# Behavioral Modeling and Simulation of Jitter and Phase Noise in Fractional-N PLL Frequency Synthesizer

Xiaojian Mao
Tsinghua University, Beijing, PRC
maoxj00@mails.tsinghua.edu.cn

Huazhong Yang
Tsinghua University, Beijing, PRC
yanghz@tsinghua.edu.cn

Hui Wang
Tsinghua University, Beijing, PRC
wanghui@tsinghua.edu.cn

## ABSTRACT

A methodology is presented for predicting the phase noise and jitter of a fractional-N PLL based frequency synthesizer. Based on the phase/jitter properties extracted from transistor level through simulation, a voltage-domain behavioral model can give phase noise performance of fractional-N PLL frequency synthesizers in system level accurately, while the simulation efficiency is also improved by merging the VCO block operated at the highest frequency into those operated at lower frequency. Comparing to phase-domain simulation, the improved voltage-domain models do a better job of capturing the details of the behavior of the loop, details such as the signal capturing and escaping traces in fractional-N frequency synthesizer.

## 1. INTRODUCTION

Phase-locked loop (PLL) based frequency synthesizers are widely used in all kinds of circuits. The settling time and frequency resolution are two trade-off key figure-of-merits. The settling time is related to the bandwidth of the loop, and to make the loop stable, the loop bandwidth should be less than 1/10 of the reference frequency. So when trying to get quicker settling, the reference frequency should be improved. In another way, in the integer-N PLL frequency synthesizer, the frequency output is integer times of the reference, and the finer is the output frequency, the lower reference frequency is needed. To relax the contradiction between them, the fractional-N frequency synthesizer (Figure. 1) is carried out. The fractional-N divider is realized by means of changing the divider ratio between two integers to get an average fractional divider ratio. Normally the changing of divider ratio will import excess spur and phase noise in the output of the loop. Sigma-delta modulation technique is proposed by Riley et al. in [1], and Miller and Conley in [2] to shaping the noise imported by dynamic divider ratio. Now this method has been widely used in a variety of applications ranging from accurate frequency generation to direct frequency modulation in transmitters [3].

Any jitter or phase noise in the output of the PLL used in these applications generally degrades the performance

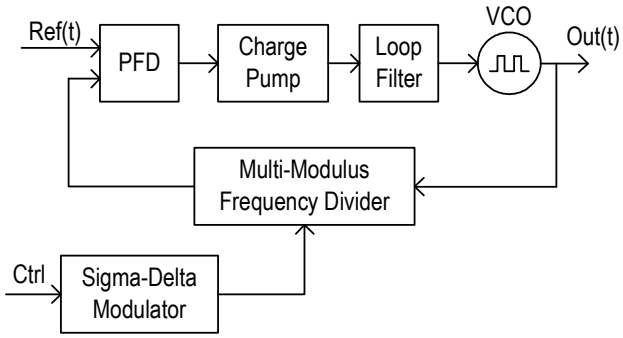


**Figure. 1 Sigma-Delta Fractional-N Frequency Synthesizer**

margins of the system in which it resides and so is of great concern to the designers of such systems. It is difficult to predict the phase noise in traditional circuit simulators [4]. SPICE [5] is useless in this situation as it can only predict the noise in circuits that have a quiescent (time-invariant) operating point. SpectreRF [6] can be used for PLL phase noise analysis in some condition, but there is a basic requirement for SpectreRF simulation that the circuits should have a periodic steady state solution. Obviously, this request can not be meted because the sigma-delta modulation is just to randomize the system and avoid periodic spurs.

As regard to behavioral model for phase noise or jitter simulation, there are two different modeling methods for phase noise analysis in PLLs. One is the popular linear phase-domain models [3, 4]. In the simplest case, these models are linear and analyzed easily in the frequency domain, making it simple to use the model to predict phase noise, even in the presence of flicker noise or other noise sources that are difficult to model in the time domain. The other is voltage-domain model, which formulates the models in terms of voltage, and the noise is added into the circuit in the form of jitter [4]. Using the model, we should get the phase property of building blocks of PLL first, and then using the approach proposed by Demir for simulating PLLs whereby a PLL is described using behavioral models simulated at a high level [7,8]. This approach allows prediction of PLL jitter behavior once the noise behavior of the blocks has been characterized. Generally, the phase-domain models are considerably more efficient, but the voltage-domain models do a better job of capturing the

details of the behavior of the loop, details such as the signal capturing and escaping traces [4].

In Kundert's paper [4], the relevant ideas of Demir were adapted to allow use of a commercial simulator, Spectre [6], and an industry standard modeling language, Verilog-A [9].

In fact, Kundert has given a comprehensive description of phase-domain and voltage-domain model in [4]. But when mention to fractional-N PLL frequency synthesizer, only the phase-domain model available. And this is the case in the other published papers [3] as the authors' read range. This is because the synthesizer's high output frequency (often in the gigahertz range) imposes a high simulation sample frequency for simulators. Unfortunately, the overall PLL dynamics have a bandwidth that is typically three to four orders of magnitude lower than the output frequency (often 100 kHz to 1 MHz of bandwidth compared to 1 to 10 GHz for the output frequency). Thus, simulators take a long time to compute the system's dynamic response, because they must process many simulation samples. In the integer-N PLL frequency synthesizer, using the property of the unchanged divide ratio, Kundert [4] proposed an approach to merging the VCO and frequency divider and lower the highest frequency of the system to same as the input reference frequency. Thus the simulation efficiency can be confirmed. However the fixed divider is not the case in fractional-N frequency synthesizers, Kundert's approach could not be applied in this case. In this paper, we will propose a tip, similar to the method in [4], to solve this problem, to provide PLL designers a voltage-domain model of the fractional-N frequency synthesizer and to make the PLL design process more efficient.

In section 2 the base concepts are reviewed. The proposed method will be discussed in section 3, and in section 4, we will carry out an experiment for the behavioral model for the phase noise and jitter in fractional-N frequency synthesizer. Section 5 is conclusion.

## 2. BASIC CONCEPTS

### 2.1 Jitter and Phase Noise

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. Displacing the time variable $t$ in a noise-free signal $v(t)$ by a stochastic process $j(t)$, we model jitter in the signal as follows,

$$v_n(t) = v(t + j(t)) \tag{1}$$

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

Generally, there are two measurements of jitter, *period jitter* and *long term jitter*. When define $t_i$ as the *i*-th transition time of signal $v(\cdot)$, $T_k = t_{k+1} - t_k$ is said to be the period of the period, and the *period jitter* of $v(\cdot)$ is the standard deviation of $T_k$ ,

$$J_{period} = \sigma(t_{k+1} - t_k) = \sqrt{\mathrm{var}(t_{k+1} - t_k)} \tag{2}$$

the *long term jitter* is :

$$J_{long-term} = \sigma(t_{i+k} - t_k) = \sqrt{\mathrm{var}(t_{i+k} - t_k)} \tag{3}$$

Now we reformulate $j(\cdot)$ as phase, and then (1) is rewritten as

$$v_n(t) = v\left(t + \frac{\phi(t)}{2 \cdot \pi \cdot f_c}\right) \tag{4}$$

where $\phi(t) = 2 \cdot \pi \cdot f_c \cdot j(t)$ is in radians and $f_c = 1/T$ . And the power spectral density $S_\phi(f)$ of *Noise Phase* $\phi(t)$ is another measure of jitter.

$S_\phi(f)$ is conceptually useful, but not directly measurable because $\phi(t)$ is not an observable quantity. The other common measure is $S_v(f)$, the power spectral density of $v(\cdot)$, 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 by the power of $V_1$, the first harmonic of $v(\cdot)$. Here is the definition of *Phase Noise*

$$L(f_m) = \frac{S_v(f_c + f_m)}{2 \cdot V_1^2} \tag{5}$$

where $f_m = f - f_c$ .

The relation between $S_v(f)$ and $L(f_m)$ [4] is approximate to

$$L(f_m) \approx \frac{1}{2} S_\phi(f_m) \tag{6}$$

Phase noise and jitter are two related quantities associated with a noisy oscillator. Phase noise is a frequency-domain view of the noise spectrum around the oscillator signal, while jitter is a time-domain measure of the timing accuracy of the oscillator period. If we know one of them, the other can be derived. In the behavioral model of PLL, the jitter is one of the major measurements used, and the output phase noise will be carried out through (6).

### 2.2 Jitter in the Blocks of PLL

As mentioned in [4], there are two types of blocks in PLL system, 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, of *PM jitter*. Autonomous blocks, such as the OSC and VCO, exhibit frequency modulation, or *FM jitter*.

PM jitter is synchronous, and has not memory effect. The noisy signal of PM jitter can be expressed like this:

$$v_n(t) = v\left(t + j_{PM}(t)\right) \tag{7}$$

When referring to the *long-term jitter* of the signal, the simple PM jitter is,

$$j_i = \sqrt{2 \cdot \mathrm{var}\left(j_{PM}\right)} \tag{8}$$

Now if we consider an oscillator with *period jitter*:

$$J = \sqrt{a \cdot T} \tag{9}$$

From [4] we can find the relationship of jitter and signal phase noise is

$$L(f_m) = \frac{1}{2} \frac{a \cdot f_c^2}{a^2 \cdot \pi^2 \cdot f_c^4 + f_m^2} \tag{10}$$

### 2.3 Jitter after an N-Divider

If we consider a signal with frequency $f_1$ after an $N$ divider, the output signal will be at the frequency of $f_2 = f_1/N$. Since the divider outputs one pulse for every $N$ pulses 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

$$J_2 = \sqrt{N} J_1 \tag{11}$$

And now we can see that if we want to get the jitter of signal at high frequency, we observe jitter of signal after an ideal divider, and using (11), we can get what we need. The phase noise has following relation,

$$S_{\phi 1} = N^2 S_{\phi 2} \tag{12}$$

## 3. PROPOSED BEHAVIORAL MODEL FOR FRACTIONAL-N PLL SYNTHEISER

The behavioral model of integer-N PLL frequency synthesizers with jitter has been discussed in [4]. Here we just give a brief description. Jitter is modeled in Verilog-A 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.

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. This approach is suitable for any model that exhibits PM jitter and generates discrete-valued outputs. The frequency divider and PFD/CP are belonging to this kind of blocks. A part of Verilog-A program file importing PM jitter is shown in Listing. 1:

**Listing. 1 PFD/CP Model with PM Jitter, Ref to [10]**

```
`include "discipline.h"
`include "constants.h"

module pfd_cp (out, ref, vco);
input ref, vco; output out; electrical ref, vco, out;
parameter real Iout=100u;
parameter integer dir=1 from [–1:1] exclude 0;
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);

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*$dist_normal(seed,0,1); // PM jitter
   end
   @(cross(V(vco), dir, ttol)) begin
      if (state < 1) state = state + 1;
      dt = 0.707*jitter*$dist_normal(seed,0,1);
   end
   I(out) <+ transition(Iout*state, td + dt, tt); // PM jitter is added.
end
endmodule
```

The delay argument of the *transition()* function can not 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 the following Listing. 2

At every output transition, the next transition is scheduled using the *timer()* function to be $T/K + J\,\delta/\sqrt{K}$ in future, where $\delta$ is the unit-variance zero-mean random process and $K$ is the number of output transitions per period. Typically, $K = 2$

.

**Listing. 2 Fixed Frequency Oscillator with FM Jitter, Ref to [10]**

```
`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); // FM Jitter
         next = next + 0.5/freq + 0.707*dT; // Next Transition Updated
        end
        V(out) <+ transition(n ? Vhi : Vlo, 0, tt);
end
endmodule
```

A VCO model is similar to fixed frequency oscillator above, expect that the interval between two adjacent transitions is determined by control voltage and jitter.

When all the building blocks are constructed, we can use the Cadence Analog Environment to carry out simulation.

### 3.1 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 number of activities. It only affects the timing of particular activities. However, if jitter causes two events that would normally occur at the same time to be displaced so that they are no longer coincident, circuit simulators will have to use more time points to resolve the distinct events and will run more slowly. For this reason, it is desirable to combine jitter sources to the degree possible.

On the other hand, in PLL system, the frequency of output signal is much higher than the reference. And the crossing transitions are so quick that cost most of the simulation time. When the output reaches to GHz, to confirm the enough observing time, the simulation time is unacceptable.

In the integer-N PLL frequency synthesizer, this problem can be solved through including PM jitter into the reference fixed frequency oscillator, and merging the frequency divider to VCO [4]. The former tip can be easy to understand, and the later one can be shown in the left of Figure. 2 and the rationality will be explained in the last part of Section 2.

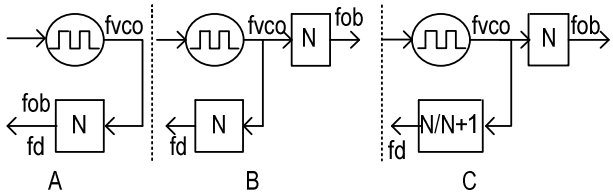


**Figure. 2 Merging Frequency Divider to VCO**

In fractional-N frequency synthesizers, the divider ratio changes according to the sigma-delta modulator and the emerging becomes workless [10] and we seem to have to spend a mass of time to simulate them. In fact the emerging has two roll. One is to produce divided signal to PFD for comparing, the other is to act as a probe for observing (Figure. 2 B). Because the divider ratio is fixed, and this two role can combine together just like Figure. 2 A. When we turn to fractional-N frequency synthesizer, the two roles will be realized in two parts (Figure. 2 C). And now we can simulate a fractional-N frequency synthesizer in relatively lower frequency and can get the expectant result through the similar simulation points. The realization of the kind of emerging is shown in the following Listing. 3

## 4. SIMULATION EXPERIMENTS

These ideas have been applied to model and simulate a fractional-N frequency synthesizer. The synthesizer is chosen with a reference frequency of $f_{REF} = 12MHz$, an output frequency of $f_{OUT} = 879.6MHz$, and a divider rate of $N = 73.3$. The bandwidth of 3rd order loop filter is $f_{lf} = 100KHz$ .The fractional-N is realized through a multi-modulus divider controlled by a 3-rd sigma-delta modulator. All the blocks are described in Verilog-A and

**Listing. 3 VCO + Multi-Modulus Frequency Divider**

```
`include "discipline.h"
`include "constants.h"
module Maoxj_vco_fd (outdiv,ctrl,in);
input in, ctrl; output outdiv;
electrical outdiv, ctrl, in;

parameter real N=1 from (0:inf);
parameter real Vlo=-1, Vhi=1;
parameter real tt=0.01*N/Fmax from (0:inf);
parameter real jitter=0 from [0:0.25*N/Fmax);
parameter real ttol=1u*N/Fmax from (0:N/Fmax);
parameter real outStart=0 from [0:inf);

real freq, phase, dT, delta, prev, Voutdiv, Next1,Next2;
integer seed,fp;
analog begin
        @(initial_step) begin
                seed = -561;dT = 0;
                prev = $abstime;
                Voutdiv = Vlo;
                fp = $fopen("./Maoxj_vco_fd.txt");
                Next1 = (N+floor(V(ctrl)+0.5))/N; Next2 = 1;
                 delta = jitter * sqrt(2*N);
        end
        // compute the freq from the input voltage
        freq = (V(in) - Vmin)*(Fmax - Fmin) / (Vmax - Vmin) + Fmin;
        if (freq > Fmax) freq = Fmax;
        if (freq < Fmin) freq = Fmin;
        // apply the frequency divider, add the phase noise
        freq = freq/N;
        freq = freq/(1 + dT*freq);
        // phase is the integral of the freq modulo 1
        phase = idt(freq,0);
        // Cross Up for Frequency Divider Output
        @(cross(phase-Next1, +1, ttol)) begin
                Next1=Next1+(N+floor(V(ctrl)+0.5))/N;
                Voutdiv = Vhi;
        end
        // Cross Up for oscillograph // Update Jitter
        @(cross(phase-Next2, +1, ttol)) begin
                Next2=Next2+1;
                dT = delta * $dist_normal(seed, 0, 1);
        end
        // Cross Down for Frequency Divider Output
        @(cross(phase-Next1+(N+floor(V(ctrl)+0.5))/2/N,+1,ttol))
        begin  Voutdiv = Vlo;
        end
        // Cross Down for oscillograph    // Update Jitter
        @(cross(phase-Next2+0.5, +1, ttol)) begin
                if ($abstime >= outStart) begin
                        $fstrobe( fp,"%0.20e ", $abstime - prev);
                end
                prev = $abstime;
                dT = delta * $dist_normal(seed, 0, 1);
        end
             @(final_step ) begin $fclose(fp); end
      V(outdiv) <+ transition(Voutdiv, 0, tt);
end
endmodule
```

shown in Figure. 3.

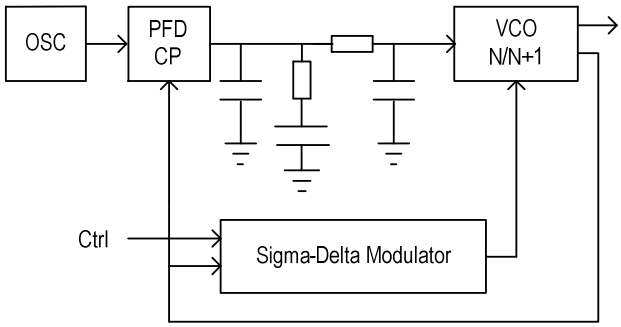


**Figure. 3 A 4-order Fractional-N PLL Frequency Synthesizer**

The noise of OSC is -140dBc/Hz at 10KHz, which corresponds to a jitter of 34fs. At center frequency 878MHz, the noise of VCO is -100dBc/Hz at 100KHz, and the period jitter is 34fs. And the PM jitters of the other blocks about 200fs. The simulation has been carried out on a Sun SpracUltra450 workstation with a CPU with frequency 450MHz. It takes 19 minutes to run a simulation without noise for 24 thousands reference cycles. When including all noise sources, the time cost is same because the runtime is determined by the number of activities and the noise sources of our behavioral model do not generate any extra activities. However, the SparcUltra450 always runs into core dump problem if the behavioral model with jitter does not take the method proposed in this paper. The simulation result without jitter is shown in Figure. 4, and Figure. 5 shows the result which includes all jitter sources. The output noise feature matches to phase domain results excellently.

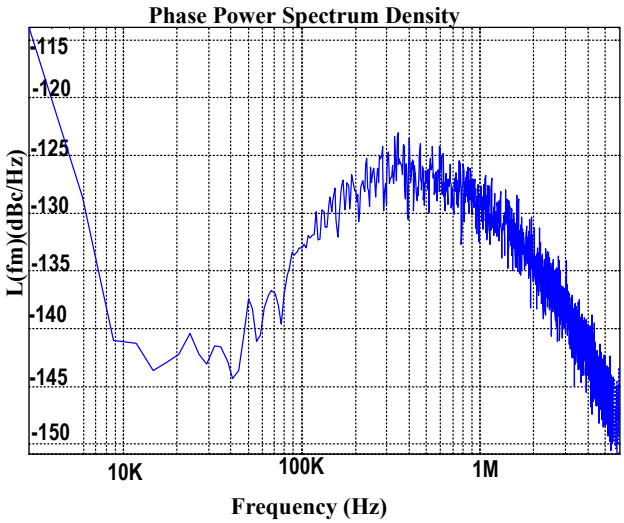


**Figure. 4 Output Phase Noise due to Sigma-Delta Modulator**

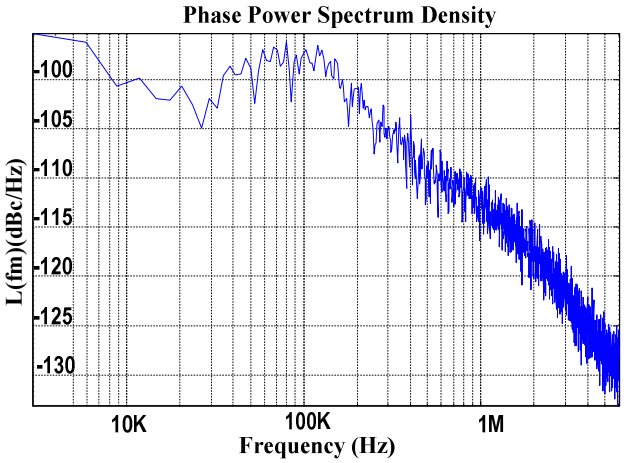


**Figure. 5 Output Phase Noise Including All Noise Sources**

## 5. CONLUSION

An improved methodology for modeling and simulating the phase noise and jitter performance of fractional-N phase-locked loops was presented. This method compensates Kundert's realization in [4, 10] to fractional-N frequency synthesizers. 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.

## REFERENCES

[1] T. A. Riley, M. A. Copeland, and T. A. Kwasniewski, "Delta-sigma modulation in fractional-N frequency syntheis," *IEEE J. Solid State Circuits*, vol. 28, pp. 553-559, May 1993.

[2] B. Miller and B. Conley, "A multiple modulator fractional divider," *IEEE Trans. Instrum. Meas.*, vol. 40, pp. 578-593, June 1991.

[3] M. H. Perrott, M. D. Trott, and C. G. Sodini, "A Modeling Approach for $\Sigma-\Delta$ Fractional-N Frequency Synthesizers Allowing Straightforward Noise Analysis," *IEEE J. Solid State Circuits*, vol. 37, pp. 1028-1038, August 2002.

[4] K. Kundert, "Predicting the Phase Noise and Jitter of PLL-Based Frequency Synthesizers," Available from *www.designers-guide.com*, May, 2003.

[5] http://bwrc.eecs.berkeley.edu/Classes/IcBook/SPICE/

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

[7] 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.

[8] 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.

[9] *Verilog-A Language Reference Manual: Analog Extensions to Verilog-HDL*, version 1.0. Open Verilog International, 1996. Available from *www.eda.org/verilogams*.

[10]K. Kundert, "Modeling and Simulation of Jitter in PLL Frequency Synthesizers," Available from *www.designers-guide.com*.