTER IN BINARY SIGNALS.

```

// Simple Limiter
#include "discipline.h"
#include "constants.h"
module limiter (out, in);
output out; input in; electrical out, in;
parameter real Vlo=-1, Vhi=1;
analog begin
    // Place time-point at threshold crossings
    @(cross(V(in) - Vlo) or cross(V(in) - Vhi));
    // Determine the output
    if (V(in) < Vlo)
        V(out) <+ Vlo;
    else if (V(in) > Vhi)
        V(out) <+ Vhi;
    else
        V(out) <+ V(in);
end
endmodule
    
```

the slew rate is approximately constant while the limiter is active. When running PNoise analysis, assure that the maxsidebands parameter is at least ten times larger than T/t_{ii} for any i . This assures that the narrow noise pulses are adequately resolved by the PNoise analysis. Jitter is inde-

pendent of T , so to reduce the number of sidebands needed, use T as small as possible. If maxsidebands is not set sufficiently large, then the extracted value of jitter will increase as T decreases.

Assume that the amplitude of the noise during each transition is the same, then

$$\text{var}(n_p, t_c) \cong \frac{\langle n_l^2 \rangle T}{t_{t1} + t_{t2}} \quad (58)$$

where $\langle n_l^2 \rangle$ is the time average of the noise power at the output of the limiter. SpectreRF computes the power spectral density of the noise at the output of the limiter, and by the Parseval's Theorem,

$$\langle n_l^2 \rangle = \lim_{T \rightarrow \infty} \frac{1}{2T} \int_{-T}^T n_l^2(t) dt = \int_{-\infty}^{\infty} S_{n_l}(f) df. \quad (59)$$

If we further assume that $t_t = t_{t1} = t_{t2}$, then

$$\dot{v}(t_c) \approx \frac{V_H - V_L}{t_t}. \quad (60)$$

And from (32) and (33)

$$J = \frac{\sqrt{2\text{var}(n_p, t_c)}}{\dot{v}(t_c)} \approx \sqrt{\frac{2\langle n_l^2 \rangle T}{K t_t}} \frac{t_t}{V_H - V_L}, \quad (61)$$

$$J \approx \frac{\sqrt{2\langle n_l^2 \rangle T t_t / K}}{V_H - V_L}, \quad (62)$$

where K is the number of transitions that occurred during the period. If $K = 2$,

$$J \approx \frac{\sqrt{\langle n_l^2 \rangle T t_t}}{V_H - V_L}. \quad (63)$$

In practice, the noise away from the transitions is usually much smaller than the noise during the transitions. So one can usually achieve reasonably accurate results by applying (62) or (63) even without using the limiter.

This general methodology for characterizing the PM jitter of driven blocks with binary outputs is extended or clarified for important special cases in the next few sections.

PM jitter of the PFD/CP: The PFD and CP work together to generate a three-level discrete-valued signal (it takes the values -1 , 0 , and $+1$) whose time average is used as the loop error signal. The average of this signal controls the VCO after it has been extracted by the LF.

There are two aspects of the PFD/CP that differ from the assumptions made above. First, the output of the CP is a current, so the limiter and the equations given in the previous section need to be adapted. Second, the output of the CP has three distinct levels rather than the two assumed

above. Thus, the CP has a ternary output rather than a binary output.

If it is necessary to apply a gating function, care must be taken because of the ternary nature of the output. A simple limiter would allow the noise associated with the middle value to pass. So the simple limiter should be replaced with a dead-band limiter. This is a limiter with a dead band in the center of its input range. The dead band rejects noise about the equilibrium point associated with the middle of the three values.

PM Jitter of a FD: With ripple counters, one can only characterize a few stages at a time because of the issue shown in Figure 5. Thus, a long ripple counter chain has to be broken into smaller chains, and characterized individually. The total jitter for the ripple counter is then computed by taking the square-root of the sum of the square of the jitter on each stage.

Unlike in ripple counters, jitter does not accumulate with synchronous counters. Jitter in a synchronous counter is independent of the number of stages. Rather, jitter of a synchronous counter is the jitter of its clock along with the jitter of the last stage.

The input to counters are generally edge sensitive, and so are only affected by jitter on either the positive-going or the negative-going clock transitions, but not both. However, this should not affect the way in which blocks are characterized as long as the behavioral models for the dividers also have edge-sensitive inputs. Then the behavioral model of the dividers only react to jitter on the proper edge and ignore jitter on the other edge.

B. Characterizing FM Jitter

The noise of a free-running oscillator is dominated by phase noise, which is a random shifting of the frequency, and hence the phase, of the oscillation signal over time. The phase of an oscillator is subject to this variation because it is free running: there is no drive signal with which to lock and so no synchronization between the signals generated by the oscillator and any reference signal.

Oscillator phase noise and FM jitter are different ways of describing the same underlying phenomenon, and so there is a direct conversion between phase noise and FM jitter, as given in (39). There is no need to invoke the use of thresholds or gating functions in order to make the conversion.

VIII. SIMULATION AND ANALYSIS

The synthesizer is simulated using the netlist from Listing XI and the Verilog-A descriptions in Listings VI-VIII, modifying them as necessary to fit the actual circuit. The

simulation should cover an interval long enough to allow accurate Fourier analysis at the lowest frequency of interest (F_{\min}). With deterministic signals, it is sufficient to simulate for K cycles after the PLL settles if $F_{\min} = 1/TK$. However, for these signals, which are stochastic, it is best to simulate for $10K$ to $100K$ cycles to allow for enough averaging to reduce the uncertainty in the result.

One should not simply apply an FFT to the output signal of the VCO/FD_N to determine $L(f_m)$ for the PLL. The result would be quite inaccurate because the FFT samples the waveform at evenly spaced points, and so misses the jitter of the transitions. Instead, $L(f_m)$ can be measured with Spectre's Fourier Analyzer, which uses a unique algorithm that does accurately resolve the jitter [12]. However, it is slow if many frequencies are needed and so is not well suited to this application.

Unlike $L(f_m)$, $S_\phi(f)$ can be computed efficiently. The Verilog-A code for the VCO/FD_N given in Listing VIII writes the length of each period to an output file named *periods.m*. Writing the periods to the file begins after an initial delay, specified using *outStart*, to allow the PLL to reach steady state. This file is then processed by Matlab from MathWorks using the script shown in Listing X. This script computes $S_\phi(f)$, the power spectral density of ϕ , using Welch's method [15]. The frequency range is from $f_{\text{out}}/2$ to $f_{\text{out}}/n_{\text{fft}}$. The script computes $S_\phi(f_m)$ with a resolution bandwidth of *rbw*.⁵ Normally, $S_\phi(f_m)$ is given with a unity resolution bandwidth. To compensate for a non-unity resolution bandwidth, broadband signals such as the noise should be divided by *rbw*. Signals with bandwidth less than *rbw*, such as the spurs generated by leakage in the CP, should not be scaled. The script processes the output of VCO/FD_N. The results of the script must be further processed using the equations in Table II to remove the effect of FD_N.

IX. EXAMPLE

These ideas were applied to model and simulate a PLL acting as a frequency synthesizer. A synthesizer was chosen with $f_{\text{ref}} = 25$ MHz, $f_{\text{out}} = 2$ GHz, and a channel spacing of 200 kHz. As such, $M = 125$ and $N = 10,000$.

The noise of OSC is -95 dBc/Hz at 100 kHz, which corresponds to 20 ps of FM period jitter. The noise of VCO is -48 dBc/Hz, which gives 6 ps of FM period jitter. The PM period jitter of the PFD/CP and FDs was found to be 2 ns. The FDs were included into the oscillators, which suppresses the high frequency signals at the input and output

5. The Hanning window used in the *psd()* function has a resolution bandwidth of 1.5 bins [10]. Assuming broadband signals, Matlab divides by 1.5 inside *psd()* to compensate. In order to resolve narrowband signals, the factor of 1.5 is removed by the script, and instead included in the reported resolution bandwidth.

LISTING X

MATLAB SCRIPT USED FOR COMPUTING $S_{\phi}(f_m)$. THESE RESULTS MUST BE FURTHER PROCESSED USING TABLE II TO MAP THEM TO THE OUTPUT OF THE VCO.

```
% Process period data to compute  $S_{\phi}(f_m)$ 
echo off;
nfft=512; % should be power of two
winLength=nfft;
overlap=nfft/2;
winNBW=1.5; % Noise bandwidth given in bins

% Load the data from the file generated by the VCO
load periods.m;

% output estimates of period and jitter
T=mean(periods);
J=std(periods);
maxdT = max(abs(periods-T))/T;
fprintf('T = %.3gs, F = %.3GHz\n', T, 1/T);
fprintf('Jabs = %.3gs, Jrel = %.2g%%\n', J, 100*J/T);
fprintf('max dT = %.2g%%\n', 100*maxdT);
fprintf('periods = %d, nfft = %d\n', length(periods), nfft);

% compute the cumulative phase of each transition
phases=2*pi*cumsum(periods)/T;

% compute power spectral density of phase
[Sphi,f]=psd(phases,nfft,1/T,winLength,overlap,'linear');

% correct for scaling in PSD due to FFT and window
Sphi=winNBW*Sphi/nfft;

% plot the results (except at DC)
K = length(f);
semilogx(f(2:K),10*log10(Sphi(2:K)));
title('Power Spectral Density of VCO Phase');
xlabel('Frequency (Hz)');
ylabel('S phi (dB/Hz)');
rbw = winNBW/(T*nfft);
RBW=sprintf('Resolution Bandwidth = %.0f Hz (%.0f dB)',
            rbw, 10*log10(rbw));
imtext(0.5,0.07, RBW);
```

of the synthesizer. The netlist is shown in Listing XI. The results (compensated for non-unity resolution bandwidth (-28 dB) and for the suppression of the dividers (80 dB)) are shown in Figures 7-10. The simulation took 7.5 minutes for 450k time-points on a HP 9000/735. The use of a large number of timepoints was motivated by the desire to reduce the level of uncertainty in the results. The period jitter in the PLL was found to be 9.8 ps at the output of the VCO. The long-term jitter is shown in Figure 11.

The low-pass filter LF blocks all high frequency signals from reaching the VCO, so the noise of the phase lock loop at high frequencies is the same as the noise generated by the open-loop VCO alone. At low frequencies, the loop gain acts to stabilize the phase of the VCO, and the noise of the PLL is dominated by the phase noise of the OSC.

LISTING XI

SPECTRE NETLIST FOR PLL SYNTHESIZER.

```
// PLL-based frequency synthesizer that models jitter
simulator lang=spectre

ahdl_include "osc.va" // Listing VI
ahdl_include "pfd_cp.va" // Listing VII
ahdl_include "vco.va" // Listing VIII

Osc (in) osc freq=25MHz ratio=125 \
fmJitter=20ps pmJitter=2ns

PFD (err in fb) pfd_cp lout=500ua
C1 (err c) capacitor c=3.125nF
R (c 0) resistor r=10k
C2 (c 0) capacitor c=625pF
VCO (fb err) vco Fmin=1GHz Fmax=3GHz \
Vmin=-4 Vmax=4 \
ratio=10000 jitter=6ps \
outStart=10ms

JitterSim tran stop=60ms
```

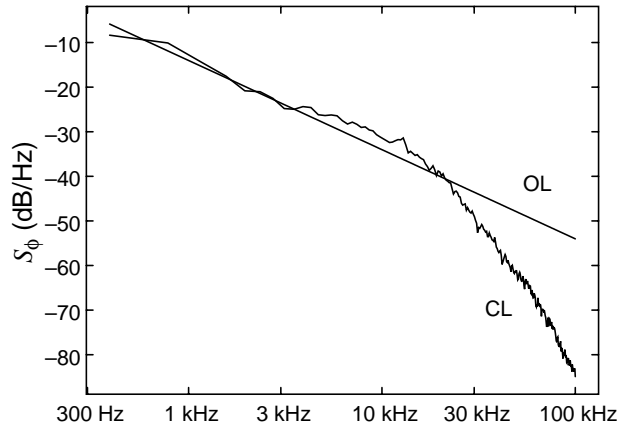
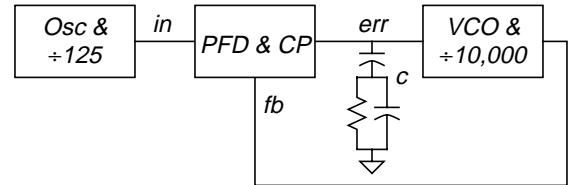


Fig. 7. Noise of the closed-loop PLL at the output of the VCO when only the reference oscillator exhibits jitter (CL) versus the noise of the reference oscillator mapped up to the VCO frequency when operated open loop (OL).

There is some contribution from the VCO, but it is diminished by the gain of the loop. In this example, noise at the middle frequencies is dominated by the PM jitter generated by the PFD/CO and FDs. The measured results agree qualitatively with these expected results. The predicted noise is higher than one would expect solely from the open-loop behavior of each block because of peaking in the response of the PLL from 5 kHz to 50 kHz. For this

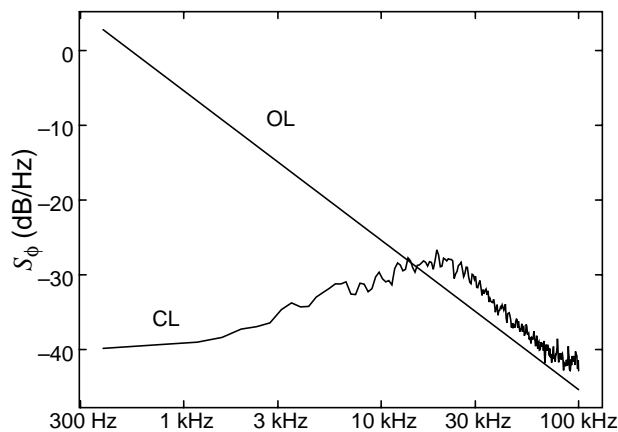


Fig. 8. Noise of the closed-loop PLL at the output of the VCO when only the VCO exhibits jitter (CL) versus the noise of the VCO when operated open loop (OL).

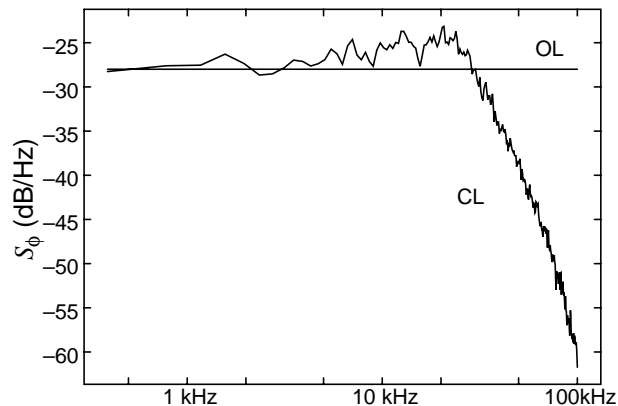


Fig. 9. Noise of the closed-loop PLL at the output of the VCO when only the PFD/CP, FD_M , and FD_N exhibit jitter (CL) versus the noise of these components mapped up to the VCO frequency when operated open loop (OL).

reason, PLLs used in synthesizers where jitter is important are usually overdamped.

X. CONCLUSION

A methodology for modeling and simulating the jitter performance of phase-locked loops was presented. The simulation is done at the behavioral level, and so is efficient enough to be applied in a wide variety of applications. The behavioral models are calibrated from circuit-level noise simulations, and so the high-level simulations are accurate. Behavioral models were presented in the Verilog-A language, however these same ideas can be used to develop behavioral models in purely event-driven languages such as Verilog-HDL and VHDL.

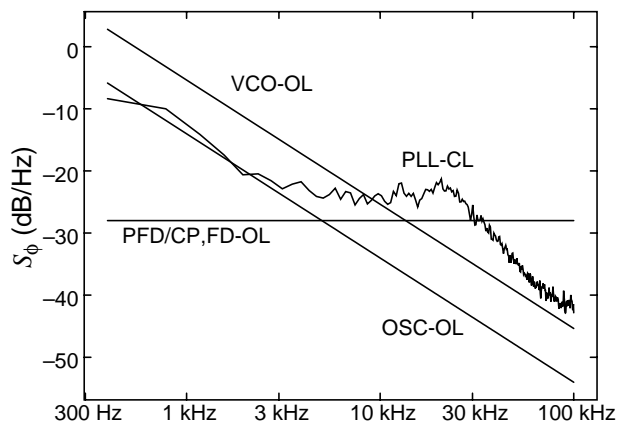


Fig. 10. Closed-loop PLL noise performance compared to the open-loop noise performance of the individual components that make up the PLL. The achieved noise is slightly larger than what is expected from the components due to peaking in the response of the PLL.

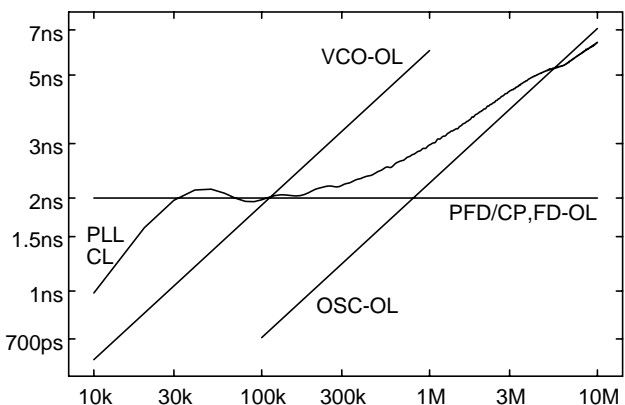


Fig. 11. The long-term jitter (J_i) of the closed-loop PLL as a function of observation interval, where the interval is measured in the number of 500 ps periods i at the output of the VCO. (These results do not agree well with expected results. This may be due to peaking in the response of the PLL, or it may be due to some yet unidentified problem).

This methodology is flexible enough to be used in a broad range of applications where jitter is important. Examples include, clock generation and recovery, sampling systems, over-sampled ADCs, digital modulation and demodulation systems, and fractional- N frequency synthesis (though it is not possible to merge the VCO and divider in this case).

ACKNOWLEDGMENTS

I would like to thank Alper Demir and Manolis Terrovitis of the University of California in Berkeley for many enlightening conversations about noise and jitter. I would

also like to thank Mark Chapman, Masayuki Takahashi, and Kimihiro Ogawa of Sony Semiconductor and Rich Davis, Frank Hellmich and Randeep Soin of Cadence Design Systems for their probing questions and insightful comments, as well as their help in validating these ideas on real frequency synthesizers.

REFERENCES

- [1] H. Chang, E. Charbon, U. Choudhury, A. Demir, E. Felt, E. Liu, E. Malavasi, A. Sangiovanni-Vincentelli, and I. Vassiliou. *A Top-Down Constraint-Driven Methodology for Analog Integrated Circuits*. Kluwer Academic Publishers, 1997.
- [2] A. Demir, E. Liu, A. Sangiovanni-Vincentelli, and I. Vassiliou. Behavioral simulation techniques for phase/delay-locked systems. *Proceedings of the IEEE Custom Integrated Circuits Conference*, pp. 453-456, May 1994.
- [3] A. Demir, E. Liu, and A. Sangiovanni-Vincentelli. Time-domain non-Monte-Carlo noise simulation for nonlinear dynamic circuits with arbitrary excitations. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 15, no. 5, pp. 493-505, May 1996.
- [4] A. Demir, A. Sangiovanni-Vincentelli. Simulation and modeling of phase noise in open-loop oscillators. *Proceedings of the IEEE Custom Integrated Circuits Conference*, pp. 445-456, May 1996.
- [5] A. Demir, A. Sangiovanni-Vincentelli. *Analysis and Simulation of Noise in Nonlinear Electronic Circuits and Systems*. Kluwer Academic Publishers, 1997.
- [6] A. Demir, A. Mehrotra, J. Roychowdhury. Phase noise in oscillators: a unifying theory and numerical methods for characterization. *Proceedings of the 35th Design Automation Conference*, June 1998.
- [7] D. FitzPatrick, I. Miller. *Analog Behavioral Modeling with the Verilog-A Language*. Kluwer Academic Publishers, 1997.
- [8] F. Gardner. *Phase-lock Techniques*. John Wiley & Sons, 1979.
- [9] W. Gardner. *Introduction to Random Processes: With Applications to Signals and Systems*. McGraw-Hill, 1989.
- [10] F. Harris. On the use of windows for harmonic analysis with the discrete Fourier transform. *Proceedings of the IEEE*, vol. 66, no. 1, January 1978.
- [11] P. R. Gray and R. G. Meyer. *Analysis and Design of Analog Integrated Circuits*. J. Wiley & Sons, Third Edition, 1993.
- [12] K. Kundert. *The Designer's Guide to SPICE and Spectre*. Kluwer Academic Publishers, 1995.
- [13] J. McNeill. Jitter in Ring Oscillators. *IEEE Journal of Solid-State Circuits*, vol. 32, no. 6, June 1997.
- [14] *Verilog-A Language Reference Manual: Analog Extensions to Verilog-HDL*, version 1.0. Open Verilog International, 1996. Available from www.ovi.org.
- [15] A. Oppenheim, R. Schaffer. *Digital Signal Processing*. Prentice-Hall, 1975.
- [16] A. Papoulis. *Probability, Random Variables, and Stochastic Processes*. McGraw-Hill, 1991.
- [17] R. Telichevesky, K. Kundert, J. White. Receiver characterization using periodic small-signal analysis. *Proceedings of the IEEE Custom Integrated Circuits Conference*, May 1996.
- [18] R. Telichevesky, K. Kundert, J. White. Efficient AC and noise analysis of two-tone RF circuits. *Proceedings of the 33rd Design Automation Conference*, June 1996.
- [19] G. Vendelin, A. Pavio, U. Rohde. *Microwave Circuit Design*. J. Wiley & Sons, 1990.