Predicting the Phase Noise of PLL-Based Frequency Synthesizers

Ken Kundert

Designer's Guide Consulting, Inc.

Version 4g, 23 October 2015

A methodology is presented for predicting the phase noise of a PLL-based frequency synthesizer using simulation that is both accurate and efficient. The methodology begins by characterizing the phase noise behavior of the blocks that make up the PLL using transistor-level RF noise simulation. For each block, the phase noise is extracted and applied to a phase-domain model for the entire PLL.

Search Terms

Phase-locked loop, PLL simulation, PLL phase-domain modeling, frequency synthesizer, oscillator phase noise, jitter, cyclostationary noise, charge-pump noise, phase-detector noise, frequency divider noise, SpectreRF, Verilog-A.

1 Introduction

Phase-locked loops (PLLs) are used to implement a variety of timing related functions, such as frequency synthesis, clock and data recovery, and clock de-skewing. Any jitter or phase noise in the output of the PLL used in these applications generally degrades the performance margins of the system in which it resides and so is of great concern to the designers of such systems. Jitter and phase noise are different ways of referring to an undesired variation in the timing of events at the output of the PLL. They are difficult to predict with traditional circuit simulators because the PLL generates repetitive switching events as an essential part of its operation, and the noise performance must be evaluated in the presence of this large-signal behavior. SPICE is useless in this situation as it can only predict the noise in circuits that have a quiescent (time-invariant) operating point. In PLLs the operating point is at best periodic, and is sometimes chaotic. Recently a new class of circuit simulators has been introduced that are capable of predicting the noise behavior about a periodic operating point [15]. Spectre®RF[1] is the most popular of this class of simulators and, because of the algorithms used in its implementation, is likely to be the best suited for this application [1]. These simulators can be used to predict the noise performance of PLLs.

The focus of this paper is frequency synthesis. Information on predicting the noise and jitter of clock and data recovery circuits can be found elsewhere [18,19].

1.1 Frequency Synthesis

The block diagram of a PLL operating as a frequency synthesizer is shown in Figure 1 [7]. It consists of a reference 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_{ref}. Given an input frequency f_{in}, the frequency at the output of the PLL is

$$f_{\text{out}} = \frac{N}{M} f_{\text{in}} \tag{1}$$

where M is the divide ratio of the input frequency divider, and N is the divide ratio of the feedback divider. By choosing the frequency divide ratios and the input frequency appropriately, the synthesizer generates an output signal at the desired frequency that inherits the long term stability of the input oscillator. In RF transceivers, this architecture is commonly used to generate the local oscillator (LO) at a programmable frequency that tunes the transceiver to the desired channel by adjusting the value of N.

FIGURE 1   *The block diagram of a frequency synthesizer.*

---
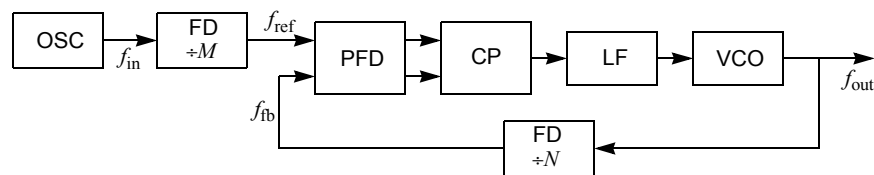
1. Spectre is a registered trademark of Cadence Design Systems.

1.2   Direct Simulation

In many circumstances, SpectreRF can be directly applied to predict the noise performance of a PLL. To make this possible, the PLL must at a minimum have a periodic steady state solution. This rules out systems such as bang-bang clock and data recovery circuits and fractional-*N* synthesizers because they behave in a chaotic way by design. It also rules out any PLL that is implemented with a phase detector that has a dead zone. A dead zone has the effect of opening the loop and letting the phase drift seemingly at random when the phase of the reference and the output of the voltage-controlled oscillator (VCO) are close. This gives these PLLs a chaotic nature.

To perform a noise analysis, SpectreRF must first compute the steady-state solution of the circuit with its periodic steady state (PSS) analysis. If the PLL does not have a periodic solution, as the cases described above do not, then it will not converge. There is an easy test that can be run to determine if a circuit has a periodic steady-state solution. Simply perform a transient analysis until the PLL approaches steady state and then observe the VCO control voltage. If this signal consists of frequency components at integer multiples of the reference frequency, then the PLL has a periodic solution. If there are other components, it does not. Sometimes it can be difficult to identify the undesirable components if the components associated with the reference frequency are large. In this case, use the strobing feature of Spectre's transient analysis to eliminate all components at frequencies that are multiples of the reference frequency. Do so by strobing at the reference frequency. In this case, if the strobed VCO control voltage varies in any significant way the PLL does not have a periodic solution.

If the PLL has a periodic solution, then in concept it is always possible to apply SpectreRF directly to perform a noise analysis. However, in some cases it may not be practical to do so. The time required for SpectreRF to compute the noise of a PLL is proportional to the number of circuit equations needed to represent the PLL in the simulator multiplied both by the number of time points needed to accurately render a single period of the solution and the number of frequencies at which the noise is desired. When applying SpectreRF to frequency synthesizers with large divide ratios, the number of time points needed to render a period can become problematic. Experience shows that divide ratios greater than ten are often not practical to simulate. Of course, this varies with the size of the PLL.

For PLLs that are candidates for direct simulation using SpectreRF, simply configure the simulator to perform a PSS analysis followed by a periodic noise (PNoise) analysis. The period of the PSS analysis should be set to be the same as the reference frequency as defined in Figure 1. The PSS stabilization time (tstab) should be set long enough to allow the PLL to reach lock. This process was successfully followed on a frequency synthesizer with a divide ratio of 40 that contained 2500 transistors, though it required several hours for the complete simulation [27].

1.3   When Direct Simulation Fails

The challenge still remains, how does one predict the phase noise of PLLs that do not fit the constraints that enable direct simulation? The remainder of this document attempts to answer that question for frequency synthesizers, though the techniques presented are general and can be applied to other types of PLLs by anyone who is sufficiently determined.

1.4   Monte Carlo-Based Methods

Demir proposed an approach for simulating PLLs whereby a PLL is described using behavioral models simulated at a high level [2,3]. The models are written such that they include jitter in an efficient way. He also devised a simulation algorithm based on solving a set of nonlinear stochastic differential equations that is capable of characterizing the circuit-level noise behavior of blocks that make up a PLL [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]. Once everything is ready, simulation of the PLL occurs with the blocks of the PLL being described with behavioral models that exhibit jitter. The actual jitter or phase noise statistics of the PLL are observed during this simulation. Generally tens to hundreds of thousands of cycles are simulated, but the models are efficient so the time required for the simulation is reasonable. This approach allows 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 to characterize the jitter of the blocks.

In an earlier series of papers [16], the relevant ideas of Demir were adapted to allow use of a commercial simulator, Spectre [13], and an industry standard modeling language, Verilog-A[2] [14,25,26]. These ideas were refined in a more recent version [17].

This document focuses on predicting the phase noise of PLLs rather than on jitter and provides a simpler approach that involves the use of phase-domain models. It starts with a brief introduction to frequency synthesis, presents the concept of phase-domain models and shows how noise can be incorporated into these models. It then introduces phase noise and shows how to extract it for the blocks that make up the synthesizer. Once available they are inserted into the phase-domain model to predict the overall noise performance of the synthesizer. Finally, modeling of fractional-N synthesizers is touched upon.

2   Phase-Domain Model

It is widely understood that simulating PLLs is expensive because the period of the VCO is almost always very short relative to the time required to reach lock. This is particularly true with frequency synthesizers, especially those with large multiplication factors. The problem is that a circuit simulator must use at least 10-20 time points for every period of the VCO for accurate rendering, and the lock process often involves hundreds or thousands of cycles at the input to the phase detector. With large divide ratios, this can translate to hundreds of thousands of cycles of the VCO. Thus, the number of time points needed for a single simulation could range into the millions.

This is all true when simulating the PLL in terms of voltages and currents. When doing so, one is said to be using *voltage-domain models*. However, that is not the only option available. It is also possible to formulate models based on the phase of the signals. In this case, one would be using *phase-domain models*. The high frequency variations associated with the voltage-domain models are not present in phase-domain models, and so simulations are considerably faster. In addition, when in lock the phase-domain-

---

2. Verilog is a registered trademark of Cadence Design Systems licensed to Accellera.

based models generally have constant-valued operating points, which simplifies small-signal analysis, making it easier to study the closed-loop dynamics and noise performance of the PLL using either AC or noise analysis.

A linear phase-domain model of a frequency synthesizer is shown in Figure 2. Such a model is suitable for modeling the behavior of the PLL to small perturbations when the PLL is in lock as long as you do not need to know the exact waveforms and instead are interested in how small perturbations affect the phase of the output. This is exactly what is needed to predict the phase noise performance of the PLL.

FIGURE 2   *Linear time-invariant phase-domain model of the synthesizer shown in Figure 1.*
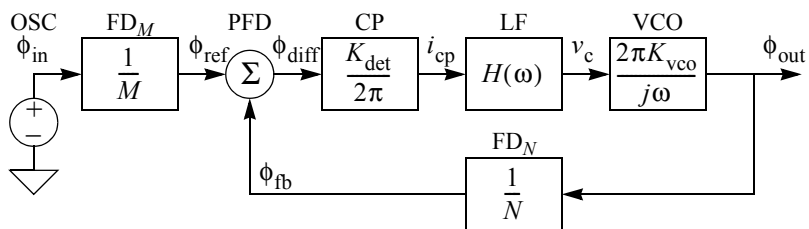
The derivation of the model begins with the identification of those signals that are best represented by their phase. Many blocks have large repetitive input signals with their outputs being primarily sensitive to the phase of their inputs. It is the signals that drive these blocks that are represented as phase. They are identified using a ϕ variable in Figure 2. Notice that this includes all signals except those at the inputs of the LF and VCO.

The models of the individual blocks will be derived by assuming that the signals associated with each of the phase variables is a pulse train. Though generally the case, it is not a requirement. It simply serves to make it easier to extract the models. Define $\Pi(t_0, \tau, T)$ to be a periodic pulse train where one of the pulses starts at t_0 and the pulses have duration τ and period T as shown in Figure 3. This signal transitions between 0 and 1 if τ is positive, and between 0 and -1 if τ is negative. The phase of this signal is defined to be $\phi = 2\pi t_0/T$. In many cases, the duration of the pulses is of no interest, in which case $\Pi(t_0, T)$ is used as a short hand. This occurs because the input that the signal is driving is edge triggered. For simplicity, we assume that such inputs are sensitive to the rising edges of the signal, that t_0 specifies the time of a rising edge, and that the signal is transitioning between 0 and ± 1.

FIGURE 3   *The pulse train waveform represented by* $\Pi(t_0, \tau, T)$.
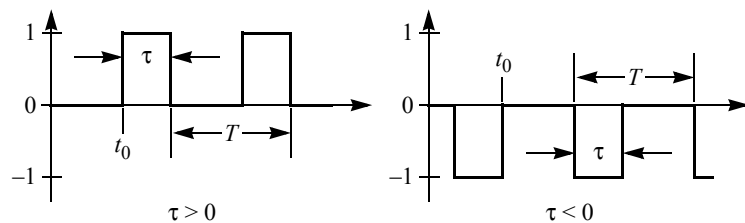
The input source produces a signal $v_{in} = \Pi(t_0, T)$. Since this is the input, t_0 is arbitrary. As such, we are free to set its phase ϕ to any value we like.

Given a signal $v_i = \Pi(t_0, T)$ a frequency divider will produce an output signal $v_o = \Pi(t_0, NT)$ where N is the divide ratio. The phase of the input is $\phi_i = 2\pi t_0/T$ and the phase of the output is $\phi_o = 2\pi t_0/(NT)$ and so the phase transfer characteristic of a divider is

$$\phi_o = \phi_i/N. \tag{2}$$

There are many different types of phase detectors that can be used, each requiring a somewhat different model. Consider a simple phase-frequency detector combined with a charge pump [23]. In this case, the detector takes two inputs, $v_1 = \Pi(t_1, T)$ and $v_0 = \Pi(t_0, T)$ and produces an output $i_{cp} = I_{max}\Pi(t_0, t_1-t_0, T)$ where I_{max} is the maximum output current of the charge pump. The output of the charge pump immediately passes through a low pass filter that is designed to suppress signals at frequencies of $1/T$ and above, so in most cases the pulse nature of this signal can be ignored in favor of its average value, $\langle i_{cp} \rangle$. Thus, the transfer characteristic of the combined PFD/CP is

$$\langle i_{cp} \rangle = I_{max}\frac{t_1 - t_0}{T} = I_{max}\frac{\phi_1 - \phi_0}{2\pi} = \frac{K_{det}}{2\pi}(\phi_1 - \phi_0) \tag{3}$$

where $K_{det} = I_{max}$. Of course, this is only valid for $|\phi_1 - \phi_2| < 2\pi$ at the most. The behavior outside this range depends strongly on the type of phase detector used [7]. Even within this range, the phase detector may be better modeled with a nonlinear transfer characteristic. For example, there can be a flat spot in the transfer characteristics near 0 if the detector has a dead zone. However it is generally not productive to model the dead zone in a phase-domain model.[3]

The model of (3) is a continuous-time approximation to what is inherently a discrete-time process. The phase detector does not continuously monitor the phase difference between its two input signals, rather it outputs one pulse per cycle whose width is proportional to the phase difference. Using a continuous time approximation is generally acceptable if the bandwidth of the loop filter is much less than f_{ref} (generally less than $f_{ref}/10$ is sufficient). In practical PLLs this is almost always the case. It is possible to develop a detailed phase-domain PFD model that includes the discrete-time effects, but it would run more slowly and the resulting phase-domain model of the PLL would not have a quiescent operating point, which makes it more difficult to analyze.

The voltage-controlled oscillator, or VCO, converts its input voltage to an output frequency, and the relationship between input voltage and output frequency can be represented as

$$f_{out} = F(v_c) \tag{4}$$

The mapping from voltage to frequency is designed to be linear, so a first-order model is often sufficient,

$$f_{out} = K_{vco}v_c. \tag{5}$$

It is the output phase that is needed in a phase-domain model,

---

3. This phase-domain model is a continuous-time model that ignores the sampling nature of the phase detector. A dead zone interacts with the sampling nature of the detector to create a chaotic limit cycle behavior that is not modeled with the phase-domain model. This chaotic behavior creates a substantial amount of jitter, and for this reason, most modern phase detectors are designed such that they do not exhibit dead zones.

$$\phi_{out}(t) = 2\pi \int K_{vco} v_c(t) dt \tag{6}$$

or in the frequency domain,

$$\phi_{out}(\omega) = \frac{2\pi K_{vco}}{j\omega} v_c(\omega). \tag{7}$$

2.1 Small-Signal Stability

This completes the derivation of the phase-domain models for each of the blocks. Now the full model is used to help predict the small-signal behavior of the PLL. Start by using Figure 2 to write a relationship for its loop gain. Start by defining

$$G_{fwd} = \frac{\phi_{out}}{\phi_{diff}} = \frac{K_{det}}{2\pi} H(\omega) \frac{2\pi K_{vco}}{j\omega} = \frac{K_{det} K_{vco} H(\omega)}{j\omega} \tag{8}$$

to be the forward gain,

$$G_{rev} = \frac{\phi_{fb}}{\phi_{out}} = \frac{1}{N} \tag{9}$$

to be the feedback factor, and

$$T = G_{fwd} G_{rev} = \frac{K_{det} K_{vco} H(\omega)}{j\omega N} \tag{10}$$

to be the loop gain. The loop gain is used to explore the small-signal stability of the loop. In particular, the phase margin is an important stability metric. It is the negative of the difference between the phase shift of the loop at unity gain and 180°, the phase shift that makes the loop unstable. It should be no less than 45° [9]. When concerned about phase noise or jitter, the phase margin is typically 60° or more to reduce peaking in the closed-loop gain, which would result in excess phase noise.
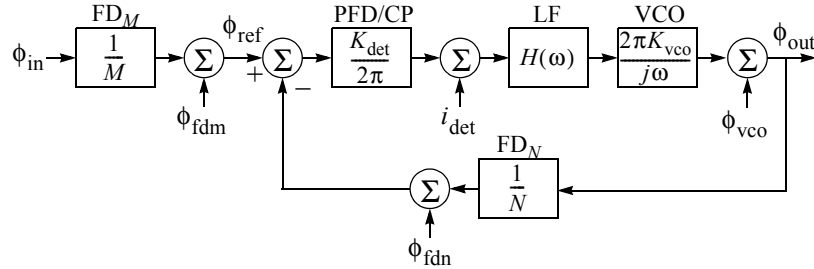
2.2 Noise Transfer Functions

In Figure 4 various sources of noise have been added. These noise sources can represent either the noise created by the blocks due to intrinsic noise sources (thermal, shot, and flicker noise sources), or the noise coupled into the blocks from external sources, such as from the power supplies, the substrate, etc. Most are sources of phase noise, and denoted ϕ_{in}, ϕ_{fdm}, ϕ_{fdn}, and ϕ_{vco}, because the circuit is only sensitive to phase at the point where the noise is injected. The one exception is the noise produced by the PFD/CP, which in this case is considered to be a current, and denoted i_{det}.

Then the transfer functions from the various noise sources to the output are

$$G_{ref} = \frac{\phi_{out}}{\phi_{ref}} = \frac{G_{fwd}}{1+T} = \frac{N G_{fwd}}{N + G_{fwd}}, \tag{11}$$

$$G_{vco} = \frac{\phi_{out}}{\phi_{vco}} = \frac{1}{1+T} = \frac{N}{N + G_{fwd}}, \tag{12}$$

FIGURE 4　　*Linear time-invariant phase-domain model of the synthesizer shown in Figure 2 with representative noise sources added. The ϕ's represent various sources of noise.*

$$G_{in} = \frac{\phi_{out}}{\phi_{in}} = \frac{1}{M}\frac{G_{fwd}}{1+T} = \frac{1}{M}\frac{NG_{fwd}}{MN+G_{fwd}} = \frac{G_{ref}}{M}, \tag{13}$$

and by inspection,

$$G_{fdn} = \frac{\phi_{out}}{\phi_{fdn}} = -G_{ref}, \tag{14}$$

$$G_{fdm} = \frac{\phi_{out}}{\phi_{fdm}} = G_{ref}, \tag{15}$$

$$G_{det} = \frac{\phi_{out}}{i_{det}} = \frac{2\pi G_{ref}}{K_{det}}. \tag{16}$$

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

These transfer functions allow certain overall characteristics of phase noise in PLLs to be identified. As $\omega \to \infty$, $G_{fwd} \to 0$ because of the VCO and the low-pass filter, and so G_{ref}, G_{det}, G_{fdm}, G_{fdn}, $G_{in} \to 0$ and $G_{vco} \to 1$. 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 $\omega \to 0$, $G_{fwd} \to \infty$ because of the $1/j\omega$ term from the VCO. So at DC, G_{ref}, G_{fdm}, $G_{fdn} \to N$, $G_{in} \to N/M$ and $G_{vco} \to 0$. 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 $(\omega \to 0)$. Oscillator phase noise in the VCO results in the power spectral density $S_{\phi_{vco}}$ being proportional to $1/\omega^2$, or $S_{\phi_{vco}} \sim 1/\omega^2$ (neglecting flicker noise). If the LF is chosen such that $H(\omega) \sim 1$, then $G_{fwd} \sim 1/\omega$, and contribution from the VCO to the output noise power, $G_{vco}^2 S_{\phi_{vco}}$, 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 $G_{fwd} \sim 1/\omega^2$ and the noise contribution to the output from the VCO goes to zero at low frequencies.

2.3 Noise Model

One predicts the phase noise exhibited by a PLL by building and applying the model shown in Figure 4. The first step in doing so is to find the various model parameters, including the level of the noise sources, which generally involves either direct measurement or simulating the various blocks with an RF simulator, such as SpectreRF. Use periodic noise (or PNoise) analysis to predict the output noise that results from stochastic noise sources contained within the blocks using simulation. Use a periodic AC or periodic transfer function (PAC or PXF) to compute the perturbation at the output of a block due to noise sources outside the block, such as on supplies.

Once the model parameters are known, it is simply a matter of computing the output phase noise of the PLL by applying the equations in Section 2.2 to compute the contributions to ϕ_{out} from every source and summing the results. Be careful to account for correlations in the noise sources. If the noise sources are perfectly correlated, as they might be if the ultimate source of noise is in the supplies or substrate, then use a direct sum. If the sources produce completely uncorrelated noise, as they would when the ultimate source of noise is random processes within the devices, use a root-mean-square sum.

Alternatively, one could build a Verilog-A model and use simulation to determine the result. The top-level of such a model is shown in Listing 1. It employs noisy phase-domain models for each of the blocks. These models are given in Listings 3-7 and are described in detail in the next few sections (3-6). In this example, the noise sources are coded into the models, but the noise parameters are not set at the top level to simplify the model. To predict the phase noise performance of the loop in lock, simply specify these parameters in the block models (given later) along with the parameters of Listing 1 and perform a noise analysis. To determine the effect of injected noise, first refer the noise to the output of one of the blocks, and then add a source into the netlist of Listing 1 at the appropriate place and perform an AC analysis.

Listings 1 and 3-7 have phase signals, and there is no phase discipline in the standard set of disciplines provided by Verilog-A or Verilog-AMS in *disciplines.vams*. There are several different resolutions for this problem. Probably the best solution is to simply add such a discipline, given in Listing 2, either to *disciplines.vams* as assumed here or to a separate file that is included as needed. Alternatively, one could use the *rotational* discipline. It is a conservative discipline that includes torque as a flow nature, and so is overkill in this situation. Finally, one could simply use either the electrical or the voltage discipline. Scaling for voltage in volts and phase in radians is similar, and so it will work fine except that the units will be reported incorrectly. Using the rotational discipline would require that all references to the phase discipline be changed to rotational in the appropriate listings. Using either the electrical or voltage discipline would require that both the name of the disciplines be changed from phase to either electrical or voltage, and the name of the access functions be changed from *Theta* to *V.*

3 Oscillators

Oscillators are responsible for most of the noise at the output of the majority of well-designed frequency synthesizers. This is because oscillators inherently tend to amplify noise found near their oscillation frequency and any of its harmonics. The reason for

LISTING 1   *Phase-domain model for a PLL configured as a frequency synthesizer.*

```
`include "disciplines.vams"

module pll(out);
output out;
phase out;
parameter integer m = 1 from [1:inf];    // input divide ratio
parameter real Kdet = 1 from (0:inf);    // phase detector gain
parameter real Kvco = 1 from (0:inf);    // VCO gain
parameter real c1 = 1n from (0:inf);     // Loop filter C1
parameter real c2 = 200p from (0:inf);   // Loop filter C2
parameter real r = 10K from (0:inf);     // Loop filter R
parameter integer n = 1 from [1:inf];    // feedback divide ratio
phase in, ref, fb;
electrical c;

oscillator OSC(in);
divider #(.ratio(m)) FDm(in, ref);
phaseDetector #(.gain(Kdet)) PD(ref, fb, c);
loopFilter #(.c1(c1), .c2(c2), .r(r)) LF(c);
vco #(.gain(Kvco)) VCO(c, out);
divider #(.ratio(n)) FDn(out, fb);

endmodule
```

LISTING 2   *Signal flow discipline definition for phase signals (the nature Angle is defined in disciplines.vams). This definition is assumed to reside in a file named "phase.vams".*

```
`include "disciplines.vams"

discipline phase
     potential Angle;
enddiscipline
```

this behavior is covered next, followed by a description of how to characterize and model the noise in an oscillator. The origins of oscillator phase noise are described in a conceptual way here. For a detailed description, see the papers by Käertner or Demir et al [6,11,12].

3.1 Oscillator Phase Noise

Nonlinear oscillators naturally produce high levels of phase noise. To see why, consider the trajectory of a fully autonomous oscillator's stable periodic orbit in state space. In steady state, the trajectory is a stable limit cycle, v. Now consider perturbing the oscillator with an impulse and assume that the deviation in the response due to the perturbation is Δv, as shown in Figure 5. Separate Δv into amplitude and phase variations,

$$\Delta v(t) \; = \; [1 + \alpha(t)]v\left(t + \frac{\phi(t)}{2\pi f_o}\right) - v(t) \, . \tag{17}$$

where v represents the unperturbed T-periodic output voltage of the oscillator, α represents the variation in amplitude, ϕ is the variation in phase, and $f_o = 1/T$ is the oscillation frequency.

FIGURE 5 *The trajectory of an oscillator shown in state space with and without a perturbation Δv. By observing the time stamps (t_0, ..., t_6) one can see that the deviation in amplitude dissipates while the deviation in phase does not.*
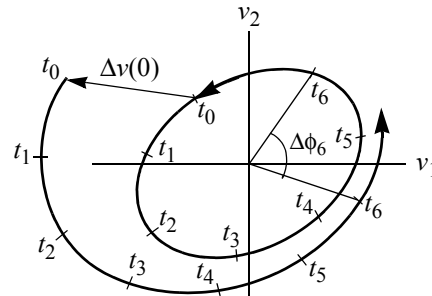
Since the oscillator is stable and the duration of the disturbance is finite, the deviation in amplitude eventually decays away and the oscillator returns to its stable orbit ($\alpha(t) \to 0$ as $t \to \infty$). In effect, there is a restoring force that tends to act against amplitude noise. This restoring force is a natural consequence of the nonlinear nature of the oscillator that acts to suppresses amplitude variations.

The oscillator is autonomous, and so any time-shifted version of the solution is also a solution. Once the phase has shifted due to a perturbation, the oscillator continues on as if never disturbed except for the shift in the phase of the oscillation. There is no restoring force on the phase and so phase deviations accumulate. A single perturbation causes the phase to permanently shift ($\phi(t) \to \Delta\phi$ as $t \to \infty$). If we neglect any short term time constants, it can be inferred that the impulse response of the phase deviation $\phi(t)$ can be approximated with a unit step $s(t)$. The phase shift over time for an arbitrary input disturbance u is

$$\phi(t) \sim \int_{-\infty}^{\infty} s(t-\tau)u(\tau)d\tau = \int_{-\infty}^{t} u(\tau)d\tau, \tag{18}$$

or the power spectral density (PSD) of the phase is

$$S_\phi(\Delta f) \sim \frac{S_u(\Delta f)}{(2\pi\Delta f)^2} \tag{19}$$

This shows that in all oscillators the response to any form of perturbation, including noise, is amplified and appears mainly in the phase. The amplification increases as the frequency of the perturbation approaches the frequency of oscillation in proportion to $1/\Delta f$ (or $1/\Delta f^2$ in power).

Notice that there is only one degree of freedom — the phase of the oscillator as a whole. There is no restoring force when the phase of all signals associated with the oscillator shift together, however there would be a restoring force if the phase of signals shifted relative to each other. This observation is significant in oscillators with multiple outputs, such as quadrature or ring oscillators. The dominant phase variations appear identically in all outputs, whereas relative phase variations between the outputs are naturally suppressed by the oscillator or added by subsequent circuitry and so tend to be much smaller [5].

3.2 Characterizing Oscillator Phase Noise

Above it was shown that oscillators tend to convert perturbations from any source into a phase variation at their output with an amplification that varies with $1/\Delta f$ (or $1/\Delta f^2$ in power). Now assume that the perturbation is from device noise in the form of white and flicker stochastic processes. The oscillator's response will be characterized first in terms of the phase noise S_ϕ, and then because phase noise is not easily measured, in terms of the normalized single-sideband noise power L. The result will be a small set of easily extracted parameters that completely describe the response of the oscillator to white and flicker noise sources. These parameters are used when modeling the oscillator.

Assume that the perturbation consists of white and flicker noise and so has the form

$$S_u(\Delta f) \sim 1 + \frac{f_c}{\Delta f}. \tag{20}$$

Then from (19) the response will take the form

$$S_\phi(\Delta f) = n\left(\frac{1}{\Delta f^2} + \frac{f_c}{\Delta f^3}\right), \tag{21}$$

where the factor of $(2\pi)^2$ in the denominator of (19) has been absorbed into the constant of proportionality n and S_ϕ is chosen to be the single-sided PSD[4]. Thus, the response of the oscillator to white and flicker noise sources is characterized using just two parameters, n and f_c, where n is the portion of S_ϕ attributable to the white noise sources alone at $\Delta f = 1$ Hz and f_c is the flicker noise corner frequency.

As shown in Figure 6, n is extracted by simply extrapolating to 1 Hz from a frequency where the noise from the white sources dominates.

FIGURE 6   *Extracting the noise parameters, n and f_c, for an oscillator from the single-sided power spectral density of its phase noise. The graph is plotted on a log-log scale.*
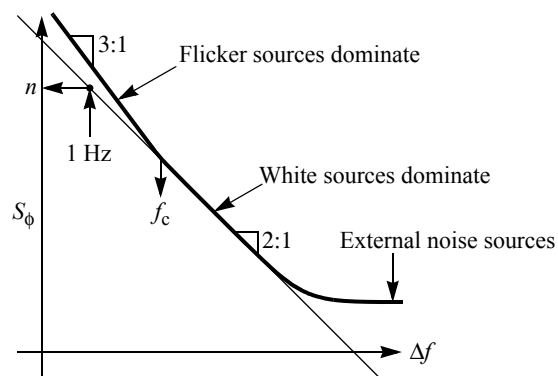
S_ϕ is not directly observable and often difficult to find. So instead oscillator phase noise is often characterized using L, the spot noise power of the output voltage S_v normalized

---

4. A single-sided PSD has a domain of $0 \le f < \infty$. The double-sided PSD is also commonly used and it has a domain of $-\infty < f < \infty$. They are related in that if S_{DS} and S_{SS} are the double- and single-sided PSDs of a signal, then $S_{DS}(0) = S_{SS}(0)$ and $S_{DS}(f) = \frac{1}{2}S_{SS}(f)$ for $f \ne 0$.

by the power in the fundamental tone. S_v is directly available from either measurement with a spectrum analyzer or from RF simulators, and L is defined as

$$L(\Delta f) = \frac{2S_v(f_o + \Delta f)}{a_1^2 + b_1^2},$$

(22)

where S_v is the single-sided PSD normalized to 1 V RMS[5] and a_1 and b_1 are the Fourier coefficients for the fundamental frequency components of v, the noise-free output signal. They satisfy

$$v(t) = \sum_{k=0}^{\infty} a_k \cos(2\pi k f_0 t) + b_k \sin(2\pi k f_0 t).$$

(23)

In (41) of [6], Demir et al shows that for a free-running oscillator perturbed only by white noise sources

$$L(\Delta f) = \frac{cf_0^2}{c^2 f_0^4 \pi^2 + \Delta f^2},$$

(24)

which is a Lorentzian process with corner frequency of

$$f_\Delta = cf_0^2 \pi.$$

(25)

The corner frequency is also known as the linewidth of the oscillator. At frequencies well above f_Δ and well below f_0,

$$L(\Delta f) = \frac{cf_0^2}{\Delta f^2}.$$

(26)

Over this same range of frequencies, Vendelin [24] showed that

$$L(\Delta f) = \frac{1}{2} S_\phi(\Delta f).$$

(27)

The empirical constants c and n are related by using this equation to combine (21) and (26).

$$\frac{cf_0^2}{\Delta f^2} = \frac{n}{2}\left(\frac{1}{\Delta f^2} + \frac{f_c}{\Delta f^3}\right)$$

(28)

This equation is valid only for $\Delta f \gg f_c$, and so the flicker noise term can be ignored, giving

---

5. While S_v is a power-spectral density, it is important to understand that the units are not W/Hz. In this case S_v has been normalized to 1 V RMS, meaning that the units are V^2/Hz (dBV/Hz if given in decibels) and that

$$P_{total} = \int_0^\infty S_v(f) df$$

is the total RMS power that would be dissipated if the signal were applied to a 1Ω resistor.

$$n = 2cf_0^2 . \tag{29}$$

The parameters n, or alternatively c, and f_c are used to describe the noise behavior of an oscillator. To extract these parameters, start by measuring either L or S_ϕ for a range of frequencies offset from the center frequency. If flicker noise is present, there will be a range of low frequencies for which the noise power drops at a rate of 30 dB per decade. Above this, the rate of drop will be 20 dB per decade. As shown in Figure 6, the frequency at which the rate switches from 30 dB to 20 dB per decade is f_c. Choose a frequency Δf well above f_c in the region where the noise is dropping at a rate of 20 dB per decade and use either (21) or (26) and (29) to determine n.

3.3   Phase-Domain Models for the Oscillators

The phase-domain models for the reference and voltage-controlled oscillators are given in Listings 3 and 4. The VCO model is based on (6). Perhaps the only thing that needs to be explained is the way that phase noise is modeled in the oscillators. Verilog-AMS provides the *flicker_noise* function for modeling flicker noise, which has a power spectral density proportional to $1/f^\alpha$ with α typically being close to 1. However, Verilog-AMS does not limit α to being close to one, making this function well suited to modeling oscillator phase noise, for which α is 2 in the white-phase noise region and close to 3 in the flicker-phase noise region (at frequencies below the flicker noise corner frequency). Alternatively, one could dispense with the noise parameters and use the *noise_table* function in lieu of the *flicker_noise* functions to use the measured noise results directly. The "wpn" and "fpn" strings passed to the noise functions are labels for the noise

---

LISTING 3   *Phase-domain oscillator noise model.*

```
`include "phase.vams"                  // from Listing 2, includes disciplines.vams.

module oscillator(out);
output out;
phase out;
parameter real n = 0 from [0:inf);     // white output phase noise at 1 Hz (rad²/Hz)
parameter real fc = 0 from [0:inf);    // flicker noise corner frequency (Hz)

analog begin
    Theta(out) <+ flicker_noise(n, 2, "wpn") + flicker_noise(n∗fc, 3, "fpn");
end
endmodule
```

---

sources. They are optional and can be chosen arbitrarily, though they should not contain any white space or special characters. *wpn* was chosen to represent white phase noise and *fpn* stands for flicker phase noise.

When interested in the effect of signals coupled into the oscillator through the supplies or the substrate, one would compute the transfer function from the interfering source to the phase output of the oscillator using either a PAC or PXF analysis. Again, one would simply assume that the perturbation in the output of the oscillator is completely in the phase, which is true except at very high offset frequencies. One then employs (12) and (13) to predict the response at the output of the PLL.

LISTING 4   *Phase-domain VCO noise model.*

```
`include "phase.vams"              // from Listing 2, includes disciplines.vams.
`include "constants.vams"

module vco(in, out);
input in; output out;
voltage in;
phase out;
parameter real gain = 1 from (0:inf);      // transfer gain, Kvco (Hz/V)
parameter real n = 0 from [0:inf);         // white output phase noise at 1 Hz (rad²/Hz)
parameter real fc = 0 from [0:inf);        // flicker noise corner frequency (Hz)

analog begin
    Theta(out) <+ 2*`M_PI*gain*idt(V(in));
    Theta(out) <+ flicker_noise(n, 2, "wpn") + flicker_noise(n*fc, 3, "fpn");
end
endmodule
```

4 Loop Filter

Even in the phase-domain model for the PLL, the loop filter remains in the voltage domain and is represented with a full circuit-level model, as shown in Listing 5. As such, the noise behavior of the filter is naturally included in the phase-domain model without any special effort assuming that the noise is properly included in the resistor model.

LISTING 5   *Loop filter model.*

```
`include "phase.vams"      // from Listing 2, includes disciplines.vams.

module loopFilter(n);
electrical n;
ground gnd;†
parameter real c1 = 1n from (0:inf);
parameter real c2 = 200p from (0:inf);
parameter real r = 10K from (0:inf);
electrical int;

capacitor #(.c(c1)) C1(n, gnd);
capacitor #(.c(c2)) C2(n, int);
resistor #(.r(r)) R(int, gnd);

endmodule
```

† The *ground* statement is not currently supported in Cadence's Verilog-A implementation, so instead ground should be explicitly passed into the module through a terminal.

5 Phase Detector and Charge Pump

As with the VCO, the noise of the PFD/CP as needed by the phase-domain model is found directly with simulation. Simply drive the block with a representative periodic signal, perform a PNoise analysis, and measure the output noise current. In this case, a

representative signal would be one that produced periodic switching in the PFD and CP. This is necessary to capture the noise present during the switching process.

The noise in the PFD/CP comes from jitter in the PFD and noise in the output current of the CP. The total noise produced by the CP will be proportional to how long it is on, while the noise from the PFD will all be in the edges and so will be independent of how long the CP is on. The CP itself has two current sources connected to its output, one that pulls up and one that pulls down. Which one is activated depends on whether the edges on the reference input lead or lag those on the feedback input. The time for which the current source stays on is equal to the difference in arrival times for the edges. Most PLLs operate in steady-state with the edges on the reference and feedback input occurring almost simultaneously (because the loop gain of the PLL is infinite at DC). In addition, most phase detectors are what is known as 'live zone' phase detectors. With these, when edges occur simultaneously on the reference and feedback inputs, both the pull up and pull down current sources will turn on for a very short period of time. From an output current perspective the pull up and pull down currents will act to cancel each other and so the effective output current is zero, however both current sources will be contributing uncorrelated noise to the output while they are on. Thus it is best to characterize the noise of the PFD/CP with simultaneous edges occurring on both the reference and feedback inputs. The output should be connected to a current probe (often in the form of an ideal voltage source) that is biased to present the expected voltage to the output of the CP.

Generally the power spectral density of the output noise current appears as in Figure 7, in which case the noise is parameterized with n and f_c. n is the noise power density at frequencies above the flicker noise corner frequency, f_c, and below the noise bandwidth of the circuit.

FIGURE 7   *Extracting the noise parameters, n and f_c, for the PFD/CP from the power spectral density of its output noise current. The graph is plotted on a log-log scale.*
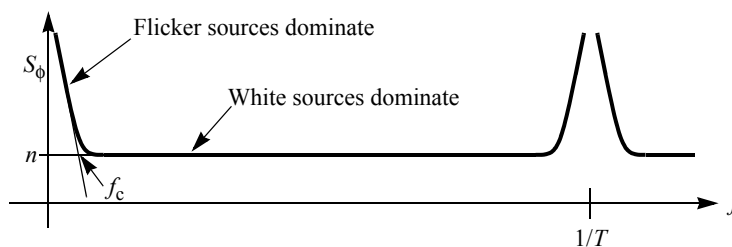
The phase-domain model for the PFD/CP is given in Listing 6. It is based on (3). Alternatively, as before one could use the *noise_table* function in lieu of the *white_noise* and *flicker_noise* functions to use the measured noise results directly.

6 Frequency Dividers

There are several reasons why the process of extracting the noise produced by the frequency dividers is more complicated than that needed for other blocks. First, the phase noise is needed and, as of the time when this document was written, SpectreRF reports on the total noise and does not yet make the phase noise available separately. Secondly,

LISTING 6   *Phase-domain phase detector noise model.*

```
`include "phase.vams"                  // from Listing 2, includes disciplines.vams.
`include "constants.vams"

module phaseDetector(pin, nin, out);
input pin, nin; output out;
phase pin, nin;
electrical out;
parameter real gain = 1 from (0:inf);    // transfer gain (A/cycle)
parameter real n = 0 from [0:inf);       // white output current noise (A²/Hz)
parameter real fc = 0 from [0:inf);      // flicker noise corner frequency (Hz)

analog begin
    I(out) <+ –gain * Theta(pin,nin) / (2*`M_PI);
    I(out) <+ white_noise(n, "wpn") + flicker_noise(n*fc, 1, "fpn");
end
endmodule
```

the frequency dividers are always followed by some form of edge-sensitive thresholding circuit, in this case the PFD, which implies that the overall noise behavior of the PLL is only influenced by the noise produced by the divider at the time when the threshold is being crossed in the proper direction. The noise produced by the frequency divider is cyclostationary, meaning that the noise power varies over time. Thus, it is important to analyze the noise behavior of the divider carefully. The second issue is discussed first.

6.1   Cyclostationary Noise.

Formally, the term cyclostationary implies that the autocorrelation function of a stochastic process varies with *t* in a periodic fashion [8,20], which in practice is associated with a periodic variation in the noise power of a signal. In general, the noise produced by all of the nonlinear blocks in a PLL is strongly cyclostationary. To understand why, consider the noise produced by a logic circuit, such as the inverter shown in Figure 8. The noise at the output of the inverter, n_{out}, comes from different sources depending on the phase of the output signal, v_{out}. When the output is high, the output is insensitive to small changes on the input. The transistor M_{P} is on and the noise at the output is predominantly due to the thermal noise from its channel. This is region *A* in the figure. When the output is low, the situation is reversed and most of the output noise is due to the thermal noise from the channel of M_{N}. This is region *B*. When the output is transitioning, thermal noise from both M_{P} and M_{N} contribute to the output. In addition, the output is sensitive to small changes in the input. In fact, any noise at the input is amplified before reaching the output. Thus, noise from the input tends to dominate over the thermal noise from the channels of M_{P} and M_{N} in this region. Noise at the input includes noise from the previous stage and noise from both devices in the form of flicker noise and thermal noise from gate resistance. This is region *C* in the figure.

The challenge in estimating the effect of noise passing through a threshold is the difficulty in estimating the noise at the point where the threshold is crossed. There are several different ways of estimating the effect of this noise, but the simplest is to use the strobed noise feature of SpectreRF.[6] When the strobed noise feature is active, the noise

---

6. The strobed-noise feature of SpectreRF is also referred to as its time-domain noise feature.

FIGURE 8   *Noise produced by an inverter (n_{out}) as a function of the output signal (v_{out}). In region A the noise is dominated by the thermal noise of M_P, in region B its dominated by the thermal noise of M_N, and in region C the output noise includes the thermal noise from both devices as well as the amplified noise from the input.*
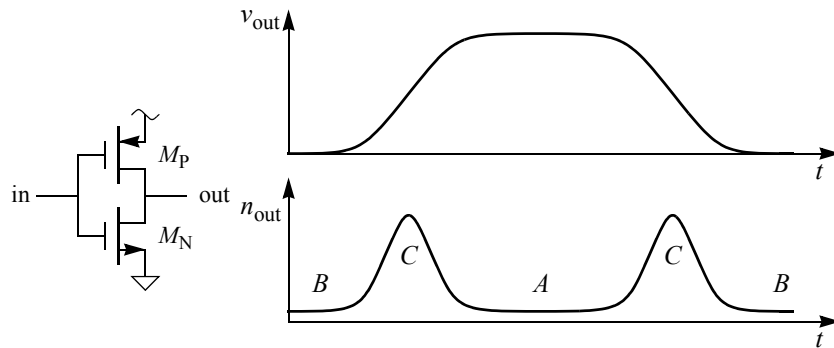
produced by the circuit is periodically sampled to create a discrete-time random sequence, as shown in Figure 9. SpectreRF then computes the power-spectral density of the sequence. The sample time should be adjusted to coincide with the desired threshold crossings. Since the T-periodic cyclostationary noise process is sampled every T seconds, the resulting noise process is stationary. Furthermore, the noise present at times other than at the sample points is completely ignored.

FIGURE 9   *Strobed noise. The lower waveform is a highly magnified view of the noise present at the strobe points in v_n, which are chosen to coincide with the threshold crossings in v.*
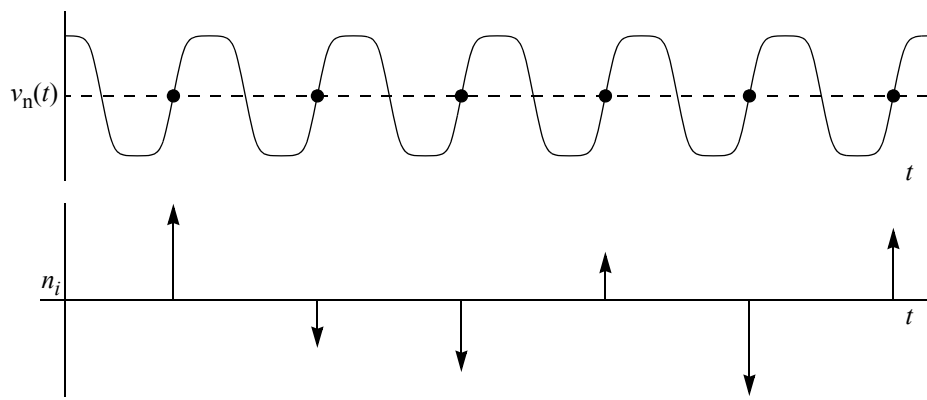
6.2 Converting to Phase Noise

The act of converting the noise from a continuous-time process to a discrete-time process by sampling at the threshold crossings makes the conversion into phase noise easier. If v_n is the continuous-time noisy response, and v is the noise-free response (response with the noise sources turned off), then[7]

$$n_i = v_n(iT) - v(iT). \tag{30}$$

Then if v_n is noisy because it is corrupted with a phase noise process ϕ, then

$$v_n(t) = v\left(t + \frac{\phi(t)}{2\pi f_0}\right). \tag{31}$$

Assume the phase noise ϕ is small and linearize v using a Taylor series expansion

$$v_n(t) \cong v(t) + \frac{dv(t)}{dt}\frac{\phi(t)}{2\pi f_0} \tag{32}$$

and

$$n_i \cong v(iT) + \frac{dv(iT)}{dt}\frac{\phi(iT)}{2\pi f_0} - v(iT) = \frac{dv(iT)}{dt}\frac{\phi(iT)}{2\pi f_0}. \tag{33}$$

Finally, ϕ_i can be found from n_i using

$$\phi_i = 2\pi f_0 n_i \Big/ \frac{dv(iT)}{dt}. \tag{34}$$

v is T periodic, which makes $dv(iT)/dt$ a constant, and so

$$S_\phi(f) = \left[2\pi f_0 \Big/ \frac{dv(iT)}{dt}\right]^2 S_n(f). \tag{35}$$

where $S_n(f)$ and $S_\phi(f)$ are the power spectral densities of the n_i and ϕ_i sequences.

6.3 Phase-Domain Model for Dividers

To extract the phase noise of a divider, drive the divider with a representative periodic input signal and perform a PSS analysis to determine the threshold crossing times and the slew rate (dv/dt) at these times. Then use SpectreRF's strobed PNoise analysis to compute $S_n(f)$. When running PNoise analysis, assure that the *maxsideband* parameter is set sufficiently large to capture all significant noise folding. A large value will slow the simulation. To reduce the number of sidebands needed, use T as small as possible. $S_\phi(f)$ is then computed from (35). Figure 10 shows the various attributes of the phase noise at the output of the divider. Notice that the noise is periodic in f with period $1/T$ because n is a discrete-time sequence with period T. The parameters n and f_c for the divider are extracted as illustrated. While shown in the figure, the high frequency roll-off always occurs well the operating frequency, and so except for flicker noise, the noise of a divider generally looks quite flat over frequency.

With ripple counters, one usually only characterizes one stage at a time and combines the phase noise from each stage by assuming that the noise in each stage is independent (true for device noise, would not be true for noise coupling into the divider from external sources). The variation due to phase noise accumulates, however it is necessary to account for the increasing period of the signals at each stage along the ripple counter. Consider an intermediate stage of a K-stage ripple counter. The total phase noise at the output of the ripple counter that results due to the phase noise S_{ϕ_k} at the output of stage k is $(T_k/T_K)^2 S_{\phi_k}$. So the total phase noise at the output of the ripple counter is

---

7. It is assumed that the sequence n_i is formed by sampling the noise at iT, which implies that the threshold crossings also occur at iT. In practice, the crossings will occur at some time offset from iT. That offset is ignored. It is done without loss of generality with the understanding that the functions v and v_n can always be reformulated to account for the offset.

FIGURE 10   *Extracting the noise parameters, n and f_c, for the divider from the power spectral density of its phase noise. The graph is plotted on a log-log scale.*
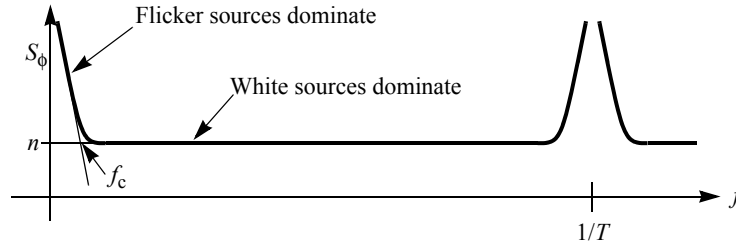
$$S_{\phi_{out}} = \frac{1}{T_K^2} \sum_{k=0}^{K} T_k^2 S_{\phi_k} \tag{36}$$

where S_{ϕ_0} and T_0 are the phase noise and signal period at the input to the first stage of the ripple counter.

With undesired variations in the supplies or in the substrate the resulting phase noise in each stage would be correlated, so one would need to compute the transfer function from the signal source to the phase noise of each stage and combine in a vector sum.

Unlike in ripple counters, phase noise does not accumulate with each stage in synchronous counters. Phase noise at the output of a synchronous counter is independent of the number of stages and consists only of the noise of its clock along with the noise of the last stage.

The phase-domain model for the divider, based on (2), is given in Listing 7. As before, one could use the *noise_table* function in lieu of the *white_noise* and *flicker_noise* functions to use the measured noise results directly.

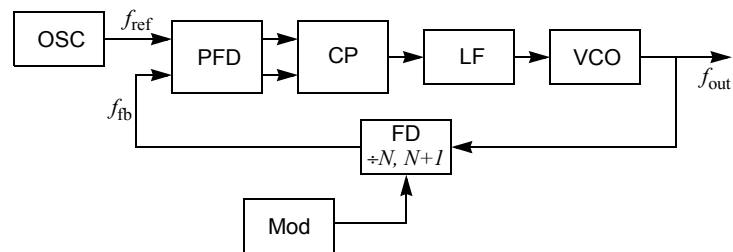LISTING 7   *Phase-domain divider noise model.*

```
`include "phase.vams"              // from Listing 2, includes disciplines.vams.

module divider(in, out);
input in; output out;
phase in, out;
parameter real ratio = 1 from (0:inf);    // divide ratio
parameter real n = 0 from [0:inf);        // white output phase noise (rads2/Hz)
parameter real fc = 0 from [0:inf);       // flicker noise corner frequency (Hz)

analog begin
    Theta(out) <+ Theta(in) / ratio;
    Theta(out) <+ white_noise(n, "wpn") + flicker_noise(n*fc, 1, "fpn");
end
endmodule
```

7 Fractional-*N* Synthesis

One of the drawbacks of a traditional frequency synthesizer, also known as an integer-*N* frequency synthesizer, is that the output frequency is constrained to be *N* times the reference frequency. If the output frequency is to be adjusted by changing *N*, which is constrained by the divider to be an integer, then the output frequency resolution is equal to the reference frequency. If fine frequency resolution is desired, then the reference frequency must be small. This in turn limits the loop bandwidth as set by the loop filter, which must be at least 10 times smaller than the reference frequency to prevent signal components at the reference frequency from reaching the input of the VCO and modulating the output frequency, creating spurs or sidebands at an offset equal to the reference frequency and its harmonics. A low loop bandwidth is undesirable because it limits the response time of the synthesizer to changes in *N*. In addition, the loop acts to suppress the phase noise in the VCO at offset frequencies within its bandwidth, so reducing the loop bandwidth acts to increase the total phase noise at the output of the VCO.

The constraint on the loop bandwidth imposed by the required frequency resolution is eliminated if the divide ratio *N* is not limited to be an integer. This is the idea behind fractional-*N* synthesis. In practice, one cannot directly implement a frequency divider that implements non-integer divide ratio except in a few very restrictive cases, so instead a divider that is capable of switching between two integer divide ratios is used, and one rapidly alternates between the two values in such a way that the time-average is equal to the desired non-integer divide ratio [23]. A block diagram for a fractional-*N* synthesizer is shown in Figure 11. Divide ratios of *N* and *N* + 1 are used, where *N* is the first integer below the desired divide ratio, and *N* + 1 is the first integer above. For example, if the desired divide ratio is 16.25, then one would alternate between the ratios of 16 and 17, with the ratio of 16 being used 75% of the time. Early attempts at fractional-*N* synthesis alternated between integer divide ratios in a repetitive manner, which resulted in noticeable spurs in the VCO output spectrum. More recently, ΔΣ modulators have been used to generate a random sequence with the desired duty cycle to control the multi-modulus dividers [22]. This has the effect of trading off the spurs for an increased noise floor, however the ΔΣ modulator can be designed so that most of the power in its output sequence is at frequencies that are above the loop bandwidth, and so are largely rejected by the loop.

FIGURE 11   *The block diagram of a fractional-N frequency synthesizer.*

The phase-domain small-signal model for the combination of a fractional-*N* divider and a ΔΣ modulator is given in Listing 8. It uses the *noise_table* function to construct a simple piece-wise linear approximation of the noise produced in an nth order ΔΣ modulator

that is parameterized with the low frequency noise generated by the modulator, along with the corner frequency and the order.

LISTING 8 *Phase-domain fractional-N divider model.*

```
`include "phase.vams"                    // from Listing 2, includes disciplines.vams.

module divider(in, out);
input in; output out;
phase in, out;
parameter real ratio = 1 from (0:inf);        // divide ratio
parameter real n = 0 from [0:inf);            // white output phase noise (rads2/Hz)
parameter real bw = 1 from (0:inf);           // ΔΣ modulator bandwidth
parameter integer order = 1 from (0:9);       // ΔΣ modulator order
parameter real fmax = 10∗bw from (bw:inf);    // maximum frequency of concern

analog begin
    Theta(out) <+ Theta(in) / ratio;
    Theta(out) <+ noise_table({
                    0,        n,
                    bw,       n,
                    fmax,     n∗pow((fmax/bw), 2∗order)
                }, "dsn"));
end
endmodule
```

8 Conclusion

A methodology for modeling and simulating the phase noise 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. This methodology is flexible enough to be used in a broad range of applications where phase noise is important.

8.1 If You Have Questions

If you have questions about what you have just read, feel free to post them on the *Forum* section of *The Designer's Guide Community* website. Do so by going to *www.designers-guide.org/Forum*. For more in depth questions, feel free to contact me in my role as a consultant at *ken@designers-guide.com*.

Acknowledgement

References

[1] Cadence Design Systems. SpectreRF simulation option. *www.cadence.com/data-sheets/spectrerf.html*.

[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, and J. Roychowdhury. Phase noise in oscillators: a unifying theory and numerical methods for characterization. *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*, vol. 47, no. 5, May 2000, pp. 655 -674.

[7] F. Gardner. *Phaselock Techniques*. John Wiley & Sons, 1979.

[8] W. Gardner. *Introduction to Random Processes: With Applications to Signals and Systems*. McGraw-Hill, 1989.

[9] Paul R. Gray and Robert G. Meyer. *Analysis and Design of Analog Integrated Circuits*. John Wiley & Sons, 1992.

[10] Emad Hegazi, Jacob Rael & Asad Abidi. *The Designer's Guide to High-Purity Oscillators*. Springer, 2004.

[11] F. Käertner. Determination of the correlation spectrum of oscillators with low noise. *IEEE Transactions on Microwave Theory and Technique*s, vol. 37, no. 1, pp. 90-101, Jan. 1989.

[12] F. X. Käertner. Analysis of white and $f^{-\alpha}$ noise in oscillators. *International Journal of Circuit Theory and Applications*, vol. 18, pp. 485–519, 1990.

[13] Kenneth S. Kundert. *The Designer's Guide to SPICE and Spectre*. Kluwer Academic Publishers, 1995.

[14] Kenneth S. Kundert. *The Designer's Guide to Verilog-AMS*. Kluwer Academic Publishers, 2004.

[15] Ken Kundert. Introduction to RF simulation and its application. *Journal of Solid-State Circuits,* vol. 34, no. 9, September 1999. Available from *www.designers-guide.org/analysis*.

[16] Ken Kundert. Modeling and simulation of jitter in phase-locked loops. In *Analog Circuit Design: RF Analog-to-Digital Converters; Sensor and Actuator Interfaces; Low-Noise Oscillators, PLLs and Synthesizers*, Rudy J. van de Plassche, Johan H. Huijsing, Willy M.C. Sansen, Kluwer Academic Publishers, November 1997.

[17] Ken Kundert. Modeling jitter in PLL-based frequency synthesizers. Available from *www.designers-guide.org/analysis*.

[18] Ken Kundert. Verification of bit-error rate in bang-bang clock and data recovery circuits. Available from *www.designers-guide.org/analysis*.

[19] Jri Lee, Kenneth S. Kundert, and Behzad Razavi. Analysis and modeling of bang-bang clock and data recovery circuits. *IEEE Journal of Solid-State Circuits*, vol. 39, no 9, September 2004, pp. 1571-1580.

[20] Joel Phillips and Ken Kundert. Noise in mixers, oscillators, samplers, and logic: an introduction to cyclostationary noise. *Proceedings of the IEEE Custom Integrated Circuits Conference*, CICC 2000. The paper and presentation are both available from *www.designers-guide.org*.

[21] J. J. Rael and A. A. Abidi. Physical processes of phase noise in differential LC oscillators. *Proceedings of the IEEE Custom Integrated Circuits Conference*, CICC 2000.

[22] T. A. D. Riley, M. A. Copeland, and T. A. Kwasniewski. Delta-sigma modulation in fractional-*N* frequency synthesis. *IEEE Journal of Solid-State Circuits*, vol. 28 no. 5, May 1993, pp. 553 -559

[23] Ulrich L. Rohde. *Digital PLL Frequency Synthesizers*. Prentice-Hall, Inc., 1983.

[24] G. Vendelin, A. Pavio, U. Rohde. *Microwave Circuit Design.* J. Wiley & Sons, 1990.

[25] *Verilog-AMS Language Reference Manual: Analog & Mixed-Signal Extensions to Verilog HDL*, version 2.1. Accellera, January 20, 2003. Available from *www.accellera.org*. An abridged version is available from *verilogams.com* or *www.designers-guide.org*.

[26] Verilog-A/MS. *verilogams.com*.

[27] D. Yee, C. Doan, D. Sobel, B. Limketkai, S. Alalusi, and R. Brodersen. A 2-GHz low-power single-chip CMOS receiver for WCDMA applications. *Proceedings of the European Solid-State Circuits Conference*, Sept. 2000.