Fast and Accurate Behavioral Simulation of Fractional-N Frequency Synthesizers and other PLL/DLL Circuits

Michael H. Perrott
Microsystems Technology Laboratory, MIT
perrott@mit.edu, http://www-mtl.mit.edu/~perrott

ABSTRACT

Techniques for fast and accurate simulation of fractional-N synthesizers at a detailed behavioral level are presented. The techniques allow a uniform time step to be used for the simulator, and can be applied to a variety of phase locked loop (PLL) and delay locked loop (DLL) circuits beyond fractional-N synthesizers, as well as to a variety of simulation frameworks such as Verilog and Matlab. Simulated results from a custom C++ simulator are shown to compare well to measured results from a prototype fractional-N synthesizer using a Σ-Δ modulator to dither its divide value.

Categories and Subject Descriptors

I.6.5 [**Simulation and Modeling**]: Model Development

General Terms

Algorithms

Keywords

fractional-N,frequency,synthesizer,sigma,delta,PLL,DLL

1. INTRODUCTION

Fractional-N frequency synthesizers provide high speed frequency sources that can be accurately set with very high resolution, which is of high value to many communication systems. Figure 1 illustrates a fractional-N synthesizer, which consists of a phase-frequency detector (PFD), charge pump, loop filter, voltage controlled oscillator (VCO), and a frequency divider that is dithered between integer values to achieve fractional divide ratios. This paper will focus on a class of fractional-N synthesizers known as Σ-Δ frequency synthesizers [12], for which the divide value is dithered according to the output of a Σ-Δ modulator [8].

Dithering of the divide value by the Σ-Δ modulator allows high frequency resolution to be achieved [12], but also has the negative side effect of introducing quantization noise that degrades the overall PLL noise performance. It is highly desirable to be able to simulate the effects of this quantization noise, along with other noise sources in the PLL shown in Figure 2, on the overall PLL performance. It is also desirable to simulate the dynamic response of the synthesizer in response to variations of the Σ-Δ input in order to evaluate stability and characterize the performance of the system when it is used as a transmitter [10].

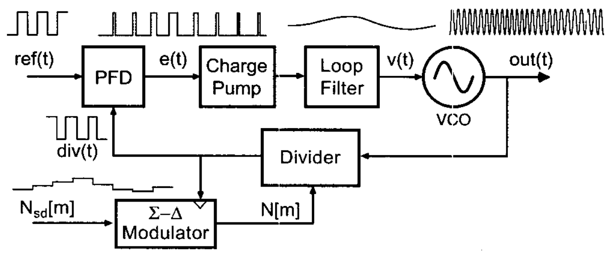

Figure 1: Σ-Δ synthesizer and associated signals.

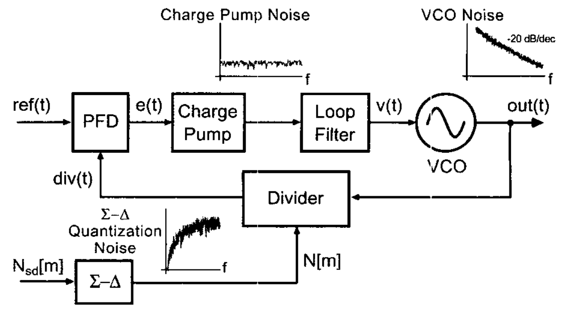

Figure 2: Spectral densities of PLL noise sources.

Simulation of fractional-N synthesizers is particularly challenging for a variety of reasons. First, the high output frequency of the synthesizer (often in the GHz range) imposes a high simulation sample frequency for traditional simulators. Unfortunately, the overall PLL dynamics have a bandwidth that is typically three to four orders of magnitude lower in frequency than the output frequency (often 100 kHz to 1 MHz bandwidth compared to a GHz output frequency). Thus, traditional simulators take a long time to compute the dynamic response of the system since many simulation samples are required. This is the classical problem that is encountered with the simulation of PLL circuits. For noise simulation, the fractional-N synthesizer adds the additional constraint that its behavior is non-periodic in steady-state due to the dithering action of the divide value, which prevents the use of methods developed for periodic steady-state conditions [7] as used with simulators such as SpectreRF.

In contrast to the above approaches, two techniques are presented in this paper that allow fast and accurate simulation of both dynamic and noise performance of fractional-N synthesizers at a detailed behavioral level. The first provides accurate representation of the continuous-time (CT) PFD output with a discrete-time (DT) sequence using an area conservation principle. The second allows a dramatic reduction of the simulation sample frequency, and therefore a longer sample period, by including the divider implementation in the VCO simulation module. Both of these methods allow a uniform time sample period to be used, and also allow non-iterative computation of the sample values of the various signals within the system. The uniform time sample period allows the results of the simulator to be readily examined in the frequency domain without resampling, and the non-iterative computation allows the technique to be easily used in mainstream simulators such as Verilog, VHDL, Matlab, and custom C/C++ programs.

An outline of the paper is as follows. Section 2 provides an overview of the discretization technique for representation of the CT PFD output with a DT sequence, and presents the corresponding mathematical analysis. Section 3 describes, at a high level, how the discretization technique can be implemented with the PFD described in terms of basic building blocks such as registers and logic gates. Section 4 focuses on the method of dramatically reducing the required simulation sample rate by combining the VCO and divider functions into one simulation block. Section 5 compares simulated results from a custom C++ simulator to measured results of an actual circuit implementation described in [10]. Finally, Section 6 concludes.

2. PFD DISCRETIZATION TECHNIQUE

For simulation based on uniform time sampling, a straightforward approach of converting the CT PFD output to a DT sequence is to apply the sampling operation shown in Figure 3 [6]. Unfortunately, this approach effectively quantizes the location of the PFD edges according to the simulation sample period, T_s. A reasonable assessment of the dynamic performance of the PLL can be achieved if T_s is made sufficiently small. However, the resulting quantization noise overpowers the true noise characteristics of the signals, and prevents proper noise analysis of the overall PLL.

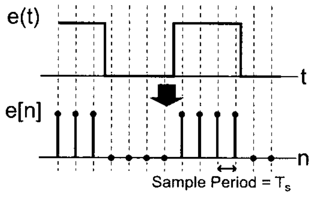

Figure 3: Classical uniform time sample method.

To solve the quantization noise issue, event-driven simulation methods have been developed for classical frequency synthesizers that align simulation samples precisely to the edges of the PFD output [2, 5, 1]. Although higher accuracy can be achieved with such methods, they are generally more complicated than uniform time sampling methods. Either closed-form calculation of the loop filter step response must be developed and then inserted into the simulation, or iterative methods, as used in SPICE or Verilog-A, must be incorporated into the simulator to calculate the loop filter response with varying time steps. The former approach is tedious and typically restricted to a low loop filter order, so that most of the recent methods focus on the latter approach [2, 5, 1]. In this case, the up-front work of the designer is minimized, but the simulation time is often longer due to the iterative calculations that are performed at each time step. Unfortunately, for either case, event-driven simulators have not yet been applied successfully to the noise analysis of fractional-N frequency synthesizers in which the divide value is dynamically varied.

In contrast to the above approaches, a constant time step method is proposed in this paper that applies an area conservation principle when converting the CT PFD output to the DT domain. This approach allows non-iterative computation of the loop filter dynamics by allowing them to be converted from CT to DT using either impulse invariance or bilinear transform methods [9]. Figure 4 illustrates an example of the resulting DT PFD signal, along with the corresponding DT loop filter impulse response. The charge pump is ignored in this analysis for simplicity; its effect can be included by simply scaling the PFD output by the value of the charge pump current. In the example, we see that the DT PFD output takes on values at its transitions that vary between 0 and 1 depending on the location of the transition edge. The DT version of the loop filter simply consists of a DT filter whose impulse response corresponds to samples of the CT impulse response of the loop filter, $h(t)$.

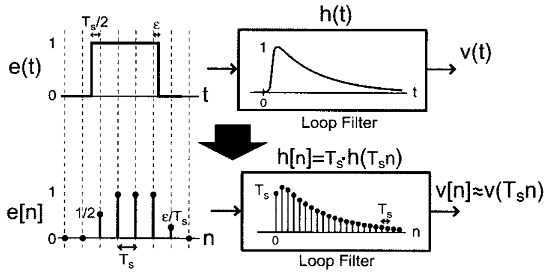

Figure 4: Proposed discretization method.

The discretization procedure is now discussed in more detail. As illustrated by Figure 5, we can view the CT PFD output, $e(t)$, as a series of rectangular pulses with height of one or zero and a width and time offset that varies according to the location of PFD edges. For rectangular pulses not associated with edges, the width corresponds to the sample period of the simulation, T_s. For rectangular pulses at edge boundaries, the width of the pulse varies between 0 and T_s as shown in the figure. In either case, these pulses look like impulses to the loop filter so that, from an intuitive standpoint, their influence can be characterized by two parameters — their area and time offset. Therefore, in line with this intuition, the corresponding DT PFD signal, $e[n]$, is chosen as samples that have amplitude proportional to the *area* of the respective rectangular pulse in that time sample interval. The area of each pulse corresponds to its associated timing parameter ϵ shown in the figure — the method of calculating ϵ for each pulse will be discussed in Section 4. It will be shown that the proposed discretization procedure yields highly accurate results, fast computation, and a simple implementation framework.

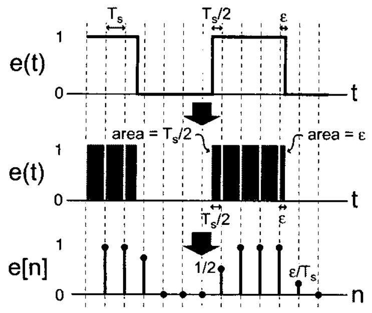

Figure 5: Details of PFD discretization technique.

To mathematically justify the technique, let us begin by specifying a notation for the rectangular pulses composing $e(t)$ that are shown in Figure 5, namely

$$\mathrm{rect}(t, \epsilon_k) = 1 \text{ for } -\epsilon_k/2 \leq t \leq \epsilon_k/2, \quad 0 \text{ elsewhere.}$$

We can describe the loop filter output as

$$v(t) = \sum_{k=-\infty}^{\infty} \mathrm{rect}(t - kT_s - \Delta t_k, \epsilon_k) * h(t), \qquad (1)$$

where $*$ denotes convolution, and Δt_k shifts its associated pulse in time according to the value of ϵ_k and whether the transition edge is rising or falling. Note that ϵ_k and Δt_k are constrained to $0 \leq \epsilon_k \leq T_s$ and $-T_s < \Delta t_k < 0$.

Taking the Fourier Transform of both sides of Equation 1, we obtain

$$V(jw) = \sum_{k=-\infty}^{\infty} e^{-jw(kT_s+\Delta t_k)} \frac{2\sin((\epsilon_k/2)w)}{w} H(jw), \qquad (2)$$

where $H(jw)$ is the loop filter frequency response.

One might be bothered that the Fourier Transform is being taken with sequences that are stochastic in nature, namely ϵ_k and Δt_k. This step is justified by noting that these sequences will be defined and finite in duration for a specific simulation run. However, we cannot infer statistical properties from Equation 2 or the analysis that follows.

Equation 2 can be simplified in light of the following assumptions:

- $H(jw)$ is a lowpass filter such that $|H(jw)| \approx 0$ for $|w| > w_o$,
- The sample frequency, $1/T_s$, is much higher than the bandwidth of $|H(jw)|$, so that $w_oT_s \ll 1$. Since $|\epsilon_k| < T_s$, we have $\sin((\epsilon_k/2)w_o) \approx (\epsilon_k/2)w_o$.

The second assumption is well justified in practice since it is typical for $w_oT_s \ll 1/100$. For instance, the author recommends sampling at a rate that is greater than a factor of 10 above the reference frequency, which is, in turn, at least a factor of 10 higher in frequency than the loop filter bandwidth in Hz, f_o, to achieve stable PLL dynamics [11]. In this case, $f_oT_s < 1/100$, so that $w_oT_s < 1/(2\pi 100)$.

Based on the above assumptions, Equation 2 is approximated as

$$V(jw) \approx \sum_{k=-\infty}^{\infty} \epsilon_k e^{-jw(kT_s+\Delta t_k)} H(jw).$$

The inverse Fourier Transform of the above expression is

$$v(t) = \sum_{k=-\infty}^{\infty} \epsilon_k h(t - kT_s - \Delta t_k). \qquad (3)$$

We are now ready to develop the DT model of the PFD/loop filter that we are seeking. We begin by sampling Equation 3:

$$v(nT_s) = \sum_{k=-\infty}^{\infty} \epsilon_k h((n-k)T_s - \Delta t_k). \qquad (4)$$

The above formulation requires nonuniform sampling of $h(t)$ due to the inclusion of Δt_k — it is preferable to remove this parameter if it can be shown that its influence is negligible. We will examine this issue using a specific form for $h(t)$, and then comment on the extension of the analysis for more general forms of $h(t)$.

Let us assume that the loop filter corresponds to a lead/lag network with transfer function of the form:

$$H(jw) = K\frac{jw + w_z}{jw(jw + w_o)}.$$

Using the method of partial fractions [9], it is straightforward to show that the corresponding loop filter impulse response is of the form

$$h(t) = K_1 e^{-w_o t}u(t) + K_2 u(t) = (K_1 e^{-w_c t} + K_2)u(t),$$

where $u(t)$ is the unit step (0 for $t < 0$, 1 for $t \geq 0$), and K_1 and K_2 are constant scale factors. Plugging the above expression into Equation 4, we obtain

$$v(nT_s) = \sum_{k=-\infty}^{\infty} \epsilon_k (K_1 e^{-w_o((n-k)T_s-\Delta t_k)} + K_2)u((n-k)T_s - \Delta t_k). \qquad (5)$$

We note that:

$$\begin{aligned} u((n-k)T_s - \Delta t_k) &= u((n-k)T_s) \\ &\quad \text{since } -T_s < \Delta t_k < 0, \end{aligned}$$

and

$$\begin{aligned} e^{-w_o((n-k)T_s-\Delta t_k)} &= e^{w_o\Delta t_k} e^{-w_o(n-k)T_s} \\ &\approx (1 + w_o\Delta t_k)e^{-w_o(n-k)T_s}. \end{aligned}$$

Therefore, the effect of the time shift operation by Δt_k has no influence on samples of $u(t)$, and only slightly modulates the amplitude of samples of the exponential response $e^{-w_o t}$. Although its effect could be incorporated into the numerical model, the author has found that it can be safely ignored given that two conditions are met. The first condition is that w_oT_s be much less than 1 so that $1 + w_o\Delta t_k \approx 1$. This condition is satisfied in practice; it was argued earlier in this section that we can typically expect that $w_oT_s \ll 1/100$. The second condition is that the sample rate of the simulator, $1/T_s$, be chosen as an integer multiple of the nominal frequency of the pulses associated with the CT PFD output. The effect of violating either of these conditions is the introduction of false spurs in the output phase noise of the synthesizer, as will be demonstrated in Section 5.

Given that the above conditions are satisfied, we can simplify Equation 5 as

$$v(nT_s) = \sum_{k=-\infty}^{\infty} \epsilon_k (K_1 e^{-w_o(n-k)T_s} + K_2) u((n-k)T_s),$$

so that we have

$$v(nT_s) = \sum_{k=-\infty}^{\infty} \frac{\epsilon_k}{T_s} T_s h((n-k)T_s). \tag{6}$$

Equation 6 is the conclusion of our effort, and matches the picture representation of the method illustrated in Figure 4 when $e[n] = \epsilon_n / T_s$.

Although the above analysis was performed for a simple lead/lag loop filter, higher order filters can be analyzed in similar fashion using the partial fraction expansion method. Specifically, high order filters have impulse responses that consist of a sum of exponentials, with each exponential corresponding to a distinct pole in the loop filter. The impact of e_k and Δt_k on each of these exponentials can be assessed in the same manner as derived above.

3. IMPLEMENTATION OF PFD

Now that it has been established that the PFD output signal can be accurately represented as a DT sequence using a principle of area conservation, let us examine the practical issue of representing the PFD topology in simulation code. As revealed by Figure 6, computation of $e[n]$ for a given PFD topology requires that transition information be passed along and processed by primitive elements such as registers and logic gates. Basic operations such as complementing signals must also be supported.

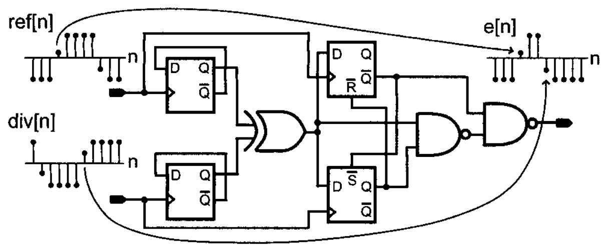

Figure 6: XOR-based PFD.

As illustrated in Figure 7, the complement operation is easily achieved as a sign change by slightly modifying the representation of $e[k]$ such that it alternates between -1 and 1 as opposed to 0 and 1. The new representation is achieved through the transformation $e[n] \Longrightarrow 2e[n] - 1$.

The transfer of transition information through primitives is illustrated in Figure 8 for a register and a representative logic gate, namely the 'and' gate. In the case of the register, the relevant timing information is contained in the clock signal. Specifically, whenever there is a transition at the output of the register, the location of that transition in time is set by the location of the rising (or falling) edge of the clock. As shown in the figure, this information is transfered to the register output by simply passing on the clk transition value when the output transitions in the same direction, and passing on the complement of the clk transition value when the output transitions in the opposite direction. In the case of the 'and' gate, either input can cause the output to transition. As gleaned from the figure, it is straightforward to determine which input is causing the transition, and appropriately pass its edge location value to the output of the 'and' gate. Similar arguments can be made for more complicated registers that include set and reset functions, and other primitives such as 'or' and 'xor' gates.

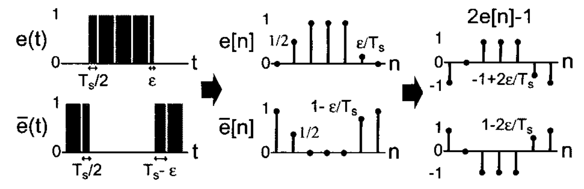

Figure 7: A better signal representation.

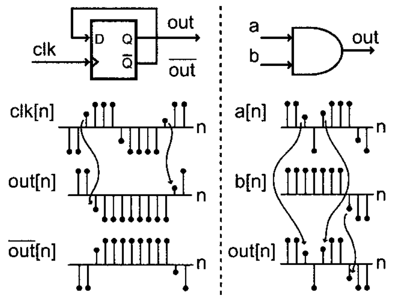

Figure 8: Example of register and logic gate signals.

4. VCO AND DIVIDER

Simulation of the VCO and divider portions of the PLL is now discussed, and a technique illustrated whereby the simulation sample period, T_s, can be set according to the reference frequency rather than the much higher VCO frequency. This technique typically allows more than two orders of magnitude speedup in simulation time of the PLL since the VCO frequency is typically more than two orders of magnitude higher than the reference frequency. Unlike a previous method that provided speedup for a PLL with no divider and a memoryless phase detector by modeling the VCO entirely in the phase domain [13], the presented technique accommodates fractional-N synthesizers that have dividers with dynamically varying value and digital PFD topologies as described in the previous sections. The key idea behind the technique is to combine the VCO and divider into one computation block.

To begin, let us define the phase of the VCO, $\Phi_{vco}(t)$, as the integral of its output frequency. Since the output frequency of the VCO is varied about its nominal frequency by its input voltage, we have:

$$\Phi_{vco}(t) = \int_{-\infty}^{t} 2\pi(K_v v(\tau) + f_c) d\tau + \Phi_{vn}(t), \tag{7}$$

where $v(t)$ is the VCO input voltage, K_v is the VCO gain (Hz/V), f_c corresponds to the nominal VCO frequency when $v(t) = 0$, and $\Phi_{vn}(t)$ is VCO noise as illustrated in Figure 2. To model a nonlinear relationship from input voltage to VCO frequency, the VCO input would be multiplied by a polynomial gain expression rather than just K_v. In general, $\Phi_{vco}(t)$ looks like a ramp in time, and rising edges of the VCO output occur each time it increments by 2π radians.

Simulation of the VCO is performed by simply discretizing Equation 7 as

$$\Phi_{vco}(nT_s) = \sum_{k=-\infty}^{n} 2\pi T_s(K_v v(kT_s) + f_c) + \Phi_{vn}(nT_s). \quad (8)$$

To prevent loss of information in the CT to DT conversion, $1/T_s$ must be higher than twice the highest frequency content of $v(t)$ and $\Phi_{vn}(t)$, as stated by the Nyquist theorem [9]. From a practical perspective, this condition will often be satisfied by meeting the sampling requirements for the PFD output. Choosing a sample rate such that $w_oT_s \ll 1/100$ is obviously sufficient for $v(t)$ since it is the output of the loop filter with bandwidth w_o rad/s. $\Phi_{vn}(t)$ is also bandlimited since it rolls off at -20 dB/decade, or more, before eventually hitting a low valued noise floor [3].

Rising edges of the divider output occur every $N[m]$ rising edges of the VCO output, where $N[m]$ corresponds to the instantaneous divide value. Therefore, as illustrated on the left side of Figure 9, the VCO phase, $\Phi_{vco}(t)$, completely specifies the location of the divider edges. As such, we can determine the value of ϵ_k at the transition points of the divider output based entirely on computed VCO phase, as shown on the right side of Figure 9 for first-order interpolation [2]. Note that the expressions assume that phase is wrapped every $2\pi N[m]$ radians. It suffices to choose a sample rate for the VCO phase computation according to the divider frequency, which equals the reference frequency, rather than the much higher VCO frequency.

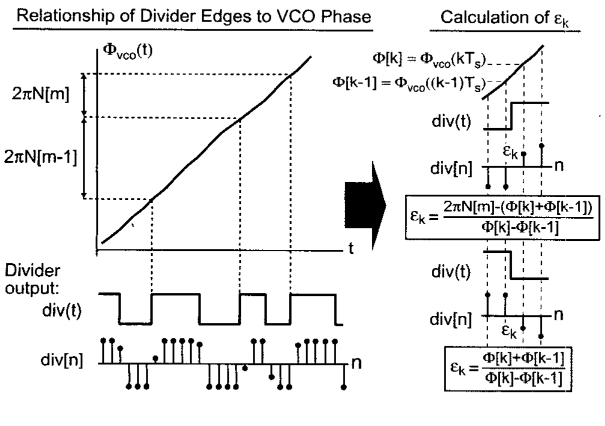
Relationship of Divider Edges to VCO Phase
Calculation of ϵ_k
$\Phi_{vco}(t)$
$2\pi N[m]$
$2\pi N[m-1]$
t
Divider output: div(t)
div[n]
n
$\Phi[k] = \Phi_{vco}(kT_s)$
$\Phi[k-1] = \Phi_{vco}((k-1)T_s)$
div(t)
div[n]
ϵ_k
$\epsilon_k = \frac{2\pi N[m]-(\Phi[k]+\Phi[k-1])}{\Phi[k]-\Phi[k-1]}$
$\epsilon_k = \frac{\Phi[k]+\Phi[k-1]}{\Phi[k]-\Phi[k-1]}$

Figure 9: VCO and divider signals.

5. RESULTS

The results of simulating the dynamic behavior and noise performance of a prototype synthesizer described in [10] using a custom C++ simulator employing the presented techniques are now presented, and simulated noise compared to measured results. Figure 1 provides a block diagram of the prototype system; the reader is referred to [10] for more details. Both dynamic and noise simulations will include the noise sources depicted in Figure 2, with VCO noise being input referred as a white noise source as described in [10]. Parameters associated with the noise sources are shown in Figure 10, which were computed from Hspice simulations and VCO measurements.

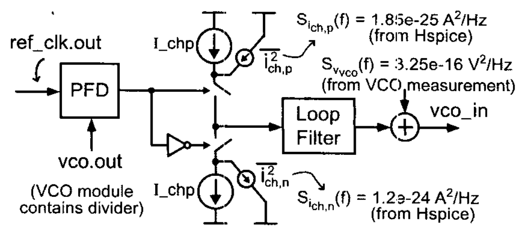
ref_clk.out
PFD
vco.out
(VCO module contains divider)
I_chp
I_chp
$\overline{i^2_{ch,p}}$
$\overline{i^2_{ch,n}}$
$S_{i_{ch,p}}(f)$ = 1.85e-25 A²/Hz (from Hspice)
$S_{v_{vco}}(f)$ = 3.25e-16 V²/Hz (from VCO measurement)
Loop Filter
vco_in
$S_{i_{ch,n}}(f)$ = 1.2e-24 A²/Hz (from Hspice)

Figure 10: Model of charge pump and VCO noise.

Relevant characteristics of the prototype include a reference frequency of 20 MHz, a VCO with f_c = 1.84 GHz and K_v = 30 MHz/V, a second order Σ-Δ modulator, a charge pump that outputs ±1.5 μamps, a nominal divide value of 92.3, a PFD topology as shown in Figure 6 [4], and a lead/lag filter with transfer function

$$H(jw) = \frac{1 + jw/(2\pi f_z)}{C_3 jw(1 + jw/(2\pi f_p))},$$

where

$$f_z = 11.6 \text{ kHz}, \quad f_p = 127.2 \text{ kHz}, \quad C_3 = 30\text{e-}12.$$

Figure 11 shows the measured synthesizer phase noise of the prototype taken from [10], along with the measured open loop VCO noise from which the input referred VCO noise variance in Figure 10 was computed.

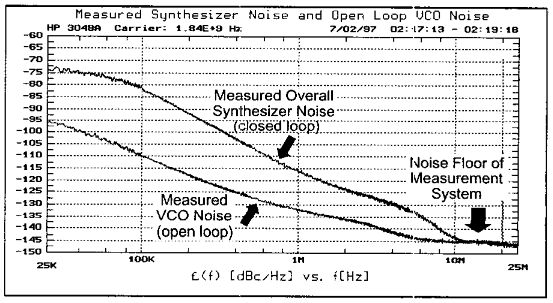

Figure 11: Measured synthesizer noise.

The simulation sample frequency was chosen as $1/T_s$ = 400 MHz, which is a factor of 20 higher than the reference frequency. The CT loop filter was converted to DT using the bilinear transform [9]. All simulations were run on a 650 MHz Pentium III laptop computer.

Beginning with dynamic behavior, Figure 12 shows the simulated VCO output frequency (constructed from the simulated VCO input) in response to variations at the input of the Σ-Δ modulator that include step and ramp functions. The step size is chosen to be large enough to knock the synthesizer out of frequency lock — the corresponding oscillations in the VCO output frequency are a result of cycle slipping before the VCO becomes frequency locked again. The subsequent ramp in divide value illustrates the high resolution of the synthesizer as its output frequency is varied over a 40 MHz range. For this simulation, 260 thousand time steps were computed in less than 5 seconds.

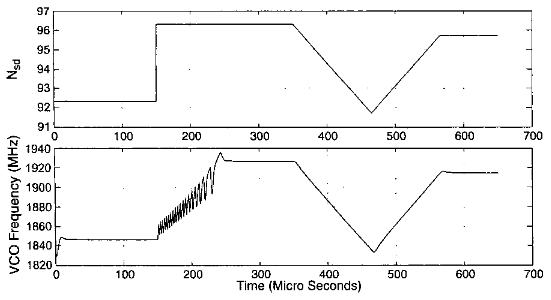

Figure 12: Simulated synthesizer dynamics.

Noise simulations of the prototype (constructed from the simulated VCO input) are shown in Figure 13 with the input to the Σ-Δ modulator being held constant. The top plot shows the simulated output noise spectral density with the simulation sample frequency, $1/T_s$, set to an integer multiple of the reference frequency, as recommended to reduce the effects of Δt_k in Equation 4. The bottom plot shows the impact of choosing $1/T_s$ to be a non-integer multiple of the reference frequency. We see that, in both cases, the simulated phase noise agrees quite well with measured results. The larger discrepancy at frequencies close to 100 kHz is probably due to non-ideal characteristics of the charge pump, such as duty cycle offset and transient dynamics, not being modeled. Such effects could be included within the given framework, but it is useful to observe that, despite ignoring such effects, the simulation results are still quite accurate for this prototype. As observed in the bottom plot, the impact of choosing $1/T_s$ to be a non-integer multiple of the reference frequency is that the reference spur at 20 MHz offset is aliased to other frequency values. This aliasing occurs due to the presence of harmonics above 402 MHz of the 20 MHz reference spur in the CT PFD output. For each simulation, 5 million time steps were computed in 80 seconds.

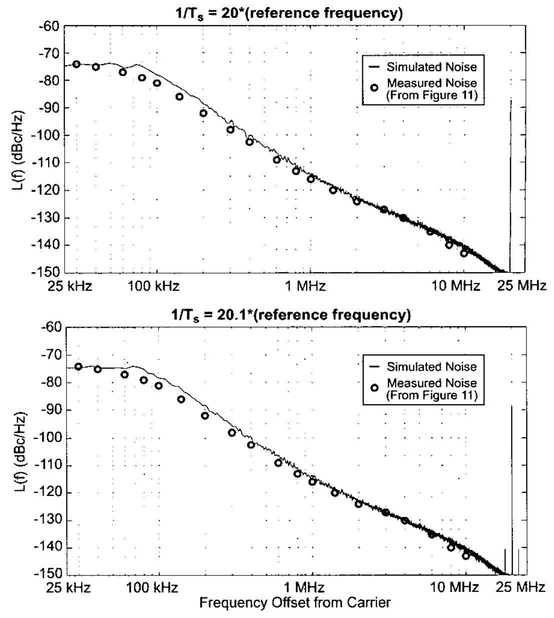

Figure 13: Simulated synthesizer phase noise.

6. CONCLUSION

Two techniques were presented in this paper that allow fast and accurate simulation of fractional-N synthesizers at a detailed behavioral level using a uniform time sample period. The first provides accurate representation of the CT PFD output with a DT sequence using an area conservation principle. The second allows a dramatic reduction of the simulation sample frequency by including the divider implementation in the VCO simulation module. The techniques were incorporated into a custom C++ simulator, which was used to simulate the dynamic and noise performance of a prototype Σ-Δ frequency synthesizer. The simulated noise performance was shown to agree quite well with measured results. The techniques can also be applied to other phase locked loop circuits, and be implemented in other simulation frameworks such as Verilog and Matlab.

7. REFERENCES

[1] B. De Smedt and G. Gielen. Nonlinear Behavioral Modeling and Phase Noise Evaluation in Phase Locked Loops. In *CICC*, pages 53–56, 1998.

[2] A. Demir, E. Liu, A. L. Sangiovanni-Vincentelli, and I. Vassiliou. Behavioral Simulation Techniques for Phase/Delay-Locked Systems. In *Custom Integrated Circuits Conference (CICC)*, pages 453–456, 1994.

[3] A. Hajimiri and T. Lee. A General Theory of Phase Noise in Electrical Oscillators. *IEEE Journal of Solid State Circuits (JSSC)*, 33(2):179–194, Feb. 1998.

[4] A. Hill and A. Surber. The PLL Dead Zone and How to Avoid It. In *RF Design*, pages 131–134, Mar. 1992.

[5] M. Hinz, I. Konenkamp, and E.-H. Horneber. Behavioral Modeling and Simulation of Phase-locked Loops for RF Front Ends. In *43rd Midwest Symp. on Circuits and Systems*, pages 194–197, 2000.

[6] D. Johns and K. Martin. *Analog Integrated Circuit Design.* Wiley, 1997.

[7] K. Kundert, J. White, and A. Sangiovanni-Vincentelli. *Steady-State Methods for Simulating Analog and Microwave Circuits.* Kluwer, Boston, 1990.

[8] S. Norsworthy, R. Schreier, and G. Temes. *Delta-Sigma Data Converters: Theory, Design, and Simulation.* IEEE Press, New York, 1997.

[9] A. V. Oppenheim and R. W. Schafer. *Discrete Time Signal Processing.* Prentice Hall, N.J., 1999.

[10] M. Perrott, T. Tewksbury, and C. Sodini. A 27 mW CMOS Fractional-N Synthesizer using Digital Compensation for 2.5 Mb/s GFSK Modulation. *JSSC*, 32(12):2048–2060, Dec. 1997.

[11] B. Razavi. *Monolithic Phase-Locked Loops and Clock Recovery Circuits: Theory and Design.* IEEE Press, New York, 1996.

[12] T. A. Riley, M. A. Copeland, and T. A. Kwasniewski. Delta-Sigma Modulation in Fractional-N Frequency Synthesis. *JSSC*, 28(5):553–559, May 1993.

[13] P. Van Halen and G. Boyle. SPICE-Compatible Behavioral Phase-Space Simulation Techniques for Phase-Locked Systems. In *38th Symposium on Circuits and Systems Conference*, volume 1, pages 53–56, 1996.