NTNU – Trondheim
Norwegian University of
Science and Technology

Python Framework for Design and Simulation of Integer-N ADPLLs

Cole Nielsen

Electronic Systems Design, Specialization Project

Submission date:	December 2019
Supervisor:	Trond Ytterdal, IET
Co-supervisor:	Carsten Wulff, IET

Norwegian University of Science and Technology
Department of Electronic Systems

Abstract.

An open source Python-language design automation and simulation framework for the design of integer-N all digital phase locked loop (ADPLL) frequency synthesizers is presented in this paper. The framework enables (1) automatic design of optimized second order discrete time digital loop filters, and (2) behavioral ADPLL simulation to verify loop filter and PLL designs for satisfactory phase noise, lock-time and stability; optionally subject to variation using Monte-Carlo sampling. Simulation is implemented with a discrete-event time domain simulator utilizing behavioral PLL component models, permitting accurate modeling of effects of time-discretization and quantization in an ADPLL. Loop filter design automation is based upon a prototype second order filter, whose parameters are optimized to minimize total integrated output phase noise for a PLL provided specifications for reference frequency, divider ratio, time-to-digital converter (TDC) resolution, digitally controlled oscillator (DCO) gain, oscillator phase noise characteristics and maximum lock time. The optimizer utilizes a continuous phase transfer function approximation of ADPLL dynamics and phase noise, allowing for faster design iteration than with direct discrete time simulation, but also requires conversion of the designed filters from continuous-to-discrete time. A post-filter design optimizer is furthermore presented in this work to map the discrete-time filters into optimal fixed-point representations that are implementable in digital hardware, with quantization noise and filter error due to finite word effects minimized.

Problem description.

The intent of this project is to develop a standalone PLL design and simulation framework to aid and facilitate a later master's thesis project regarding the design of an all-digital, ultra-low power PLL frequency synthesizer. This framework is intended to address and ease all-digital PLL design and simulation challenges at a high level. Specifically it should enable speedy PLL simulation defined with system and component- level specifications (e.g. desired frequency, gain of digitally controlled oscillator, phase detector resolution, divider ratio). This is to allow for development of component level specifications and verification of PLL performance (phase noise, lock time, stability) under ideal circumstances before transistor level implementation, thus accelerating overall implementation time for hardware.

Contents

List of Figures

List of Tables

Abbreviations.

ADPLL	All digital phase locked loop
AR	Autoregressive
BBPD	Bang-bang phase detector
BW	Bandwidth
CI	Confidence interval
CLK	Clock
CMOS	Complementary metal oxide semiconductor
DCO	Digitally controlled oscillator
DFT	Discrete Fourier transform
DIV	Divider
FFT	Fast Fourier transform
GSS	Golden section search
IIR	Infinite impulse response
KDCO	DCO Gain
LF	Loop filter
LSB	Least significant bit
MMSE	Minimum mean squared error
MSE	Mean squared error
NF	Noise figure
OTW	Oscillator tuning word
PD	Phase detector
PI	Proportional-integral
PID	Proportional-integral-derivative
PLL	Phase locked loop
PM	Phase margin
PSD	Power spectral density
PVT	Process, voltage and temperature
RMS	Root mean squared
SFDR	Spurious free dynamic range
TDC	Time to digital converter
TF	Transfer function
VCO	Voltage controlled oscillator

1 Introduction

Phase locked loops are extraordinarily useful frequency synthesizers that are vital to the operation of virtually all wired and wireless communication systems of today. The trend towards increasingly lower power wireless devices poses an acute need to reduce PLL power consumption. This is a challenge as PLLs typically rank among the highest power consuming components of a radio, and are necessarily so to limit oscillator phase noise. A sampling of literature on ultra-low power 2.4GHz radios finds oscillator power consumption as a portion of total radio consumption to be 53% for the receiver in [1], 88% of the transmitter in [2], 52% of the transmitter in [3], and 50% of the receiver in [4]. Reducing analog PLL power consumption can be a prohibitive challenge as the performance of analog loop filters degrade as a result of unavoidably lower charge pump current. However, recent CMOS process nodes with minimum gate lengths as small as 7nm allow for all-digital loop filters and PLLs to be a possible alternative to analog designs due to increasingly low power consumption associated with their implementation. Digital loop-filters have the unique advantage where they can be scaled indefinitely as process nodes advance, suffering no loss in performance, while also having greatly reduced sensitivities to process, voltage, temperature (PVT) variations compared to analog implementations.

All-digital PLLs (ADPLLs) introduce new challenges in the process of design in the ultra-low power domain. Low power design is complemented by low complexity design, which in a PLL transcribes to low resolution of phase detectors, digitally tunable oscillators, and loop filter digital data paths. In other words, effects of quantization are strong. Quantization is an inherently nonlinear process, thus strong quantization is tied to strong nonlinear effects. Consequently, where high resolution digital PLLs with low quantization effects can be effectively analyzed with linear-time invariant transfer functions in the Z-domain, simulation and numerical analysis is necessary to accurately capture quantization effects in low power, low resolution ADPLLs. This, of course, presents a challenge in manual loop filter design of PLLs if simulation is an integral part of the most basic modeling, and thus motivates the creation of an automated design solution. Currently, no openly available PLL design framework automates loop filter design for the needs of ultra low power all-digital PLLs as characterized here.

Thus, in this paper, a new framework, `pllsim`, written in Python[1] is introduced (this framework is available on GitHub[2]), which uniquely addresses issues of ultra-low power ADPLL design. Specifically, design of integer-N type PLLs is focused on, as the impetus of this work is an integer-N PLL design project. Topics presented are (a) automatic design and optimization of ADPLL loop filters given target system and component level specifications for the PLL, and (b) behavioral time domain PLL simulation for accurate analysis and verification of loop filter and PLL performance, with an integrated Monte-Carlo sampling variation analysis engine. Due

[1]Python Software Foundation `https://www.python.org/`.
[2]`pllsim` codebase: `https://github.com/nielscol/pllsim`.

to high phase noise associated with low power design, the optimization approach introduced in this paper focuses on the minimization of total integrated phase noise power to allow for maximum PLL performance on a given power budget.

A brief outline of the paper is as follows. An introduction to PLL theory is in section 2. Simulation and optimization methods are discussed in sections 3, 4, 5 and 6. An example design exercise with the framework, a comparison to existing solutions, and general discussion considerations for using the framework are in section 7. Finally, section 8 concludes.

The main contributions of this design framework to PLL design are:

1. Fully automatic loop filter design and optimization for all digital integer-N PLLs in an open source Python framework.
2. Generation of ready-for-hardware digital filter coefficients from high level PLL specifications (maximum lock time, reference frequency, DCO gain, divider ratio, oscillator phase noise).
3. Filter design optimization considering minimization of total phase noise, lock time and quantization errors in the final digital loop filter computed.
4. An integrated behavioral time domain simulator coupled with the loop filter designer allowing for verification of automatically designed filters for lock time and phase noise performance.
5. Included parametric sweep and variational analysis in the simulator to verify process tolerance ranges and statistical distributions of PLL performance with process variation.

2 Theory

In its most basic form, a phase locked loop is a phase-based feedback system whose output tracks or maintains a fixed phase relationship to an input signal. As will be shown, such a system is uniquely suited to the task of frequency synthesis, which is the process of generating derivative frequencies from some reference frequency. Given reference and output phase signals Φ_{ref} and Φ_{out}, a PLL can be modeled as in figure 1, with feedforward and feedback networks A(s) and B(s).

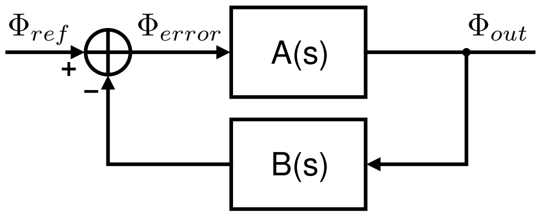


Figure 1: Basic phase feedback network.

The closed loop phase response T(s) for Φ_{ref} to Φ_{out} is therefore:

$$\mathrm{T}(s) = \frac{\Phi_{out}(s)}{\Phi_{ref}(s)} = \frac{A(s)}{1 + A(s)B(s)} \tag{1}$$

A particular case of interest is when B(s) = 1/N, where N is a constant, and the loop gain L(s) = A(s)B(s) $>>$ 1. The closed loop response for this case is:

$$\frac{\Phi_{out}(s)}{\Phi_{ref}(s)} \approx \frac{A(s)}{A(s)B(s)} = \frac{1}{B(s)} = N \tag{2}$$

We see that the phase through the PLL is multiplied by a factor of N. If the input phase signal is a sinusoid with frequency ω_{ref}, and likewise the output with ω_{out}, then $\phi_{ref}(t) = \omega_{ref}t$ and $\phi_{out}(t) = \omega_{out}t$. Accordingly:

$$\frac{\Phi_{out}(t)}{\Phi_{ref}(t)} = \frac{\omega_{out}t}{\omega_{ref}t} \approx N \quad \rightarrow \quad \omega_{out} \approx N\omega_{ref} \tag{3}$$

Therefore, it is observed that a PLL allows for the generation, i.e. synthesis, of a new frequency from a reference frequency signal. Given a feedback divider ratio of 1/N, the PLL multiplies the reference frequency by a factor of N. In the following sections, more advanced models for PLL will be developed, extending the concept introduced here. Specifically, the theory of digital, discrete-time PLLs will be developed and extended from a continuous phase model of a basic PLL.

2.1 Continuous PLL Model

Although this work is interested in discrete time sampling PLL design, it will later be seen that continuous models can still be of use in the analysis and design of such systems. Thus a continuous PLL model is developed in this section.

2.1.1 PLL Synthesizer Architecture

The traditional architecture for implementing a PLL frequency synthesizer [5] is shown in figure 2. This basic PLL is comprised of four components: (1) a phase detector, denoted by PD, (2) a loop filter, denoted by H$_{LF}$(s), (3) a voltage controlled oscillator, denoted by VCO, and (4) and phase divider, denoted by "$\div$ N" in the figure. These components are explained in the following sections.

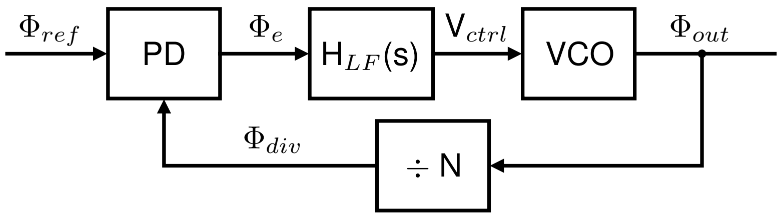


Figure 2: Basic PLL.

2.1.2 Divider

The phase divider is used as the feedback path in the PLL, where the division ratio N controls the phase and associated frequency multiplication of the PLL. The transfer function of the divider is:

$$\mathrm{H}_{div}(s) = \frac{\Phi_{div}(s)}{\Phi_{out}(s)} = \frac{1}{\mathrm{N}} \tag{4}$$

2.1.3 Phase Detector

The phase detector is used to measure the phase of the feedback signal (i.e. divider output) in relation to the reference phase, in order to establish a phase error signal Φ_e used to control the tuning of the PLL.

$$\Phi_e(s) = \Phi_{ref}(s) - \Phi_{div}(s) \tag{5}$$

2.1.4 Loop Filter

The PLL loop filter is used to control the phase-frequency response of PLL, which affects transient PLL behavior, as well as phase noise performance (see section 2.3). This can be designed to have P poles and Z zeros, and can be represented in the canonical form of equation 6 as a rational function of polynomials of s with coefficients given with $\{a_0, ..., a_P\}$ and $\{b_0, ..., b_Z\}$.

$$\mathrm{H}_{LF}(s) = \frac{\sum_{j=0}^{Z} b_j s^j}{\sum_{k=0}^{P} a_k s^k} \tag{6}$$

2.1.5 Voltage Controlled Oscillator

The voltage controlled oscillator (VCO) is an oscillator with frequency controlled by an input signal V_{ctrl}. The VCO is characterized by its gain $K_{VCO} = \partial f / \partial \mathrm{V}_{ctrl}$, and the nominal oscillation frequency f_0. Analyzed in terms of phase, an oscillator can be seen as a time-phase integrator:

$$\Phi_{VCO}(t) = \Phi_{out}(t) = \int 2\pi(K_{VCO}\mathrm{V}_{ctrl}(t) + f_0)\mathrm{dt} \tag{7}$$

In the s-domain, frequency offsets from steady state can be represented via initial conditions for modeling purposes. The VCO transfer function is therefore:

$$\mathrm{H}_{VCO}(s) = \frac{\Phi_{VCO}(s)}{\mathrm{V}_{ctrl}(s)} = \frac{\Phi_{out}(s)}{\mathrm{V}_{ctrl}(s)} = \frac{2\pi K_{VCO}}{s} \tag{8}$$

2.1.6 Continuous PLL Transfer Function

Now that the continuous PLL synthesizer is understood at a component level, the closed loop dynamics of the PLL can be analyzed. First the PLL loop gain L(s) is determined:

$$\mathrm{L}(s) = \mathrm{H}_{LF}(s)\mathrm{H}_{VCO}(s)\mathrm{H}_{div}(s) = \frac{2\pi K_{VCO}}{\mathrm{N}} \frac{1}{s} \frac{\sum_{j=0}^{Z} b_j s^j}{\sum_{k=0}^{P} a_k s^k} \tag{9}$$

With the phase detector as the feedback summation point, the closed loop response of the PLL from reference to output is in equation 10.

$$\mathrm{T}(s) = \frac{\Phi_{out}(s)}{\Phi_{ref}(s)} = \frac{2\pi K_{VCO} \sum_{j=0}^{Z} b_j s^j}{\sum_{k=0}^{P} a_k s^{k+1} + \frac{2\pi K_{VCO}}{\mathrm{N}} \sum_{j=0}^{Z} b_j s^j} = \mathrm{N}\frac{\mathrm{L}(s)}{1+\mathrm{L}(s)} \tag{10}$$

2.2 ADPLL - All Digital, Discrete Time PLL Model

Based on the continuous PLL theory, a model for a discrete time sampled all digital PLL (ADPLL) can be adapted. The general approach here utilizes the bilinear transformation [6] to bridge between continuous s-domain models to discrete z-domain models. To ensure translation from s-to-z domains with negligible granularity effects due to time-sampling, a constraint for PLL loop sampling frequency commonly cited in PLL literature from [7] is imposed here. This is to constrain the loop sampling frequency $f_s > 10 \cdot \mathrm{BW}_{loop}$, where BW_{loop} is the PLL closed loop bandwidth.

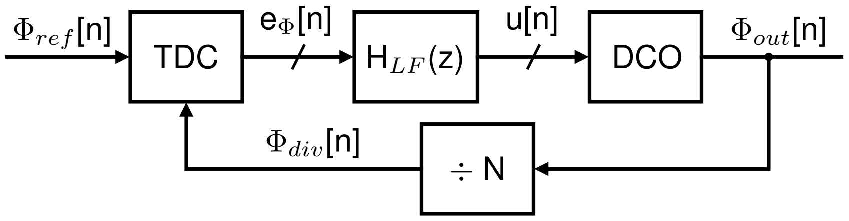


Figure 3: Basic ADPLL.

A basic architecture for implementing an all digital PLL is shown in figure 3, based from [8][9], which replaces the phase detector of continuous PLL in figure 2 with a time to digital converter (TDC). Furthermore, the loop filter $\mathrm{H}_{LF}(s)$ is replaced with a discrete-time loop filter $\mathrm{H}_{LF}(z)$, and the VCO with a digitally controlled oscillator (DCO). In this architecture, all components are fully or partially digital in implementation.

2.2.1 Divider

A digital phase divider (also called frequency divider) is commonly implemented with a modulo-N counter which counts oscillator cycles [10]. With a divider ratio N, the output of the divider will have an active edge transition (considered to be rising edge as shown in figure 4) every N input cycles. Phase information is inferred from the output edge timing, which occurs with time interval N/f_{osc}, and is equal to the point at which output phase equals a multiple of 2π. Thus a digital divider does not provider continuous phase information, but rather a sampled phase signal with rate f_{osc}/N.

For PLL transfer function modeling, a digital divider behaves identically to the continuous case:

$$\Phi_{div}[n] = \frac{\Phi_{out}[n]}{\mathrm{N}} \tag{11}$$

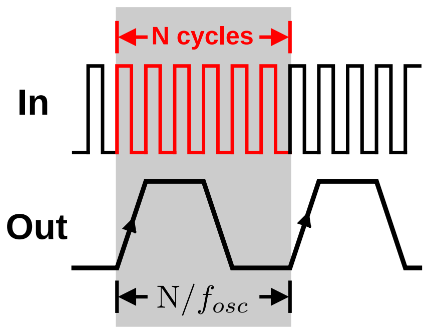


Figure 4: Digital divider signals.

In the z- and s-domain the divider transfer model is:

$$\mathrm{H}_{div}(z) = \mathrm{H}_{div}(s) = \frac{\Phi_{div}(z)}{\Phi_{out}(z)} = \frac{1}{\mathrm{N}} \tag{12}$$

2.2.2 Time to Digital Converter

A time to digital converter (TDC) measures the the difference Δt in arrival of two input signal [11], which in the ADPLL synthesizer is the differential between the divider signal Φ_{div}[n] and the reference frequency signal Φ_{ref}[n]. With a known reference frequency f_{ref}, the measured time differential Δt corresponds to a phase difference $\Delta\Phi = 2\pi f_{ref}\Delta t$, thus the TDC acts as a phase detector. Being digitized, a TDC will have limited resolution in time (phase), defined here to be M steps per reference cycle. Figure 5 shows the basic TDC phase detector model architecture. This has a minimum step size in time of $\Delta t_{step} = 1/M f_{ref}$. To map the output to digitized value, the model applies a scale factor M/2π and floor rounding, so one least significant bit (LSB) of the output $e_\Phi[n]$ equates to Δt_{step} timing error between inputs Φ_{div}[n] and Φ_{ref}[n]. The described TDC model in sampled-time equation form is in equation 13.

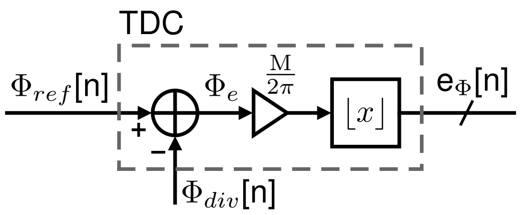


Figure 5: TDC model.

$$e_\Phi[n] = \left\lfloor \frac{\mathrm{M}}{2\pi}(\Phi_{ref}[n] - \Phi_{div}[n]) \right\rfloor \tag{13}$$

The TDC z-domain representation is equation 14. For purposes of PLL loop gain calculation, the TDC transfer function is in equation 15, which accounts for phase-to-digital domain conversion gain. Effects of quantization will be handled separately in section 2.3.

$$e_{\Phi}(z) = \frac{\mathrm{M}}{2\pi}(\Phi_{ref}(z) - \Phi_{div}(z)) \tag{14}$$

$$\mathrm{H}_{TDC}(z) = \mathrm{H}_{TDC}(s) = \frac{\mathrm{M}}{2\pi} \tag{15}$$

2.2.3 Discrete Time Loop Filter

The discrete-time loop filter design will be derived from the continuous canonical loop filter (equation 6) via application of a s-to-z domain transformation. The bilinear transform [6] allows for such conversion from continuous transfer function to discrete representation. In the case presented in this work, the loop filter sampling rate f_s=f_{ref} is constrained to be much greater than the PLL loop bandwidth ($f_s > 10 \cdot \mathrm{BW}_{loop}$). Thus, a simpler transformation is permissible, derived through Taylor series approximation of the definition of complex variable z=$re^{s\Delta T}$ [12] for values on the unit circle, i.e. r=1. This will be referred to as the approximate bilinear transform in this work. Given the $1/\Delta T_s$=f_s as the relation for sampling rate, then:

$$\begin{aligned}
z^{-1} &= e^{-s\Delta T_s} && \text{(definition of z on unit circle)}\\
&= \sum_{k=0}^{\infty} \frac{(-s\Delta T_s)^k}{k!} && \text{(exponential Taylor series)}\\
&\approx 1 - s\Delta T_s && (\text{if } |s\Delta T_s| = 2\pi \mathrm{BW}_{loop} \cdot \Delta T_s << 1)
\end{aligned}$$

Thus the s-to-z and z-to-s identities for the approximated bilinear transform are:

$$z^{-1} = 1 - s\Delta T_s \tag{16}$$

$$s = \frac{1}{\Delta T_s}(1 - z^{-1}) \tag{17}$$

Applying equation 17 to the general loop filter of equation 6 yields the z-domain loop filter:

$$\mathrm{H}_{LF}(z) = \mathrm{H}_{LF}(s)|_{s=\frac{1}{\Delta T_s}(1-z^{-1})} = \left.\frac{\sum_{j=0}^{Z} b_j s^j}{\sum_{k=0}^{P} a_k s^k}\right|_{s=\frac{1}{\Delta T_s}(1-z^{-1})} \tag{18}$$

$$= \frac{\sum_{j=0}^{Z} \frac{b_j}{\Delta T_s^j}(1 - z^{-1})^j}{\sum_{k=0}^{P} \frac{a_k}{\Delta T_k}(1 - z^{-1})^k} \tag{19}$$

Equation 19 is transformed into a digitally implementable form by reorganizing into the canonical representation of equation 20, which then determines the tap coefficients for the sampled-time difference equation in equation 21.

$$\mathrm{H}_{LF}(z) = \frac{\sum_{j=0}^{P} b_j^{'} z^{-j}}{1 + \sum_{k=1}^{Z} a_k^{'} z^{-k}} \tag{20}$$

$$y[n] = -\sum_{k=1}^{P} a_k^{'} y[n-k] + \sum_{j=0}^{Z} b_j^{'} x[n-j] \tag{21}$$

The obtained difference equation is directly implementable in digital hardware with a direct form-I IIR filter [13] shown in figure 6. The filter coefficients $\{a_1^{'}, ..., a_P^{'}\}$ and $\{b_0^{'}, ..., b_Z^{'}\}$ must be quantized into finite resolution fixed point words for a complete digital implementation. The delay elements (z^{-1} blocks) are implementable digitally as registers, the coefficient gains are implementable with array multipliers, and the adders are implementable with digital adders. Effects of quantization will be discussed in section 2.3.

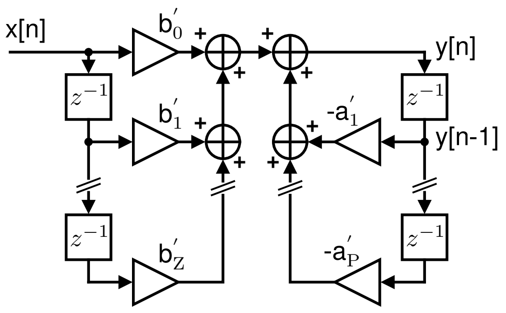


Figure 6: Direct form I implementation of IIR filter.

2.2.4 Digitally Controlled Oscillator

The digitally controlled oscillator (DCO) varies from a VCO by only accepting a digital frequency tuning signal, called the oscillator tuning word (OTW). A DCO is modeled in discrete time as a recursive phase integrator, dependent on (a) the DCO gain K_{DCO}, equal to the frequency tuning of the oscillator per LSB of the OTW, (b) the current state of the OTW u[n], and (c) the PLL sampling period ΔT_s=f_{ref}^{-1}.

$$\Phi_{out}[n] = \Phi_{out}[n-1] + 2\pi K_{DCO} u[n] \Delta T_s \tag{22}$$

Application of the z-transform yields equation 23, and successive application of the bilinear transform to the DCO transfer function yields the continuous approximation of equation 24.

$$\mathrm{H}_{DCO}(z) = \frac{\Phi_{out}(z)}{u(z)} = \frac{2\pi K_{DCO} \Delta T_s}{1 - z^{-1}} \tag{23}$$

$$\mathrm{H}_{DCO}(s) = \frac{\Phi_{out}(s)}{u(s)} = \frac{2\pi K_{DCO} \Delta T_s}{1 - (1 - s\Delta T_s)} = \frac{2\pi K_{DCO}}{s} \tag{24}$$

2.2.5 Discrete Time PLL Transfer Function

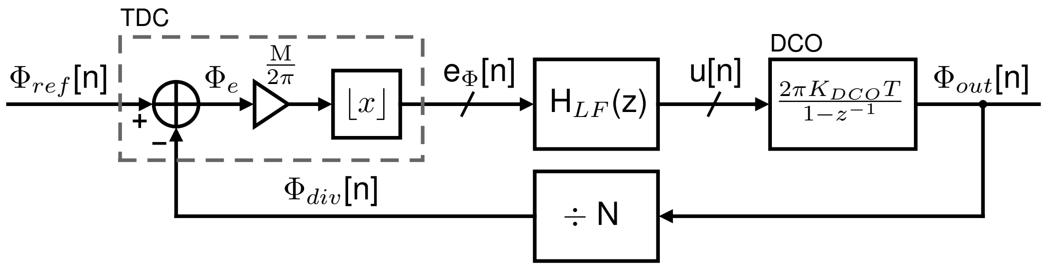


Figure 7: Discrete time PLL model.

The transfer function for the full discrete-time PLL of figure 7 can be computed in the z-domain, and also approximated continuously. The open loop z-domain transfer function is:

$$\mathrm{L}(z) = \mathrm{H}_{TDC}(z)\mathrm{H}_{LF}(z)\mathrm{H}_{DCO}(z)\mathrm{H}_{DIV}(z) = \frac{\mathrm{M}}{\mathrm{N}}\frac{K_{DCO}\Delta T_s}{(1-z^{-1})}\frac{\sum_{j=0}^{Z} b_j(1-z^{-1})^j}{\sum_{k=0}^{P} a_k(1-z^{-1})^k} \tag{25}$$

The closed loop z-domain PLL phase transfer function is:

$$\mathrm{T}(z) = \frac{\Phi_{out}(z)}{\Phi_{ref}(z)} = \frac{\mathrm{M}K_{DCO}\Delta T_s \sum_{j=0}^{Z} b_j(1-z^{-1})^j}{\sum_{k=0}^{P} a_k(1-z^{-1})^{k+1} + K_{DCO}\Delta T_s \frac{\mathrm{M}}{\mathrm{N}} \sum_{j=0}^{Z} b_j(1-z^{-1})^j} \tag{26}$$

The s-domain approximation of the transfer function is:

$$\mathrm{L}(s) = \mathrm{H}_{TDC}(s)\mathrm{H}_{LF}(s)\mathrm{H}_{DCO}(s)\mathrm{H}_{DIV}(s) = \frac{\mathrm{M}}{\mathrm{N}}\frac{K_{DCO}}{s}\frac{\sum_{j=0}^{Z} b_j s^j}{\sum_{k=0}^{P} a_k s^k} \tag{27}$$

And in closed loop configuration the s-domain PLL phase transfer function is:

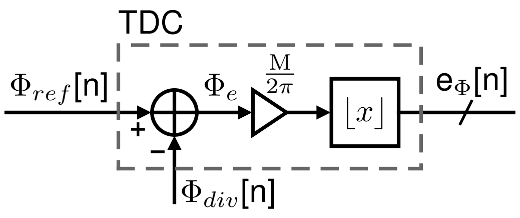
$$\mathrm{T}(s) = \frac{\Phi_{out}(s)}{\Phi_{ref}(s)} = \frac{\mathrm{M}K_{DCO}\sum_{j=0}^{Z} b_j s^j}{\sum_{k=0}^{P} a_k s^{k+1} + \frac{\mathrm{M}}{\mathrm{N}} K_{DCO}\sum_{j=0}^{Z} b_j s^j} = \mathrm{N}\frac{\mathrm{L}(s)}{1+\mathrm{L}(s)} \tag{28}$$

2.3 ADPLL Noise Model

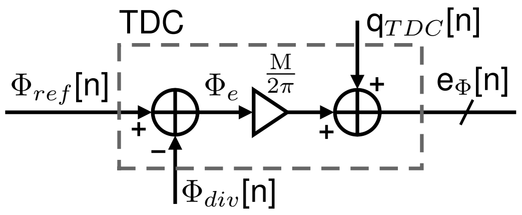
The noise in the discrete-time ADPLL results from quantization and from stochastic sources. Quantization results from round off errors introduced in the digitization of PLL components (in the TDC, loop filter, and DCO). Stochastic noise results from thermal and flicker noise in the PLL components when considered in an analog viewpoint (present in the DCO, divider and TDC). The noise generated by these quantization sources will be discussed in the following sections.

2.3.1 TDC Noise

The predominant noise source of within a TDC is due to quantization. This source will dominate overall noise within a PLL's closed loop bandwidth [14][15][16], so its consideration is critical for PLL performance. As will be seen, a straightforward approach to model quantization noise is to treat it as an additive signal component as shown in figure 8b.



(a)



(b)

Figure 8: (a) TDC Model, **(b)** TDC additive noise model.

Suppose the quantized signal e_Φ[n] out of the TDC can be represented as the sum of its unquantized representation $\Phi_e \frac{M}{2\pi}$ with a quantization error signal $\mathrm{q}_{TDC}[n]$, as in figure 9.

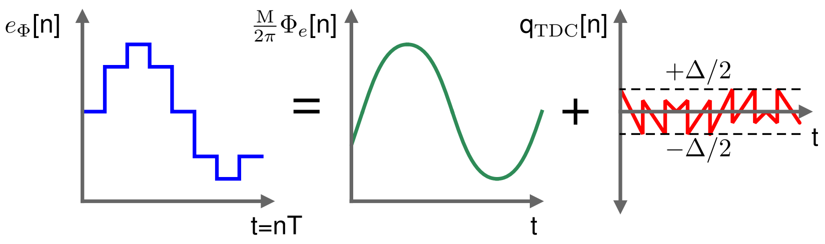


Figure 9: Quantization as via additive error signal.

Based on [17], if the quantization noise signal $\mathrm{q}_{TDC}[n]$ is uncorrelated to unquantized signal $\Phi_e \frac{M}{2\pi}$, it has the statistical property that it is uniformly distributed in the range $[-\Delta/2, \Delta/2]$, i.e. $P_q(Q = q) = \mathrm{U}(-\Delta/2, \Delta/2)$. Δ is the quantization step size. The variance, or power, of the TDC quantization noise signal is:

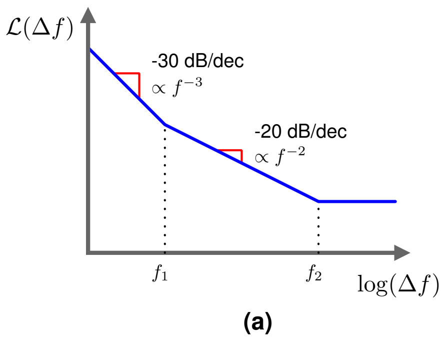
$$\sigma^2_{q_{TDC}} = \int_{-\infty}^{\infty} q^2 P_q(Q = q) dq = \int_{-\Delta/2}^{\Delta/2} \frac{q^2}{\Delta} dq = \frac{\Delta^2}{12} \tag{29}$$

Since e_Φ[n] is a digital signal, the quantization step size is Δ=1 LSB. The TDC quantization noise power is therefore $\sigma^2_{q_{TDC}} = 1/12\,\mathrm{LSB}^2$. If the quantization noise is presumed to be white, and the TDC sampling rate is f_{ref}, the quantization noise power spectral density (PSD) is then given in equation 30.

$$S_{qn_{TDC}}(f) = \frac{P_{q_{TDC}}}{\Delta f} = \frac{\sigma^2_{q_{TDC}}}{f_{ref}} = \frac{\Delta^2}{12 f_{ref}} = \frac{1}{12 f_{ref}} \quad \frac{[\mathrm{LSB}]^2}{[\mathrm{Hz}]} \tag{30}$$

2.3.2 DCO Noise

Noise in a DCO is resulting from (a) quantization of the oscillator tuning word u[n], and (b) from thermal and stochastic sources. In the digital PLL, the OTW quantization occurs in the loop filter, so this will be analyzed in the later loop filter section (2.3.4). Thus oscillator thermal/stochastic noise will be considered, based on Leeson's model for oscillator phase noise [18]. Leeson's model considers noise power density at an offset Δf from the oscillator tone (carrier). Noise power density is represented with the function $\mathcal{L}(\Delta f)$, which is the noise power density normalized to the power of the oscillator carrier tone, in other words in units of dBc/Hz. Leeson's model divides phase noise into three regions, illustrated in figure 10a: (1) flicker-noise dominated, with a slope of -30 dB/decade, (2) white frequency-noise dominated, with -20 dB per decade, and (3) a flat region, limited by the thermal noise floor or amplitude noise. It is noted that phase noise components are at frequencies different than the carrier, hence are orthogonal, and can be treated as independent components that are added to the main oscillator tone signal for analysis. Figure 10b demonstrates the application of this principle for modeling of the DCO phase noise $\Phi_{n_{DCO}}$ as an additive process to the oscillator phase Φ_{osc}, thus $\Phi_{out} = \Phi_{osc} + \Phi_{n_{DCO}}$.



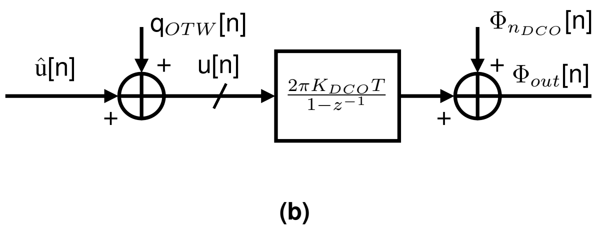


Figure 10: (a) Leeson model for phase noise, **(b)** DCO additive noise model.

The equation for $\mathcal{L}(\Delta f)$ (from [19]) is in equation 31, and is dependent on temperature T, excess noise factor F, oscillator power P, oscillator Q factor, and the transition frequencies f_1 and f_2 that separate the different noise regions. It is of interest to note that the phase noise relative to the carrier will increase as power decreases, which provides challenge for creating low power oscillators with acceptable phase noise characteristics.

$$\mathcal{L}(\Delta f) = 10\log_{10}\left[\frac{2\mathrm{F}k_B\mathrm{T}}{\mathrm{P}}\left(1+\left(\frac{f_2}{2Q\Delta f}\right)^2\right)\left(1+\frac{f_1}{|\Delta f|}\right)\right] = S_{\Phi n_{DCO}}(\Delta f) \qquad (31)$$

For notational consistency, the following redefinition is used in the remainder of this paper: $S_{\Phi n_{DCO}}(f) = \mathcal{L}(\Delta f)|_{\Delta f = f}$

2.3.3 Divider Noise

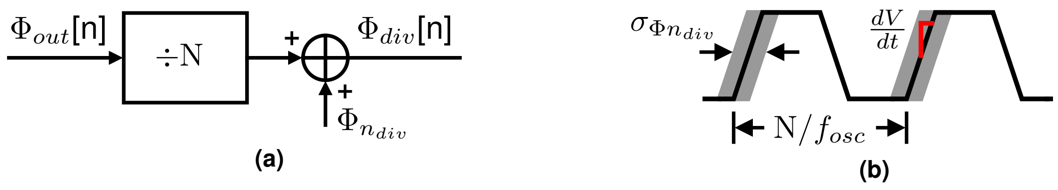


Figure 11: (a) Divider noise model, **(b)** Digital divider output jitter.

Divider noise is manifested as jitter (with RMS distribution in time of $\sigma_{tn_{div}}$) on the divider output. If the divider is a digital circuit, with edge rate dV/dt, and subject to thermal noise in the form of an additive voltage v_n, with noise power of $\sigma^2_{v_n}$, the divider phase noise power added to the divider output is:

$$\sigma^2_{\Phi n_{div}} = \omega^2_{ref}\sigma^2_{tn_{div}} = \omega^2_{ref}\left(\frac{dV}{dt}\right)^{-2}\sigma^2_{v_n} \tag{32}$$

At lock, the output of a digital divider will have an update rate $f_{osc}/\mathrm{N} \approx f_{ref}$, which can be treated as the sampling rate of the output phase signal $\Phi_{div}[n]$ as mentioned in section 2.2.1. If the divider phase noise power is confined into a bandwidth equal to the PLL reference frequency f_{ref}, the spectral density of divider noise is:

$$S_{\Phi n_{div}}(f) = \frac{\sigma^2_{\Phi n_{div}}}{f_{ref}} = 2\pi\omega_{ref}\sigma^2_{tn_{div}} = 2\pi\omega_{ref}\left(\frac{dV}{dt}\right)^{-2}\sigma^2_{v_n} \quad \frac{[\mathrm{rad}]^2}{[\mathrm{Hz}]} \tag{33}$$

2.3.4 Loop Filter Noise - Direct Form I

In a digital loop filter, quantization noise arises from rounding errors due to finite precision in the digital arithmetic circuits that implement the filter [20]. Quantization noise power here will be derived under the assumption of a direct form I structure implementating the filter, with B bits in each fixed point data word throughout the digital loop filter. In a digital implementation of the canonical z-domain transfer function in equation 34 as the direct form I structure of figure 12a, delays are constructed using registers, adders with digital adders, and the filter coefficient gain terms $\{a_1, ...a_N; b_0, ..., b_M\}$ with digital multipliers. The registers and adders do not introduce extra round-off error beyond that already existing (if overflows do not occur). However, the multipliers will if the products resulting from two B bit words (nominally 2B bits), are mapped back onto B bit words.

$$\mathrm{H}_{LF}(z) = \frac{\sum_{j=0}^{Z} b_j z^{-j}}{1 + \sum_{k=1}^{P} a_k z^{-k}} \tag{34}$$

Quantization in this case can be represented by adding a quantization error signal q_x[n] to the result of each ideal multiplication, as shown in figure 12b. This is the same approach for TDC quantization noise in section 2.3.1. The noise power associated with each q_x[n] is identical, with $\sigma^2_{qx} = 1/12\ \mathrm{LSB}^2$.

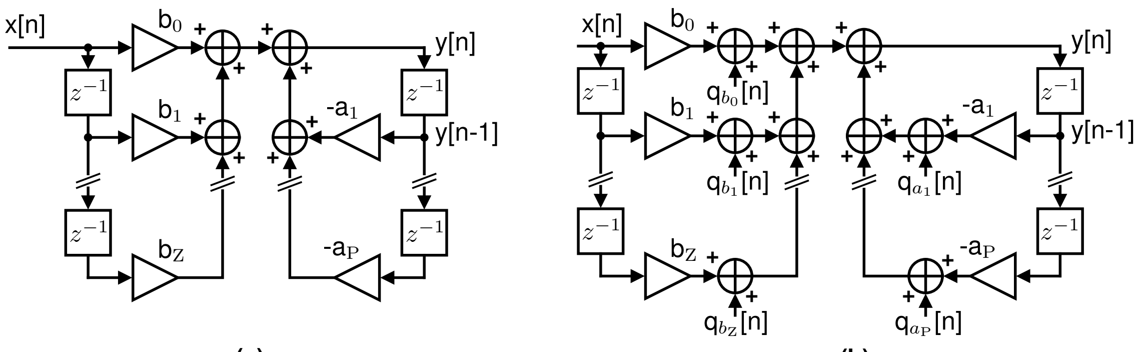


Figure 12: (a) Direct form I filter implementation, **(b)** With added quantization error signals.

Assuming the quantization error signals of each multiplier are uncorrelated with all other multipliers, the output-referred noise power of the filter can be computed as the sum of the output-referred individual contributions. These contributions can be determined via solving for the transfer function from each quantization noise source q$_x$[n] to the output y[n]. In the case of the direct form I filter structure, all quantization sources q$_x$[n] have the same transfer characteristic to the output y[n] given in equation 35.

$$\frac{Y(z)}{Q_x(z)} = \frac{1}{1+\sum_{k=1}^{P} a_k z^{-k}} \tag{35}$$

Applying the bilinear transform to equation 35, with high oversampling where P·BW$_{loop}10 < f_{ref}$, given P is the number of poles in the system:

$$\left.\frac{Y(z)}{Q_x(z)}\right|_{z^{-1}=1-sT} \approx \frac{1}{1+\sum_{k=1}^{P} a_k - s\sum_{k=1}^{P} ka_k} \tag{36}$$

The output power spectral density for each error source is, confined to a bandwidth defined by the (sampling) reference frequency f_{ref}:

$$S_{qx}(f) = \frac{\sigma_{qx}^2}{f_{ref}}\left|\frac{Y(s)}{Q_x(s)}\right|^2_{s=j2\pi f} \approx \frac{1}{12f_{ref}}\left|\frac{1}{1+\sum_{k=1}^{P} a_k - j2\pi f\sum_{k=1}^{P} ka_k}\right|^2 \quad \frac{[\text{LSB}]^2}{[\text{Hz}]} \tag{37}$$

Given P poles and Z zeros in the filter, the total output quantization PSD of the filter is in equation 38. The total loop filter noise PSD linearly scales with the number of multipliers in the direct form I filter implementation.

$$S_{qn_{LF}}(f) = (P+Z+1)S_{qx}(f) \quad \frac{[\text{LSB}]^2}{[\text{Hz}]} \tag{38}$$

2.3.5 PLL Noise Sensitivity Transfer Functions

Having established models for noise of generated by each PLL component, noise sensitivity transfer functions must be computed to refer each noise source to the PLL output in terms of phase. In the noise theory presented here, all noise sources have been modeled as additive signal components. The full system model including all noise sources is in figure 13.

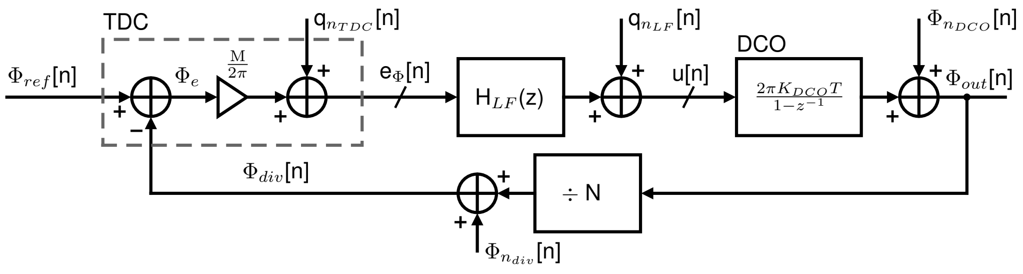


Figure 13: Full PLL additive noise model.

Following the approach of [21], it is useful to define a transfer function $\hat{\mathrm{T}}(s)$ as in equation 39 which characterizes the normalized closed loop response from reference input to output of the PLL. Normalized refers to the zero-frequency response being 1, i.e. $|\hat{\mathrm{T}}(0)| = 1$. The noise transfer functions will be defined in this work in terms of $\hat{\mathrm{T}}(s)$. L(s) is the PLL loop gain.

$$\hat{\mathrm{T}}(s) = \frac{\mathrm{L}(s)}{1+\mathrm{L}(s)} \quad \text{s.t.} \quad \mathrm{T}(s) = \frac{\Phi_{out}}{\Phi_{ref}} = \mathrm{N}\hat{\mathrm{T}}(s) \tag{39}$$

Solving for the closed transfer functions between each noise source ($q_{n_{TDC}}$, $q_{n_{LF}}$, $\Phi_{n_{DCO}}$ and $\Phi_{n_{div}}$) to the output Φ_{out} in the s-domain yields equations 40-43.

$$\frac{\Phi_{out}(s)}{q_{n_{TDC}}(s)} = \frac{2\pi\frac{K_{DCO}}{s}\mathrm{H}_{LF}(s)}{1+\mathrm{L}(s)} = 2\pi\frac{\mathrm{N}}{\mathrm{M}}\frac{\mathrm{L}(s)}{1+\mathrm{L}(s)} = 2\pi\frac{\mathrm{N}}{\mathrm{M}}\hat{\mathrm{T}}(s) \tag{40}$$

$$\frac{\Phi_{out}(s)}{\Phi_{n_{DCO}}(s)} = \frac{1}{1+\mathrm{L}(s)} = 1-\hat{\mathrm{T}}(s) \tag{41}$$

$$\frac{\Phi_{out}(s)}{q_{n_{LF}}(s)} = \frac{2\pi\frac{K_{DCO}}{s}}{1+\mathrm{L}(s)} = 2\pi\frac{K_{DCO}}{s}(1-\hat{\mathrm{T}}(s)) \tag{42}$$

$$\frac{\Phi_{out}(s)}{\Phi_{n_{div}}(s)} = \frac{\mathrm{M}\frac{K_{DCO}}{s}\mathrm{H}_{LF}(s)}{1+\mathrm{L}(s)} = \mathrm{N}\frac{\mathrm{L}(s)}{1+\mathrm{L}(s)} = \mathrm{N}\hat{\mathrm{T}}(s) \tag{43}$$

2.3.6 PLL Phase Signal and Output PSD Relationship for Noise

When analyzing PLL noise in hardware, the noise power spectral density of the PLL output waveform is the most relevant metric. Up to this point, noise has been defined in terms of phase signal Φ_n, or an unwanted added component to the oscillator phase signal $\Phi_{osc} = \omega_{osc}t$. The PLL output phase signal Φ_{out} is thus:

$$\Phi_{out}(t) = \Phi_{osc}(t) + \Phi_n(t) = \omega_{osc}t + \Phi_n(t) \tag{44}$$

To determine the PSD of the PLL output waveform, a definition of the PLL output voltage waveform will be made in terms of a sinusoid, i.e. the real portion of a complex exponential. Given an oscillation amplitude A_0:

$$V_{out} = \Re\left(A_0 e^{j\Phi_{out}(t)}\right) = \Re\left(A_0 e^{j\omega_{osc}t} e^{j\Phi_n(t)}\right) \tag{45}$$

Assuming the phase noise signal is zero mean, $\mathbb{E}[\Phi_n(t)] = 0$, and the power of phase noise signal is small, $\mathrm{Var}[\Phi_n(t)] << 1$, then the approximation $e^{j\Phi_n(t)} = 1 + j\Phi_n(t)$ can be applied by truncating the exponential Taylor series expansion.

$$\begin{aligned} V_{out} &= \Re\left(A_0 e^{j\omega_{osc}t} e^{j\Phi_n(t)}\right) = \Re\left(A_0 e^{j\omega_{osc}t} + j\Phi_n(t) A_0 e^{j\omega_{osc}t}\right) && (46)\\ &= A_0 \cos(\omega_{osc}t) - \Phi_n(t) A_0 \sin(\omega_{osc}t) && (47) \end{aligned}$$

The result is a carrier cosine signal, and an orthogonal sine signal modulated by the phase noise Φ_n. From this, the spectral density of the phase noise relative to the carrier can be estimated. The power spectral density $S_{V_{out}}$ is computed in equations 48-50. Due to orthogonality of the sine/cosine components of equation 47, the cross terms that appear in the PSD computation are zero.

$$\begin{aligned} S_{V_{out}}(f) &= \lim_{\Delta T\to\infty} \frac{1}{\Delta T} \left|\mathcal{F}\{V_{out}(t)\cdot \mathrm{rect}(t/\Delta T)\}\right|^2 && (48)\\ &= \lim_{\Delta T\to\infty} \frac{A_0^2}{\Delta T} \left|\mathcal{F}\{\cos(\omega_{osc}t)\cdot \mathrm{rect}(t/\Delta T)\}\right|^2 && (49)\\ &\quad + \lim_{\Delta T\to\infty} \frac{A_0^2}{\Delta T} \left|\mathcal{F}\{\Phi_n(t)\cdot \mathrm{rect}(t/\Delta T)\} * \mathcal{F}\{\sin(\omega_{osc}t)\cdot \mathrm{rect}(t/\Delta T)\}\right|^2 && (50) \end{aligned}$$

The noise power spectral density function of the output waveform $\mathcal{L}(\Delta f)$ is defined as the noise PSD at offset Δf from the carrier frequency f_{osc}, normalized to the carrier power. Here the PSD of the carrier component is given by equation 49, and the noise component by equation 50. Shifting equation 50 by $-\omega_{osc}$ and performing normalization for carrier power results in:

$$\mathcal{L}(\Delta f) = \lim_{\Delta T\to\infty} \frac{1}{\Delta T} \left|\mathcal{F}\{\Phi_n(t)\cdot \mathrm{rect}(t/\Delta T)\}\right|^2 \Bigg|_{f=\Delta f} = S_{\Phi_n}(\Delta f) \tag{51}$$

Thus, the noise PSD $\mathcal{L}(\Delta f)$ of the PLL output waveform relative to the carrier is equal to the PSD of the phase noise signal $\Phi_n(t)$, provided $\mathrm{Var}[\Phi_n(t)] << 1$. The PSD of $\Phi_n(t)$ is notated as $S_{\Phi_n}(\Delta f)$.

2.3.7 PLL Output-Referred Noise PSD

To compute the PLL output noise PSD, the individual PLL noise sources must be referred to the PLL output. Thus far the following have been established: (a) noise spectrum generated by each individual PLL component, (b) the PLL phase noise sensitivity functions, and (c) the relationship between output phase noise and the PSD of the PLL output waveform. These can be combined to provide a final result for total PLL output-referred noise PSD. An assumption here is all noise sources are uncorrelated, so their independent noise power contributions may be summed to find the total noise PSD. The PLL output phase noise PSD for each noise source is simply found by multiplying the magnitude squared of the respective noise sensitivity function with the noise source PSD. Thus:

$$S_{\Phi n_{TDC,out}}(f) = S_{qn_{TDC}}(f)\left|\frac{\Phi_{out}(f)}{q_{n_{TDC}}(f)}\right|^2 = \frac{1}{12f_{ref}}\left|2\pi\frac{\mathrm{N}}{\mathrm{M}}\hat{\mathrm{T}}(f)\right|^2 \tag{52}$$

$$S_{\Phi n_{DCO,out}}(f) = S_{\Phi n_{DCO}}(f)\left|\frac{\Phi_{out}(f)}{\Phi_{n_{DCO}}(f)}\right|^2 = \frac{S_{0\Phi n_{DCO}}}{f^2}\left|1-\hat{\mathrm{T}}(f)\right|^2 \tag{53}$$

$$S_{\Phi n_{LF,out}}(f) = S_{qn_{LF}}(f)\left|\frac{\Phi_{out}(f)}{q_{n_{LF}}(f)}\right|^2 \approx \frac{K_{DCO}^2}{12f_{ref}f^2}\frac{(P+Z+1)|1-\hat{\mathrm{T}}(f)|^2}{\left|1+\sum_{k=1}^{P}a_k - j2\pi f\sum_{k=1}^{P}ka_k\right|^2} \tag{54}$$

$$S_{\Phi n_{div,out}}(f) = S_{\Phi n_{div}}(f)\left|\frac{\Phi_{out}(f)}{\Phi_{n_{div}}(f)}\right|^2 = f_{ref}\left|2\pi\sigma_{tn_{div}}\mathrm{N}\hat{\mathrm{T}}(f)\right|^2 \tag{55}$$

The final PLL output noise PSD at offset Δf relative to the carrier frequency and normalized to carrier power of PLL will be:

$$\mathcal{L}(\Delta f) = S_{\Phi n_{TDC,out}}(\Delta f) + S_{\Phi n_{DCO,out}}(\Delta f) + S_{\Phi n_{LF,out}}(\Delta f) + S_{\Phi n_{div,out}}(\Delta f) \tag{56}$$

2.4 Bang-bang phase detector PLL

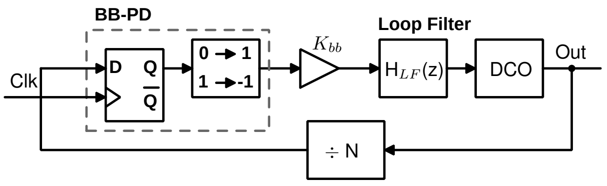


Figure 14: PLL with bang-bang phase detector.

An alternative digital phase detector that is not resolution limited like a TDC is the bang-bang phase detector (BBPD) [22]. A BBPD is implemented in a PLL as shown in figure 14. The BBPD in principle functions by outputting a 1 if the divider signal is late relative to the clock, and -1 if it is early. A BBPD exhibits hard nonlinearity in its transfer characteristics. However, as shown in [23], a linearized model for BBPD gain can be established, given in equation 57 if the signal variance $\sigma_{\Phi_e}^2$ into the detector is constant. Here the input to the detector is phase error signal $\Phi_e = \Phi_{CLK} - \Phi_{DIV}$ and the output y valued as ± 1 (its variance σ_y^2=1).

$$K = \frac{\mathbb{E}[\Phi_e(t)\cdot \mathrm{y}(t)]}{\mathbb{E}[\Phi_e^2(t)]} = \sqrt{\frac{2}{\pi}}\frac{1}{\sigma_{\Phi_e}} \tag{57}$$

The noise power of the BBPD can be determined as $\sigma_{n_{BBPD}}^2$ = σ_y^2 - $K^2\sigma_{\Phi_e}^2$ = $1-\frac{2}{\pi}$. It is observed that the BBPD noise power is fixed, unlike the TDC, which can reduce the limitations on detector phase noise compared to the quantization prone TDC. Given the BBPD samples the divider phase at a rate equal to f_{ref}, the BBPD noise spectral density is in equation 58. The gain of the BBPD, however, changes with input signal variance, which affects the PLL loop gain and thus closed loop PLL response. The variance of the signal into the phase detector should be expected to vary across PLL process, voltage and temperature conditions, thus the BBPD is limited in terms of gain accuracy. The linearized gain and noise values determined here can be substituted for the TDC model established for PLL transfer function and phase noise in sections 2.2 and 2.3 to solve for BBPD-PLL dynamics. This yields the closed loop response of equation 59, the noise transfer function in equation 60, and the output-referred noise spectral density in equation 61.

$$S_{n_{BBPD}(f)} = \frac{\sigma^2_{n_{BBPD}}}{\Delta f} = \frac{\left(1 - \frac{2}{\pi}\right)}{f_{ref}} \tag{58}$$

$$\mathrm{T}(s) = \frac{\Phi_{out}(s)}{\Phi_{ref}(s)} = \frac{2\pi\sqrt{\frac{2}{\pi}}\frac{1}{\sigma_{\Phi_e}}K_{DCO}\sum_{j=0}^{Z} b_j s^j}{\sum_{k=0}^{P} a_k s^{k+1} + 2\pi\sqrt{\frac{2}{\pi}}\frac{1}{\sigma_{\Phi_e}\mathrm{N}}K_{DCO}\sum_{j=0}^{Z} b_j s^j} = \mathrm{N}\frac{\mathrm{L}(s)}{1+\mathrm{L}(s)} \tag{59}$$

$$\frac{\Phi_{out}(f)}{n_{BBPD}(f)} = \sqrt{\frac{\pi}{2}}\sigma_{\Phi_e}\mathrm{N}\frac{\mathrm{L}(f)}{1+\mathrm{L}(f)} = \sqrt{\frac{\pi}{2}}\sigma_{\Phi_e}\mathrm{N}\hat{\mathrm{T}}(f) \tag{60}$$

$$S_{\Phi n_{BBPD,out}}(f) = S_{n_{BBPD}}(f)\left|\frac{\Phi_{out}(f)}{q_{n_{BBPD}}(f)}\right|^2 = \frac{\left(\frac{\pi}{2} - 1\right)}{f_{ref}}\left|\sigma_{\Phi_e}\mathrm{N}\hat{\mathrm{T}}(f)\right|^2 \tag{61}$$

3 Proposed ADPLL Architecture

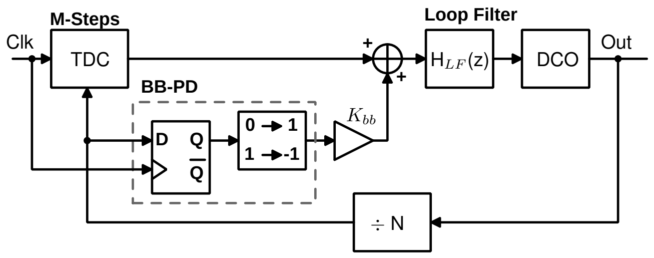


Figure 15: PLL with parallel bang-bang phase detector and TDC.

To gain linearity advantages associated with a TDC, and low phase noise and high phase resolution of a BBPD, the PLL architecture in figure 15 is utilized in this work. The TDC and BBPD are configured in a parallel configuration, where the TDC output is summed with the BBPD output. The BBPD output has a gain term K_{bb} to adjust the BBPD gain relative to the TDC gain. This architecture is suggested as this work focuses on applications of low power PLLs, where a low resolution TDC is expected. Thus, it is expected the TDC resolution will limit the ability to detect phase (or equivalently frequency) errors at the output in steady state. With M TDC steps per reference cycle, a divider ratio of N and a frequency error at the PLL output of Δf, the time needed for enough phase error to accumulate for a 1 LSB change of the TDC output is in equation 62.

$$t = \frac{\mathrm{N}}{\mathrm{M}\Delta f} \tag{62}$$

If M $<$ N, the response time t of the TDC to output frequency disturbance Δf from phase noise is greater than the period of that disturbance $1/\Delta f$. Under these conditions, the PLL will have impaired ability to correct phase error and the phase noise performance will be degraded in steady state. Thus, the BBPD is added in order to introduce higher resolution phase-feedback in steady state, where the phase error signal is small. Methods for selecting K_{bb} will be considered later, however an approximate method will be described here, whose result is given in equation 64. This method assumes that in steady state (a) the intrinsic jitter of the BBPD with time variance $\sigma_{t_e}^2$ is the dominant PLL noise component, and (b) BBPD gain shall be set to equal the gain provided by the TDC. Given the established gain of the BBPD in equation 57:

$$\sqrt{\frac{2}{\pi}}\frac{1}{\sigma_{\Phi_e}} \cdot K_{bb} = \sqrt{\frac{2}{\pi}}\frac{1}{2\pi f_{ref}\sigma_{t_e}} \cdot K_{bb} = \frac{\mathrm{M}}{2\pi} \tag{63}$$

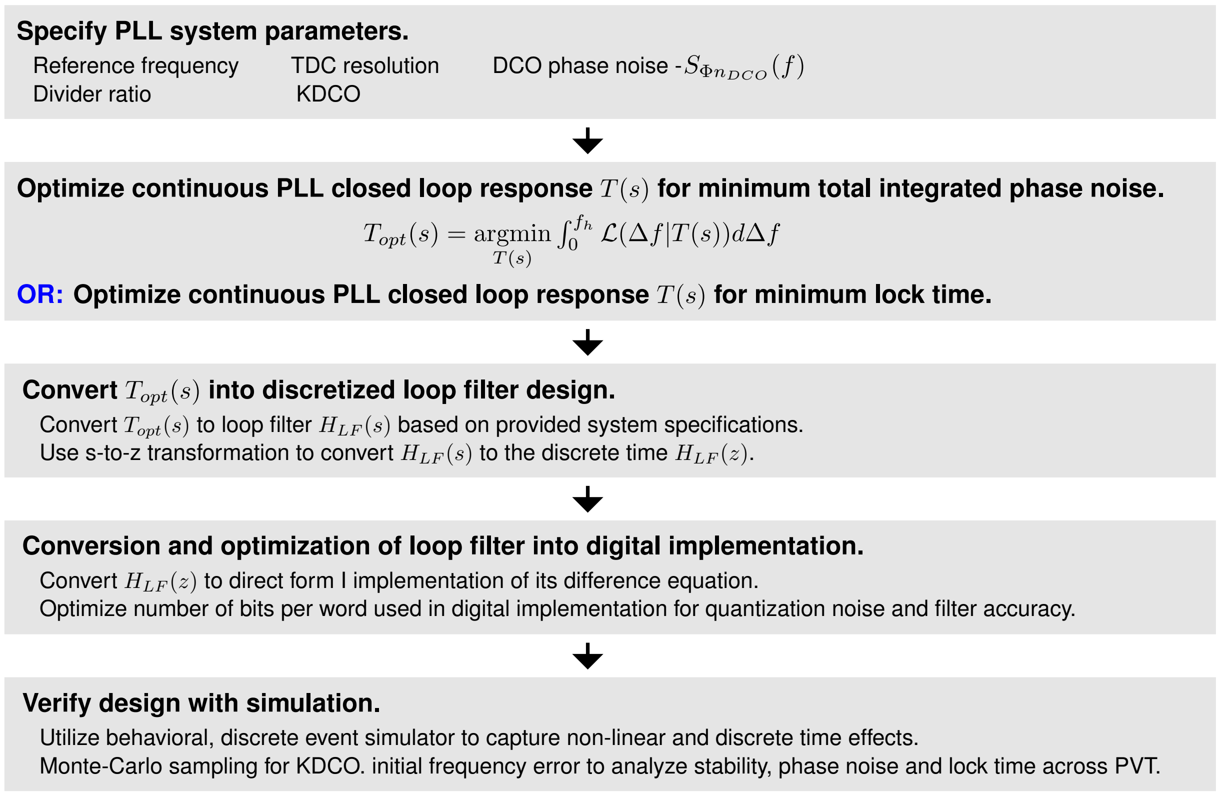
$$K_{bb} = \sqrt{\frac{\pi}{2}} f_{ref}\sigma_{t_e}\mathrm{M} \tag{64}$$

The proposed architecture enables a form of fast-locking via gear switching of the loop filter [24]. In this work, the TDC is initially used for large frequency and phase errors with a loop filter optimized for speed, which after locking can be changed (gear-switched) to a BBPD optimized loop filter chosen to minimize total phase-noise. Methods for optimized loop filter design for phase noise and lock time will be considered in the following sections.

4 Loop Filter Design Automation

4.1 Design Sequence

Design automation for ADPLL loop filter design is implemented with a strategy that is illustrated in figure 16, that utilizes a continuous time-approximation based model of the PLL to generate a loop filter design which either minimizes the total integrated phase noise power out of the PLL, or lock time. The optimized continuous filter then undergoes discrete time conversion and mapping into a digital representation of the design. A post-optimization process is then applied to the discretized and digitized filter implementation, to minimize the effects of quantization error and filter design error due to finite word effects. A discrete-time, behavioral PLL simulator is then used to verify the filter design for proper lock-time, phase noise and stability. The following sections will detail these processes.



Specify PLL system parameters.
Reference frequency, TDC resolution, DCO phase noise -$S_{\Phi n_{DCO}}(f)$
Divider ratio, KDCO

Optimize continuous PLL closed loop response $T(s)$ for minimum total integrated phase noise.

$$T_{opt}(s) = \underset{T(s)}{\text{argmin}} \int_0^{f_h} \mathcal{L}(\Delta f | T(s)) d\Delta f$$

OR: Optimize continuous PLL closed loop response $T(s)$ for minimum lock time.

Convert $T_{opt}(s)$ into discretized loop filter design.
Convert $T_{opt}(s)$ to loop filter $H_{LF}(s)$ based on provided system specifications.
Use s-to-z transformation to convert $H_{LF}(s)$ to the discrete time $H_{LF}(z)$.

Conversion and optimization of loop filter into digital implementation.
Convert $H_{LF}(z)$ to direct form I implementation of its difference equation.
Optimize number of bits per word used in digital implementation for quantization noise and filter accuracy.

Verify design with simulation.
Utilize behavioral, discrete event simulator to capture non-linear and discrete time effects.
Monte-Carlo sampling for KDCO. initial frequency error to analyze stability, phase noise and lock time across PVT.

Figure 16: Filter design sequence.

4.2 Loop Filter Prototype

The design automation approach developed utilizes a fixed loop filter as a prototype for all loop filter designs. This choice was derived from several criteria that were established for desirable

ADPLL operation:

1. Zero steady state phase error, to ensure accuracy of synthesized frequency.
2. Loop filter output should remain constant if input (phase error) goes to 0.
3. Minimize complexity of implemented logic, i.e. minimize the number of poles and zeros.
4. Low pass response of PLL in closed-loop.

To satisfy criterion 1, the loop filter response must include an integrator (1/s) term in the transfer function [25]. A rational starting point is to consider a proportional-integral-derivative (PID) controller [26] for the loop filter, whose transfer function is given in equation 65. This filter contains coefficients K_d, K_p, K_i which set the gain of the derivative (s), proportional and integral (1/s) terms respectively.

$$\mathrm{H}_{LF}(s) = sK_d + K_p + \frac{K_i}{s} = \frac{K_d}{s}\left(s^2 + s\frac{K_p}{K_d} + \frac{K_i}{K_d}\right) \tag{65}$$

Application of the loop filter to the continuous ADPLL transfer function model from section 2.2.5 produces the PLL loop gain in equation 66 and the closed loop ADPLL response in equation 67.

$$\mathrm{L}(s) = \mathrm{H}_{TDC}(s)\mathrm{H}_{LF}(s)\mathrm{H}_{DCO}(s)\mathrm{H}_{DIV}(s) = \frac{\mathrm{M}}{\mathrm{N}}\frac{K_{DCO}K_d}{s^2}\left(s^2 + s\frac{K_p}{K_d} + \frac{K_i}{K_d}\right) \tag{66}$$

$$\mathrm{T}(s) = \frac{\Phi_{out}(s)}{\Phi_{ref}(s)} = \frac{\mathrm{M}K_{DCO}K_d\left(s^2K_d + sK_p + K_i\right)}{s^2\left(1 + \frac{\mathrm{M}}{\mathrm{N}}K_{DCO}K_d\right) + \frac{\mathrm{M}}{\mathrm{N}}K_{DCO}K_d\left(sK_p + K_i\right)} = \mathrm{N}\frac{\mathrm{L}(s)}{1 + \mathrm{L}(s)} \tag{67}$$

It should be noted that in the closed loop configuration, this PLL phase transfer function contains two poles and two zeros. This is not a low pass response as desired per criterion 4, needed for satisfactory PLL phase noise power spectrum as will later be discussed. In order achieve low pass operation, the derivative term K_d must be set to zero, yielding a proportional-integral (PI) controller for the loop filter, with closed loop response in equation 68.

$$\mathrm{T}(s) = \frac{\Phi_{out}(s)}{\Phi_{ref}(s)} = \frac{\mathrm{M}K_{DCO}\left(sK_p + K_i\right)}{s^2 + \frac{\mathrm{M}}{\mathrm{N}}K_{DCO}\left(sK_p + K_i\right)} = \mathrm{N}\frac{\mathrm{L}(s)}{1 + \mathrm{L}(s)} \tag{68}$$

Steady state zero phase error can be verified for the PI-controller PLL by solving the closed loop phase error $\Phi_e(s)$ for s=0:

$$\Phi_e(s)|_{s=0} = \left(\Phi_{ref}(0) - \frac{\Phi_{out}(0)}{\mathrm{N}}\right) = \Phi_{ref}(0)\left(1 - \frac{\Phi_{out}(0)}{\mathrm{N}\Phi_{ref}(0)}\right) = \Phi_{ref}(0)\left(1 - \frac{\mathrm{N}}{\mathrm{N}}\right) = 0 \tag{69}$$

4.2.1 Continuous PI-loop filter design

Given a PI-controller loop filter, which can be optionally represented using a pole at zero and a zero with $\omega_z = K_i/K_p$:

$$\mathrm{H}_{LF}(s) = K_p + \frac{K_i}{s} = \frac{K_i}{s}\left(\frac{s}{\omega_z} + 1\right) \tag{70}$$

Selection of (not-necessarily optimal) PI controller gains can be easily derived from overall PLL settling time requirements. Suppose that settling time t_s is defined such that the PLL settles within $\pm\delta$ of the final value for a step input. If the initial and final PLL output frequencies are f_i and Nf_{ref}, and a frequency tolerance of $\pm f_{tol}$ is desired for the lock criteria, $\delta = f_{tol}/|f_i - Nf_{ref}|$. Setting time is therefore:

$$t_s = -\tau \ln(\delta) \tag{71}$$

Thus, to find settling time, a value for the PLL time constant τ must be derived. Rewriting equation 68 with substitutions $\omega_z = K_i/K_p$ and $\mathrm{K} = MK_{DCO}K_i/\mathrm{N}$:

$$\frac{\Phi_{out}(s)}{\Phi_{ref}(s)} = \mathrm{N} \cdot \frac{s\frac{K}{\omega_z} + K}{s^2 + s\frac{K}{\omega_z} + K} \tag{72}$$

If the second order denominator can be redefined in terms of a natural frequency ω_n and damping ratio ζ, such that:

$$s^2 + s\frac{K}{\omega_z} + K = s^2 + s2\zeta\omega_n + \omega_n^2 \tag{73}$$

It is then found that $\omega_n = \sqrt{K}$, and $\omega_z = \sqrt{K}/2\zeta$. The poles of equation 72 are then located at s = $\zeta\sqrt{K} \pm \sqrt{K}\sqrt{1-\zeta^2}$. The settling time of the PLL will be determined by the real portion of dominant pole of equation 72, specifically $\tau = 1/|\min(\Re(\{s_{p1}, s_{p2}\}))|$. Based on the pole-zero plot of figure 17, it can be observed that the dominant pole location is maximized with $\zeta = 1$. The shown pole-zero loci orientations are based on increasing ζ values. According to Razavi [27], ζ is usually "chosen to be $> \sqrt{2}/2$ or even 1 to avoid excessive ringing."

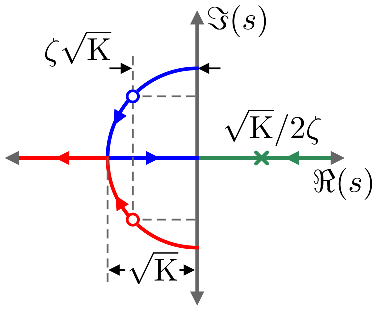


Figure 17: PI-controller PLL pole-zero locations.

To illustrate the effect of the damping ratio ζ, figure 18 illustrates the example frequency and step responses of a PI-controlled PLL with N=1. Notice the excessive peaking and ringing for $\zeta < \sqrt{2}/2$. The peaking observed in the frequency response is unavoidable with the PI-PLL due to the inherent zero in the transfer function. Its effect can be reduced with large ζ, however this will increase PLL settling time.

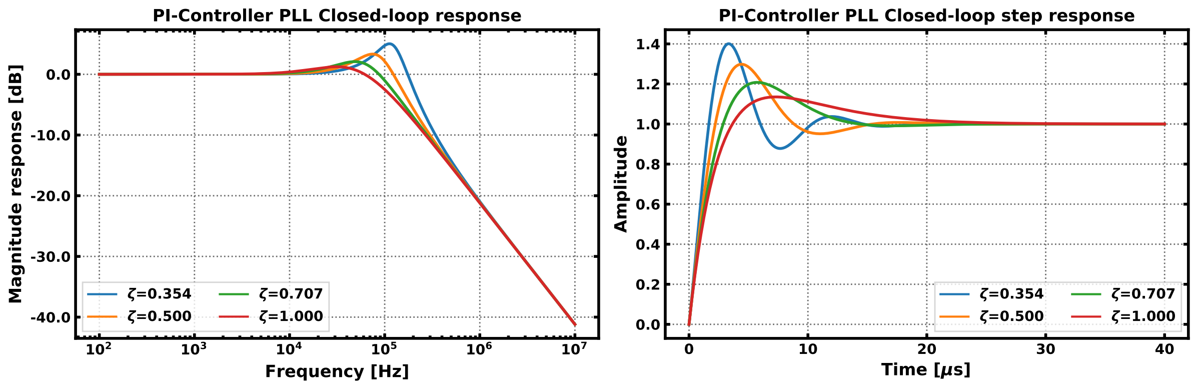


Figure 18: Example PI-PLL responses with varied ζ.

If ζ is constrained to ≤ 1:

$$\tau = \frac{1}{|\min(\Re(\{s_{p1}, s_{p2}\}))|} = \frac{1}{\zeta\sqrt{K}} \tag{74}$$

Thus settling time is:

$$t_s = \frac{-\ln(\delta)}{\zeta\sqrt{K}} = \frac{-\ln\left(\frac{f_{tol}}{|f_i - Nf_{ref}|}\right)}{\zeta\sqrt{K}} \tag{75}$$

Based on specification for settling time and damping ratio ζ, the PI-PLL model values K and ω_z can be determined. If K_{DCO} and N are also specified, the PI gain coefficients can be solved additionally.

$$\omega_z = \frac{-\ln(\delta)}{2t_s} = \frac{-\ln\left(\frac{f_{tol}}{|f_i - Nf_{ref}|}\right)}{2t_s} \tag{76}$$

$$K = \frac{\ln^2(\delta)}{\zeta^2 t_s^2} = \frac{\ln^2\left(\frac{f_{tol}}{|f_i - Nf_{ref}|}\right)}{\zeta^2 t_s^2} \tag{77}$$

$$K_i = \frac{\text{N}}{\text{M}}\frac{\text{K}}{K_{DCO}} \tag{78}$$

$$K_p = \frac{K_i}{\omega_z} \tag{79}$$

4.2.2 PI-controller phase margin

A considerable advantage to using the PI-controller PLL design methodology of section 4.2.1 is the resulting phase margin of the PLLs is only determined by the damping ratio ζ. The phase margin for such a PI-controller PLL was determined to be equation 80. This was computationally verified, resulting in the plot of figure 19 showing phase margin versus $0 \geq \zeta \geq 1$ of the PI-controller PLL. It is recommended to use at least 30-60 degrees in phase margin to achieve stability [28]. Thus, it is expected the PI-controller can yield stable ADPLLs for any closed loop bandwidth conditioned on the preceding the phase margin recommendation and the constraint that PLL loop bandwidth must be greater than 10 times the loop filter sampling rate as mentioned in section 2.2.

$$\angle \mathrm{L}(\omega_{ug}) = \frac{180}{2\pi} \arctan\left(2\zeta\sqrt{2\zeta^2 + \sqrt{4\zeta^4 + 1}}\right) \quad \text{[degrees]} \tag{80}$$

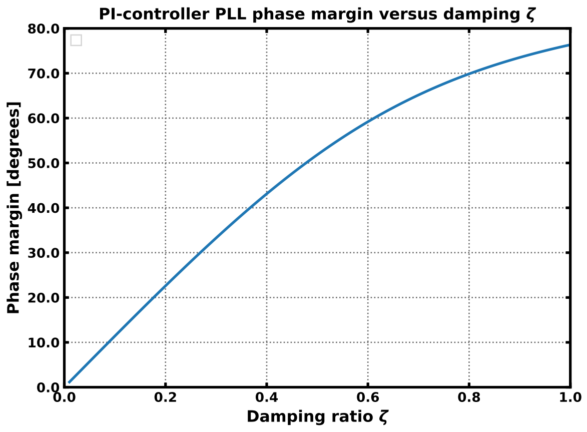


Figure 19: PI-controller PLL phase margin versus damping ratio.

4.2.3 PI-Controller Peaking Compensation

To compensate for closed loop peaking, the original PI-controller loop filter of equation 70 can be modified with the addition of a single tunable pole at ω_p. The closed loop response becomes third order, which complicates direct analysis and design of the loop filter, but can be handled utilizing the numerical optimization approach described in this work. Due to the third order nature, the stability of this PLL configuration is not guaranteed based on a closed-form selection of a single parameter like with the PI-controller PLL.

$$\mathrm{H}_{LF}(s) = \frac{K_i}{s} \frac{\left(\frac{s}{\omega_z} + 1\right)}{\left(\frac{s}{\omega_p} + 1\right)} \tag{81}$$

4.2.4 Alternative PID Controller Permutations

If individual terms within the PID-controller are dropped, different controller permutations (PD, ID, PI, P, I, D) can be achieved. As mentioned before, inclusion of an integral term is needed to ensure the desired zero steady state error for a PLL, and the derivative term must be removed to achieve low pass response in the PLL. This leaves an integral-only controller as the remaining candidate here for PID controller design. Thus, setting the K_p and K_d terms of equation 67 to zero yields:

$$\frac{\Phi_{out}(s)}{\Phi_{ref}(s)} = \frac{2\pi K_{VCO}K_i}{s^2 + \frac{2\pi K_{VCO}}{\mathrm{N}}K_i} \tag{82}$$

This closed loop transfer function results in a pair of poles at $\pm\sqrt{2\pi K_{VCO}K_i/\mathrm{N}}$. This is not stable, as it can only be manifested as (1) a pair of poles on the imaginary axis, which is an oscillator, or (2) a real pole in the right-half plane and a real pole in the left-half plane, the former of which is not causally stable. Thus a PI-controller is the only viable PID-controller permutation for use in the PLL loop filter of this work.

4.2.5 Discretized Loop Filter Prototype

Using the continuous filter discretization approach described in section 2.2.3 on the loop filter of equation 81 results in equation 83.

$$\mathrm{H}_{LF}(z) = \left.\frac{K_i}{s}\frac{\left(\frac{s}{\omega_z}+1\right)}{\left(\frac{s}{\omega_p}+1\right)}\right|_{s=\frac{1}{\Delta T_s}(1-z^{-1})} = k_i\Delta T_s\frac{\omega_p}{\omega_z}\frac{(1+\omega_z\Delta T_s)-z^{-1}}{(1+\omega_p\Delta T_s)-z^{-1}(2+\omega_p\Delta T_s)+z^{-2}} \tag{83}$$

The transformation of equation 83 into a digitally implementable design as a direct form 1 IIR filter shown in figure 20. Its filter coefficients given by equations 84-85.

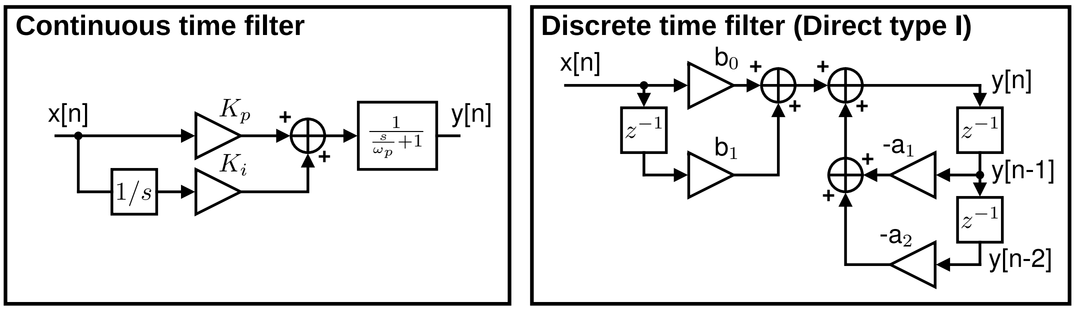


Figure 20: Implementation of filter.

$$a_1 = -\frac{2+\omega_p\Delta T_s}{1+\omega_p\Delta T_s} \qquad a_2 = \frac{1}{1+\omega_p\Delta T_s} \tag{84}$$

$$b_0 = \frac{K_i\omega_p\Delta T_s}{\omega_z}\frac{1+\omega_z\Delta T_s}{1+\omega_p\Delta T_s} \qquad b_1 = \frac{K_i\omega_p\Delta T_s}{\omega_z}\frac{1}{1+\omega_p\Delta T_s} \tag{85}$$

4.2.6 Prototype PLL Response

Based on the prototype loop filter developed, the PLL closed loop response is developed for usage in the loop filter optimizer discussed in this work. Applying the discrete-time PLL transfer function model developed in section 2.2.5, the PLL loop gain is in equation 87, and the closed loop transfer function of the PLL is in 88.

$$\mathrm{L}(z) = \mathrm{H}_{TDC}(z)\mathrm{H}_{LF}(z)\mathrm{H}_{DCO}(z)\mathrm{H}_{DIV}(z) \tag{86}$$

$$= 2\pi K_{DCO}K_i\Delta T_s^2\frac{\mathrm{M}}{\mathrm{N}}\frac{\omega_p}{\omega_z}\frac{(1+\omega_z\Delta T_s)-z^{-1}}{(1+\omega_p\Delta T_s)-z^{-1}(3+2\omega_p\Delta T_s)+z^{-2}(3+\omega_p\Delta T_s)-z^{-3}} \tag{87}$$

The closed loop z-domain PLL phase transfer function is:

$$\mathrm{T}(z) = \frac{\Phi_{out}(z)}{\Phi_{ref}(z)} = \frac{2\pi K_{DCO}K_i\Delta T_s^2\mathrm{M}\frac{\omega_p}{\omega_z}(1+\omega_z\Delta T_s)-z^{-1}}{\left(\begin{array}{l}(1+\omega_p\Delta T_s+2\pi K_{DCO}K_i\Delta T_s^2\frac{\mathrm{M}}{\mathrm{N}}\frac{\omega_p}{\omega_z}(1+\omega_z\Delta T_s))-z^{-1}(3+\\ 2\omega_p\Delta T_s+2\pi K_{DCO}K_i\Delta T_s^2\frac{\mathrm{M}}{\mathrm{N}}\frac{\omega_p}{\omega_z})+z^{-2}(3+\omega_p\Delta T_s)-z^{-3}\end{array}\right)} \tag{88}$$

Applying z-to-s domain transformation, the continuous s-domain approximation of the loop gain is given in equation 89, and the closed loop transfer function in equation 90.

$$\mathrm{L}(s) = \mathrm{H}_{TDC}(s)\mathrm{H}_{LF}(s)\mathrm{H}_{DCO}(s)\mathrm{H}_{DIV}(s) = \frac{\mathrm{M}}{\mathrm{N}}\frac{K_{DCO}K_i}{s^2}\frac{\left(\frac{s}{\omega_z}+1\right)}{\left(\frac{s}{\omega_p}+1\right)} \tag{89}$$

$$\mathrm{T}(s) = \frac{\Phi_{out}(s)}{\Phi_{ref}(s)} = \frac{\mathrm{M}K_{DCO}K_i\left(\frac{s}{\omega_z}+1\right)}{s^3\frac{1}{\omega_z}+s^2+\frac{\mathrm{M}}{\mathrm{N}}K_{DCO}K_i\left(\frac{s}{\omega_z}+1\right)} = \mathrm{N}\frac{\mathrm{L}(s)}{1+\mathrm{L}(s)} \tag{90}$$

5 Behavioral ADPLL Simulation

To fully capture the effects of a discrete time PLL with digital quantization effects, the implementation of a behavioral, discrete event PLL simulator in Python is described in the following. The simulator utilizes behavioral models to describe the individual components which comprise the PLL. These components are (1) a clock reference, (2) a TDC, (3) a loop filter, (4) a DCO, and (5) a divider. These behavioral models fully encapsulate effects of time-quantization and digitization. The simulator operates by iterating in fixed time steps of Δt=1/`fs`=$1/f_{ref}$, where each node in the PLL is updated based upon the previous node values in a manner that is defined by each PLL component behavioral model. Each behavioral model is represented programmatically utilizing Python classes. In the simulation, each PLL component is represented by an object instance of the respective model class, constructed with any initial parameters (such as division ratio) that describe the component.

5.1 Simulation engine

The simulator engine for this work follows the sequence illustrated in figure 21 for running a simulation. Given specifications for simulation conditions (listed in the figure), the simulator accordingly initializes model objects that constitute the PLL components in simulation space. Then the simulation is run by entering a simulation loop.

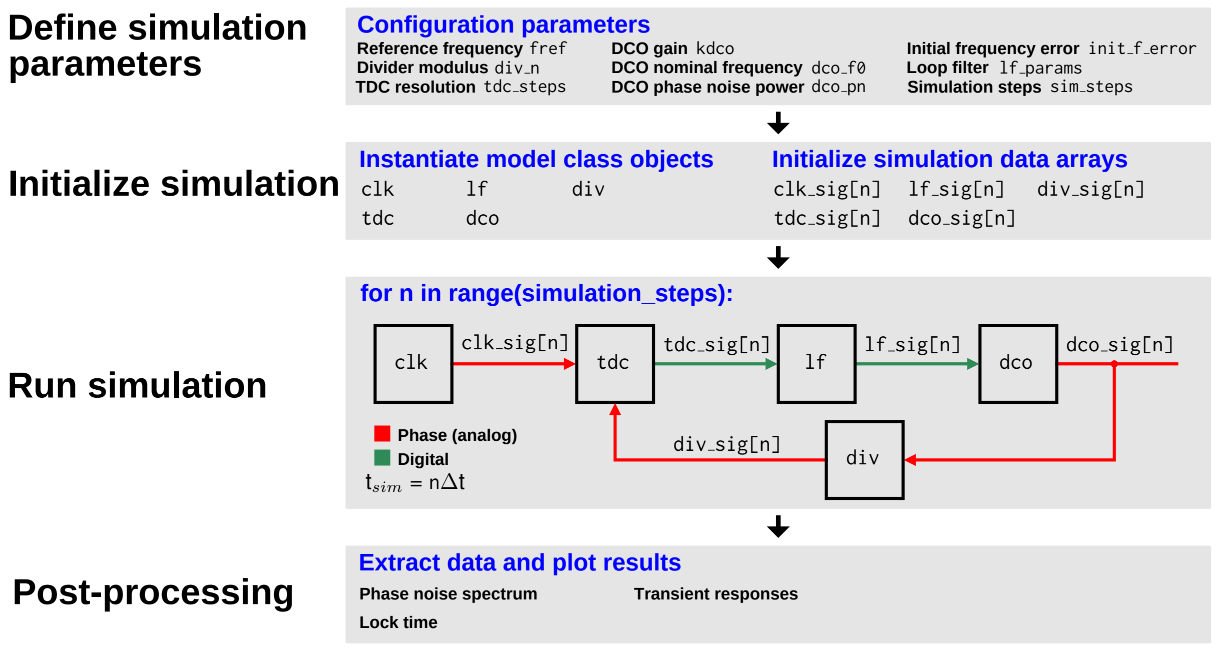


Figure 21: Simulation process.

The discrete event simulator loop is given in the following pseudocode of listing 1, with component model objects {`clk, tdc, lf, dco, div`}, and simulation data arrays {`clk_sig, tdc_sig, lf_sig, dco_sig, div_sig`}. The simulation operates with a fixed, discrete time step. Each model object is updated at each simulation step (loop iteration) utilizing the class method `update`, passing any relevant simulation data as arguments to the method. The output state of each component is saved into the respective array instance for each simulation step.

```python
for n in range(simulation_steps):
    clk_sig[n] = clk.update()
    tdc_sig[n] = tdc.update(clk_sig[n-1], div_sig[n-1]))
    lf_sig[n]  = lf.update(tdc_sig[n-1], clk_sig[n-1]) #loop filter
    osc_sig[n] = dco.update(lf_sig[n-1])
    div_sig[n] = div.update(osc_sig[n-1], div_n)
```

Listing 1: PLL simulation loop Python pseudocode

After the simulation loop reaches completion, the results stored in the simulation data arrays can be post-processed to extract phase noise data, transient behavior and lock time. The following sections will discuss in more detail the implemented behavioral model classes and post-processing.

5.2 Clock Behavioral Model

An ideal behavioral clock model is utilized, given in the pseudocode of listing 2. The model is instantiated with the clock frequency `f` and the simulator time step `dt`. The model increments its phase every simulation step by a fixed amount $\Delta\Phi = 2\pi\texttt{f}\cdot\texttt{dt}$, as in equation 91. The model outputs an analog phase signal.

$$\Phi_{clk}[n] = \Phi_{clk}[n-1] + 2\pi\texttt{f}\cdot\texttt{dt} \tag{91}$$

```python
class Clock:
  def __init__(self, f, dt):
    self.f = f        # clock frequency
    self.dt = dt      # simulation time step
    self.phase = 0.0  # clock phase state variable

  def update(self):
    self.phase += 2*pi*self.f*self.dt   # increment phase
    return self.phase
```

Listing 2: Ideal clock behavioral model Python pseudocode.

5.3 TDC Behavioral Model

The TDC behavioral model takes two analog inputs x and y that are in units of phase, and outputs a digital word that quantifies the phase separation of the signals. The model is instantiated with a resolution parameter tdc_steps, which defines the number of phase steps per cycle of the reference input x that the TDC can resolve. The method round quantizes a floating point argument to the nearest integer.

$$\mathrm{out}[n] = \left\lfloor 0.5 + \texttt{tdc\_steps}\frac{\mathrm{x}[n]-\mathrm{y}[n]}{2\pi} \right\rfloor \tag{92}$$

```
class TDC:
  def __init__(self, tdc_steps):
    self.tdc_steps = tdc_steps

  def update(x, y):
    ph_error = wrap(x-y)  # wraps phase to be within [0, 2*pi]
    return round(self.tdc_steps*(ph_error/(2*pi)))
```

Listing 3: TDC behavioral model Python pseudocode.

5.4 Loop Filter Behavioral Model

The loop filter model implements a discrete-time filter via difference equation that operates on input x. The equivalent of one zero and two poles are modeled, described using the filter coefficients $\{a_1, a_2; b_0, b_1\}$:

$$\mathrm{H}_{LF}(z) = \frac{b_0 + b_1 z^{-1}}{a_0 + a_1 z^{-1} + a_2 z^{-2}} \tag{93}$$

$$\mathrm{y}[n] = -a_1\mathrm{y}[n-1] - a_2\mathrm{y}[n-2] + b_0\mathrm{x}[n] + b_1\mathrm{x}[n-1] \tag{94}$$

Pseudocode for the implementation of the model class is as follows. Data is to be represented with two's complement signed fixed point format, with number of fractional bits frac_bits and number of integer bits int_bits. The method fixed_point rounds floating point values to be equivalent to the nearest representation in the desired fixed point format. The filter coefficients are assumed to be pre-converted to the desired fixed-point equivalent values, and input x is assumed to be integer-valued.

```
class LoopFilter:
  def __init__(self, a1, a2, b0, b1, int_bits, frac_bits):
    self.a1 = a1; self.a2 = a2
    self.b0 = b0; self.b1 = b1
    self.xprev1 = 0;
    self.yprev1 = 0; self.yprev2 = 0
```

```python
    self.int_bits=int_bits; self.frac_bits=frac_bits

  def update(x):
    ynew = -self.a1*self.yprev1 - self.a2*self.yprev2 \\
         + self.b0*x + self.b1*self.xprev1  # difference equation
    self.yprev2 = self.yprev1
    self.yprev1 = fixed_point(ynew, self.int_bits, self.frac_bits)
    self.xprev1 = x
    return round(self.yprev1) # convert to integer
```

Listing 4: Loop filter behavioral model Python pseudocode.

5.5 DCO Behavioral Model

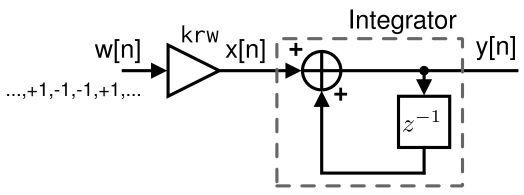
The DCO is modeled similar to the clock model, with the inclusion of a digital input otw for tuning the frequency. Nominal oscillator frequency is given by f0, and the DCO gain in Hz/LSB of otw is kdco. Additionally, modeling of the $\propto f^{-}2$ oscillator phase noise component, which is equivalent to random phase walk [29], is included utilizing additive random phase walk. Random walk is implemented by stochastically either adding or subtracting a fixed magnitude random walk phase increment krw to the oscillator phase every simulation step. The sign of krw is randomly chosen with equal probability for positive and negative (sampling a bi-delta distribution), implemented with the method choice.

$$\Phi_{osc}[n] = \Phi_{osc}[n-1] + 2\pi(\texttt{f0} + \texttt{otw}\cdot\texttt{kdco})\cdot\texttt{dt} \pm \texttt{krw} \tag{95}$$

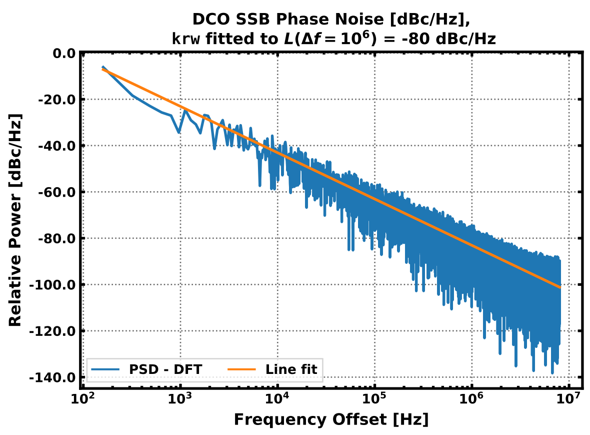
```python
class DCO:
  def __init__(self, f0, kdco, krw, dt):
    self.f0 = f0     # nominal frequency
    self.kdco = kdco  # DCO gain
    self.krw       # random phase walk gain
    self.dt        # simulation time step size
    self.phase = 0    # phase state variable

  def update(otw):
    self.phase += 2*pi*(self.f0 + otw*self.kdco)*self.dt + krw*choice
       ([-1,1])
    return self.phase
```

Listing 5: DCO behavioral model Python pseudocode.

Selection of the parameter krw to achieve a target phase noise level $\mathcal{L}(\Delta f)$ at offset Δf from the carrier is as follows.



(a)



(b)

Figure 22: (a) Discrete model for oscillator random walk, **(b)** Simulated phase noise of behavioral model.

The random walk process described in the behavioral model is represented in figure 22a, where a random white-spectrum input sequence w[n] taking on values $\pm$ 1 with equal probability are multiplied by gain krw, yielding signal x[n] that is passed into a discrete integrator. The output y[n] is then the phase noise signal that is summed with the oscillator phase trajectory. If the simulation is limited to sim_steps samples, application of Parseval's theorem for the discrete Fourier transform (DFT) of x[n] in equation 96 results in the energy of x[n] in equation 97, having a constant value σ_x^2.

$$\sum_{n=0}^{\texttt{sim\_steps}-1} |\texttt{krw} \cdot w[n]|^2 = \frac{1}{\texttt{sim\_steps}} \sum_{k=0}^{\texttt{sim\_steps}-1} |X[k]|^2 \tag{96}$$

$$\sigma_x^2 = |X[k]|^2 = \texttt{sim\_steps} \cdot \texttt{krw}^2 \tag{97}$$

Computation of the spectrum of the output $y[n]$ of figure 22a can be computed as follows, recalling that $x[n]$ is white with energy σ_x^2, and approximating $z = 1 - s\texttt{dt}$ as shown in equation 98. Normalizing $Y(f)$ by the number of samples sim_steps to compute spectral density is performed in 99.

$$|Y(f)|^2 = |X(z)|^2 \frac{1}{|1 - z^{-1}|^2}\bigg|_{z^{-1}=(1-j2\pi f\texttt{dt})} = \frac{\sigma_x^2}{|j2\pi f\texttt{dt}|^2} = \frac{\texttt{sim\_steps} \cdot \texttt{krw}^2}{(2\pi f\texttt{dt})^2} \tag{98}$$

$$|\hat{Y}(f)|^2 = \frac{\texttt{krw}^2}{\texttt{sim\_steps} \cdot (2\pi f\texttt{dt})^2} \tag{99}$$

Given the target phase noise $\mathcal{L}(\Delta f)$ at offset Δf, setting $f = \Delta f$ and $|\hat{Y}(\Delta f)|^2$=$\mathcal{L}(\Delta f)$, a reorganization of equation 99 results in an expression for krw in equation 100. Figure 22b demonstrates a simulated test of this behavioral model, for krw fitted to $\mathcal{L}(\Delta f = 10^6)$ = -80 dBc/Hz.

$$\texttt{krw} = 2\pi\Delta f \cdot \texttt{dt}\sqrt{\mathcal{L}(\Delta f) \cdot \texttt{sim\_steps}} \tag{100}$$

5.6 Divider Behavioral Model

The divider model is defined with the divider ratio `div_n` and only performs a simple division of input phase `x`.

```
class Divider:
  def update(x, div_n):
    return x/div_n
```

Listing 6: Divider behavioral model Python pseudocode.

5.7 Post-Processing: Lock Time Detection

Lock time detection of the PLL start-up transient can be determined from the simulation data conditioned on a tolerance band for acceptable frequency error `lock_f_tol` is provided to the simulator by the user. Since the desired frequency of the PLL `fosc = div_n*fref`, nominal oscillator frequency `dco_f0` and simulation conditions of the PLL are known by the simulator, the ideal value of the loop filter at lock, `lf_lock_ideal` can be computed directly. Given the DCO gain value in the simulation `kdco`, the value of `lf_lock_ideal = round((fosc - dco_f0)/kdco)`. Lock is then detected at the first simulation step for which the current and all later time steps meet the following criteria for the loop filter signal `lf_sig` (i.e. frequency of oscillator is within the frequency tolerance band):

$$\texttt{abs(lf\_sig[n] - lf\_lock\_ideal)*kdco < lock\_f\_tol} \tag{101}$$

This measurement is predicated on the simulation being fully complete at the time of measurement. Lock time is computed by multiplying the simulation index of lock with the simulation time step, `1/fs`. Figure 23 demonstrates the lock detection technique.

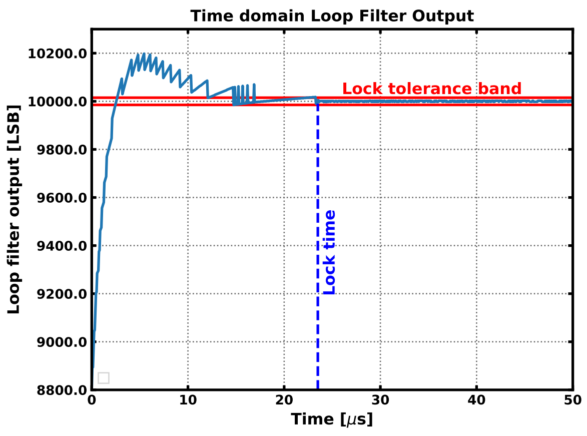


Figure 23: Detection of lock time from simulated PLL transient at loop filter.

5.8 Post Processing: Phase Noise Power Spectrum Estimate

The simulation data can be used to make an estimate of the phase noise power spectrum $\mathcal{L}(\Delta f)$ of the PLL (normalized to the carrier power). First, the phase error signal of the simulated PLL must be computed, `phase_error = div_n*clk_sig - osc_sig`. The phase error signal includes the phase noise signal in addition to any transient phase error components of the PLL. If the simulated PLL is in lock, the phase error is dominated by phase noise components, as the transient components become negligible. If `phase_error` is reduced to the simulation signal span after the detection of lock, with a span in samples of `n_steps`, the power spectral density of the phase noise normalized to the carrier tone, utilizing the fast Fourier transform (FFT) is in equation 102. This definition is based on the FFT implementation available in the `numpy.fft`[3] package.

$$\texttt{psd} = \left| \frac{1}{\texttt{n\_steps}} \cdot \mathcal{FFT}\{\texttt{div\_n*clk\_sig - osc\_sig}\} \right|^2 \tag{102}$$

Provided the simulation sampling rate `fs`, the indices [0, `n_steps/2-1`] of the FFT data and consequently `psd` correspond to frequencies [0, `fbin`·(`n_steps/2-1`)], where `fbin = fs/n_steps`. Slicing the power spectrum data `psd` with indices [1, `fbin`·(`n_steps/2-1`)] will yield the single side band spectrum of the oscillator, with the corresponding offset frequency of each index `k` being $\Delta f \cdot \texttt{k} \cdot \texttt{fbin}$. Thus:

$$\mathcal{L}(\Delta f) = \texttt{psd[round(}\Delta f\texttt{/fbin)]} \tag{103}$$

Computation of power spectrum based on the DFT has high variance about the true spectrum, thus an additional approach utilized here is parametric spectral estimation with autoregressive (AR) modeling [30], which allows for lower variance of the estimate. An AR(p) model is simply a transfer function with p poles, which can be minimum mean square error (MMSE) optimized to fit the spectrum of a signal. The MMSE optimization for the AR model is performed in this work using linear algebra with the Yule-Walker equations, which is based on the autocorrelation sequence of the signal to be fitted (see appendix A). Figure 24 shows example power spectrum estimates utilizing the two implemented methods discussed here on a test data set. It is observed the AR model has low variance.

[3]numpy.fft.fft - NumPy v1.17 Manual, https://docs.scipy.org/doc/numpy/reference/generated/numpy.fft.fft.html.

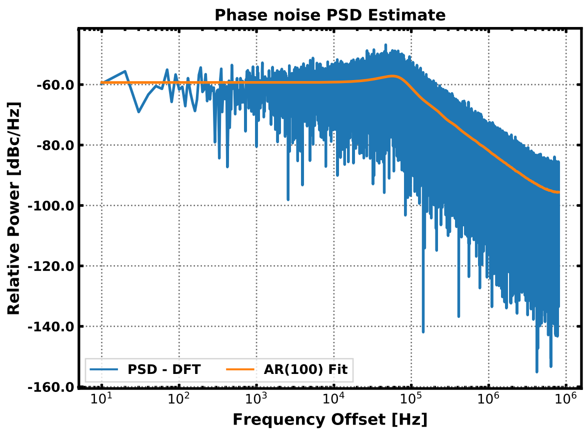


Figure 24: Power spectrum estimates example.

5.9 Monte-Carlo Sampling

The simulation code implemented uses a Python dictionary containing the simulation parameters and the corresponding parameters for which the simulation engine should simulate the PLL. This format makes the introduction of Monte-Carlo sampling into the simulator straightforward. This can be implementing utilizing a dictionary with the nominal simulation configuration, and then a second dictionary to define the parameters to be varied along with the corresponding parameter variance. Given a sample size of N for the Monte-Carlo simulation, a loop is implemented which creates a new simulation configuration dictionary each loop iteration with stochastically sampled values from a normal distribution based on the provided nominal configuration and variance dictionaries. That unique configuration dictionary instance is used to spawn a PLL simulation, and is stored. After N iterations, N sets of data are created with varied parameters.

6 Loop Filter Optimization

The loop filter optimization engine implemented either minimizes (a) total phase noise power, or (b) lock time of a PLL subject to a maximum lock time constraint. The optimizer uses a fixed prototype design for the loop filter in the form of equation 104 arrived at in section 4.2.3 with parameters $\{K_i,\ \omega_p,\ \omega_z\}$. Based on section 4.2.6, the PLL developed in this work will have open loop and normalized and continuous closed loop transfer functions $\mathrm{L}(s)$ and $\hat{\mathrm{T}}(s)$ in equation 105. The parameter K is equivalent to $\frac{\mathrm{M}}{\mathrm{N}}K_{DCO}K_i$, where M is the TDC resolution in steps, N is the divider ratio, K_{DCO} is the DCO gain in Hz/LSB and K_i is the loop filter integral term gain.

$$\mathrm{H}_{LF}(s) = \frac{K_i}{s}\frac{\left(\frac{s}{\omega_z}+1\right)}{\left(\frac{s}{\omega_p}+1\right)} \tag{104}$$

$$\mathrm{L}(s) = \frac{K}{s^2}\frac{\left(\frac{s}{\omega_z}+1\right)}{\left(\frac{s}{\omega_p}+1\right)}, \qquad K = \frac{\mathrm{M}}{\mathrm{N}}K_{DCO}K_i, \qquad \hat{\mathrm{T}}(s) = \frac{\mathrm{L}(s)}{1+\mathrm{L}(s)} \tag{105}$$

The filter optimization and design sequence is detailed below. The following sections cover this in more detail.

1. Optimize parameters $\{K, \omega_p, \omega_z\}$ of $\hat{\mathrm{T}}(s)$ for minimize total PLL phase noise power or lock time, using the phase noise model developed in section 2.3.7 parameterized by the transfer function $\hat{\mathrm{T}}(s)$ and PLL parameters $\{\mathrm{M}, \mathrm{N}, K_{DCO}, S_{0\Phi n_{DCO}}\}$.
2. Translate optimal parameters $\{K, \omega_p, \omega_z\}$ to prototype loop filter parameters $\{K_i, \omega_p, \omega_z\}$, perform continuous to discrete time filter conversion.
3. Optimize fixed point resolution of digital loop filter implementation for finite word effects.

6.1 Phase Noise Optimization Rationale

The ADPLL transfer function in use (equation 105), coupled with the phase noise theory of section 2.3.7 results in an output PLL phase noise spectrum with component-wise breakdown of figure 25a. The shown figure uses an arbitrary selection of PLL and phase noise parameters, however, varying the model parameters will only shift the individual components up and in power and frequency. It is important to note in the presented PLL design, TDC, divider and loop filter phase noise components are manifested with a low pass response, and the DCO noise with a band pass response. Also note the overall PLL response is low pass, owing to the low pass nature of the loop filter. Figure 25b shows the result of integrating the phase noise spectrum to yield total PLL phase noise power for the system modeled in figure 25a. The parameters of the PLL were varied to change the half-power bandwidth of the closed loop response, resulting in a plot of total phase noise power versus PLL loop bandwidth. Here, it is seen that there is an *optimal* selection of bandwidth that minimizes phase noise power. Also, the function that defines the phase noise power-bandwidth relation is *convex*, and thus is easily optimized.

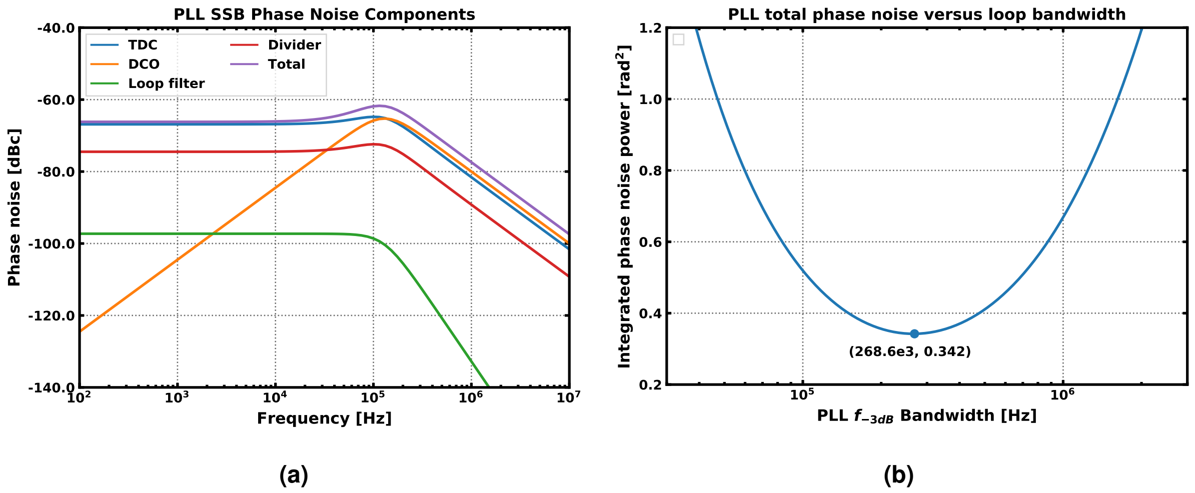


Figure 25: (a) Example PLL phase noise from models in this work, **(b)** integrated phase noise power versus bandwidth for the same PLL.

The take-away here is the PLL will have an optimal set of parameters, which will minimize the overall PLL phase noise. There is then a question of criteria for optimization. In the optimizer introduced here, the TDC and DCO phase noise are assumed to be the principal phase noise components, and are used as the sole criteria for establishing phase noise optimality. This assumption follows that the loop filter and divider implementations can be tuned easily to have less phase noise than DCO and TDC. This will be discussed in more detail later. If the TDC and DCO parameters are kept fixed, but the loop filter is varied resulting in a change of PLL bandwidth, figure 26 illustrates the corresponding changes in the DCO and TDC phase noise components. At low bandwidths, DCO noise because dominant, and at high bandwidths the TDC noise is dominant. In both cases, the total integrated phase noise is high (sub-optimal) due to excessive contributions from either component. Optimal bandwidth occurs when the contributions from both sources are approximately equal. It is possible that the optimal bandwidth for phase noise does not result in the desired transient lock time for the PLL, thus the final criteria for optimality is defined in this work as *the filter configuration that minimizes integrated TDC and DCO phase noise subject to a constraint for maximum lock time.*

6.2 Loop Filter Optimization Algorithm - Phase Noise

Based on the developed criteria for PLL optimality, the following algorithm is defined to establish the optimal parameters $\{K, \omega_p, \omega_z\}$ of $\hat{\mathrm{T}}(s)$. In order to verify phase noise and lock time, algorithms based on the continuous transfer function $\hat{\mathrm{T}}(s)$ have been implemented (see sections

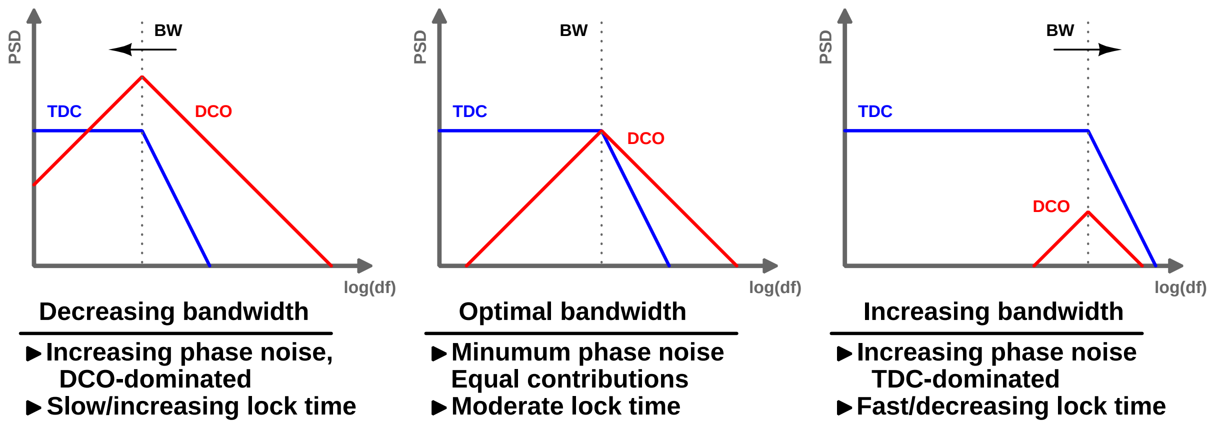


Figure 26: Bandwidth versus total integrated phase noise of PLL.

6.5 and 6.6). Usage of the continuous transfer function $\hat{T}(s)$ for these estimates is done as it is computationally lower complexity than direct time-domain simulation of the ADPLL, thus allowing for faster optimization.

To allow the phase noise minimization and the constraint on maximum lock time to both have an impact in the loop filter optimization, the optimizer is implemented with a divided approach. This is realized with a lower level optimizer (optimizer B), which tunes the loop filter to minimize the total PLL output phase noise power given a fixed target for settling time. There is also a higher level optimizer (optimizer A) that applies a constrained search for settling time that minimizes phase noise, using optimizer B to determine the optimal filter and minimum phase noise at each settling time value. Figure 27 illustrates this algorithm.

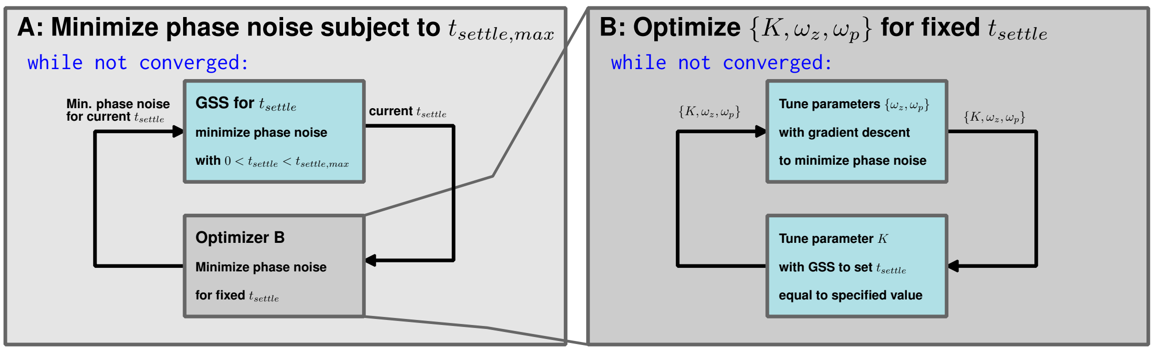

A: Minimize phase noise subject to $t_{settle,max}$
while not converged:
Min. phase noise for current t_{settle}
GSS for t_{settle}
minimize phase noise
with $0 < t_{settle} < t_{settle,max}$
current t_{settle}
Optimizer B
Minimize phase noise
for fixed t_{settle}
B: Optimize $\{K, \omega_z, \omega_p\}$ for fixed t_{settle}
while not converged:
$\{K, \omega_z, \omega_p\}$
Tune parameters $\{\omega_z, \omega_p\}$
with gradient descent
to minimize phase noise
$\{K, \omega_z, \omega_p\}$
Tune parameter K
with GSS to set t_{settle}
equal to specified value

Figure 27: Optimizer visualized.

Optimizer A - Minimize PLL phase noise given a maximal constraint on settling time.

- Provided a maximum settling time $t_{settle,max}$ and PLL parameters $\{$M,N,K_{DCO},$S_{0\Phi n_{DCO}}\}$ one-dimensional golden section search (GSS) [31] is used with settling time t_{settle} as the varied parameter, trying to minimize total phase noise power. The cost function for this optimization is the minimum phase noise for each t_{settle} tried found using optimizer B.

Optimizer B - Minimize PLL phase noise for fixed settling time.

- Provided a fixed target for settling time = t_{settle}, and PLL parameters $\{\mathrm{M}, \mathrm{N}, K_{DCO}, S_{0\Phi n_{DCO}}\}$, evaluate the two following steps until satisfactory convergence of optimization parameters is observed:

  (1) Minimize total phase noise power using pole/zeros locations $\{\omega_p, \omega_z\}$ using gradient descent with a two-point step size method from [32].

  (2) Tune K such that settling time is equal to the fixed target value using unconstrained golden section search.

Convergence for gradient descent, GSS and optimizer B algorithms is decided by comparing the parameter values in each optimizer iteration to the previous. If the normalized change in a given iteration for all optimization parameters is below a tolerance specified by the user, convergence is detected and the optimization ends.

Once the optimal parameters $\{K, \omega_p, \omega_z\}$ for $\hat{\mathrm{T}}(s)$ are determined, the parameter K can be translated onto K_i with equation 106. The loop filter design then can be mapped into a difference equation which is implementable in digital hardware as a direct form-I IIR architecture, as outlined in section 4.2.5.

$$K_i = \frac{\mathrm{M}}{\mathrm{N}} \frac{K}{K_{DCO}} \tag{106}$$

6.3 Optimization of Phase Noise with BBPD

The nonlinear nature of the bang-bang phase detector complicates the optimization of phase noise in a PLL. Based on the BBPD-PLL theory in section 2.4, it is seen that the closed loop transfer function $\hat{\mathrm{T}}(s)$ of the BBPD-PLL is dependent on the variance $\sigma^2_{\Phi_e}$ of the phase error signal which the BBPD is sampling. In steady state, $\sigma^2_{\Phi_e} = \sigma^2_{\Phi_n}/\mathrm{N}$, where $\sigma^2_{\Phi_n}$ is the PLL output phase noise power and N is the PLL divider ratio. To calculate $\sigma^2_{\Phi_n}$, however, requires $\hat{\mathrm{T}}(s)$ to integrate the PLL phase noise PSD determined by section 2.3.7. Therefore, it is seen that $\sigma^2_{\Phi_n}$ and $\hat{\mathrm{T}}(s)$ have a circular definition in steady state. To handle optimization of the loop response $\hat{\mathrm{T}}(s)$ in light of this, the following method is used. First a cost function for the optimization is defined:

Cost function with input $\sigma^2_{\Phi_e}$:

(a) Find optimal PLL response $\hat{\mathrm{T}}_{opt}(s)$ using the method of section 6.2 that minimizes PLL output phase noise power, with the BBPD-PLL transfer function and noise spectral density (from section 2.4) defined using the input value for $\sigma^2_{\Phi_e}$.

(b) Integrate the phase noise power based on $\sigma^2_{\Phi_e}$ and $\hat{\mathrm{T}}_{opt}(s)$, to yield $\hat{\sigma}^2_{\Phi_n}$.

(c) Compute the error between the expected phase noise power from Φ^2_e and that from the phase noise integration, i.e. $\sigma^2_{\Phi_n}$ - $\hat{\sigma}^2_{\Phi_n}$ = N$\sigma^2_{\Phi_e}$ - $\hat{\sigma}^2_{\Phi_n}$

Golden section search is then used with the aforementioned cost function varying parameter $\sigma^2_{\Phi_e}$ to find the optimal $\hat{\mathrm{T}}_{opt}(s)$ for which the PLL is self-consistent, i.e. $\sigma^2_{\Phi_n}$-$\hat{\sigma}^2_{\Phi_n}$ = 0 in the cost function.

6.4 Optimization of Settling Time

The algorithm for optimization of settling time is based on a user specified target value for phase margin PM and a value δ for lock tolerance. δ is given as $f_{tol}/|f_{osc} - \Delta f_i|$, where f_{tol} is frequency tolerance range for lock, f_{osc} is the oscillator frequency, and f_i is the initial frequency error. A PI-controller loop filter is used for the prototype design due to the predictable relationship between the damping ratio ζ of the PLL closed loop response and the PLL phase margin, given in equation 80. A constraint is upheld that closed loop PLL bandwidth $\leq$ 20 times the PLL loop sampling rate $f_s = f_{ref}$, a conservative increase from the 10 times factor previously mentioned from [7], to reduce the risk of instability due to discrete time-conversion effects. Using the PI-controller theory of section 4.2.1, the normalized PLL response $\hat{\mathrm{T}}(s)$ in terms of δ, damping ratio ζ and settling time t_s is:

$$\hat{\mathrm{T}}(s,\zeta,\delta,t_s) = \frac{-2s\frac{\ln(\delta)}{\zeta^2 t_s} + \frac{\ln^2(\delta)}{\zeta^2 t_s^2}}{s^2 - 2s\frac{\ln(\delta)}{\zeta^2 t_s} + \frac{\ln^2(\delta)}{\zeta^2 t_s^2}} \tag{107}$$

The optimal selection of ζ for the specified phase margin can be determined from equation 80 with numerical root finding. The following two equations are then used to determine the optimizal settling time $t_{s_{opt}}$. Equation 108 determines the PLL closed loop half-power bandwidth, and equation 109 determines the settling time based corresponding settling time $t_{s_{opt}}$ for the maximum allowed bandwidth of $f_{ref}/20$. The optimal values of $t_{s_{opt}}$ and ζ may be translated to PI-controller parameters in equations 76-79, which is discretized using section 4.2.5.

$$\mathrm{BW}(\zeta,t_s,\delta) = \arg\min_f \left|\frac{1}{2} - \left|\hat{\mathrm{T}}(f,\zeta,t_s,\delta)\right|^2\right| \tag{108}$$

$$t_{s_{opt}}(\zeta,t_s,\delta) = \arg\min_f \left|\frac{f_{ref}}{20} - \mathrm{BW}(\zeta,t_s,\delta)\right| \tag{109}$$

6.5 Fast Estimation of PLL Settling Time

The following is the method implemented to estimate settling time from the PLL closed loop phase transfer function $\hat{\mathrm{T}}(s)$. $\hat{\mathrm{T}}(s)$ can be represented as a rational function of two polynomial functions of s, with P poles and Z zeros. Such a transfer function is computationally represented with arrays A and B, containing the denominator and numerator polynomial coefficients.

$$\hat{\mathrm{T}}(s) = \frac{\sum_{j=0}^{Z} b_j s^j}{\sum_{k=0}^{P} a_n s^n} \tag{110}$$

An estimate of the step response settling time of $\mathrm{T}(s)$ can by utilizing its representation in state space [33]. This is given in equations 111 and 112, with input vector $\mathrm{U}(s)$, state vector $\mathbf{X}(s)$, and output $Y(s)$. The state-space representation from a s-domain transfer function can be quickly solved computationally with available signal processing packages, such as `scipy.signal`[4] used in this work, given the coefficient arrays A and B.

$$s\mathbf{X}(s) = \mathbf{A}\mathbf{X}(s) + \mathbf{B}\mathrm{U}(s) \tag{111}$$

$$Y(s) = \mathbf{C}\mathbf{X}(s) + \mathbf{D}\mathrm{U}(s) \tag{112}$$

The set of k eigenvalues $\{\lambda_1, ..., \lambda_N\}$ corresponding to poles for the system are found as the roots of equation 113 [34]. Matrix $\mathbf{A}$ is referred to as the state matrix.

$$|\mathbf{A} - \lambda\mathbf{I}| = 0 \tag{113}$$

Imposing the constraint of number of poles P $>$ number of zeros Z, the system $\mathrm{T}(s)$ may be represented via partial fraction decomposition using the poles from the eigenvalues of state matrix $\mathbf{A}$ $\{\lambda_1, ..., \lambda_N\}$:

$$T(s) = \sum_{k=1}^{P} \frac{c_k}{s - \lambda_k} \tag{114}$$

Inverse Laplace transformation shows the step response of the system will be a sum of exponentials:

$$y(t) = c_1 e^{\lambda_1 t} + ... + c_k e^{\lambda_k t} \tag{115}$$

The dynamics of the step response are governed by the exponential components of y(t). If $\{\lambda_1, ..., \lambda_N\} \in \mathbb{C}$ where $\lambda_k = 1/\tau_k + j\omega_K$, the real portion of each λ_k will describe the transient behavior of each exponential, having time constant τ_k. The long term settling of y(t) will be dominated by the exponential with the largest τ_k, i.e. the dominant pole of the system. This estimate of settling time uses the dominant pole τ_k as a heuristic estimate for overall time constant of the system, τ. Finally, settling time t_s can be considered as the time interval required for the signal to drop within a tolerance band $\pm\delta_{tol}\mathrm{y}(\infty)$ about the final value $\mathrm{y}(\infty)$. Thus:

$$t_s = \tau \ln(\delta_{tol}) = \frac{\ln(\delta_{tol})}{\min(|\Re(\{\lambda_1, ..., \lambda_k\})|)} \tag{116}$$

This settling time estimate is computationally fast, as it requires only (a) computation of state matrix $\mathbf{A}$ from the coefficients of $\hat{\mathrm{T}}(s)$, (b) computation of the eigenvalues of $\mathbf{A}$, and (c) computation of settling time from the eigenvalue with minimum real component.

[4]Signal processing (scipy.signal) - SciPy v1.3.3 Reference Guide, https://docs.scipy.org/doc/scipy/reference/signal.html.

6.6 Fast Estimation of PLL Phase Noise

The following is the method implemented to estimate total integrated phase noise power from the PLL closed loop phase transfer function $\hat{\mathrm{T}}(s)$, and PLL parameters $\{\mathrm{M}, \mathrm{N}, K_{DCO}, S_{0\Phi n_{DCO}}\}$. As has been stated, this optimizer assumes the dominant output-referred phase noise contributions of the PLL are due to the DCO thermal noise and TDC quantization. If S_{TDC} and S_{DCO} are the PLL output-referred noise PSD respectively for the TDC and DCO noise sources from section 2.3, the total PLL output noise PSD $S_\Sigma(f)$ is estimated as equation 117.

$$S_\Sigma(f) = S_{\Phi n_{TDC,out}}(f) + S_{\Phi n_{DCO,out}}(f) \tag{117}$$

$$= \frac{1}{12 f_{ref}} \left| 2\pi \frac{\mathrm{N}}{\mathrm{M}} \hat{\mathrm{T}}(f) \right|^2 + \frac{S_{0\Phi n_{DCO}}}{f^2} \left| 1 - \hat{\mathrm{T}}(f) \right|^2 \tag{118}$$

Given a bandwidth of interest Δf (i.e. baseband bandwidth for radio applications), the total integrated phase noise power is:

$$P_{\phi noise} = 2 \int_0^{\Delta f} S_\Sigma(f) df \tag{119}$$

This can be computed via numerical integration on a grid of K values in the interval Δf, where each point represents a frequency bin f_{bin} = Δf/K. The Romberg method for numerical integration [35] is utilized in this work as it has high accuracy for smooth integrands.

6.7 Loop Filter Optimization - Finite Word Effects

Once a filter design has been optimized in the continuous domain following the algorithm of section 6.2, post-filter design optimization is carried out to ensure the digitized, discrete implementation performs as expected in the presence of finite word effects. The implemented post-optimization considers both the effect of loop filter quantization noise and transfer function error due to quantization.

This optimizer works by converting the ideal discrete filter design into a fixed point implementation with direct form-I structure. All components of the data path in the optimization are constrained to have the same signed fixed point format, in two's complement format with 1 sign bit, a number of integer bits `int_bits`, and a number of fraction bits `frac_bits`. Thus the total data word representation bits is B = 1 + `int_bits` + `frac_bits`. The optimization is constrained to a range of bits for data word size specified by the user, for which the number of bits will be found that yields the minimum mean squared error (MMSE) for filter accuracy and quantization noise.

Selection of `int_bits` is determined by considering the integer part of the discrete filter coefficients $\{a_1, a_2, b_0, b_1\}$. If the integer portions are divided into positive and negative valued sets pos_ints and neg_ints, the number of integer bits is therefore:

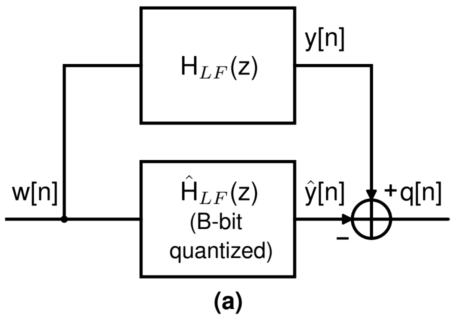
$$\texttt{pos\_int\_bits} = \lfloor \log_2 \left(\max\left(\lfloor \texttt{pos\_ints} \rfloor\right)\right) \rfloor + 1 \tag{120}$$

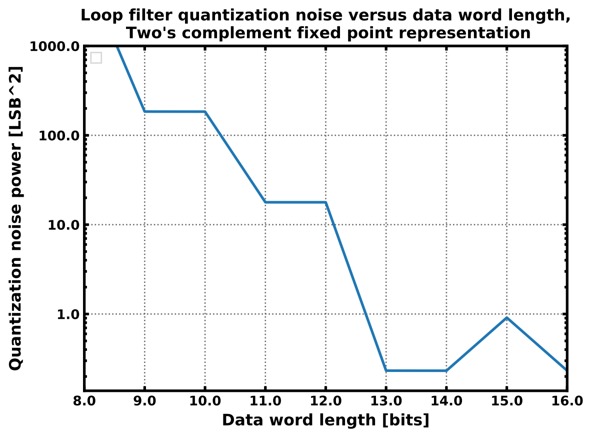
$$\texttt{neg\_int\_bits} = \lceil \log_2 \left(\max\left(\lfloor \texttt{neg\_ints} \rfloor\right)\right) \rceil \tag{121}$$

$$\texttt{int\_bits} = \max(\{\texttt{pos\_int\_bits}, \texttt{neg\_int\_bits}\}) \tag{122}$$

6.7.1 Loop Filter Quantization Noise Optimization

Quantization noise for the loop filter design is determined by discrete-time simulation with of the unquantized representation and the quantized representations of the filter for a range of data word bits constrained by the user. The input signal w[n] used is a Gaussian-distributed white noise signal, whose variance is significantly larger than the maximum loop filter LSB size tested. The quantization error signal q[n] is then computed for each quantized realization utilizing the approach in figure 28a. The power (variance) of the quantization noise signal q[n] for each filter realization is then computed. Figure 28b shows the result of the aforementioned for a hypothetical loop filter design; quantization power is seen to decrease significantly with data word resolution.





(b)

Figure 28: (a) Model for determining quantization noise of loop filter, **(b)** Quantization noise power power out of an example loop filter versus data word resolution.

The optimal number of bits is chosen on a constraint that loop filter noise components should be less than the detector noise components. It was observed in figure 25a that for the loop filter prototype design used in this work, the phase noise spectral densities of the detector (TDC in the figure) and the loop filter are both low pass shaped at the PLL output. Thus as the loop filter noise is reduced by increasing the number of data word bits, it is possible to reduce the loop filter entirely noise below that of the detector noise. The strategy employed here to calculate this is to refer the detector noise via closed loop transfer function calculation to the loop filter output node, and then to integrate the total TDC phase noise power observed at that node. The spectral density of TDC noise at the loop filter output node is in equation 123, and for the BBPD noise it is equation 124.

$$S_{\Phi n_{TDC,LFout}}(f) = \frac{1}{12 f_{ref}} \left| \frac{\mathrm{N}}{\mathrm{M}} \frac{2\pi f}{K_{DCO}} \hat{\mathrm{T}}(f) \right|^2 \tag{123}$$

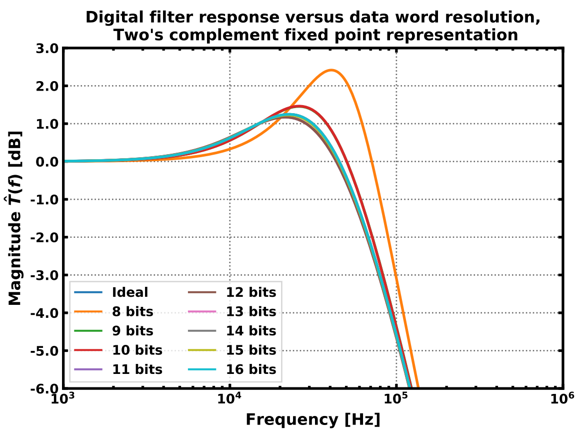
$$S_{\Phi n_{BBPD,LFout}}(f) = \frac{\left(\frac{\pi}{2} - 1\right)}{f_{ref}} \left| \sigma_{\Phi_e} \mathrm{N} \frac{f}{K_{DCO}} \hat{\mathrm{T}}(f) \right|^2 \tag{124}$$

Evaluation of the phase noise in this work is using the Romberg method for numerical integration within a bandwidth defined by the PLL reference frequency f_{ref}. The resulting TDC noise power at the loop filter output $\sigma^2_{n_{TDC,LFout}}$ can then be used to define a cutoff threshold for maximum loop filter output noise. If a maximum degredation of of the phase noise is given by a loop filter noise figure NF_{LF} in dB, then the cutoff threshold for loop filter quantization noise power σ^2_{cutoff} is equation 125. The minimum number of filter realization bits from the quantization error simulation meeting this cutoff requirement is the optimal word size selection.

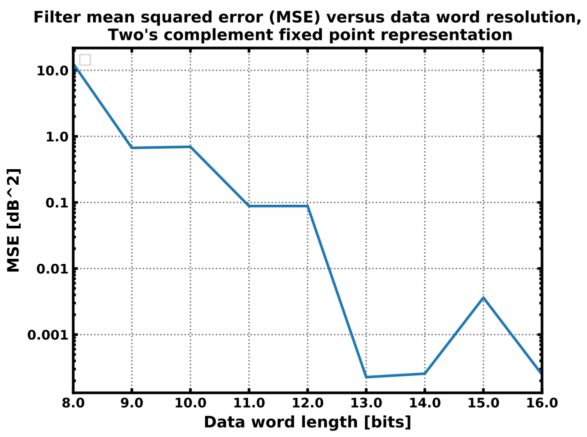
$$\sigma^2_{cutoff} = \sigma^2_{n_{TDC,LFout}} \cdot \left(10^{\mathrm{NF}_{LF}/10} - 1\right) \tag{125}$$

6.7.2 Loop Filter Transfer Function Error Optimization

Rounding of the discrete PLL filter coefficients changes the response of the implemented digital filter. Figure 29a demonstrates this effect for a hypothetical selection of parameters, where filter error increases with decreasing number of bits used for the digital data word size. Optimization here is based on minimum mean square error (MMSE) optimization of the filter response for a range of quantized filter realizations defined by the user (in terms of minimum and maximum number of data word bits). Given a target value for MMSE by the user, the optimizer computes each quantized transfer function realization having B bit data words, and the frequency domain response $\hat{\mathrm{H}}_{LF}(f|\mathrm{B})$ within a bandwidth equal to $f_{ref}/2$. Then then ideal ideal, unquantized $\mathrm{H}_{LF}(f)$ is used to compute the mean squared error for each realization, i.e. $\mathbb{E}[(\mathrm{H}_{LF}(f) - \hat{\mathrm{H}}_{LF}(f|\mathrm{B}))^2]$. Figure 29b illustrates such a computation for hypothetical loop filter parameters. The optimal loop filter selection is then the minimum number of bits which satisfies the user's specified MMSE value.



(a)



(b)

Figure 29: (a) Example filter error due to coefficient quantization, **(b)** Example MSE error of filter design due to coefficient quantization.

6.7.3 Final Optimal Loop Filter Digital Word Size

Once the optimal word size for both quantization and filter error has been computed, the largest of the two is taken as the final value for optimal word size.

7 Results and Discussion

In this discussion, the performance of the presented design solution will first be evaluated with a design example. Then, a comparison of the presented solution will be made to existing solutions in literature, pointing out advantages and disadvantages of this work. Finally, a general discussion will be made covering areas of improvement, reasoning for the central design choices made, and considerations for usage the framework.

7.1 Design Exercise Using This Work

The design of a loop filter for the PLL with the system level specifications of table 1 will be considered here, with the intent of optimizing total phase noise power (notated in this discussion as residual phase modulation). These specifications, where the TDC resolution in steps equals the divider ratio, is equivalent to a special case of a TDC-less PLL where a 150-step synchronous counter is used as a divider, and the loop filter directly samples the state of the synchronous counter. For the loop filter design, a PI-controller loop filter prototype was utilized in the optimizer due to its predictable phase margin and stability characteristics. Results of two design approaches with this framework will be presented. Approach 1 minimizes the residual phase modulation based on the TDC-phase detector PLL model. The BBPD gain in this case is optimized to minimize phase noise spectrum error in steady state for that expected from the TDC-PLL model. Approach 2 demonstrates loop filter optimization for both fast locking and low phase noise using loop filter gear switching. The first gear loop filter is optimized for minimum lock time using TDC feedback, and the second gear is optimized to minimize residual phase modulation in steady state using BBPD feedback.

Parameter	Value	Unit
Output Frequency	2.4	GHz
Ref. frequency	16	MHz
Divider ratio	150	
TDC resolution	150	steps/reference cycle
DCO gain K_{DCO}	10^4	Hz/LSB
DCO Phase noise	-80	dBc/Hz at $\Delta f = 10^6$ Hz
Lock Time	$\leq$ 25	μs
Lock Δf tolerance	10^5	Hz
Digital filter word resolution	$\leq$ 16	bits
Residual phase modulation	minimize	

Table 1: System-level specifications

7.1.1 Approach 1: Results of Filter Optimization.

Table 2 contains the optimized filter parameters obtained from the filter design optimizer developed in this work using the TDC based PLL model. $\{K, K_i, K_p, f_z\}$ are the continuous model parameters of section 4.2.1, and $\{a_0, a_1, a_2, b_0, b_1\}$ are the filter coefficients for the discrete-time direct form-I filter structure of section 4.2.5. The estimated bandwidth for the filter is 144.8 kHz, and the lock time is estimated to be 19.3 μs. Table 3 contains the digitized version of the discrete time filter, based on a selection for word size determined via optimization for finite word effects. The final data words are 13 bits in length.

Parameter	Value	Unit
K	1.343792×10^{11}	
K_i	1.343792×10^{7}	
K_p	7.331074×10^{1}	
f_z	2.917324×10^{4}	Hz
b_0	$7.4150613906 \times 10^{1}$	
b_1	$-7.3310743796 \times 10^{1}$	
a_0	1.0×10^{0}	
a_1	-1.0×10^{0}	
a_2	0.0×10^{0}	
K_{bb}	3.007953×10^{-2}	
Estimated bandwidth	1.448234×10^{5}	Hz
Estimated lock time	1.934253×10^{-5}	seconds

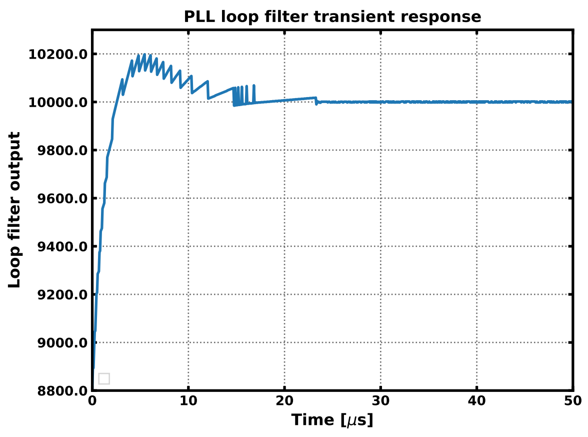
Table 2: PLL parameters determined from filter design and optimization process.

Parameter	Value	Value (digital)	Value Error
Total dataword bits	13		
Sign bits	1		
Integer bits	7		
Fractional bits	5		
b_0	7.415625×10^{1}	`0b0100101000101`	$+5.636094 \times 10^{-3}$
b_1	-7.331250×10^{1}	`0b1111011010110`	-1.756204×10^{-3}
a_0	1.0×10^{0}	`0b0000000100000`	0.0×10^{0}
a_1	-1.0×10^{0}	`0b1111111100000`	0.0×10^{0}
a_2	0.0×10^{0}	`0b0000000000000`	0.0×10^{0}
K_{bb}	3.125×10^{-2}	`0b0000000000001`	$+1.170461 \times 10^-3$

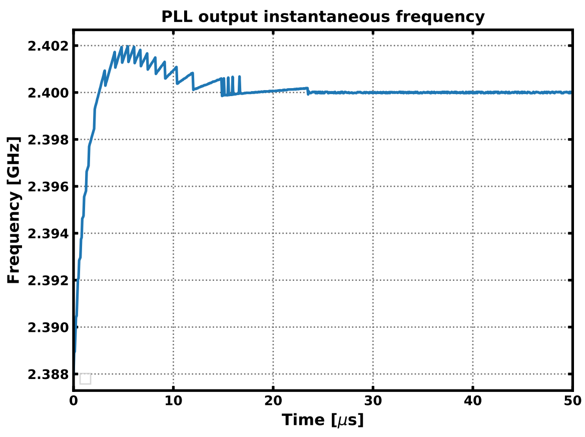
Table 3: Loop filter parameters after digitization and optimization for data word length.

7.1.2 Approach 1: Results of Transient and Phase Noise Simulation.

The simulation engine implemented in this work was utilized to run a time domain simulation to verify the designed filter. Figures 30a and 30b demonstrate the transient response of the PLL with an initial frequency error of 12 MHz (0.5% of final frequency). It is observed that the PLL design stably approaches the target frequency in approximately 23 μs. Figures 31a illustrate the BBPD and TDC outputs during this initial transient. It is observed that the TDC output gives the dominant feedback, until it reaches an output word of 0, where the BBPD then begins providing feedback in the steady state condition of the PLL. Figure 31b is the result of a phase noise calculation for the simulated PLL in an interval beginning immediately after detection of lock. The spectrum closely approximates that designed for, with some small additional peaking.

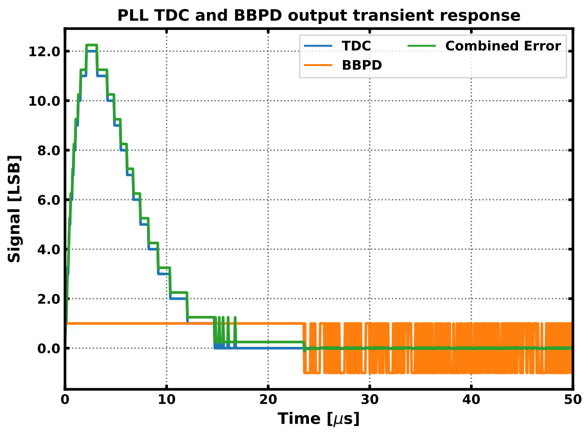


(a)

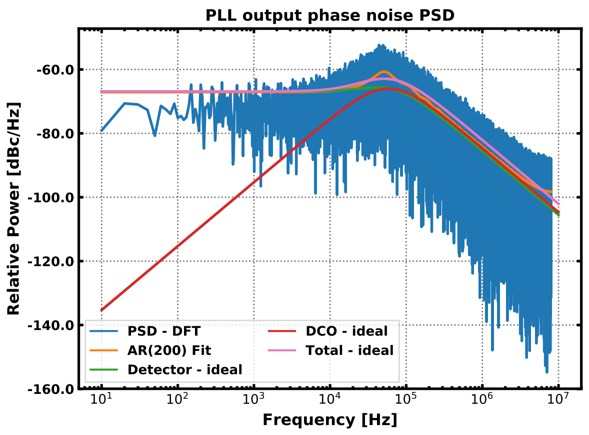


(b)

Figure 30: Simulation with 0.5% initial frequency error: **(a)** Loop filter transient response, **(b)** PLL output instantaneous frequency.



(a)

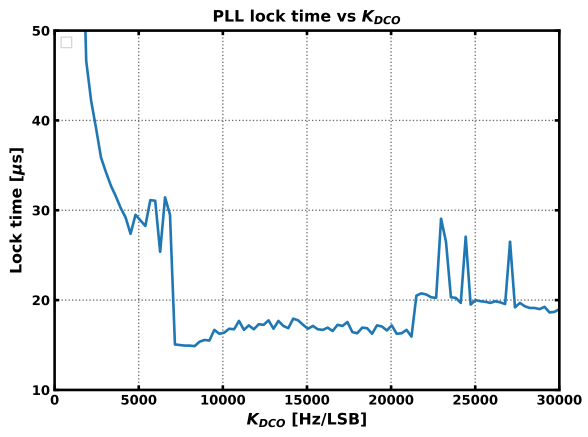


(b)

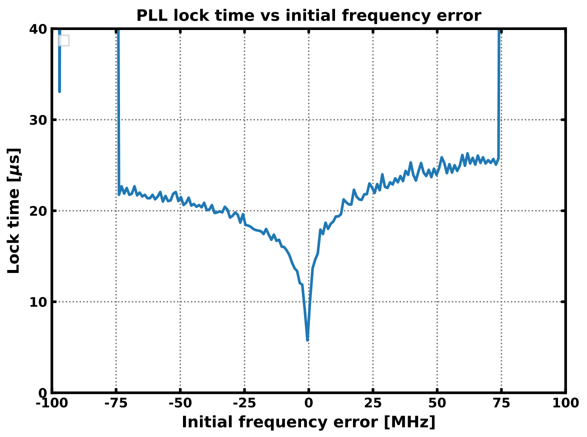
Figure 31: Simulation with 12 MHz (0.5%) initial frequency error: **(a)** BBPD/TDC detector responses, **(b)** PLL output phase noise power spectrum.

7.1.3 Approach 1: Results of Parameter Sweep and Variational Analysis.

The Monte-Carlo sampling and parametric sweep capabilities of the simulator designed in this work were used to run analysis for effects of variation of DCO gain K_{DCO} and initial frequency error of the PLL. Figure 32a shows a parametric sweep of K_{DCO}, with lock time being measured. It is observed that the lock time is nearly flat in the range 7100-21200 Hz/LSB, meaning that there is a tolerance of -2900/+11200 Hz LSB for K_{DCO} about the nominal 10000 Hz/LSB specified. This specification can be utilized in the design of a physical DCO to constrain maximum acceptable variation across PVT. Figure 32b demonstrates the simulated effect of initial frequency error on lock time. It is seen that PLL stably locks to the target frequency with an initial frequency error within the interval of $\pm$ 74 MHz from 2.4GHz, implying that the capture range of the PLL is 148 MHz. This specification can be translated into a requirement for initial coarse frequency calibration needed before starting the PLL. Figures 33a and 33b are the results of a variation analysis simulation utilizing the Monte-Carlo sampling engine implemented in this work. The simulation was configured to vary K_{DCO} with a standard deviation of 20 % of the nominal value, and to vary the initial starting frequency with a standard deviation of 60 MHz (2.5 % of the final frequency). The simulation sample size is 1000. The transient responses from the simulation figure 33a are all stable, and figure 33b shows the histogram for measured lock time subject to the described variation. The mean lock time was 19.33 μs, which correlates well with the estimated 19.34 μs from the filter design process and meets the 25 μs lock time set for the PLL. The upper bound for a 99% confidence interval on the data is 34.75 μs. A set of extracted parameters from these simulations are in table 4. The Monte-Carlo simulations presented here are useful to analyze the range of variation in which the designed PLL can tolerate, as well as determine the expected performance variation, so well informed decisions on a PLL design can be made before moving into transistor level implementation and simulation.

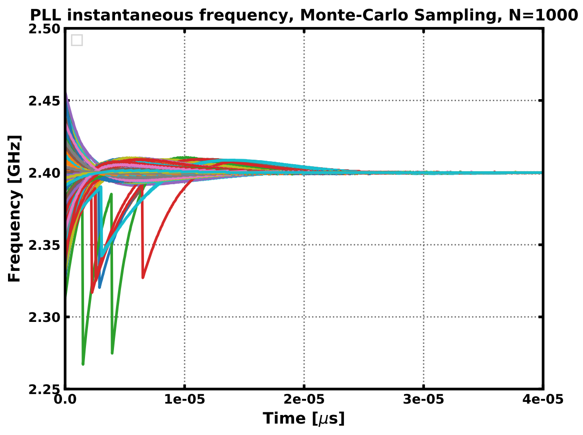


(a)

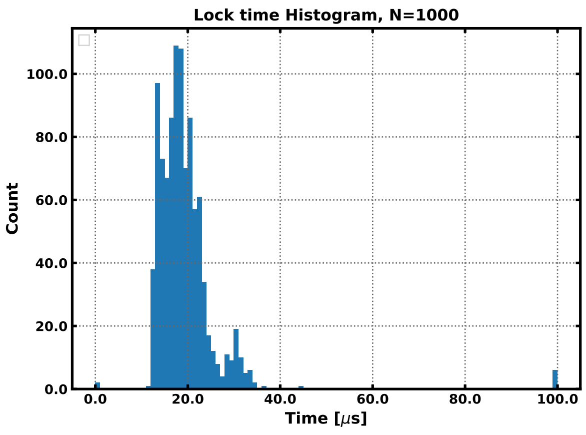


(b)

Figure 32: (a) PLL lock time simulation with KDCO swept, 12 MHz (0.5%) initial frequency error, **(b)** PLL lock time simulation with initial frequency error swept.



(a)



(b)

Figure 33: Monte-Carlo simulation with 1000 samples, 20% RMS deviation in KDCO, and 60 MHz (2.5%) RMS deviation in initial frequency error **(a)** Frequency transient responses, **(b)** Lock time histogram.

Parameter	**Value**	**Unit**
K_{DCO} **Tolerance**	-2900/+11200	Hz/LSB
Capture range	148 ($\pm$74)	MHz
Mean lock time	19.32769	μs
Lock time σ	7.802302	μs
Lock time 99 % CI upper bound	34.75	μs
Residual phase modulation	3.374731×10^{-1}	rad^2

Table 4: PLL parameters extracted from variance and parameter sweep simulations.

7.1.4 Approach 2: Results of Filter Optimization

The same PLL design has been approached with a second optimization approach utilizing gear switching in the loop filter. The first gear loop filter is optimized for lock speed, using TDC feedback, so high initial frequency and phase errors may be tolerated. After initial lock is achieved with the first filter, the filter is switched gears by changing the digital filter coefficients immediately to a second filter optimized for minimum total phase noise using a BBPD. This combination allow both fast lock and low phase noise. The results of the optimization process are in tables 5 and 6, and the optimal digitized filter designs for finite word effects are in table 7. The total lock time is estimated to be 5.52 μs. The method for determining number of bits in this work currently assumes the decimal point in the fixed point representation does not change position, so here a total data word size of 20 bits was arrived at, albeit the individual optimization for gear 1 and gear 2 resulted in 10 and 16 bits respectively. This optimization could be improved by considering a filter datapath architecture that handles changing of the decimal position in the loop filter data words during gear switching.

Parameter	Value	Unit
K	2.982197×10^{12}	
K_i	2.982197×10^{8}	
K_p	2.115052×10^{2}	
f_z	2.244064×10^{5}	Hz
b_0	$2.3014397180 \times 10^{2}$	
b_1	$-2.1150524223 \times 10^{2}$	
a_0	1.0×10^{0}	
a_1	-1.0×10^{0}	
a_2	0.0×10^{0}	
Estimated bandwidth	5.333423×10^{5}	Hz
Estimated lock time	4.527067×10^{-6}	seconds

Table 5: PLL parameters determined from filter design and optimization process for fast lock speed with TDC feedback.

Parameter	Value	Unit
K	5.325862×12^{12}	
K_i	1.271456×10^{10}	
K_p	1.101813×10^{4}	
f_z	1.836596×10^{5}	Hz
b_0	$1.1812790734 \times 10^{4}$	
b_1	$-1.1018130778 \times 10^{4}$	
a_0	1.0×10^{0}	
a_1	-1.0×10^{0}	
a_2	0.0×10^{0}	
K_{bb}	1.0×10^{0}	
Estimated bandwidth	9.117332×10^{5}	Hz
Estimated lock time	9.978130×10^{-7}	seconds

Table 6: PLL parameters determined from filter design and optimization process for minimum phase noise with BBPD.

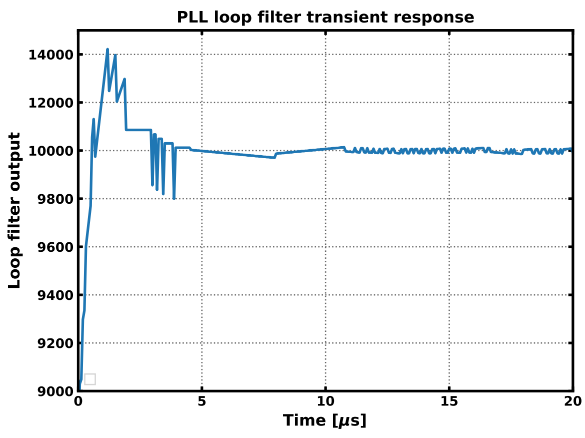
7.1.5 Approach 2: Results of Transient and Phase Noise Simulation

It is observed that the fast lock gear converges the PLL to steady state in approximately 6 μs as seen in the transient step simulation in figures 34a and 34b, where an initial frequency error of 0.5% 12 MHz is used. Figure 35a shows the output of the TDC and BBPD, it is seen that BBPD feedback has a high density at approximately 11 μs, implicating steady state conditions. Finally, the computed phase noise spectrum of figure 35b demonstrates that there is some discrepancy between the ideal designed response and that simulated, albeit the bandwidth appears correct.

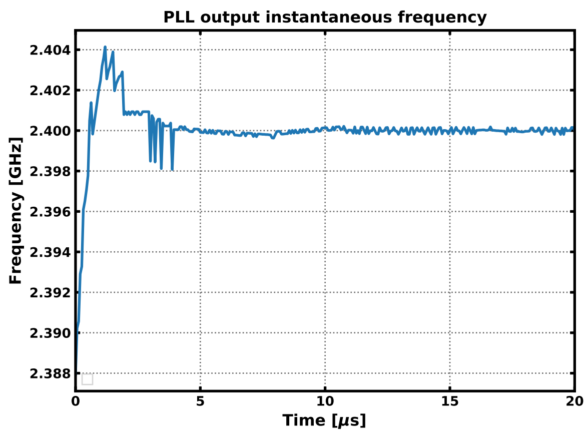
Parameter	Value	Value (digital)	Value Error
Total dataword bits	20		
Sign bits	1		
Integer bits	8		
Fractional bits	11		
b_0 (gear 1)	2.301440×10^2	`0b01110011000100100111`	$+7.116930 \times 10^{-5}$
b_1 (gear 1)	-2.115054×10^2	`0b11110110001111110101`	-1.288619×10^{-4}
a_0 (gear 1)	1.0×10^0	`0b00000000100000000000`	0.0×10^0
a_1 (gear 1)	-1.0×10^0	`0b11111111100000000000`	0.0×10^0
a_2 (gear 1)	0.0×10^0	`0b00000000000000000000`	0.0×10^0
K_{bb} (gear 1)	0.0×10^0	`0b00000000000000000000`	0.0×10^0
b_0 (gear 2)	8.899902×10^0	`0b00000100011100110011`	$+4.616306 \times 10^{-5}$
b_1 (gear 2)	-8.301270×10^0	`0b11111011110110010111`	-1.168639×10^{-4}
a_0 (gear 2)	1.0×10^0	`0b00000000100000000000`	0.0×10^0
a_1 (gear 2)	-1.0×10^0	`0b11111111100000000000`	0.0×10^0
a_2 (gear 2)	0.0×10^0	`0b00000000000000000000`	0.0×10^0
K_{bb} (gear 2)	1.0×10^0	`0b00000000100000000000`	1×10^0

Table 7: Loop filter parameters after digitization and optimization for data word length, gear 1 and gear 2.

The phase noise spectrum discrepancy is likely due to the model used in the optimization for the BBPD, which uses a linearized model of the BBPD with only idealized DCO and detector noise considered. This does not accurately account for all quantization and discretization emergent behaviors in the discrete time behavioral simulation possibly being manifested here.

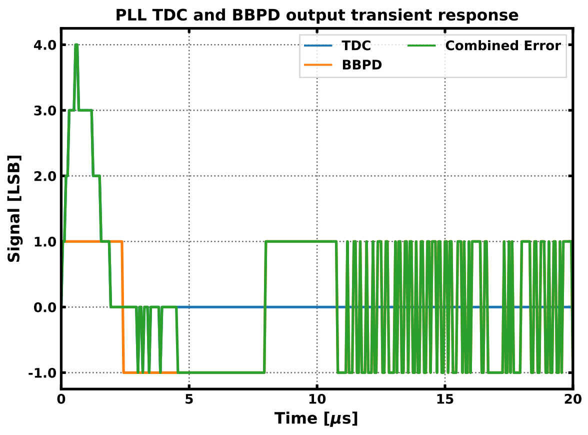


(a)

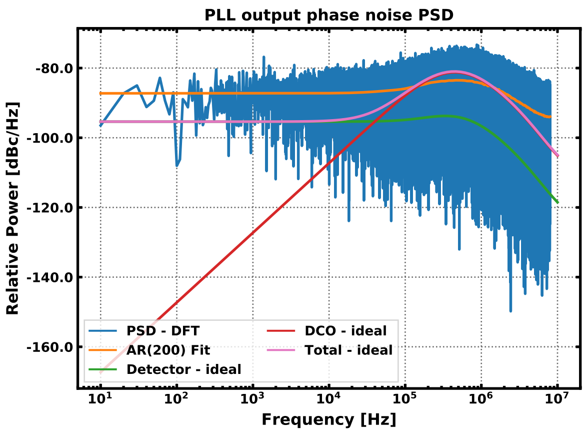


(b)

Figure 34: Simulation with 0.5% initial frequency error: **(a)** Loop filter transient response, **(b)** PLL output instantaneous frequency.



(a)

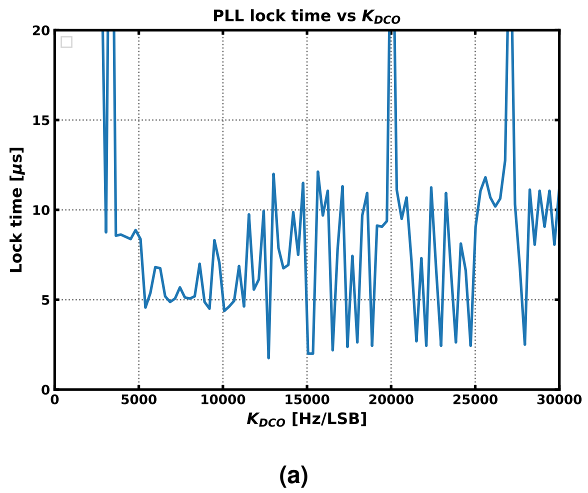


(b)

Figure 35: Simulation with 12 MHz (0.5%) initial frequency error: **(a)** BBPD/TDC detector responses, **(b)** PLL output phase noise power spectrum.

7.1.6 Approach 2: Results of Parameter Sweep and Variational Analysis

The same parameter sweeps for K_{DCO} and initial frequency error of approach 1 were applied here. In figures 36a and 36b, the results of these sweeps are shown. The loop filter gear switching mechanism as modeled in the simulation waits a fixed time interval before switching gears, in this case chosen to be 2.0 times the estimated lock time given in table 5. This interval has provided sufficient settling within the simulated parameter range of K_{DCO} and initial frequency error that lock time stability is observed under most conditions, except under high offset for K_{DCO} and initial frequency error. It was established for this simulation results that tolerance range for K_{DCO} is -6950/+9750 Hz/LSB from the nominal 10000 Hz/LSB value, and the capture range is 168 MHz. Figures 37a and 37b demonstrate a variational simulation of the PLL using Monte-Carlo sampling, with 1000 samples. The simulation was configured to vary K_{DCO} with a standard deviation of 20 % of the nominal value, and to vary the initial starting frequency with a standard deviation of 60 MHz (2.5 % of the final frequency). It was observed that the PLL stably locked for all simulation instances, a mean lock time of 5.96 μs was achieved. This value correlates well with the estimate of 5.52 μs from the filter design process. The upper bound for a 99% confidence interval of lock time is 11.625 μs, thus meeting the 25μs lock time specification with considerable margin. The extracted PLL performance parameters from these simulations of this gear-switching PLL is in table 8.



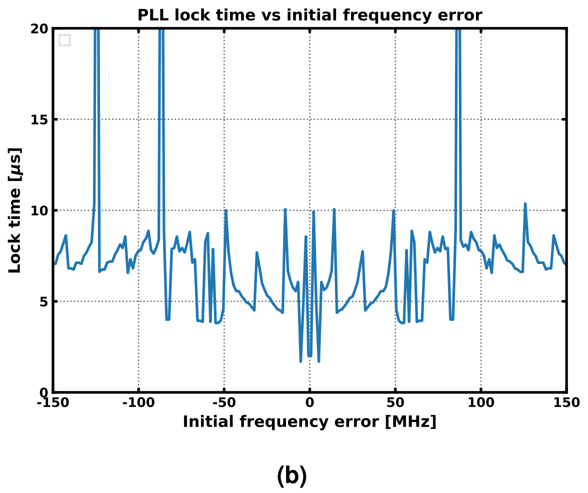
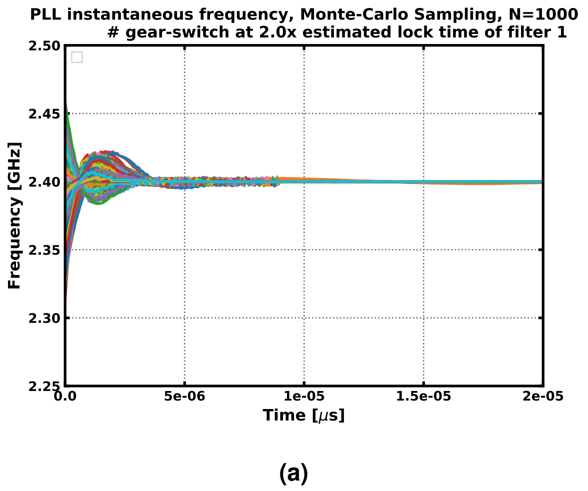


Figure 36: **(a)** PLL lock time simulation with KDCO swept, 12 MHz (0.5%) initial frequency error, **(b)** PLL lock time simulation with initial frequency error swept.



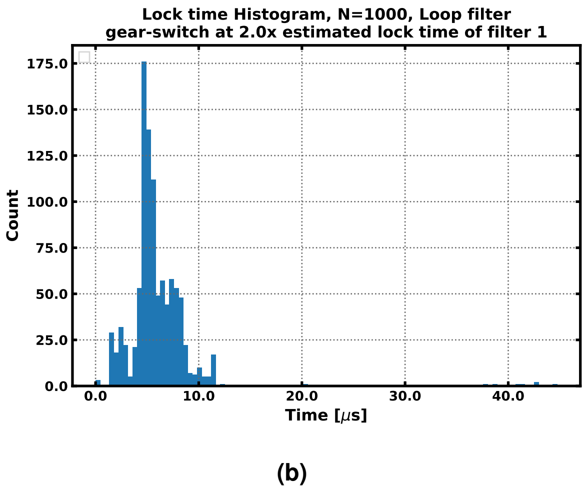


Figure 37: Monte-Carlo simulation with 1000 samples, 20% RMS deviation in KDCO, and 60 MHz (2.5%) RMS deviation in initial frequency error **(a)** Frequency transient responses, **(b)** Lock time histogram.

Parameter	Value	Unit
K_{DCO} **Tolerance**	-6950/+9750	Hz/LSB
Capture range	168 (±84)	MHz
Mean lock time	5.961688	μs
Lock time σ	3.611130	μs
Lock time 99 % CI upper bound	11.625	μs
Residual phase modulation	4.367535×10^{-2}	rad^2

Table 8: PLL parameters extracted from variance and parameter sweep simulations.

7.1.7 Results Comparison of Design Approaches

A comparison of the two design approaches shows that the PLL utilizing the gear switching results in a superior result using the design automation framework of this work. The usage of gear switching results in lower phase noise than approach 1, with total phase noise power (residual phase modulation) of 4.36×10^{-2} rad^2 versus 3.37×10^{-1} rad^2. Additionally, the lock time is significantly lower in the PLL with gear switching, at an average of 5.96 μs versus 19.32 μs. Therefore it has been shown that this framework offers the capability to design gear-switching PLLs with performance advantages to static loop filter PLL designs.

7.2 Comparison to Existing Solutions

Due to the relative youth of all digital PLL design, and discontinuity between the continuous theory of analog PLL and digital PLL design, a smaller body of work exist pertaining to the loop filters for exclusively ADPLLs. Of the existing literature, the majority design approaches utilize discrete-time transformed PI-controller loop filters with either bang-bang phase detectors [22][23][36][37] or phase-frequency detectors [38][39]. This work takes a similar approach to the aforementioned existing works, focusing on a fixed filter architecture to limit the scope of the problem at hand, and also uses a bang-bang phase detector in its PLL architecture. This work, though, uniquely considers the optimization of digital loop filter design to use a TDC and BBPD in conjunction. Also, the approach to filter design here, which is through numerical methods to find an optimal configuration, differs from the other works that largely focus on closed-form mathematical analysis. The usage of numerical methods has allowed for implementation in this work of (a) full automation of the loop filter design process, including generation of optimal digitized filter coefficients, and (b) automatic verification through the simulator integrated into this filter design framework. These aspects of this work reduces overall implementation work for the PLL designer, and are not paralleled by any existing literature or openly available code bases.

The criteria and motivation for loop design varies across the surveyed works. Of the surveyed works, several [36][38][39][37] do not consider optimization for phase noise as this work principally does, but rather consider stability/phase margin [36][38][37], lock time [39][37] and even approach simplicity [38]. The methods that consider optimization of phase noise [22][23] both provide a similar model of PLL dynamics based on a linearized model of the bang-bang phase detector. The design methods of the latter two works are presented with closed-form mathematical theory, using closed loop transfer function modeling to estimate phase noise from BBPD and oscillator contributions. The simulation results presented and the demonstration of correlation of their model to measurement in [23], which correlate better than that seen in this work, perhaps imply these methods provide for better optimization of BBPD-PLL phase noise than this work. The simulation methods used in those works are not described, so it is unknown if the simulations capture the presence of discrete-time and quantization-related emergent behaviors not in with their linearized PLL models. In the case of [23] it appears that the loop filter is treated to be analog in nature, which differs from this work. Since the simulator used in this work accounts for full discrete-time and quantization effects, perhaps this may explain some level of observed discrepancy in the model versus simulation results compared to that in literature.

Few of the existing works consider the implementation of the loop filter design into digital hardware, rather just provide continuous valued coefficients for the filter designs. Of those surveyed, only [38] provides an analysis for quantization noise in terms of spurious-free dynamic range SFDR out of the loop filter design, but lacks the connection to output phase noise. This work uniquely considers (a) the impact of quantization of the digitized filter design, in terms of filter accuracy and output phase noise, and (b) attempts to optimize the digital implementation of the filter in terms of number of bits the various components of the filter (i.e. filter coefficients, multipliers, adders) are implemented with for complexity and performance.

A final advantage of this work not observed in the surveyed literature is the integrated PLL simulator with Monte-Carlo sampling which allows for verification of the designed loop filter with accurate modeling of time-discretization and digital quantization effects. This allows for lock time, phase noise and stability to be verified on the design to ensure it meets the designer's requirements before moving onto testing with transistor level implementations and testing.

7.3 Design Choices and Areas of Improvement

In the design of the filter optimization in this framework for phase noise, a choice was made to only consider phase noise components resulting from detector (TDC and BBPD) noise and oscillator noise. This choice has been motivated from the reasoning that other PLL noise components, being loop filter and divider noise, can be reduced below that of these components with relative ease. In the case of the loop filter noise in a digital PLL, noise results from quantization within the filter implementation are connected with finite word effects. It shown in section 6.7.1 that the quantization noise can be reduced considerably by increasing the number of bits used to represent data in the digital filter representation. As it was seen in figure 25a, the detector phase noise referred to the PLL output in the PLL architecture of this work is low pass with higher order roll than the TDC, thus a sufficient number of bits utilized in the loop filter implementation data words will reduce all output-referred loop filter noise components below detector noise. An optimizer was accordingly implemented in this work to select the optimal number of bits automatically for loop filter noise to be insignificant. In the case of divider noise, it is seen in the theory of section 2.3.7 in equations 52 and 55 that the detector noise (in this case TDC) and divider noise PSD referred to the PLL output both have a frequency dependence proportional to the closed loop PLL response, i.e. $\propto \hat{\mathrm{T}}(s)$. As demonstrated in appendix B, by constraining the output PSD of the divider to be below detector, it is found that the maximum allowed divider RMS jitter $\sigma_{tn_{div}}$ is in equation 126, for a TDC with with M steps per reference cycle. This is observed to be equal to the time resolution of the detector, in the case of the TDC $\Delta t_{step_{TDC}}$. This shows that the divider jitter requirement is rather loose, for example with a 10 MHz reference frequency and a 1000-step resolution TDC, the maximum RMS jitter limit is 100 ps.

$$\sigma_{tn_{div}} < \frac{1}{\mathrm{M}f_{ref}} = \Delta t_{step_{TDC}} \tag{126}$$

A further choice in the modeling and optimization of phase noise in this work was to only utilize $\propto f^{-2}$ components of oscillator phase noise from the Leeson model discussed in section 2.3.2, ignoring flicker frequency ($\propto f^{-1}$) components. The rationale for the exclusion of flicker noise components was that it is expected that flicker components of the reference oscillator will be significantly larger than the DCO contributions. Analyzing the PLL phase noise sensitivity transfer functions of section 2.3.5 shows that divider and reference noise is transferred to PLL output with sensitivity N$\hat{\mathrm{T}}(s)$, and DCO noise with 1-$\hat{\mathrm{T}}(s)$. The normalized PLL transfer function $\hat{\mathrm{T}}(s)$ has a zero-frequency gain of 1, i.e. $|\hat{\mathrm{T}}(0)|$ = 1. Thus, it is expected that near zero frequency, the flicker noise components of the DCO noise will be rejected as the low frequency region is the dominant flicker noise region. However, the reference flicker phase noise will be multiplied by the divider ratio N. Thus, DCO flicker contributions will be minor compared to the reference, and are not considered in the loop filter optimization code, nor the PLL behavioral simulator. Despite the significance of reference flicker noise, reference noise components are not included in the optimization and simulation processes in this work owing to reasoning that the reference noise is not easily improved with loop filter tuning. This is due to the fixed low frequency reference noise sensitivity of N$\hat{\mathrm{T}}(f) \approx N$, which will not improve flicker noise unless the bandwidth of $\hat{\mathrm{T}}(s)$ is made very small.

The loop filter design optimization in this work is limited to operating on a fixed architecture, being a proportional-integral controller with optional additional pole to compensate for closed-loop peaking. This choice was selected to limit the scope of the optimization problem to a subset for which convergence of the optimizer and stability of the designs was predictable. Furthermore, the fixed architecture allowed for simple optimization of its digital implementation by using a fixed design of second order direct-type I IIR filter section. For more generality, achieving optimal higher order filter design would require exploring a range of parallel and cascade digital filter structures to minimize finite word effects [40], significantly complicating the digitization process for filter than for the fixed case. It is possible gains in performance can be achieved with higher order and unconstrained architectures, so this is a possible area of improvement for this work.

To consider the limitation of the scope of this work, the simulation and optimization here only handles integer-N PLLs, which is a shortcoming as many PLL designs require fractional-N synthesis. To increase the generality of the work by extending to fractional-N PLLs, introduction of a fractional-N behavioral divider model and a major decrease in simulation time step size (much smaller than the reference cycle time used in integer-N case of this work) is a possible improvement. Reduction of the time step is required to decrease simulation time-quantization noise effects to low enough levels for the fractional divider noise characteristics to not be overpowered [41]. Such an improvement may prove to be limited by the current simulator, as overall simulation time will be increased by the factor that the simulation step is divided to support fractional-N models. Another possible improvement to consider is the implementation of a non-uniform step event-driven behavioral simulator to reduce simulation steps and thus simulation time.

8 Conclusion

In this work, an automatic framework for the design and optimization of all digital PLL loop filters has been introduced. The framework allows for input of target PLL specifications, for which a digital implementation-ready filter design will be generated having (a) minimized lock time or (b) minimized phase noise in terms of integrated power, subject to specified lock time constraints. The framework additionally performs post-optimization on the generated loop filter design to ensure performance including effects of quantization in the digital loop filter. A time domain PLL simulation engine is included within the framework to verify the performance of the designed filters for acceptable phase noise, lock time and stability. It was shown in this work that the designed and optimized filters correlate with the results found with the implemented simulator. Furthermore, the design of a fast-locking and phase noise-minimized PLL using loop filter gear-shifting was demonstrated and shown to correlate with simulation.

This work will be applied in a later thesis project undertaken by the author of this work concerning the design of an ultra-low power ADPLL, in order to assist and accelerate the design process.

References

[1] Sesha Sairam Regulagadda, Nagaveni S, and Ashudeb Dutta. “A 550-μW, 2.4-GHz ZigBee/BLE Receiver Front End for IoT applications in 180-nm CMOS”. In: *2018 16th IEEE International New Circuits and Systems Conference (NEWCAS)* (2018). DOI: 10.1109/newcas.2018.8585629.

[2] Yao Shi et al. “28.3 A 606μW mm-Scale Bluetooth Low-Energy Transmitter Using Co-Designed 3.5x3.5mm2 Loop Antenna and Transformer-Boost Power Oscillator”. In: *2019 IEEE International Solid-State Circuits Conference - (ISSCC)* (2019). DOI: 10.1109/isscc.2019.8662333.

[3] Xing Chen et al. “Analysis and Design of an Ultra-Low-Power Bluetooth Low-Energy Transmitter With Ring Oscillator-Based ADPLL and 4$\times$ Frequency Edge Combiner”. In: *IEEE Journal of Solid-State Circuits* 54.5 (2019), pp. 1339–350. DOI: 10.1109/jssc.2019.2896404.

[4] Franz Pengg et al. “A low power miniaturized 1.95mm2 fully integrated transceiver with fastPLL mode for IEEE 802.15.4/bluetooth smart and proprietary 2.4GHz applications”. In: *2013 IEEE Radio Frequency Integrated Circuits Symposium (RFIC)* (2013). DOI: 10.1109/rfic.2013.6569525.

[5] Behzad Razavi. “Design of monolithic phase-locked loops and clock recovery circuits tutorial”. In: 1996.

[6] John G. Proakis and Dimitris G. Manolakis. “8.3.3 IIR Filter Design by the Bilinear Transformation”. In: *Digital signal processing: principles, algorithms, and applications*. Macmillan, 1993.

[7] F. Gardner. “Charge-Pump Phase-Lock Loops”. In: *IEEE Transactions on Communications* 28.11 (Nov. 1980), pp. 1849–1858. DOI: 10.1109/tcom.1980.1094619.

[8] Chun-Ming Hsu, Matthew Z. Straayer, and Michael H. Perrott. “A Low-Noise Wide-BW 3.6-GHz Digital $\Delta\Sigma$ Fractional-N Frequency Synthesizer With a Noise-Shaping Time-to-Digital Converter and Quantization Noise Cancellation”. In: *IEEE Journal of Solid-State Circuits* 43.12 (2008), pp. 2776–2786. DOI: 10.1109/jssc.2008.2005704.

[9] Enrico Temporiti et al. “A 3 GHz Fractional All-Digital PLL With a 1.8 MHz Bandwidth Implementing Spur Reduction Techniques”. In: *IEEE Journal of Solid-State Circuits* 44.3 (2009), pp. 824–834. DOI: 10.1109/jssc.2008.2012363.

[10] Neil H. E. Weste and David Money Harris. “13.5.1.2 Divider”. In: *CMOS VLSI design: a circuits and systems perspective*. Addison Wesley, 2011.

[11] Rui Machado, Jorge Cabral, and Filipe Serra Alves. “All-Digital Time-to-Digital Converter Design Methodology Based on Structured Data Paths”. In: *IEEE Access* 7 (2019), pp. 108447–108457. DOI: 10.1109/access.2019.2933496.

[12] John G. Proakis and Dimitris G. Manolakis. “3.1 The z-Transform”. In: *Digital signal processing: principles, algorithms, and applications*. Macmillan, 1993.

[13] John G. Proakis and Dimitris G. Manolakis. “7.3 Structures for IIR Systems”. In: *Digital signal processing: principles, algorithms, and applications*. Macmillan, 1993.

[14] Jue Shen et al. “Phase noise improvement and noise modeling of type-I ADPLL with non-linear quantization effects”. In: *2014 Norchip* (2014). DOI: 10.1109/norchip.2014.7004732.

[15] Koji Takinami et al. “A Distributed Oscillator Based All-Digital PLL With a 32-Phase Embedded Phase-to-Digital Converter”. In: *IEEE Journal of Solid-State Circuits* 46.11 (2011), pp. 2650–660. DOI: 10.1109/jssc.2011.2164011.

[16] Ping Lu and Pietro Andreani. “A high-resolution Vernier Gated-Ring-Oscillator TDC in 90-nm CMOS”. In: *Norchip 2010* (2010). DOI: 10.1109/norchip.2010.5669467.

[17] B. Widrow. “Statistical analysis of amplitude-quantized sampled-data systems”. In: *Transactions of the American Institute of Electrical Engineers, Part II: Applications and Industry* 79.6 (1961), pp. 555–568. DOI: 10.1109/tai.1961.6371702.

[18] D.b. Leeson. “A simple model of feedback oscillator noise spectrum”. In: *Proceedings of the IEEE* 54.2 (1966), pp. 329–330. DOI: 10.1109/proc.1966.4682.

[19] T.h. Lee and A. Hajimiri. “Oscillator phase noise: a tutorial”. In: *IEEE Journal of Solid-State Circuits* 35.3 (2000), pp. 326–336. DOI: 10.1109/4.826814.

[20] John G. Proakis and Dimitris G. Manolakis. “7.7 ROUND-OFF EFFECTS IN DIGITAL FILTERS”. In: *Digital signal processing: principles, algorithms, and applications*. Macmillan, 1993.

[21] M.h. Perrott, M.d. Trott, and C.g. Sodini. “A modeling approach for $\Sigma - \Delta$ fractional-N frequency synthesizers allowing straightforward noise analysis”. In: *IEEE Journal of Solid-State Circuits* 37.8 (2002), pp. 1028–1038. DOI: 10.1109/jssc.2002.800925.

[22] M. Zanuso et al. “Noise Analysis and Minimization in Bang-Bang Digital PLLs”. In: *IEEE Transactions on Circuits and Systems II: Express Briefs* 56.11 (2009), pp. 835–839. DOI: 10.1109/tcsii.2009.2032470.

[23] Hao Xu and Asad A. Abidi. “Design Methodology for Phase-Locked Loops Using Binary (Bang-Bang) Phase Detectors”. In: *IEEE Transactions on Circuits and Systems I: Regular Papers* 64.7 (2017), pp. 1637–1650. DOI: 10.1109/tcsi.2017.2679683.

[24] Robert Bogdan Staszewski and Poras T. Balsara. “All-Digital PLL With Ultra Fast Settling”. In: *IEEE Transactions on Circuits and Systems II: Express Briefs* 54.2 (2007), pp. 181–185. DOI: 10.1109/tcsii.2006.886896.

[25] Katsuhiko Ogata. “5-7 Effects of Integral and Derivative Control Actions on System Performance”. In: *Modern control engineering*. Prentice Hall, 2010.

[26] Katsuhiko Ogata. “Chapter 8 PID Controllers and Modified PID Controllers”. In: *Modern control engineering*. Prentice Hall, 2010.

[27] Behzad Razavi and Behzad Razavi. “16 Phase-Locked Loops”. In: *Design of analog CMOS integrated circuits*. McGraw-Hill.

[28] Katsuhiko Ogata. “7-6 Stability Analysis”. In: *Modern control engineering*. Prentice Hall, 2010.

[29] V. Vannicola and P. Varshney. “Spectral Dispersion of Modulated Signals Due to Oscillator Phase Instability: White and Random Walk Phase Model”. In: *IEEE Transactions on Communications* 31.7 (1983), pp. 886–895. DOI: 10.1109/tcom.1983.1095902.

[30] John G. Proakis and Dimitris G. Manolakis. “12 Power Spectrum Estimation”. In: *Digital signal processing: principles, algorithms, and applications*. Macmillan, 1993.

[31] William H. Press et al. “10.2 Golden Section Search in One Dimension”. In: *Numerical recipes: the art of scientific computing*. Cambridge University Press, 2007.

[32] Jonathan Barzilai and Jonathan M. Borwein. “Two-Point Step Size Gradient Methods”. In: *IMA Journal of Numerical Analysis* 8.1 (1988), pp. 141–148. DOI: 10.1093/imanum/8.1.141.

[33] Katsuhiko Ogata. “2-4 Modeling in State Space”. In: *Modern control engineering*. Prentice Hall, 2010.

[34] R. Brockett. “Poles, zeros, and feedback: State space interpretation”. In: *IEEE Transactions on Automatic Control* 10.2 (1965), pp. 129–135. DOI: 10.1109/tac.1965.1098118.

[35] “Chapter 15: Numerical Integration”. In: *A First Course in Numerical Methods* (2011), pp. 441–479. DOI: 10.1137/9780898719987.ch15.

[36] Volodymyr Kratyuk et al. “A Design Procedure for All-Digital Phase-Locked Loops Based on a Charge-Pump Phase-Locked-Loop Analogy”. In: *IEEE Transactions on Circuits and Systems II: Express Briefs* 54.3 (2007), pp. 247–251. DOI: 10.1109/tcsii.2006.889443.

[37] Sally Safwat, Maged Ghoneima, and Yehea Ismail. “A design methodology for a low power bang-bang all digital PLL based on digital loop filter programmable coefficients”. In: *2011 International Conference on Energy Aware Computing* (2011). DOI: 10.1109/iceac.2011.6136676.

[38] Martin Kumm, Harald Klingbeil, and Peter Zipf. “An FPGA-Based Linear All-Digital Phase-Locked Loop”. In: *IEEE Transactions on Circuits and Systems I: Regular Papers* 57.9 (2010), pp. 2487–2497. DOI: 10.1109/tcsi.2010.2046237.

[39] Yawgeng A. Chau and Chen-Feng Chen. “All-digital phase-locked loop with an adaptive bandwidth design procedure”. In: *2009 International Symposium on Intelligent Signal Processing and Communication Systems (ISPACS)* (2009). DOI: 10.1109/ispacs.2009.5383893.

[40] John G. Proakis and Dimitris G. Manolakis. “7 Implementation of Discrete-Time Systems”. In: *Digital signal processing: principles, algorithms, and applications*. Macmillan, 1993.

[41] M.h. Perrott. “Behavioral simulation of fractional-N frequency synthesizers and other PLL circuits”. In: *IEEE Design & Test of Computers* 19.4 (2002), pp. 74–83. DOI: 10.1109/mdt.2002.1018136.

A Estimating PSD with Autoregressive Model

The following is based on [30]. Given a signal x[n] whose power spectrum should be estimated, its autocorrelation sequence $r_{xx}[l]$ with lag l must be computed:

$$r_{xx}[l] = \sum_{n=-\infty}^{\infty} x[n]x[n-l] \tag{127}$$

The autoregressive model for power spectrum, with p poles, that shall be fitted is given in 128

$$S_{XX}(f) = \frac{1}{|1+\sum_{n=1}^{p} a_n z^{-1}|^2}\Bigg|_{z^{-1}=e^{-j2\pi f\Delta T}} \tag{128}$$

MMSE optimization of the distribution for coefficients $\{a_1, ..., a_p\}$ is done by solving the Yule-Walker equation in 129.

$$\begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_p \end{bmatrix} = -\mathbf{R}_{xx}^{-1}\mathbf{r}_{xx} = -\begin{bmatrix} r_{xx}[0] & r_{xx}[1] & \dots & r_{xx}[p-1] \\ r_{xx}[1] & r_{xx}[0] & \dots & r_{xx}[p-2] \\ \vdots & \vdots & & \\ r_{xx}[p-1] & r_{xx}[p-2] & & r_{xx}[0] \end{bmatrix}^{-1} \begin{bmatrix} r_{xx}[1] \\ r_{xx}[2] \\ \vdots \\ r_{xx}[p] \end{bmatrix} \tag{129}$$

B Divider Noise Constraint

Output referred phase noise PSD of TDC:

$$S_{\Phi n_{TDC,out}} = \frac{1}{12 f_{ref}} \left|2\pi \frac{N}{M} G(f)\right|^2 \tag{130}$$

Output referred phase noise PSD of divider:

$$S_{\Phi n_{div,out}} = f_{ref} \left|2\pi N \sigma_{tn_{div}} G(f)\right|^2 \tag{131}$$

The output-referred phase noise for the TDC and divider have the same frequency dependence. So by setting $S_{\Phi n_{div,out}} < S_{\Phi n_{TDC,out}}$, a constraint to force PLL output divider less than TDC noise can be found:

$$\sigma_{tn_{div}} < \frac{1}{\mathrm{M} f_{ref}} = \Delta t_{step_{TDC}} \tag{132}$$

Must simply ensure that jitter of divider is much less than TDC resolution, which is a reasonable demand. Thus, it is reasonable to ignore divider noise in the phase noise optimization if divider noise can reasonably be made insignificant in the overall output phase noise.

$\square$