Design and Verification Methods for Digital-Centric RF Circuits and Systems

Von der Fakultät für Elektrotechnik und Informationstechnik
der Rheinisch-Westfälischen Technischen Hochschule Aachen
zur Erlangung des akademischen Grades eines
Doktors der Ingenieurwissenschaften
genehmigte Dissertation

vorgelegt von

Jonas Meier,
M. Sc.
aus Hamburg

Berichter: Univ.- Prof. Dr.-Ing. Stefan Heinen
Univ.- Prof. Dr.-Ing. Gerd Ascheid

Tag der mündlichen Prüfung: 30.09.2022

ACKNOWLEDGMENT

This work has been created during my time as research assistant at the Institute for Integrated Analog Circuits and RF Systems at RWTH Aachen University.

My special thanks go to Prof. Dr-Ing. Stefan Heinen, who made this work possible as he gave me the opportunity to work on these exciting topics. I would also like to thank Prof. Dr-Ing. Gerd Ascheid for taking over as co-supervisor and his interest in this work.

I wish to thank all my colleagues for our time together at the insitute during weekly meetings and interesting discussions – technical and otherwise – which have contributed to this work. Christoph Beyerstedt, Fabian Speicher and Markus Scholl should be mentioned explicitly. Thanks for all your hints, help, ideas and support.

Special thanks go to Andreas Köllmann and Ulrich Möhlmann for their continuous insights into PLL design during our many meetings in preparation for the PLL tape-out. Further, a big thank you to Johannes Bastl, Tim Lauber, Alexander Meyer, Christopher Nardi and Lantao Wang, who significantly contributed to the successful tape-out.

I would like to show my appreciation to all the students, who supported me with their bachelor and master theses as well as their work as student researchers. Especially, Fabian Maul, Florian Menke, Gregor Boronowski and Ozan Yildirim shall be mentioned here.

Additionally, I thank my family and friends for their constant support. In particular, my gratitude goes to my wife Carola, who patiently bore with me on the way to finish this dissertation and in all other situations as well.

Hamburg, May 2022
Jonas Meier

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF LISTINGS

LIST OF ABBREVIATIONS

AC	alternating current
ACQ	acquisition
ADC	analog-to-digital converter
ADPLL	all-digital phase-locked loop
AMS	analog mixed signal
API	application programming interface
AWGN	additive white gausian noise
C2I3	control, configuration, interface, interconnect and integration
CKR	retimed reference clock
CKV	variable clock
CMOS	complementary metal oxid semiconductor
DAC	digital-to-analog converter
DC	direct current
DCO	digitally controlled oscillator
DCW	discrete control word
DPI	Direct Programming Interface
DSP	digital signal processing
DTC	digital-to-time converter
DUT	device under test
EA	error amplifier
FCW	frequency command word
FFT	fast Fourier transform
FIFO	First In – First Out
FREF	reference frequency
GUI	graphical user interface
HB	harmonic balance
HDL	hardware description language
IAS	Institute for Integrated Analog Circuits
IC	integrated circuit
IDE	integrated development environment

IF	intermediate frequency
IFFT	inverse fast Fourier transform
IIR	inifinite impulse response
ISM	industrial, scientific and medical
LDO	low-dropout regulator
LF	loop filter
LFSR	linear feedback shift register
LMS	least-mean-square
LO	local oscillator
LSB	least significant bit
LTI	linear time invariant
MvS	model vs. schematic
NTF	noise transfer function
OPAMP	operational amplifier
OTW	oscillator tuning word
PCB	printed circuit board
PFD	phase-frequency detector
PHE	phase error word
PHR	reference phase word
PHV	variable phase word
PLL	phase-locked loop
PMOS	p-type metal oxid semiconductor
PSN	power supply noise
PSRR	power supply rejection ratio
PSS	periodic-steady-state
PVT	process, voltage, temperature
RAM	random-access memory
RF	radio frequency
RF-DAC	RF-digital-to-analog converter
RMS	root-mean-square
RNM	real number modeling
RO	ring oscillator
RTL	register transfer level
SC	SystemC
SDR	software-defined radio
SNR	signal-to-noise ratio
SoC	System-on-Chip
SPI	serial peripheral interface
SRAM	static random-access memory

SSD single-sided spectral density
SSN simultaneous switching noise
STF signal transfer function
SV SystemVerilog
TDC time-to-digital converter
TIEH tie-high
TIEL tie-low
TRK tracking
UDN user-defined net
UDR user-defined resolution function
UDT user-defined data type
VCO voltage controlled oscillator
VDD supply voltage
VHDL very high speed integrated circuit hardware description language

CHAPTER 1

INTRODUCTION

1.1 Motivation and Goal

The demand for reliable, affordable, low-power electronics has caused a wide adoption of so called Systems-on-Chip (SoCs). These combine analog, digital and radio frequency (RF) components on the same silicon die. Especially, advances in communication technology are a driver of design complexity, demanding ever-growing data rates on a limited power budget of a battery-powered device. Currently used communication standards are based on digital signal modulation. Smaller technology nodes combined with digital-centric designs are of advantage here. They provide the possibility to use smaller supply voltages and allow to reach higher operating frequencies.

But with this shift, the accompanying rising integration densities and an increasing interaction between the sub-systems, new error sources gain importance. For instance, switching noise and cross-coupling over supply from the growing digital circuits to the remaining analog and RF blocks present an issue that needs to be considered during design and verification. The low supply voltage poses a challenge for remaining analog components. The voltage headroom is reduced, which furthers the susceptibility to noise and disturbances on supply lines. A critical component in digital-centric designs is the reference clock generation. Phase-locked loops (PLLs) are commonly used to provide low phase noise, high accuracy clock signals. A degradation in performance directly influences the functionality of all other parts in the system.

Traditional analog mixed signal (AMS) simulators verify the system behavior by co-simulation of digital and analog circuits. But especially the analog simulator becomes a bottleneck when simulating high frequency varying signals. They are thus not able to handle the requirements for a full-system verification in an adequate time-frame, especially on chip level. Event-driven simulations as known from digital simulators provide a promising solution. With the recent addition of real number modeling they allow the development

of analog models for efficient co-simulation with digital circuits. Nevertheless, the analog modeling capabilities are still limited and often focus on the main signal path. Also, their advantage is limited for high frequency radio signals.

Goal of this thesis is the development of a modeling strategy that allows in digital-centric designs to simulate the remaining analog and RF circuits efficiently together with the digital system. A focus of these models is the ability to consider aggressors from other parts of the system e.g. by coupling over the supply network. The developed modeling approaches are verified by applying them to sub-circuits of an RF transceiver developed at the institute Institute for Integrated Analog Circuits (IAS) to enable pre-tape-out verification of the design. They are further used in the development of an all-digital phase-locked loop (ADPLL). Using parameterized models enables to consider the effects of supply voltage modulation in an early design phase and to adjust the system partitioning accordingly.

1.2 Proceeding

To enable the examination of different modeling approaches, a generic simulation framework is developed that provides a seamless integration in the design process of full custom circuits. This thesis proposes the use of SystemVerilog (SV) as modeling basis. The SV Direct Programming Interface (DPI) allows the use of a C++ class-based architecture and benefits from the available standard libraries to implement efficient signal descriptions. The native netlister is utilized to reuse the analog design hierarchy and only replace bottom-level functional building blocks and signal representations with for the specific use-case optimized, computational efficient implementations. A library of parameterized typical building blocks is developed to enable a fast and widely automatic module generation.

Using specialized signal descriptions for analog and RF signals along with the signal processing capabilities available in C++ allows to grow the simulation effectiveness. Different approaches from literature are evaluated with regard to their suitability for system-level mixed-signal verification. During the process not only the main signal path is considered but also the peculiarities when modeling a variable supply voltage are discussed. The class-based architecture allows for a modular extension of the simulation framework.

The developed models and simulation approaches are applied to the verification of a RF transceiver developed at the IAS and enable to find errors and

optimization potential in the subsystems that would otherwise only be seen in the lab during measurements.

The methodology is further applied to the development of a fractional ADPLL. The ADPLL is designed using a 2D-Vernier time-to-digital converter (TDC) for fractional phase measurement. A characterizing phenomenon in fractional PLLs is the occurrence of periodical phase errors that cause so called spurious tone when looking at the output phase noise. Two different approaches to suppress these fractional spurs are implemented in the digital loop filter. The first is based on the digital cancellation of expected sampling errors at the TDC, the second uses pseudo-random delay dithering and correction of the reference clock. Using parameterized models for the few remaining analog circuits in the design, the simulation method is used to analyze the susceptibility to cross-coupling and switching noise of the ADPLL.

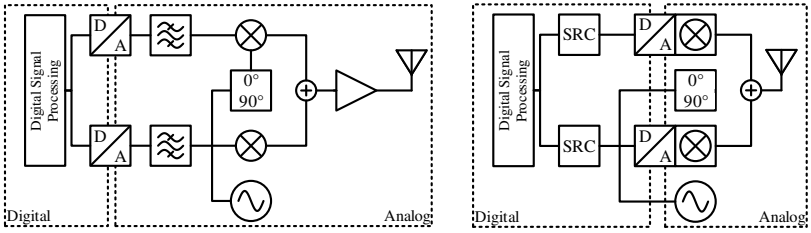
CHAPTER 2

FUNDAMENTALS

2.1 RF Systems and Components

The conventional architecture for RF transmitter is depicted in figure 2.1a. It consists of a digital signal processor that handles the coding and modulation of the data to be transmitted at baseband frequency. This generated baseband signal is converted to an analog system using a digital-to-analog converter (DAC). The following low pass filter removes any alias spectra. An analog mixer converts the baseband signal to the RF carrier frequency. The subsequent power amplifier is connected to the antenna to transmit the signal [Zim11].

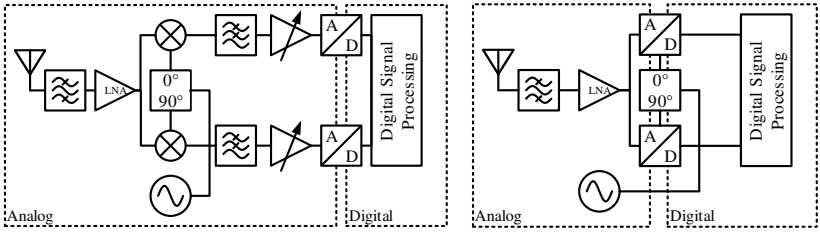
Analog based transmitters as described above do not scale with the advances to smaller technology nodes. To keep the smaller transistors working reliably, the supply voltage is reduced. Since the threshold voltage of the transistors stays nearly the same, this reduces the effective voltage margin and therefore the linearity of analog transistors. On the other hand digital circuits benefit greatly from smaller technology nodes. With decreasing size the circuit



(a) Conventional RF transmitter architecture.

(b) Digital-centric RF transmitter architecture.

Figure 2.1: Comparison of conventional and digital-centric transmitter architectures.



(a) Conventional homodyne receiver architecture.

(b) Software-defined radio receiver architecture.

Figure 2.2: Advances in receiver architectures.

consumes proportionally less power while the speed increases linear [SA11]. Therefore, digital-centric transmitters have been proposed as an alternative [Zim11]. Figure 2.1b illustrates a typical architecture [Ala14]. The baseband signal from the digital signal processing (DSP) unit is up-sampled and fed to an RF-digital-to-analog converter (RF-DAC). This converter takes a digital clock at RF frequency and mixes it with the upsampled control word. The resulting output is a signal with an amplitude proportional to the digital value at the input. This output is then either transmitted to a subsequent power amplifier or drives the antenna directly.

Since almost all building blocks are digital, such an architecture utilizes the advantages of smaller processing nodes. Additionally, the software based implementation of signal processing and filtering allows the flexibility to develop transmitters that support multiple wireless communication standards on the same hardware [MHN16].

Similar advances are noted on the receiving side[Riv+07; CC11; Lie+14; Kim+10]. Software-defined radio (SDR) receivers in contrast to conventional architectures use faster analog-to-digital converters (ADCs) to shift the boundary between RF and digital domain closer towards the antenna as seen in figure 2.2. This allows more and more filters to be implemented in digital post-processing, making the design more versatile and adjustable via software.

2.1.1 Analog-to-Digital Conversion

While most physical phenomena interacting with electronics, like wireless transmission using electromagnetic waves, are analog by nature, signal pro-

cessing is shifted more and more towards the digital domain. In contrast to analog signals, which are continuous in time and value, digital signals are discrete in both. Thus, a conversion of the signal is needed quantizing it in both domains. This is typically done using ADCs. Their performance must keep up with the advances of digital circuitry due to scaling of the technology nodes, as not to have a bottleneck at the interface between digital signal processing and analog environment [ST04].

Analog-to-digital conversion in practice means sampling the analog waveform at specific, typically equidistant points in time, using a reference clock to provide a stable period. The minimum necessary sampling rate is given by

$$f_{Nyquist} = 2 \cdot f_{in} \,, \tag{2.1}$$

where f_{in} is the highest input frequency of the signal being sampled. Sampling at a lower rate than the Nyquist Rate causes aliasing. Multiple frequency components are mapped to the same discrete frequency bin and information is lost due to the sampling. Thus, the sampled signal can not be reconstructed correctly [OL07].

Quantizing the analog value is usually realized by limiting the signal in its range and rounding its amplitude to the next predefined discrete value. The number of available digital levels determines the resolution of the ADC expressed as the number of bits of the digital output word [All02].

This conversion process causes a quantization error due to rounding and the finite sampling rate. The error is often described as uniformly distributed noise. Assuming a noiseless sinusoidal input signal, this leads to an output signal-to-noise ratio (SNR) of

$$SNR = (N \cdot 6.02 + 1.76)\ \text{dB} \,. \tag{2.2}$$

To further improve the performance of ADCs, oversampling can be applied. Instead of sampling with a frequency close to the necessary Nyquist Rate, a much higher frequency is used. The oversampling ratio is defined as

$$OSR = \frac{f_{sample}}{f_{Nyquist}} \tag{2.3}$$

and is typically in the order of 8 to 512.

Oversampling allows to reduce the number of bits of the quantizer without

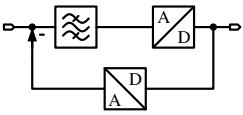


Figure 2.3: ΔΣ ADC.

losing resolution. The SNR with oversampling is

$$SNR = (N \cdot 6.02 + 1.76 + 10 \log OSR)\ \text{dB}\ . \tag{2.4}$$

ΔΣ-modulators belong to the group of oversampling converters. In contrast to classical ADCs, they are not so much limited by non-linearity and matching issues of analog circuitry [All02]. Furthermore, they provide intrinsic anti-aliasing at low power and high speed [RTL09].

Figure 2.3 shows such a modulator. It contains a feedback loop combined with an internal low resolution ADC and DAC as well as a loop filter [ST04]. This can be used to apply noise-shaping by choosing an appropriate filter structure. The ΔΣ-ADC inherently has two different transfer functions for the propagation of quantization noise and the signal to the ADC's output called noise transfer function (NTF) and signal transfer function (STF):

$$NTF(s) = \frac{1}{1 + H(s)}\ , \tag{2.5}$$

$$STF(s) = \frac{H(s)}{1 + H(s)}\ . \tag{2.6}$$

Choosing the loop filter transfer function $H(s)$ allows to model the NTF and STF to improve the SNR by moving the noise out of the desired frequency signal band:

$$SNR = 20 \log \left[(2^n - 1) \sqrt{\frac{3}{2}} \frac{\sqrt{2m+1}}{\pi^m} OSR^{\,m+\frac{1}{2}} \right] \text{dB}\ . \tag{2.7}$$

Here, m is the loop order of the ΔΣ-modulator [RTL09].

Instead of quantizing each sample with a high resolution, a delta-sigma modulated signal encodes amplitude information in the time duration similar to pulse-width modulation. A decimation filter in the digital domain afterwards reduces the signal rate back to the Nyquist frequency decoding the amplitude

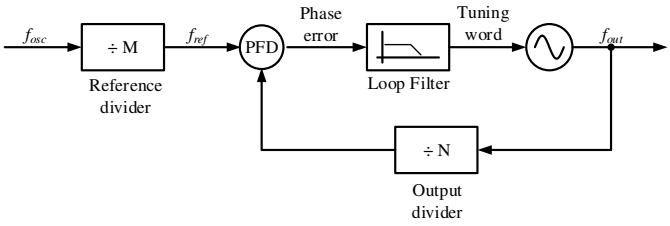


Figure 2.4: Integer-N PLL block diagram.

information.

The ΔΣ-modulator is very sensitive to non-linearity in the feedback path, since distortions caused there are not shaped by the NTF. To avoid that problem, often only single-bit ADCs and DACs are used, which are inherently linear. When internally using multi-bit converters, extra care has to be taken to minimize the non-linearity, but it is rewarded with better stability and a higher SNR, even with reduced oversampling rate [ST04].

2.1.2 Frequency Synthesis Using Phase-Locked Loops

Regardless if conventional or digital-centric, integrated transceivers require an accurate frequency synthesis to supply the local oscillator (LO) and clock signals. Different system topologies are available that can be categorized as direct, indirect and hybrid structures, each coming with individual advantages as well as disadvantages. Indirect frequency synthesizers such as PLL are preferably used in mobile communication systems since they satisfy most of the strict phase noise and spur requirements, which are mandated by modern communication standards.

The classical charge-pump based analog PLL utilizes a stable low frequency reference and a voltage controlled oscillator (VCO) to generate a high frequency output f_{out}. The output is divided by a constant factor N to a lower frequency domain close to the input reference. The phase-frequency detector (PFD) compares the phases of subdivided output and reference frequency to return the momentary phase error between the two. Due to the feedback structure as shown in figure 2.4 even small output drifts are detected and corrected in the loop [SB06]. The continuous counteractions of the PFD cause noise and spurious tones close to the center frequency of the system. The loop

filter (LF) is designed to reduces these as well as the noise introduced by the reference oscillator itself which can cause unwanted frequency modulation in the VCO. To this end, the LF needs to have a rather small bandwidth. However, a low corner frequency f_{BW} results in slower loop dynamics and long settling times. Therefore, these systems are especially suited for narrow-band applications with a high demand for low out-of-band noise. To achieve low in-band phase noise, an oscillator with adequately high quality factor Q and hence a low phase noise characteristics, such as crystal oscillators are required. An additional frequency divider with factor M can be utilized as a prescaler of the input frequency f_{osc}. The PLL output then follows (2.8).

$$\begin{aligned} f_{out} &= f_{osc} \cdot \frac{N}{M} \\ &= f_{ref} \cdot N \,. \end{aligned} \tag{2.8}$$

The architecture is thus referred to as *integer-N PLL*. Varying the divider factor N, allows to tune the output frequency. The minimum spacing Δf_{ch} between neighboring frequency settings is bound by:

$$\Delta f_{ch} = \frac{f_{out}}{N} = f_{ref} \,. \tag{2.9}$$

Modern communication standards require channel spacing in the range of kHz to MHz around the carrier in the GHz region. Thus, large values for N are necessary. Unfortunately, the phase noise S_ϕ of the reference oscillator is raised due to the frequency multiplication as in (2.10):

$$\Delta S_\phi = 20 \log(N) \,[\mathrm{dB}] \,. \tag{2.10}$$

Thus it is hard to maintain small channel spacings Δf_{ch} and the specified noise limits at the same time. Additionally, in integer-N PLLs spurious tones occur at multiples of f_{ref}, which happen to be neighboring channels. Therefore, the LF must be designed with $f_{BW} \approx \Delta f_{ch}$ and steep roll-off to eliminate such spurs appropriately. Traditional analog RC-filters with such requirements require large capacitors and resistors making monolithic integration practically impossible.

A fractional-N PLL can be considered as an alternative approach which tackles most of the above problems. In contrast to the interger-N PLL the divider in the feedback path is replaced with a fractional divider as seen in figure 2.5. Being able to tune the output frequency by fractions of the input reference f_{ref} eases many design constraints such as the minimum channel spacing and loop filter bandwidth. To improve the system dynamics, the reference

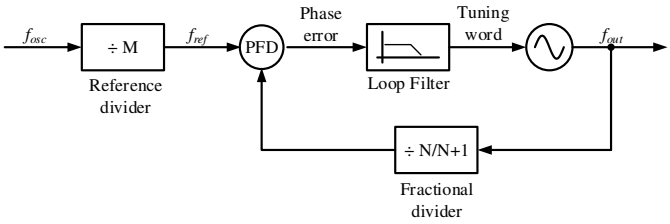


Figure 2.5: Fractional-N PLL block diagram.

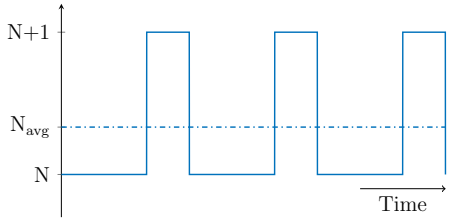


Figure 2.6: Time averaged fractional divider ratio.

frequency f_{ref} and filter bandwidth f_{BW} can be chosen much higher than Δf_{ch} to allow a faster update rate an swifter response to phase errors. Additionally, an increased f_{ref} allows to reduce the divisor N which yields a lower phase noise contribution ΔS_{ϕ} as in (2.10).

Since there is no intrinsic block which implements a fractional frequency divider in analog PLL circuits, all techniques rely on the principle of time averaging as depicted in figure 2.6. Two techniques have proven suitable for this application: Fractional Accumulation and ΣΔ-Modulation. They modulate the divider value in the feedback path over time such that the average value equals the desired frequency ratio $\{N.f\}$, where $\{N\}$ accounts for the integer and $\{.f\}$ for the fractional value. The fractional portion $\{.f\}$ of the divider ratio is directly derived from the duty cycle D_f given by

$$D_f = \frac{T_{N+1}}{T_N + T_{N+1}} = \{.f\}\,, \tag{2.11}$$

where T_{N+1} and T_N are the duration of each state within a single cycle. Thus,

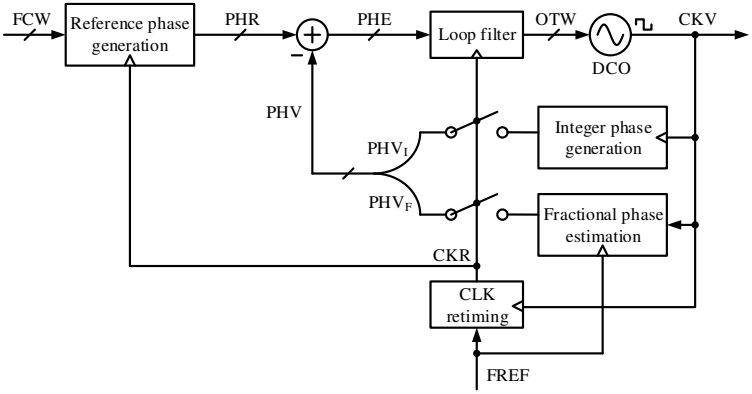


Figure 2.7: Block diagram of a fractional-N ADPLL.

the output frequency follows

$$f_{out} = f_{ref} \cdot (N + D_f) = f_{ref} \cdot \{N.f\} \,. \tag{2.12}$$

2.1.3 All-Digital Phase-Locked Loops

With the progress towards smaller nodes and submicrometer technologies, traditional charge-pump based PLLs are not amenable to integration anymore due to the necessary analog control signals and bulky filter structures. Replacing the analog signals with digital counterparts remedies the situation by using DSP units to handle the filtering of binary encoded phase and frequency information. The schematic in figure 2.7 depicts an ADPLL. All blocks within the control loop have digital ports, including the oscillator tuning word (OTW) controlling the digitally controlled oscillator (DCO). Hence, the system completely relinquishes the need for additional signal conversions using ADCs or DACs [SB05].

Short rise and fall time of nanoscale logic allow exact reconstruction of signal times at rising and falling edges. The system proposed in figure 2.7, counts the events of either rising or falling transitions of the rectangular shaped output waveform variable clock (CKV) of the DCO to gain the integer phase (PHV_I) information. Accumulating these events converts the output frequency into phase information, which is the equivalent discrete operation for phase

integration in continuous time-domain:

$$\begin{aligned}\phi(t) &= 2\pi \int_{-\infty}^{t} f(\tau) d\tau \\ &\approx \phi_0 + 2\pi \sum_{0}^{k} f_k(t) \ .\end{aligned} \tag{2.13}$$

The fractional phase $\mathrm{PHV_F}$ is estimated with respect to the external reference clock reference frequency (FREF) and the composition of $\mathrm{PHV_I}$ and $\mathrm{PHV_F}$ yields a binary representation of the momentary variable output phase variable phase word (PHV). This signal is the equivalent digital description of the down-divided clock in the loop feedback of the previously described analog-based PLL architectures.

The frequency command word (FCW) controls the DCO frequency. It is defined as the ratio of the desired output f_{DCO} and applied input reference f_{ref} given in (2.14), similarly to the divider ratio in 2.1.2:

$$\mathrm{FCW} = \frac{f_{DCO}}{f_{ref}} = \{N.f\} \ . \tag{2.14}$$

In contrast to the analog implementations, the fractional portion of FCW is not gained through time averaging but, instead, it is directly processed in the control loop.

Another accumulation stage translates the frequency command word, which is time-invariant during normal operation, to the momentary reference phase word (PHR) on every instance of the significant edge. The PFD in digital PLL designs is implemented as a simple arithmetic block performing a subtraction of PHR and the feedback signal PHV. Compared to charge-pump based PFD implementations of analog PLLs, which in general suffer from non-linearities, the signal subtraction is a linear operation without any need for approximation and inherently reduces spurs. The resulting phase error word (PHE) feeds a digital LF whose result is applied to the DCO as oscillator tuning word OTW. A retiming block in the schematic indicates a sampling process of the low-frequency reference signal FREF with the DCO output CKV, which is usually several orders of magnitudes higher in frequency. The so gained retimed reference clock (CKR) allows the operation in a system synchronized to the DCO output signal but operating the same rate as FREF.

2.1.3.1 Operation Principle in Phase-Domain

To gain knowledge of the absolute phase and thereby of the momentary frequency, accurate timing information of the zero-crossings of either rising or falling signal edges are necessary. By definition, the DCO output signal has a variable period duration T_v while the reference oscillator period T_r should be constant. Further, it is assumed, that $T_v \ll T_r$ which allows to use T_v as unit interval and assume both periods as time invariant. Clock edges, indexed with i and k, appear at the absolute times t_v and t_r, at multiples of their respective periods as defined in (2.15). t_0 denotes a possible initial time offset between both clock domains and is referred to FREF.

$$\begin{aligned} t_v &= i \cdot T_v \,, \\ t_r &= k \cdot T_r + t_0 \end{aligned} \tag{2.15}$$

Normalizing the time steps using the declared unit interval T_v results in a dimensionless expression for reference (Φ_r) and variable (Φ_v) phase similar to radian or angular representation:

$$\begin{aligned} \Phi_v &= \frac{t_v}{T_v} \,, \\ \Phi_r &= \frac{t_r}{T_v} \,. \end{aligned} \tag{2.16}$$

Applying the normalization to (2.15) and evaluating the phases at their corresponding clock edges yields (2.17). The ratio of the two period durations T_v and T_r is the same as defined in (2.14).

$$\begin{aligned} \Phi_v[i] &= i \,, \\ \Phi_r[k] &= k \cdot \frac{T_r}{T_v} + \frac{t_0}{T_v} = k \cdot \mathrm{FCW} + \Phi_0 \end{aligned} \tag{2.17}$$

Thus, the discrete frequency-to-phase conversion can be formulated as summation of the sampled data. To calculate the phases, for *PHV* ones are accumulated on each significant CKV edge, while *PHR* is incremented by FCW on each reference edge:

$$\mathrm{PHV}[i] = \sum_{l=0}^{i} 1 \,, \tag{2.18}$$

$$\mathrm{PHR}[k] = \sum_{l=0}^{k} \mathrm{FCW} \,. \tag{2.19}$$

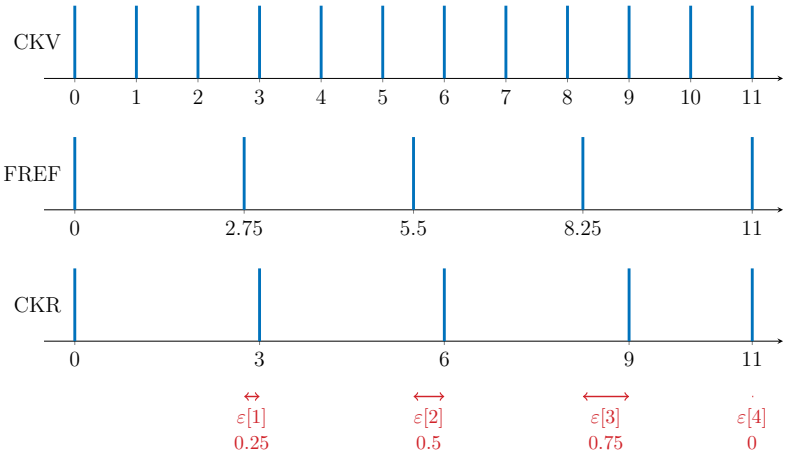


Figure 2.8: Retiming of fractional-N ratio with FCW = 2.75.

The initial phase offset Φ_0 in (2.17) will be constant in case of a type-I PLL or decay completely if a type-II PLL is implemented.

2.1.3.2 Reference Retiming

The sampling operations in two asynchronous clock domains, as described above, lead to a remaining phase discrepancy that needs to be resolved in order to avoid metastability issues. [SB06] proposes a clock retiming mechanism to synchronise the clock domains, i.e. create a common time stamp variable for the indices i and k. The high-frequency signal CKV is used as sampling clock for the low-frequency FREF input. An example for a fractional ratio of FCW = 2.75 is shown in figure 2.8. Ideally the sampling process is implemented as a simple D-flip-flop that updates the FREF level on each successive CKV edge. Thus, the normalized phase expressions from (2.17) have a common index k. However, the resampling introduces a delay between FREF and CKV that originates from the fractional portion of the FCW.

$$\begin{aligned} \Phi_v[k] &= \lceil k \cdot \mathrm{FCW} \rceil \\ \Phi_r[k] &= k \cdot \mathrm{FCW} + \Phi_0 + \varepsilon[k] \end{aligned} \tag{2.20}$$

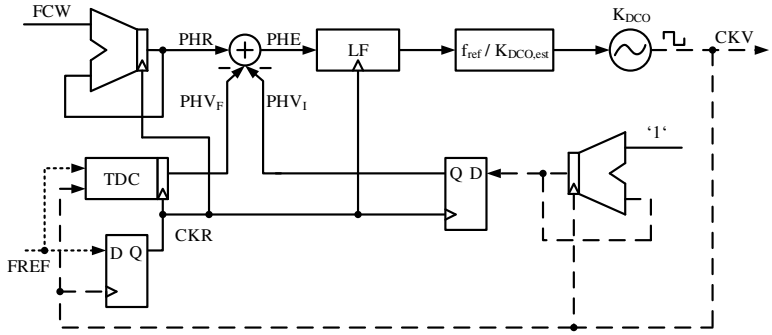


Figure 2.9: ADPLL system overview with clock domain highlighting.

It can be seen as a quantization or rounding error $\varepsilon[k]$ between the actual FREF edge and the following CKV edge. It should be noted that

$$\mathrm{PHV_F} = -\varepsilon\,. \tag{2.21}$$

Figure 2.9 depicts an ADPLL implementation with retiming mechanism. The signal paths are coded with respect to their coresponding clock domains. The main operations, marked as solid lines, are synchronized to the retimed clock CKR. The integer phase conuter runs in the CKV domain (large dashed). A TDC is used to asynchronously estimate the quantization error between the integer phase counter and the FREF domain (small dashed). Additional registers at the TDC output and integer phase accumulator synchronize these signals to the CKR domain.

2.1.4 Controlled Oscillators

The proposed PLL architectures all need a high-frequency oscillator to synthesize the RF output frequency. Two common design choices are ring oscillator (RO) and LC resonator based circuits.

RO based circuits utilize an odd number of inverters with the output of the last one fed back to the first inverter to force a free running oscillation (figure 2.10). By limiting the amount of current available to the inverter stages, the propagation delay and thus the oscillation frequency can be tuned.

However, process variation and the high amount of transistors introduce additional noise to the system making it unsuitable for the strict requirements needed in communication circuits.

An alternative are LC resonator based designs. A possible design is shown in figure 2.11. The resonant frequency of the oscillating circuit is given by:

$$f_{OUT} = \frac{1}{2\pi\sqrt{LC_{tune}}} . \tag{2.22}$$

Using varactors as capacitance allows to tune the output frequency using an analog voltage. The cross coupled transistors provide a negative impedance to counter the ohmic losses and damping of the oscillation.

A variation of the above design, using switchable capacitor unit cells instead of varactors, allows a digital control of the output frequency. Thus, the circuit is called DCO and an exemplary implmentation is shown in figure 2.12. It is preferable in a digital PLL since it alleviates the need for intermediate signal conversion by a DAC. Furthermore, it addresses a major concern emerging from deep-submicrometer PLL implementations. The supply voltage range is constrained by the feature size of the complementary metal oxid semiconductor (CMOS) process [Sta+03]. Hence, the dynamic range of an analog varactor control signal is bound and makes the frequency synthesizer prone to noise degrading the overall SNR. On the downside, DCOs generally suffer from discontinuities due to the nature of the discrete digital control.

Each tuning-bit modulates the driving inverter of a capacitor cell, switching it either on or off. The oscillator output frequency is thus altered according to the capacitor states, which corresponds to a digital-to-frequency conversion. Therefore, the LC tank can be modeled as in figure 2.13. The capacitors should be matched and are typically either binary or unitary weighted. A

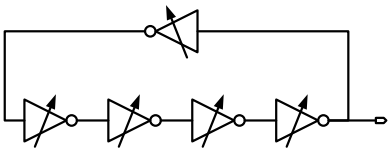


Figure 2.10: Schematic of a basic ring oscillator.

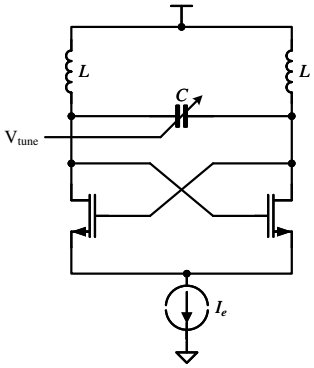


Figure 2.11: LC resonator.

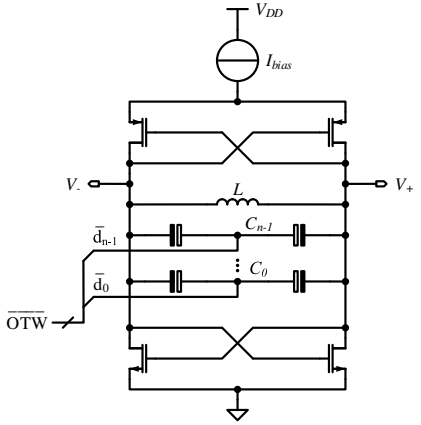


Figure 2.12: Example implementation of a DCO.

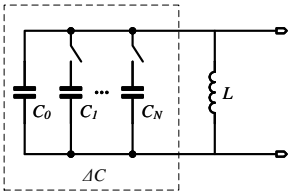


Figure 2.13: Implementation of a capacitor bank for the LC tank of a DCO.

shunt capacitor C_0 of unit weight is employed to ensure DCO operation at default frequency for start-up when tuning word is not yet set to a valid value.

$$C_{tot} = \sum_{k=0}^{N_{on}} C_k \tag{2.23}$$

$$= C_0 + \sum_{k=1}^{N} 2^{k-1} \cdot \bar{d}_k \Delta C \tag{2.24}$$

The total capacitance C_{tot} of a binary weighted capacitor bank is calculated as the sum of all turned-on capacitors – see (2.23) – including the shunt capacitor. This term can be enlarged to (2.24) which considers a binary control word OTW. $\bar{d}_k$ represents the k-th control bit of the capacitor bank. It should be noted that the inversion of control bits is necessary to accommodate the direct digital-to-frequency conversion with inverse relation between capacity and frequency given by

$$f = \frac{1}{2\pi \cdot \sqrt{LC_{tot}}} \,. \tag{2.25}$$

The frequency steps are non-linear in regard to a single capacitor step ΔC. However, applying the first derivative one can linearize the tuning steps $\delta f / \delta C$ for small deviations of the capacitor value:

$$\Delta f(f) = \frac{\delta f}{\delta C} = -f \frac{\Delta C}{2C_{tot}} \,. \tag{2.26}$$

The total capacitance C_{tot} can be substituted with (2.25) resulting in a lin-

earized frequency offset Δf around a center frequency f:

$$\Delta f(f) = -2\pi^2 L \Delta C f^3 \,. \tag{2.27}$$

2.1.5 Time-to-Digital Converters

As described in section 2.1.2, fractional-N PLLs allow to use reference frequencies f_{ref} greater than the channel spacing Δf_{ch}. The integer phase counter in the feedback path captures significant edges at multiples of the output frequency f_{dco}. The remaining fractional error, $\varepsilon[k]$ as depicted in the retiming scheme in figure 2.8, appears as quantization noise, which increases the phase noise at the DCO output. For mobile applications circuits should keep the in-band phase noise below −100 dBc/Hz [VLC10b]. To fulfill the given restrictions, the fractional portion of the output phase needs to be measured and evaluated. Therefore, the concept of time-to-digital conversion is introduced.

One of the most simple implementation of a TDC is a counter based structure. Two modes of operation exist. Either the edges of the signal to be sampled are counted during one reference period or the reference signal is a high frequency signal operating on frequency $f_{count} \gg f_{signal}$ and the reference periods between successive signal edges are counted. The result is an integer multiple of the higher frequency signal fitting in one slower signal period. The highest possible time resolution is therefore limited by the high frequency sampling signal. Some scalability in time resolution can be achieved by adjusting the frequencies or measuring over multiple periods at the cost of higher latency. The counter based TDC structure introduces quantization error when the start and stop edges are not in phase with the signal edges. The introduced measurement error is therefore up to two sampling signal period.

An alternative method of converting time difference to a digital quantity is utilizing delay elements and chains. Digital delay elements are buffers having a control signal to slow the propagation of a signal edge. *RS* flip-flops are often used as 1 bit TDCs able to detect which input is triggered first. The operational table can be seen in table 2.1. The internal operating structure consists of either cross coupled NAND or NOR gates, as can be seen in figure 2.14. They are especially suited for TDCs because of the symmetric input impedance [VLC10b].

S	R	Q	$\overline{Q}$	
0	0	Q_{-1}	$\overline{Q_{-1}}$	hold
0	1	0	1	reset
1	0	1	0	set
1	1	1	1	not allowed

Table 2.1: RS flip-flop operation

(a) $\overline{RS}$ flip-flop. (b) RS flip-flop.

Figure 2.14: Flip-flop implementations.

A flash structure is the simplest delay-based TDC architecture. As depicted in figure 2.15, the time comparison of the edge delays is accomplished by $\overline{RS}$ flip-flops. The first edge, called start signal, propagates the delay chain

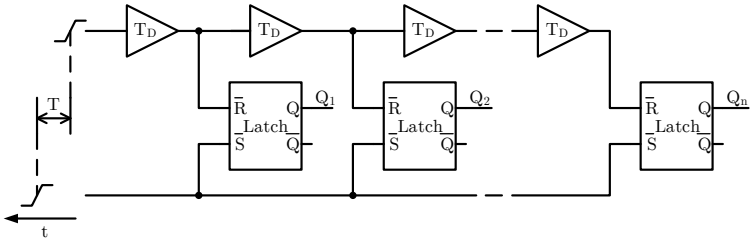


Figure 2.15: Flash-TDC with unit delay T_D and $\overline{RS}$ flip-flops as time comparators.

adding a delay of T_D after each stage. The second edge, the stop signal, is directly connected to the inverted set input of the flip-flops. The output Q_i of the i-th stage toggles from '1' to '0' if the start signal has not yet arrived at its inverted reset-port. However, if the stop edge arrives after the start edge at the flip-flop, the previous '1' is registered in Q_i. The resulting output array $\{Q_1 ... Q_N\}$ contains the thermometer encoded time difference T in terms of

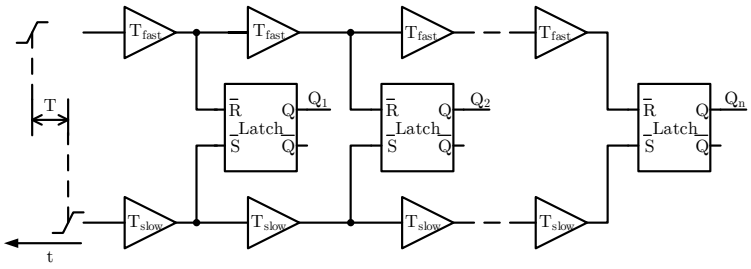


Figure 2.16: Vernier TDC.

T_D unit-delays:

$$T = T_D \cdot \sum_{i=1}^{N} Q_i \,. \tag{2.28}$$

The resolution is limited by the smallest achievable time delay of the technology while the dynamic range is only limited by the number of delay cells connected in series.

$$T_{LSB,flash} = T_D \tag{2.29}$$
$$DR_{flash} = n \cdot T_D \tag{2.30}$$

To allow finer time resolution the Vernier method can be employed [Bar57]. Here both edges – start and stop signal – are fed into delay lines with different unit delays. The outputs are again connected to $\overline{RS}$ flip-flops (figure 2.16). Thus, using this differential design, the resolution is no longer limited by the absolute delay of the buffer elements but instead by the delay difference between the two chains.

$$T_{LSB,Vernier} = T_{slow} - T_{fast} \tag{2.31}$$
$$DR_{Vernier} = n \cdot T_{LSB,Vernier} = n \cdot (T_{slow} - T_{fast}) \tag{2.32}$$

Additionally to allowing much smaller unit delay to be resolved, the structure is also more process, voltage, temperature (PVT) variation resistant because these quantities affect both delay chains in the same way.

Other architectures also focus on improving the smallest possible time resolution. E.g. pulse shrinking TDCs as presented in [Gra+18] utilize differences

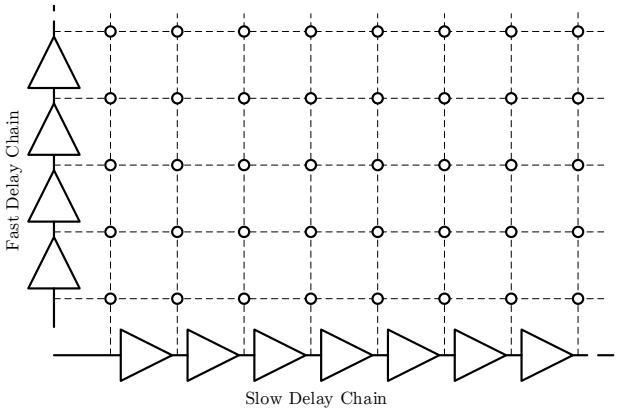


Figure 2.17: Two-Dimensional Vernier TDC Structure.

in delay of rising and falling edges to generate a slowly decaying pulse. Alternatively several architectures exist to amplify the phase difference at the input [LA08; Kwo+11].

To overcome the dynamic range limitations in Flash and Vernier TDC structures, several possibilities exist. One solution is to close the delay chains to a feedback loop creating a ring oscillator. An inverter is connected to the end of the delay chain and its output connected to the input of the first delay element. Now, the start and stop signal can propagate endlessly through the oscillator resulting in an infinite dynamic range.

Another method is arranging the latches in a two-dimensional matrix [VLC10a]. Each delay element's output is evaluated against every element's output from the other chain as can be seen in figure 2.17. The $\overline{RS}$-flip-flops will trigger in diagonal succession while the delay difference accumulates. The fold back point to the next diagonal line needs to occur after N_{fold} stages to ensure a continuous conversion as given in (2.33):

$$N_{fold} = \frac{2T_{slow} - T_{fast}}{T_{LSB}} \,. \tag{2.33}$$

Thus, the dynamic range per delay element is increased but non-linearities are introduced at the folding points [VLC10a]. They are caused by a change of number in delay elements used for the conversion. A solution to handle this problem is presented in [WDW17] and [WD17]. The matrix is reconfigured by placing subsequent evaluation points at the end of diagonal lines in a

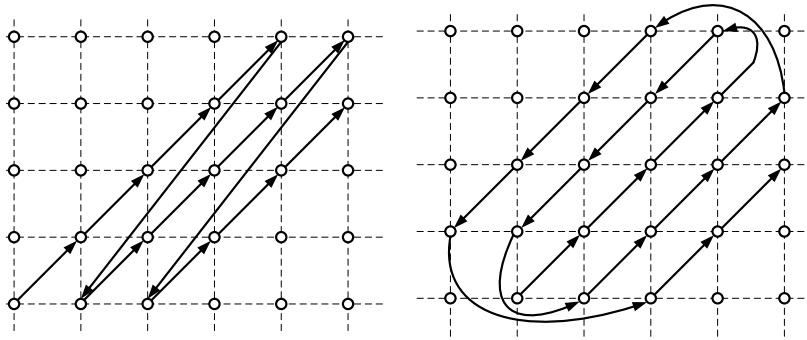


Figure 2.18: 2D Matrix structures.

spiral configuration as shown in figure 2.18. This minimizes the difference of added non-linearities at the end of diagonal lines compared to the small non-linearities at the beginning of the diagonal lines.

2.1.6 Noise in RF Circuits

Noise is caused by random processes in the devices. It includes but is not limited to flicker and thermal noise. Random electron movement in the device causes fluctuations of the voltages across the device. This thermal noise affects all electronic devices. Its spectral density given in (2.34) is constant over frequency. Flicker noise, also called $\frac{1}{f}$ noise, occurs in MOSFETs at the interface between the gate oxide and the silicon. It is inversely proportional to the frequency as can be seen in (2.35) [Raz01].

$$S_{thermal}(f) = 4kTR \tag{2.34}$$

$$S_{flicker}(f) = \frac{K}{C_{ox}WL} * \frac{1}{f} \tag{2.35}$$

Both contributions will effect the circuits in the form of jitter or phase noise.

2.1.6.1 Power Supply Noise

Additionally, to noise processes in the devices, random fluctuations of the supply voltage also affect the performance of electronic circuits. This so-called power supply noise can have various sources such as:

- electromagnetic interference from external sources
- ground bounces on-chip or from the external printed circuit board (PCB)
- simultaneous switching noise (SSN) from other circuits

Due to the close proximity of different subcircuits in integrated systems and a lowered supply voltage, SSN is gaining importance in advanced submicrometer nodes.

For digital logic cells, the current consumption depends on the switching activity, transition time and output load. Charging and discharging the transistor's gate capacity leads to an IR-drop on the supply voltage. Each activity triggers further switching in successive cells. Since the digital signal processing is in general clocked by a reference clock, the current drawn by the circuit is periodic, causing a regular disturbance of the supply voltage that can be described as power supply noise (PSN). This noise may also affect analog and RF blocks on the design through direct cross-coupling. Additionally, a similar affect can be observed from the analog circuits themselves. Due to input signals controlled by a clocked circuit, thier activity is also changing periodically. The power consumption of e.g. power amplifiers, dividers, etc. is thus not constant but varies with the periodicity of the input signals.

2.1.6.2 Phase Noise

An ideal oscillator operates at a single frequency f_0. The output $v(t)$ can be described by a sinusoidal function as written in (2.36) with amplitude A and a fixed phase reference ϕ [SFB05]. A Fourier transform of this signal results in the single-sided spectrum consisting of a Dirac impulse at f_0 defined by (2.37) as shown in figure 2.19.

$$v(t) = A \cdot \cos(2\pi f_0 t + \phi) \tag{2.36}$$

$$S_v(t) = \frac{A}{2} \cdot \delta(f - f_0) \tag{2.37}$$

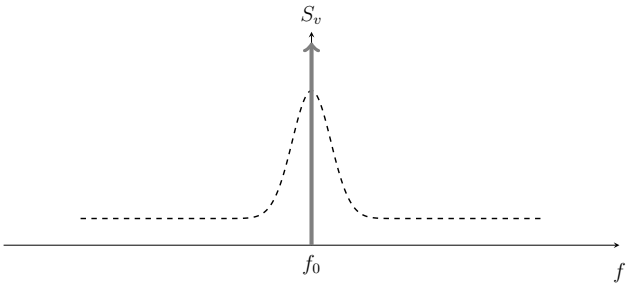


Figure 2.19: Output spectrum of ideal (thick) and real (dashed) oscillator.

Noise and non-linearity in real oscillators exhibit small random phase variations $\phi(t)$. These variations can be translated to frequency alterations by (2.38), if the arbitrary assumption $\phi(t) \ll 1\,\text{rad}$ holds:

$$v(t) \approx A \cdot \cos(2\pi f_0 t) - A\phi(t) \cdot \sin(2\pi f_0 t)\,. \tag{2.38}$$

This appears as smearing of the fundamental signal power in a real oscillator over neighboring frequencies, shown in figure 2.19 by the dashed line, and causes degradation of the overall SNR [Kes]. According to [Kun15], any form of perturbation is amplified in the oscillator and appears mainly in the phase. In steady-state operation, amplitude noise eventually decays over time, whereas phase shifts are accumulated during operation leading to a permanent phase shift $\Delta\phi$. The power of the phase perturbation increases with $1/\Delta f^2$ in close vicinity to the oscillating frequency f_0. Phase noise is usually quantified within a 1-Hz unit bandwidth in a Δf offset from the carrier relative to the carrier power resulting in a single-sided spectral density (SSD) given by (2.39):

$$\mathcal{L}\{\Delta f\} = 10\log\left(\frac{\text{noise power in 1-Hz BW at } f_0 + \Delta f}{\text{carrier power}}\right)\,. \tag{2.39}$$

By definition in [IEE08], the single-sided phase noise $\mathcal{L}\{\Delta f\}$ is one half of the noise spectrum $S_\phi(\Delta f)$.

The idealized phase noise spectrum of an oscillator is plotted in figure 2.20. It shows the phase noise S_ϕ normalized to the carrier signal power in Δf distance. Following four different regions can be identified:

1. *Broadband thermal noise (f^0):* Out-of-band uncorrelated timing fluctuations with flat spectral characteristic.

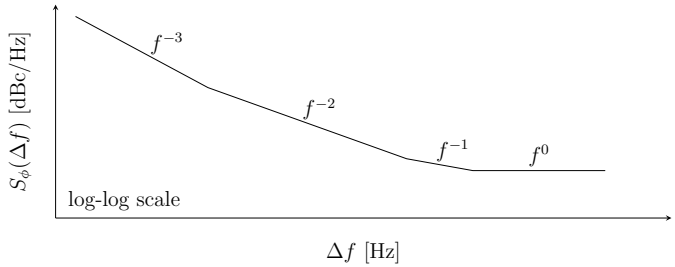


Figure 2.20: Idealized phase noise regions in Δf distance to carrier.

2. *Flicker* $1/f$ *noise* (f^{-1})*:* Especially present in transistors due to random generation and recombination of charge carriers and impurities in the channel. It features a −10 dB/dec slope for frequencies in vicinity of the carrier.

3. *Upconverted thermal noise* (f^{-2})*:* Thermal fluctuations or white noise upconverted by oscillator with −20 dB/dec slope.

4. *Upconverted* $1/f$ *noise* (f^{-3})*:* Frequency-to-phase conversion (integration) adds a 20 dB/dec attenuation on top yielding a total of −30 dB/dec.

2.1.6.3 Jitter

An equivalent description of noise contributions in time-domain is called jitter. It is typically used when talking about clocked circuits. At each sampling edge the circuit takes a snapshot of the input signal. Since the input signal is generally not constant over time, any deviation of the exact sampling point will create an error in the sampled value. These timing variation are called aperture jitter, when referring to ADC. Random jitter causes a random deviation in the sampled values, which can be described as additive noise and thus a deterioration of the SNR [RTL09].

Electric thermal noise is causing timing deviations Δt_j following an additive white gausian noise (AWGN) distribution. The deviation is additive to the corresponding period T_j as in (2.40) but will affect consecutive periods. Hence, it is referred to as non-accumulative jitter depicted in figure 2.21:

$$t_j = j \cdot T + \Delta t_j \,. \tag{2.40}$$

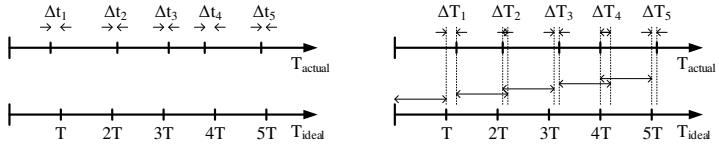


Figure 2.21: Timing jitter as non-accumulative (left) and accumulative (right) jitter.

The variance $\sigma_{\Delta t}$ can be derived from the total noise power. The single-sided spectral density $\mathcal{L}$ of the noise floor is multiplied by two and integrated up to the Nyquist frequency. The square root of this power is called rms jitter which then is normalized with $T_0/2\pi$. Rearrangement yields

$$\sigma_{\Delta t} = \frac{1}{2\pi}\sqrt{\frac{10^{\mathcal{L}/10}}{f}}\,. \tag{2.41}$$

Upconverted thermal noise can be described similarly. As before, jitter following an AWGN distribution is added to each period T_j. However, the excess phase close to the carrier does not diminish. Therefore, this random jitter is additive to each following period as seen in (2.42) and called accumulative jitter (figure 2.21):

$$t_j = j \cdot T + \sum_{i=0}^{j} \Delta T_i\,. \tag{2.42}$$

The rms jitter of this noise type is computed with the following equation [WK08]:

$$\sigma_{\Delta T} = \frac{\Delta f}{f}\sqrt{\frac{10^{\mathcal{L}/10}}{f}}\,. \tag{2.43}$$

Transforming flicker noise to an equivalent jitter distribution can be done with different techniques such as shaping flat white spectral noise with a filter chain to gain a −10 dB/dec slope [SFB05] or by utilizing a random number generator with the Voss-McCartney algorithm [Vos78].

Susceptibility to clock jitter is a known weakness of continuous-time delta sigma ADC. Firstly, it causes a sampling error at the internal ADC, although

this noise has a comparably small effect on the overall performance, because it is shaped by the NTF and therefore attenuated. Secondly, the jitter causes an error in the internal DAC. This noise in turn is not attenuated in signal band, but transferred by the STF. Thus, its influence is much more severe. By using internal multi-bit ADCs and DACs the effect of jitter can be reduced [OG06].

2.2 Simulation Methods

To ensure the correct functionality of the developed systems, a circuit verification is necessary. Since verification on real prototypes is in general impractical and expensive for integrated circuits, a computer aided method needs to be used. There are two major categories to classify these approaches [Seg+04]:

- formal verification
- simulation-based analysis

Formal verification tries to mathematically prove the equivalence between two system descriptions [Lam05], e.g. the circuit model and its specification or the high-level model and its synthesized version. It is well suited to be used in digital systems, which posses a limited number of possible states. Analog signals do not have discrete states but a continuous state space [WA14]. Therefore, formal verification methods are mostly unsuited for analog or mixed-signal circuits, although there exist some approaches to discretize the available state space [SH08; Ste11].

In contrast to formal verification methods, simulation based circuit analysis does not check the mathematical correctness of the system but analyses the circuit behavior for a defined set of input stimuli. Since it is impossible to simulate all input stimuli in an acceptable amount of time, test scenarios have to be selected that allow conclusions of the circuit's behavior and performance to be drawn. A typical virtual test environment is shown in figure 2.22. The set of needed stimuli to address the device under test (DUT) is derived from the product specification. To allow an automated verification of the DUT output the driving signals are passed to a golden reference model created directly from the specification [Ber12]. A scoreboard compares the reference output with the monitored values from the DUT. If the reference model does not exist, the signals need to be checked in the monitor if they adhere to the allowed specification.

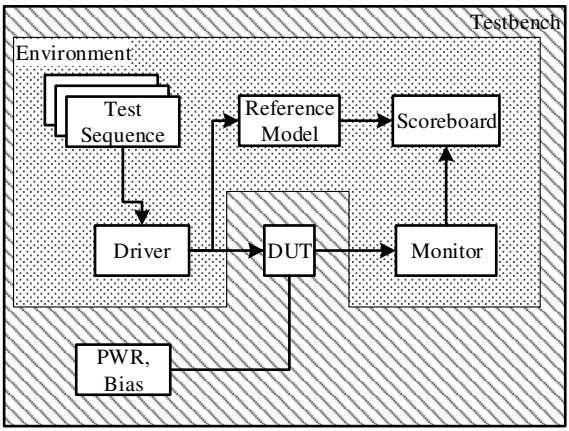


Figure 2.22: Typical testbench environment.

Different simulation methods exist depending on the abstraction level. For circuit level simulation matrix-based analog simulations and event-driven digital simulations are the most common.

2.2.1 Matrix-based Analog Simulations

Matrix-based circuit simulations use nodal analysis to create a formalized description of the electrical circuit and its components. The electrical system is a conservative system where each node is described by its voltage and the currents flowing between them [FM98]. The relationship between current and voltage is given by the characteristic equation of the electrical components. As an example, for a resistance, Ohm's law describes the linear relation between voltage difference and current. Following Kirchhoff's law, which states that the sum of all currents in a node is equal to zero,a nonlinear system of differential equations that describe the circuit is created [War09]. The Modified Nodal Analysis extends the nodal analysis with a few exception for ideal or controlled voltage sources, which allows a completely systematic approach to building the circuit description that can be implemented in a computer.

The equation system needs to be solved during simulation. Several analysis methods can be used. One of the straight forward approaches is called transient simulation which solves the differential equations in time-domain. The equations are transformed to a set of difference equations using integration methods [Kun06]. Next the system of equations is solved numerically by iterative methods such as the Newton-Raphson algorithm [Ypm95]. While the transient system analysis can be run on any circuit with no limitations on the linearity or periodicity of the signals, the time steps chosen typically depend on the highest frequency present in the system. Thus it is not well suited for long time constants in systems with high frequency components.

Direct current (DC) simulations assume a steady state of the system and calculate a static operating point for the circuit. Thus, all inductors are treated as shorts while the capacitances are regarded as open clamps [War09]. The simplified set of equation can be solved with the same algorithms as transient simulations. The results can be used for a starting point for other simulations like the transient simulations described above or the small signal analysis.

The small signal or alternating current (AC) analysis calculates the circuits response for a linearized system in the frequency domain. It is assumed that nonlinearities of the circuit can be neglected since the amplitudes of the input signals are small. Typically it is run after a DC simulation which determines the operating point around which the system is linearized. Due to the linearization AC simulations are suitable to quickly provide transfer functions for circuit blocks, such as the power supply rejection ratio of a low-dropout regulator (LDO). However, the operating point needs to be stable during the operation and the effect of nonlinearities can not be considered.

Thus, the AC analysis is often not sufficient for RF circuits. On the other hand, transient simulations are also not well suited since the highest frequencies in the system often lie in the GHz-range while other subsystems have time constants of a few milliseconds. Knowing the circuit will end up in a steady state, RF simulation methods exploit these characteristics to find a periodic stable operating point. The two main RF simualtion methods are *Periodic-Steady-State* and *Harmonic Balance* [Kun97].

The Periodic-steady-state (PSS) assumes a periodicity with a duration T for all currents and voltages in the system. Thus the system is in a steady state. It can be solved in time domain similarly to a transient simulation with the additional boundary condition:

$$v(t_0 + T) = v(t_0) \; . \tag{2.44}$$

Additional small-signal analyses can be applied to the so found steady-state solution, e.g. *pac* or *pnoise* [Kun97].

Harmonic balance (HB) simulation method works similarly, but in contrast to PSS the initial steady-state solution is computed in the frequency domain. It is assumed that all variables can be written in the form

$$v(t) = \sum_k \sin(k \cdot \omega \cdot t) \,. \tag{2.45}$$

The approach can also be extended to multiple fundamental frequencies [PC03]. Similarly to PSS, extending small-signal simulations exist. Which simulation approach is better suited depends on the circuit. Typically, HB is preferred if only a small number of harmonics is of interest, while PSS is used for systems with large discontinuities [Kun97].

2.2.2 Event-driven Digital Simulations

Matrix-based simulation methods as described above are only suited to handle small to medium sized circuits. For each time step the complete set of equations needs to be solved. In general this is of complexity $O(n^3)$ [Sto13]. Using an intelligent re-partitioning of the typically sparsely populated matrix, allows to reduce the effort to $\sim O(n^{1.5})$ [Shi10]. Nevertheless, even slow changing nodes need to be recalculated in every time step which makes it inefficient especially in circuits with largely different time constants.

Event-driven simulations try to solve this by applying an unidirectional signal flow. Thus, there is no influence from the output of a component on its inputs. This allows to only reevaluate the components equations if one of the input changes. Since the influence from a logic circuit on its input gates is negligible, this technique is especially suited for digital circuits. A further abstraction to Boolean logic reduces the solution space drastically.

Hardware description languages (HDLs) are used to describe the systems to be simulated. The most common languages are very high speed integrated circuit hardware description language (VHDL), SystemVerilog (SV) and SystemC (SC). Components are defined by their input and output ports. Sensitivity lists handle which outputs need to be reevaluated if an input changes. A global event queue schedules which evaluations and updates need to be executed. It is visualized in figure 2.23. A problem occurs if two signals simultaneously trigger an update of a component. In real hardware both effects happen concurrently but for simulation on a computer they need to be executed

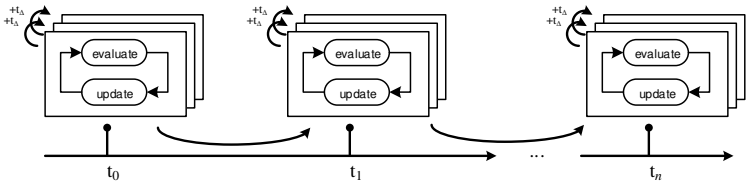


Figure 2.23: Event-driven simulation process.

sequentially. Therefore, infinitesimal simulation time steps t_Δ are introduced. Output signals are typically only updated after all Δ cycles are processed, thus ensuring a predictable and repeatable behavior.

Main advantage of event-driven simulations is the reduced computational effort. Signals do not need to be recalculated at each time step but only after a change in a related signal [Kes12]. The effects can be solved locally in a single time step. Thus, it is possible to simulate periods with little activity with a small number of simulation steps. The gain in simulation speed is traded for some loss in accuracy compared to SPICE-like matrix-based simulations [Acu+90].

2.2.3 Mixed-Signal Simulations

Today's SoCs are typically mixed-signal systems containing analog and digital parts. The verification of these systems is a challenging task since both digital and analog as well as their interactions need to be simulated [Shi10]. Mixed-mode simulators handle these tasks by running both the matrix-based analog simulation and the event-driven digital simulation in parallel [Acu+90]. To be able to do so two main challenges have to be resolved:

1. different time concept of both simulators
2. translation of the signal representation between simulators

As described in section 2.2.2 digital simulators use a set of events at discrete times to update the systems state while analog simulators have a continuous time that is sampled at variable time steps depending on the involved signals. A mechanism is needed to synchronize these two time concepts. One solution is to run both simulations independently. If an event in either simulation

affects the other one, this simulation has to retrace to that time and restart with the changed input conditions. This method is called backtracking [Acu+90]. In an alternative approach called lockstep [Zwo+95] both simulators can run on a fixed time step at which re-synchronization happens. The analog solver can further reduce the time steps if necessary.

For signal representation a translation between the continuous analog and discrete digital signals is needed. *Connect modules* are inserted automatically by the simulator at the border between simulation domains [FO00]. To allow the translation some assumptions regrading current drive and input impedance are needed. To enable a more fine-grained control of the analog properties, extensions to the classical HDLs have been specified, e.g. Verilog AMS [EDA14] and VHDL AMS [IEE99]. They can further be used to replace transistor-level implementations, which are simulated in the analog solver, with higher level macro models to speed up the overall simulation.

The mixed-mode simulators are well suited for medium-sized circuits. However, for large systems, especially with large analog parts, the analog solver dominates the simulation speed [VOM15]. While the performance of the digital simulator scales linear with size due to the signal flow approach, the performance of the analog solver scales superlinear. Therefore, real number modeling (RNM) and the development of signal flow based analog macro models has been widely adopted over the last years [Shi10; VOM15; Dav10]. For instance, SV [IEE13] and VHDL [IEE09] also support floating point variables. The analog circuit blocks are simplified in such a way, that outputs can be calculated directly from the input values. Continuous changes need to be sampled instead. These models are well suited to allow fast functional verification of large and complex systems [Dav10].

2.3 Circuit Modeling in SystemVerilog

SystemVerilog is a HDL, that can be used for describing hardware designs as well as implementing testbenches. It is an extension to the Verilog Hardware Description Language as standardized in the IEEE Standard 1364-2001 [IEE01]. Syntactically SV is similar to C++ [Spe08].

Complex design structures are typically described in a hierarchical way, combining common building blocks. As Verilog, SV uses modules to describe hierarchical structures. Typically modules are either described as a set of

```
1  module EXAMPLE (
2    A, B, C, Y, Z
3  )
4    //port and parameter definitions
5    input A,B,C;
6    output Y, Z;
7    inout VDD,VSS;
8
9    logic A,B,Z;
10
11   real C, Y;
12
13   parameter real multiplicator = 2.3;
14
15   //variable definitions
16   real pi = 3.14;
17
18   initial begin
19     // description of initial condition
20     Z <= '1;
21   end
22
23   // continuous assignment
24   assign Y = C * multiplicator + pi;
25
26   always @(A,B) begin
27     // code to update module
28     Z <= A + B;
29   end
30
31 endmodule
```

Listing 2.1: SV module description.

submodules and their connections or directly characterized by their functionality [SDF06]. To communicate with the outside world modules contain input and output ports. These ports have a data type associated with them and are connected via nets of the same data types.

Listing 2.1 shows the structure of a typical SV module. At first the ports are declared (ln. 2). Afterwards the port type and directions as well as additional parameters and variables are defined (ln. 5-16). For an event-driven description of a module's functionality two blocks of code are used. The *initial* block (ln. 18-22) is evaluated at module creation. It therefore describes the starting conditions of the module. During simulation signal changes trigger *always @()* blocks (ln. 24-28). The code in these blocks models the functionality of the module. The sensitivity list *@(A,B)* is used to define which signal changes trigger the execution of the block. Multiple *always* blocks with different sensitivity lists can be used to model different aspects of the blocks behavior separately. Alternatively, the *assign* statement can be used for continuous

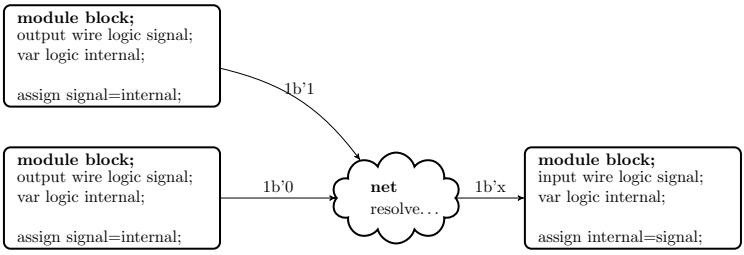


Figure 2.24: SystemVerilog net functionality.

assignments.

To be able to model hardware behavior of signals SV includes two diferent types: variables and wire. Variables behave similar to what is known from sequential programming, in the sense, that a variable's value is equal to the value lastly written to it. Wires, on the other hand, allow multiple different drivers writing to them simultaneously. An internal net resolution function resolves these multiple drivers and calculates the resulting output value as shown in figure 2.24.

The goal of SV is to have a unified HDL to describe verification and design. Therefore, several features were added to the Verilog standard that was most suited for design descriptions [Spe08].

New data types like enumerations, logic and structs can be used. Especially structs are an advantage because they allow storing multiple information in the same variable. The newly added user-defined nets and resolution functions allow passing this information over a single port. An example declaration of these user-defined data type (UDT) and user-defined resolution function (UDR) is presented in listing 2.2. Also, SV introduces object-oriented programming concepts. While Verilog had a strong division between data and functions that use these data, an object-oriented approach defines data structures and methods that work on these data together. The structuring of code in classes simplifies the reuse of code in different scenarios. This is especially useful when implementing testbenches or models to be used in different projects. It allows to describe the functionality on a higher abstraction level, decoupling it from the design details.

Additionally, the DPI gives access to C/C++ functions allowing the reuse of common system libraries. These libraries include several functionalities already implemented in an efficient way which can be used for modeling the

```
typdef struct{
  int a;
  real b;
  bit [0:3] data;
} exampleStruct;

function automatic exampleStruct res_function (input exampleStruct driver[])
  // Implementation of the resolution function goes here
  res_function.a = driver[0].a;

  res_function.b = 0;
  foreach (driver[i]) begin
    resfunction.b += driver[i].b;
    resfunction.data = (res_function.data ~= driver[i].data);
  end

endfunction : res_function

nettype exampleStruct exampleNet with res_function;
```

Listing 2.2: Definition of a UDT and UDR in SV.

system behavior. Also custom written C++ code can be integrated into SV functions. To use an external function in SV it has to be declared:

```
import "DPI-C" pure function int add(input int a,b);
```

Listing 2.3: Declaration of a C function in SV.

Most SV data types translate to an equivalent data type in C. The complete table can be found in [Spe08]. An important data type to be passed back from C++ is a pointer. A special SV data type *chandle* exists to store pointers. Listing 2.4 shows the SV declaration of a function to create a new C++ class object.

```
import "DPI-C" function chandle LogicCell_create(input string CellName);
```

Listing 2.4: Declaration of a function to create a new C++ object in SV.

The DPI can only connect to C++ functions known at link time. Thus C++ class methods, that depend on objects created at runtime, need to be exposed as static C functions. As an example the C++ constructor for the class object from Listing 2.4 is given in Listing 2.5

```
#include "LogicCell.h"
external C void* LogicCell_create(char* CellName) {
  return new LogicCell(std::string(CellName));
}
```

Listing 2.5: C++ constructor exposed for SV DPI.

CHAPTER 3

AMS MODELING FRAMEWORK

3.1 Generic Simulation Framework

As already introduced in section 2.2.2 several hardware description languages exist to model digital circuits. Most have introduced extensions to also allow modeling analog behavior, though these are still solved in a node-based simulator, thus, negating the speed advantages of a signal flow based simulation. RNM, as explained in [VOM15], limits itself to a pure event-driven simulator to keep this advantage. This approach is further extended in this thesis to enable an accurate and efficient description of analog signals. Part of the results have already been published in [Spe+18] and are further expanded in this work.

3.1.1 Framework Structure

The developed framework uses a co-implementation in SystemVerilog and C++ to gain the advantages of both programming languages. SV is well supported and integrated in industry standard integrated development environments (IDEs) and verification environments, which allows for an easy inclusion of the framework in existing setups. Furthermore, SV already includes a native implementation of concurrently running task as well as a notion of time that can be reused in the framework.

However, the native analog signal processing capabilities of SV are limited. Therefore, the signal descriptions and behavior of the blocks are implemented in C++ classes. The SV DPI defines a way to bind precompiled C++ code which allows the SV models to access the data and functions on C++ side and vice-versa.

Figure 3.1 shows the implementation structure. The SystemVerilog modules provide pin-equivalent views of the analog building blocks. They should be abstracted on a low hierarchical level — ideally just above transistor level

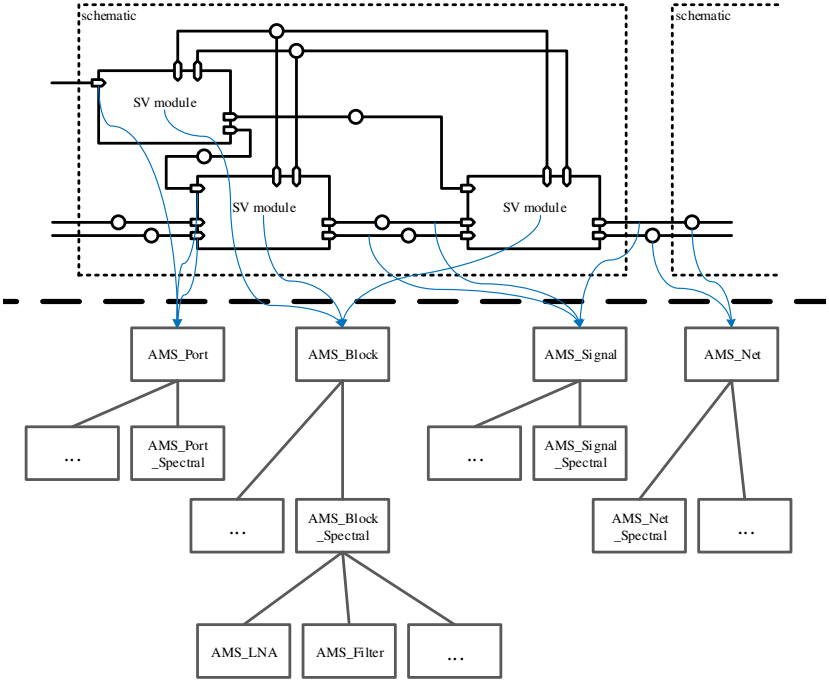


Figure 3.1: Overview of AMS Simulation Framework.

— to allow the verification of the genuine analog toplevel view. This way the actual connections from the higher level schematics are simulated and verified. Furthermore, the low abstraction level simplifies the complexity of the circuits to be modeled. For each modeled element a corresponding block, port, net and signal C++ object is created. Specialized classes exist for different usecases. They are structured hierarchically under abstract parent classes. The parent classes provide a unified application programming interface (API) and allow a structured, standardized way of creating and interacting with the C++ objects. Also they collect common code, that can be reused in the child classes to avoid error-rone re-implementations.

To enable use case specific electrical data types, SV supports user-defined data types (UDTs) and user-defined nets (UDNs). Unfortunately, the UDN comes with some restrictions. E.g. the data type of the net needs to have a predefined

size, it does not support structures with a dynamic amount of components. But most noticeably pointers are not directly supported as well. Thus, for a generic framework a workaround is needed. The framework implements an ID based signaling system. Unique IDs are passed from the ports to nets and from nets to successive ports. The IDs can be used to query a TransactionItem object on C++ side which holds the pointers to all port, net and signal objects involved. Further, the sign of the IDs are used to indicate activity on the net and schedule a reevaluation of the blocks state.

3.1.2 Framework Initialization

The C++ classes are implemented in a pre-compiled library that needs to be included in the event-driven simulator. While for a given simulation the SV modules are statically compiled when invoking the simulator, the C++ objects need to be created dynamically during the simulation. Thus, at the beginning of each simulation an initialization stage is necessary.

In a first step the `AMS_Block` and `AMS_Port` objects for each SV module are created and prepared in an initial state. The `AMS_Ports` create the initial output `AMS_Signal` objects and `AMS_TransactionItems` with unique IDs to be written on the net (see figure 3.2a).

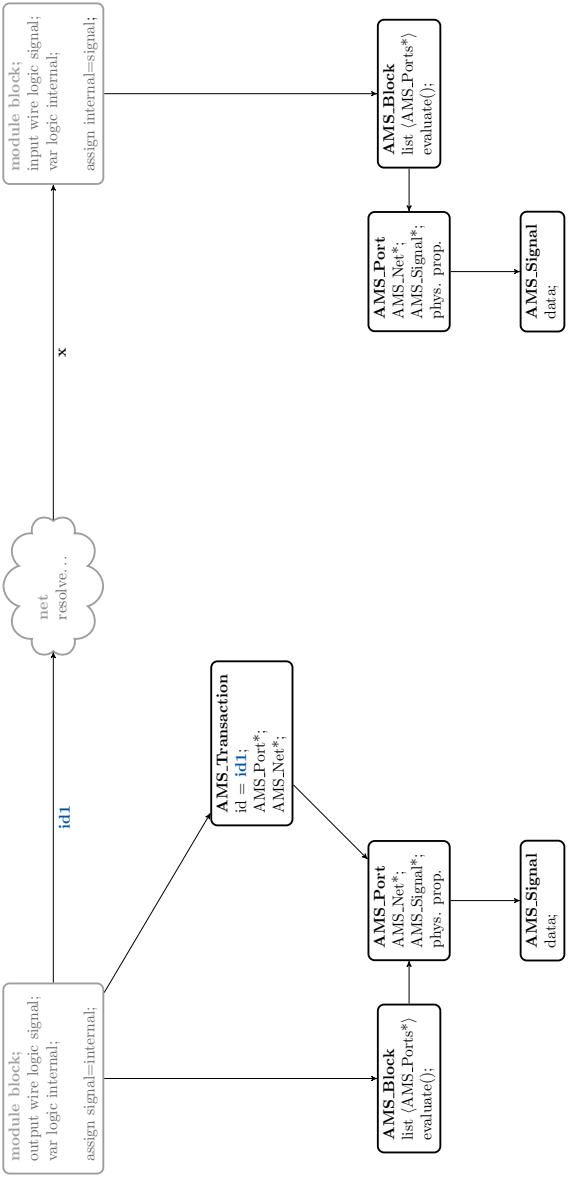
The writing of the IDs activates the net resolution function. The function is shown in listing 3.1. Since the transaction items do not yet hold any net pointer, the creation of `AMS_Net` objects is triggered. The nets create new `AMS_Signals`, that hold the resolved signal if needed, and a new ID for them. They are then passed to the successive SV modules (figure 3.2b).

The new available signals are received by the SV modules due to the change of the ID. The input ports are now able to retrieve the net information from the transaction item and connect themselves (figure 3.2c). This allows to pass e.g. needed electrical information contrary to the main signal flow direction. Having all electrical and connect information available, next the first valid signals can be written by the outputs and nets. From now on during the simulation only an evaluation routine needs to be called to update the block's state.

,

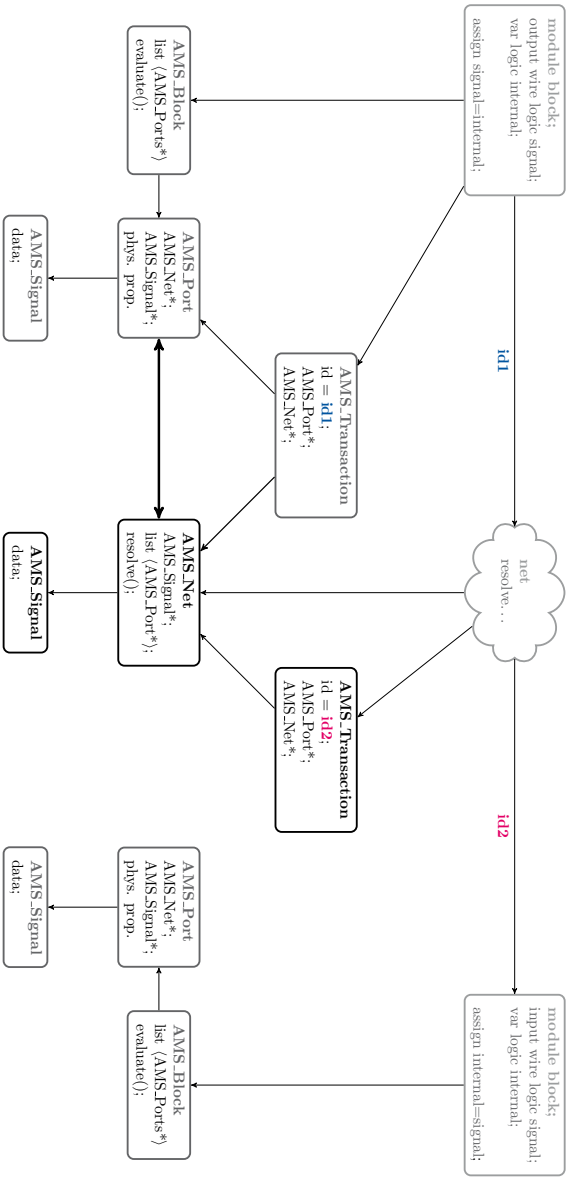
```
typedef real signal_type;
function automatic signal_type net_eval ( input signal_type driver[] );
  chandle net_pointer, transitem_pointer;
  if (driver[0] != 0) begin
    if (AMS_Transaction_get_port( AMS_Transaction_get_ptr(driver[0])) == null) begin
      net_eval = driver[0];
    end else begin
    net_pointer = AMS_Transaction_get_net( AMS_Transaction_get_ptr(driver[0]));
      if (net_pointer == null) begin : Creation
        net_pointer = AMS_Net_new_by_port_type( AMS_Transaction_get_port(
        ↪  AMS_Transaction_get_ptr(driver[0]) ));
        foreach (driver[i]) begin : Connect_Ports
          if (driver[i] != 0 && AMS_Transaction_get_port( AMS_Transaction_get_ptr(driver[i])) != null)
          ↪  begin
            transitem_pointer = AMS_Transaction_get_ptr(driver[i]);
            if(AMS_Transaction_get_net(transitem_pointer) == null) begin
              AMS_Transaction_set_net(transitem_pointer,net_pointer);
              AMS_Net_add_port(net_pointer, AMS_Transaction_get_port(transitem_pointer));
            end
          end
        end : Connect_Ports
        AMS_Object_prepare(net_pointer);
        transitem_pointer = AMS_new("AMS_Transaction");
        AMS_Transaction_set_net(transitem_pointer,net_pointer);
        net_eval = AMS_Net_get_transitem_id_number(net_pointer);
      end : Creation
      else begin
        if(AMS_Object_get_initialized(net_pointer) == 0) begin : Initialization
          AMS_Object_initialize(net_pointer);
          net_eval = AMS_Net_get_transitem_id_number(net_pointer);
        end : Initialization
        else begin : Evaluation
          AMS_Net_evaluate(net_pointer);
          net_eval = AMS_Net_get_transitem_id_number(net_pointer);
        end : Evaluation
      end
    end
  end
endfunction
nettype signal_type signal_net with net_eval;
```

Listing 3.1: AMS signal net evaluation routine.



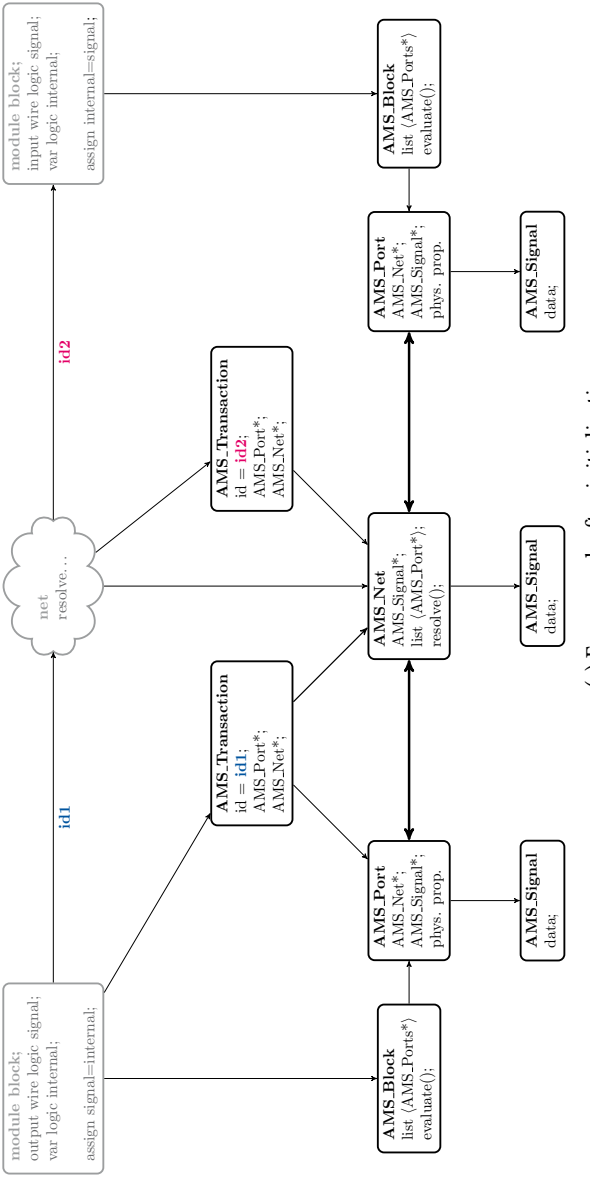
(a) Framework during inital phase.

Figure 3.2: Framework initialization steps.



(b) Framework during first net evaluation.

Framework initialization steps.



(c) Framework after initialization.

Framework initialization steps.

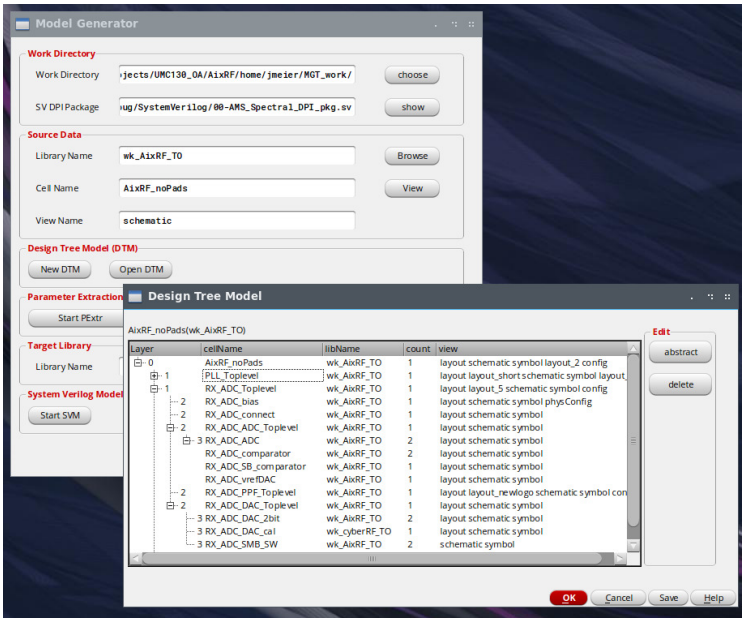


Figure 3.3: GUI to select toplevel and leaf cells to be modeled.

3.1.3 Model Creation

As can be seen from the previous section, the setup allows for a well defined and clearly structured SV module description. An example can be found in listing A.1 in the appendix. After the definition of some parameters and variables, the SV ports are defined. It is followed by a section that holds the pointers to the C++ `AMS_Block` and `AMS_Port` objects. The initial block contains the setup procedure, while the always block holds the initialization and evaluation routine. An additional block allows the probing of the C++ `AMS_Signal` objects for plotting with the integrated waveform viewer.

As can be seen the code is long and repetitive but only contains small parts that are block dependent. Thus, the whole description can be generated automatically from only the port list and `AMS_Block` properties. For this purpose a graphical user interface (GUI) has been developed. In a first step, the design hierarchy is extracted from the schematic and the leaf cells to be modeled are chosen (figure 3.3). For each model then the appropriate `AMS_Block` implementation has to be decided (figure 3.4). The available blocks

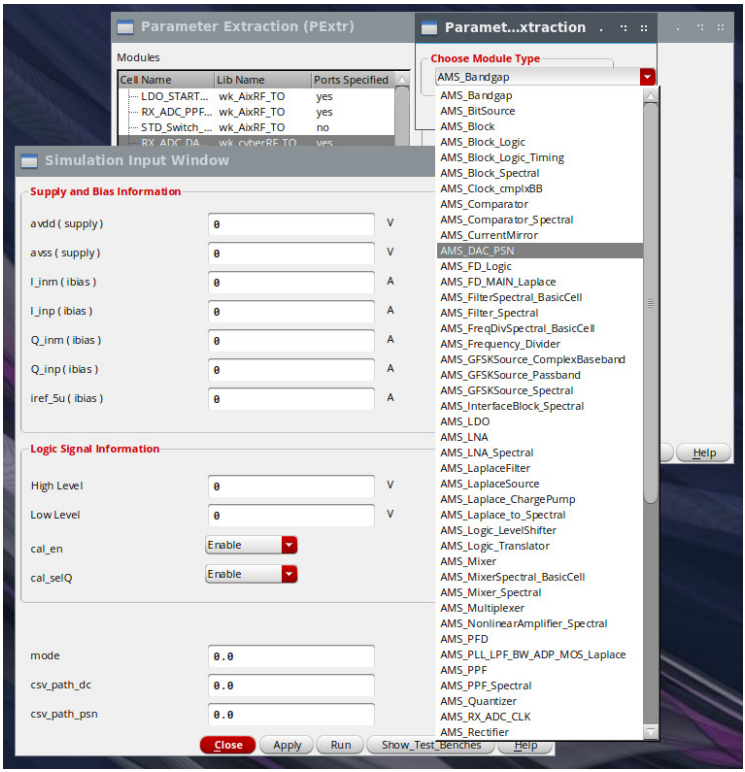


Figure 3.4: GUI to parameterize and generate SV model implementation.

and necessary parameters are extracted from the DPI description. Thus, they adapt automatically if new blocks or parameters are added.

3.2 Signal Descriptions for RF Systems

Signals hold information about the system's behavior. They are time varying functions that hold properties that need to be exchanged between different parts of the system. On the lowest abstraction level the signals directly hold the relevant electrical properties like voltage, current or charge. For higher-level modeling also non-electrical properties, e.g. frequency or phase, are possible.

To efficiently simulate large systems, choosing a suitable signal domain for high-level models can simplify the system description considerably. Looking for example on a PLL, the transfer function from the reference input to the PLL's output is highly nonlinear in the voltage domain but the frequency of the output is almost constant and the phase quasi linear. A description in the phase domain is thus beneficial to circumvent the description of the nonlinearity. Fourier and Laplace transforms can be used to change the signal from a time-dependent to a frequency or s-domain representation and vice-versa. Thus, they can be utilized to translate signals to a representation that is most suited for the application. E.g. while nonlinearities are beneficial to be described in time-domain, the frequency domain efficiently allows the description of linear systems.

In the simulation framework the `AMS_Signal` class serves as a common base to other domain specific implementations. Four different classes have been implemented for specific use-cases:

1. **Digital:** 4-state logic for configuration etc.
2. **Biasing:** electrical description for quasi static signals
3. **Spectral:** Fourier domain signal suitable for modulated RF signals
4. **Laplace Representation:** for aperiodic processes

Each signal class is accompanied by an `AMS_Port` class that holds the physical properties that affect the signal and an `AMS_Net` class to resolve conflicting drivers and multiple loads.

3.2.1 Time-Varying Digital Signal

To describe discrete-valued signals a 4-state logic data type similar to the existing data types in VHDL or SV is implemented. It can take the following four states:

- **'0'**: logical low level value
- **'1'**: logical high level value
- **'Z'**: high ohmic or floating state
- **'X'**: unknown or conflicting state

The propagation delay of asynchronous logic through its implementation blocks depends on the connected gate capacitance as well as the driving strength of the cell itself. For static timing analysis during synthesis timings in digital standard cells are characterized dependent on output load capacitance, supply voltage and input transition steepness. To allow a reuse of this characterization data to dynamically adapt propagation delays during the simulation, the Logic signal type is extended by additional variables holding the rise/fall time. The capacitance value is supplied by the port object while the block can provide the current supply voltage.

3.2.2 Electrical Biasing Signal

To offer non-binary control signals, e.g. biasing currents or reference voltages, a separate signal class has been implemented. It provides an electrical pass-band signal description. A simple piece-wise-constant implementation was chosen. An important feature of the net type is to resolve situations with multiple drivers or loads. For instance, a reference voltage can be connected to multiple sub-circuits or alternative sources for biasing currents during start-up and normal operation might be present. Thus, the ports are implemented as Norton equivalent providing an output current and an input resistance. The net then resolves all current sources and connected loads to a net voltage.

3.2.3 Baseband Equivalent and Spectral Signal

Pass-band signals as described in section 3.2.2 are a suitable signal description for slow-varying signals, but not ideal for high frequency communications. To correctly reconstruct RF signals in the GHz-range they need to be sampled with at least double the highest frequency component to fulfill the Nyquist theorem. Thus sampling time steps of only a couple picoseconds are needed. But adding some additional knowledge about the RF signal properties allows to find a more suitable representation. RF signals are typically band-pass signals which means all information is centered in a small band around a constant carrier frequency. Thus, all modulated RF signals can be described by (3.2) [Che09]. The fast oscillating carrier $\exp(j\omega t)$ has a constant frequency while the complex envelope $c(t)$ is only slowly varying.

$$x(t) = I(t) \cdot \cos(\omega t) + Q(t) \cdot \sin(\omega t) \tag{3.1}$$

$$= \Re\left\{c(t) \cdot \exp(j\omega t)\right\} \tag{3.2}$$

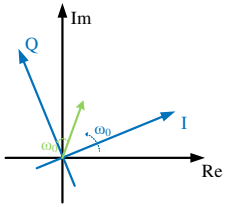


Figure 3.5: Visualization of the coordinate change by baseband equivalent signals.

The description can be visualized as a change in the coordinate system. While $x(t)$ is rotating with $\omega(t)$ in a fixed system, the baseband equivalent system itself is rotating with ω_0 as shown in figure 3.5. Thus, the remaining variation of $x(t)$ is relatively slow. To reconstruct $c(t)$ only a sampling rate of double the bandwidth of the original signal is needed, which is often lower by three to four orders of magnitude.

The baseband equivalent signal description offers a compact and fast representation for ideal modulated RF systems. It can efficiently describe the signal by only passing the center frequency ω and the two components of the complex envelope $I(t)$ and $Q(t)$. Considering nonlinearites, however, the representation is not ideal. A nonlinearity causes harmonic frequency components at multiples of the base frequency. If they were to be modulated as part of the envelope signal, it negates the advantages gained through the small bandwidth of the system. A similar scenario arises if two input tones need to be considered, as the intermodulation products also add further frequency components.

To keep the advantages of the baseband equivalent signal description, the signal needs to be extended to a sum of multiple baseband signals as shown in (3.3). The idea has been published in [Spe+18].

$$x(t) = \sum_k c_k(t) \cdot exp(j\omega_k t) \tag{3.3}$$

The number of frequency components depends on the number of independent fundamental tones and the number of harmonics per tone to be considered. $c_k(t)$ holds the time-varying amplitudes of the corresponding center frequency. They can be visualized as a multidimensional plane, where each fundamental tone adds a new dimension. The number of harmonics are give

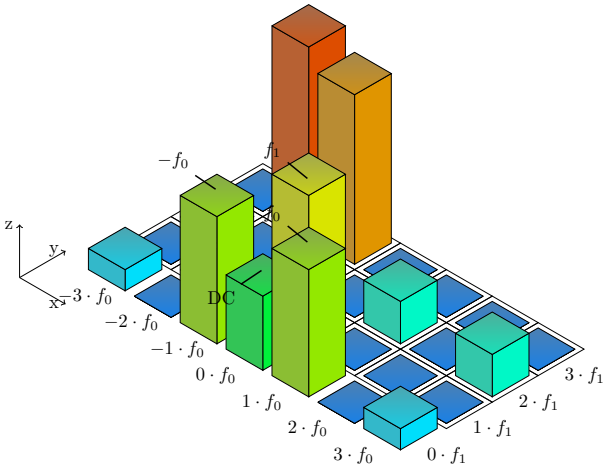


Figure 3.6: Two-dimensional spectral signal representation.

by the samples per dimension. c_k then fills the sampling points. While the fundamental tones and harmonics lie on the axis, the intermoduation products fill the other matrix positions. An example for two dimensions is shown in figure 3.6.

To be able to efficiently evaluate nonlinearites the signal needs to be transferred to the time-domain and solved there, as is the case in most HB algorithms as well [Kun97]. Interpreting the factors $c_k(t)$ as Fourier coefficients of the signal, a multidimensional inverse fast Fourier transform (IFFT) can be applied to transform the spectral signal to the time-domain.

3.2.4 Laplace Domain Based Signal

The frequency domain modeling approach described in section 3.2.3 assumes a quasi-periodic state of the modeled signal. Thus, it is not suited to analyze the transient behavior of the investigated systems. The piece-wise-constant signaling introduced in section 3.2.2 is a good real number approximation for transient signals. Using higher order polynomials for modeling, e.g. 1^{st} order or piece-wise-linear approximation, it is possible to increase the accuracy to some extent but in general a high sampling rate is needed to decrease the approximation error.

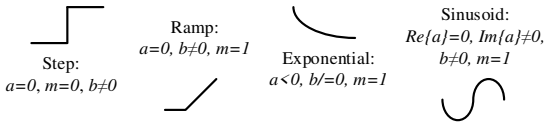


Figure 3.7: Example functions representable by the XREAL group introduced in [Jan+12].

A different concept was proposed by Jang, Park, Lee, and Kim in [Jan+12]. They assume that all relevant signals can be described by a piece-wise superposition from a group of base functions that in Laplace domain are represented as rational function. The group includes e.g. step function, polynomials, sinusoidal and exponential functions as shown in figure 3.7. A compact and parameterized version in the s-domain is shown in (3.4). The time-domain equivalent description is shown in (3.5).

$$X(s) = \sum_k \frac{b_k}{(s - a_k)^{m_k}} \qquad \text{occuring at} \quad t_{0,k}\,, \tag{3.4}$$

$$x(t) = \sum_k \frac{b_k}{(m_k - 1)!} \cdot (t - t_{0,k})^{m_k} \cdot \exp(a_k \cdot (t - t_{0,k})) \cdot u(t - t_{0,k}) \tag{3.5}$$

The C++ class `AMS_Signal_LaplaceRepresentation` implements the XREAL type as std::vector as shown in listing 3.2. This allows an efficient implementation of common base operations on the signal. For example the sum of two signals is implemented by merging both component vectors while the multiplication with a constant is implemented by scaling the coefficient b of all components.

3.3 Modeling Power Supply

The previous sections have focused on modeling techniques for the main signal path. But since most bugs introduced in integrated circuits are control, configuration, interface, interconnect and integration (C2I3) errors [Shi10], it is important to create pin-equivalent models including the supply connections.

,

```
#pragma once

#include "AMS_Signal.h"
#include <vector>
#include <complex>

class AMS_Signal_LaplaceRepresentation: public AMS_Signal {
public:
  /** \brief struct definition for a base function representation in Laplace domain
   *
   * All base functions are of the form \f$ b \cdot (t - t_0)^{m-1} \cdot \exp{(-a \cdot (t-t_0))} \cdot
↪  \mathrm{u}(t-t_0) + initValue \f$
   * or in Laplace domain \f$ \frac{b}{(s + a)^m} \f$.
   */
  typedef struct {
    complex<double>  a,    ///< pol of the Laplace representation
                     b;    ///< amplitude of the Laplace representation
    double m;              ///< exponent m of Laplace representation
    double t_0;            ///< last update time
  } parameter_set;

protected:
  /// vector holding all base function contributions to the signal
  vector<parameter_set> signal_representation;

public:
  AMS_Signal_LaplaceRepresentation();
  virtual ~AMS_Signal_LaplaceRepresentation();

  void add_parameter_set(const complex<double> a, const complex<double> b, const double m, const double
↪  t_0);
  void add_parameter_set(const parameter_set& set);
  void clear_parameters();
  void manipulate_parameter_set(int set_number, complex<double> amplitude, double sv_time);

  void append(const AMS_Signal_LaplaceRepresentation* const signal);

  complex<double> calculate_value(double sv_time) const;
  complex<double> get_settle_value();

  complex<double> calculate_nth_derivative(double sv_time, int n) const;
  void signal_multiplication_by_constant(complex<double> factor);

  double return_crossing_time(double crossing_value, double tolerance, double sv_time);
  void get_resampled_signal(const AMS_Signal_LaplaceRepresentation* const input_signal, double sv_time);
};
```

Listing 3.2: `AMS_Signal_LaplaceRepresentation` header file with class definition.

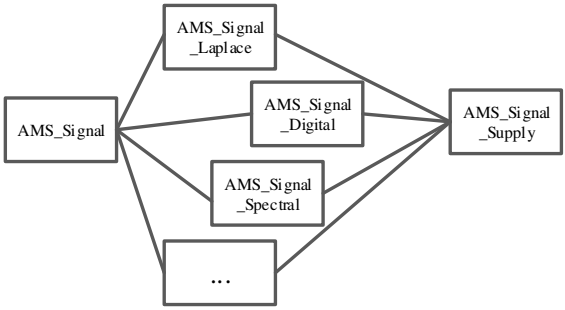


Figure 3.8: Inheritance diagram for supply signal.

Otherwise subtle errors on the boarders of two supply domains cannot be detected. Furthermore, cross-coupling and PSN is gaining impact in small technology nodes and can affect the overall circuit performance.

3.3.1 Power Supply Signal

Compared to other connections in an integrated circuit the supply takes a special position as it can serve different purposes. Mainly, of course, it has to supply power to the block. Here the transient behavior of the supply voltage and currents are of interest, especially during turn-on and -off of separate subsystems. But the stable supply can also be reused to provide an always-high – or in the case of the ground lines an always-low – logic level. It can be reused as a biasing input. Lastly it also provides an unwanted coupling path between different subcircuit that needs to be analyzed to ensure proper operation of the complete system. For each of the described purposes a different signal description from the previous section 3.2 is best suited. The transient settling is efficiently described by the Laplace representation, cross-coupling possibly using the spectral signals, while for tie-high (TIEH) and tie-low (TIEL) functionalities logic signals are needed. C++ inheritance allows for multiple parent classes. Thus, the AMS_Supply class is created as a child class of all other AMS_Signal classes. This enables the underlying blocks to choose, which signal representation to query. It, therefore, allows to choose the most efficient implementation on a block-by-block basis.

Having a separate signal class for the supply allows to add further properties to the signal. E.g. by adding variables to hold the domain and voltage level,

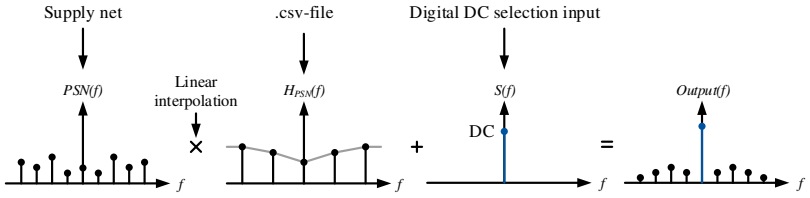


Figure 3.9: Implementation of transfer function for supply noise in [Mei+19b].

domain crossing errors can be detected. The supply generating circuit provides a unique domain name and its nominal voltage level. On the receiving side, the block can now be checked if they are designed for this voltage level and connected correctly. Furthermore, by passing this information to the output ports, missing level shifters can easily be checked for at simulation start. Mismatches between the input signal levels and supply can be detected and should only be accepted in the level shifter blocks.

3.3.2 Power Supply Noise

Possible sources of power supply induced noise and its growing importance in submicrometer designs have been described in section 2.1.6.1. Approaches to consider the effects of PSN on block-level and SSN in particular can be categorized in two groups:

1. transfer function based models
2. parameter variation approaches

Transfer function based approaches treat the supply as an additional input signal. An example, where this method naturally applies, is the modeling in linear drop-out regulators. Since the power supply rejection ratio (PSRR) is already a characteristic performance parameter it can easily be considered in the model behavior. The same approach can also be transferred to other analog circuits. Results of this approach used for modeling a ΔΣ-ADC have already been published in [Mei+19b]. The noisy supply signal is convoluted with the power supply transfer function. The result is then superpositioned with the signal from the main path as shown in figure 3.9.

In the simplest case a linear relation is assumed, but the idea can also be extended to cover nonlinear circuit behavior. Cascading the linear time invariant

(LTI) transfer functions with nonlinear transfer functions allows to describe a wide range of nonlinear system behavior. These Wiener-Hammerstein models are well suited to describe analog circuit behavior.

Volterra series are an alternative to describe nonlinear circuits, which can be thought of as Taylor series with memory [PC03]. In [Gib+16] they are modified such that they account for dynamic supply deviations. A drawback of this approach are the large number of parameters that need to be extracted in order to describe the system.

In addition to the block oriented models, such as LTI and Wiener-Hammerstein, as well as the Volterra series, there are several black-box approaches. A recent example is the popular use of neural networks [Xu+02; YCS19; Yu+19]. Main disadvantage of this approach is that they give little insight into the behavior of the circuit. Therefore, special care has to be taken to prove the trained models cover the circuit behavior correctly in all corner cases.

The second group of methods to handle PSN on block level analyses the influence of a varying supply voltage on the main signal path. Therefore, the signal transfer function needs to be modeled in a parameterized way. A varying supply voltage then affects these parameters. E.g. power supply induced jitter is commonly modeled this way by adjusting the propagation delay according to the deviation from a nominal supply level [WM13]. Assuming only a small voltage deviation, a linear approximation as used in the α-delay model is often sufficient:

$$\alpha = \frac{\Delta t_d / t_d}{\Delta V_{dd} / V_{dd}} \,. \tag{3.6}$$

As shown in figure 3.10 the delay variation is measured at nominal voltage and with an offset applied to extract the value of α.

$$\Delta t_d = \alpha \frac{\Delta V_{dd}}{V_{dd}} t_d \,. \tag{3.7}$$

In [WM13] this technique is used to model the inverter delay variations to analyze the jitter in a ring-oscillator.

A similar approach can be used to vary other performance parameters, as has been published in [Mei+19a]. Similarly to what has been described in section 3.2.4 the accuracy can be increased by using piece-wise-linear or higher order approximations. Alternatively, if the assumption of a quasi-static supply does not hold, the use of sensitivity functions [Ier+08; KLD05] is a possibility. The sensitivity function of the modeled parameter is extracted by applying a unit impulse at different times relative to the input event.

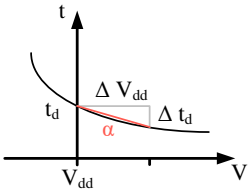


Figure 3.10: Delay calculation of alpha factor.

Convolution of this transfer function with an arbitrary noise source during simulation, allows considering previous supply voltage levels while evaluating the effect of the supply noise variation on the model behavior.

Both groups of methods presented here serve their purpose in finding a efficient modeling solution for RF systems. While the transfer function based approaches are intuitively well suited to model cross-coupling, they are efficiently usable in receiver chains, where the influence of noise on the signal filtering is the main concern. Parameter variation solutions on the other hand are well suited for generating concise and accurate signal representations and are thus preferably used on the transmitting circuits.

CHAPTER 4

RF SYSTEMS UNDER TEST

The modeling methods and simulation environment introduced in Chapter 3 are used to verify several circuits developed at the IAS. First an integrated wireless multiband transceiver is investigated. While the framework has been applied to verify the pre-tapeout design of the complete system, they are applied in this thesis to investigate the supply noise influence on the integrated ΔΣ-ADC. The main focus lies on a second circuit, an all-digital phase-locked loop. It was developed for low phase noise with the introduced methods also available during early design stages to optimize the circuit for robust operation.

4.1 Multiband Transceiver

An integrated triple-band wireless transceiver for robust and reliable communication has been developed at the IAS. The integrated circuit (IC) is designed in a 130 nm CMOS technology and supports the 433 MHz, 868 MHz and 2.4 GHz industrial, scientific and medical (ISM) bands. It consists of RF transmitter and receiver frontends, a fractional analog PLL as local RF oscillator and a ΔΣ-ADC to pass the received signals to the digital post-processing. Furthermore, a power management unit, a clock management block with quartz oscillator and a low power wake-up receiver to monitor the bands for signals and consecutive start-up of the system are present. The on-chip digital processing consists of units for modulation and demodulation for different wireless standards, a static random-access memory (SRAM) FIFO to store data to be send and received as well as an serial peripheral interface (SPI) to communicate with the outside.

4.1.1 ΔΣ-ADC

The receiver architecture implemented on the transceiver is a low intermediate frequency receiver that mixes the arriving RF signal to an intermediate frequency (IF) band at 1 MHz. It is followed by a complex-valued ΔΣ-ADC

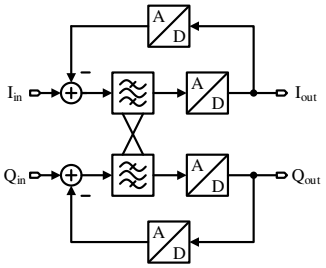
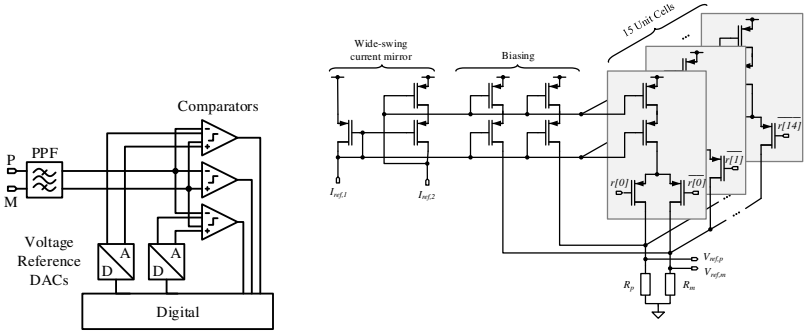


Figure 4.1: System overview of ΔΣ-ADC.



(a) Multi-bit tracking ADC.

(b) Reference Voltage DAC.

Figure 4.2: Analog-to-Digital Converter.

with a sample rate of 96 MHz. A system overview of the analyzed circuit is given in figure 4.1. It is presented in depth in [Saa+16]. The design is fully differential and consists of a poly-phase quadrature loop filter, a multi-bit tracking ADC and a current cell based DAC in the feedback path.

The internal ADC structure is shown in figure 4.2a. In single-bit mode only the center comparator is used, while the 2-bit mode uses all three comparators as flash ADC. For the four-bit mode a tracking ADC concept is used instead, utilizing the outer comparators. Due to the high oversampling rate, it is guaranteed that the input signal does not change more than 1 bit between two samples. Thus, only two comparators are used to detect this possible deviation. The reference voltage levels are then updated accordingly for the next sample. They are generated by the DAC presented in figure 4.2b. A current mirror provides 15 switchable output currents that can be delivered

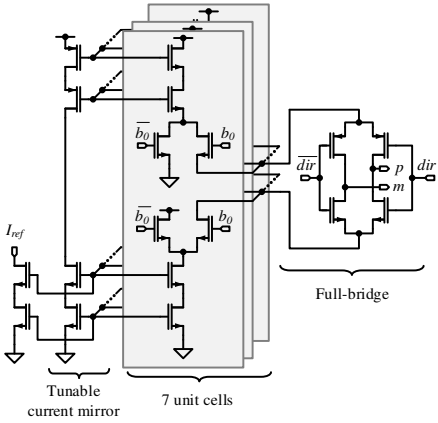


Figure 4.3: Feedback DAC.

to either of two shunt resistors, where the resulting current difference is translated to a reference voltage level. The comparators are implemented by a differential amplifier with current mirror load and a successive latch for sampling.

The DACs in the feedback path are realized as current cell DACs as presented in figure 4.3. Each of the unit currents can either be fed in an output full bridge, that decides the sign of the current, or drained directly into the supply nodes. Additionally, the unit current is tunable to adjust offsets due to process variation.

The filter used in the ADC is a third order quadrature band-pass filter with an intermediate frequency of 1 MHz and bandwidth of 1 MHz [Ata+13]. The mixer used to shift the received signal from the RF to IF generally produces an image component at the negative intermediate frequency. This image can not be filtered by a real filter, due to its symmetric transfer function around zero, but a quadrature filter can [ST04].

Table 4.1: Summarized specifications of the ADPLL.

Symbol	Parameter	Min	Typ.	Max	Unit
f_{ref}	Reference frequency	40	55	100	MHz
f_{DCO}	Oscillator frequency	9	10	11	GHz
Δf_{ch}	Channel spacing	10	-	20	kHz
f_{BW}	Filter bandwidth	-	100	-	kHz
$\lvert\mathcal{L}_{\phi}(\Delta f)\rvert$	$\Delta f < 100\,\text{kHz}$	-	-	-100	dBc/Hz
	$\Delta f > 100\,\text{MHz}$	-	-	-155	dBc/Hz

4.2 All-Digital PLL

4.2.1 System Design

The ADPLL designed in this thesis is developed for direct frequency generation in the GHz range with a low phase noise requirement. The main design parameters are summarized in table 4.1.

The channel spacing with values lower than the reference frequency calls for a fractional-N PLL design. The targeted DCO operating range spans 2 GHz while the channel spacing in the kHz range is 5 orders of magnitude lower. Since a linear relation between capacitance and frequency of the LC resonator are only valid within a certain range, differently coarse operating modes and gear-shifting is implemented. This also allows a trade-off between PLL settling time and output noise.

The loop filter design has a major influence on the noise performance of the system, as it shapes the different noise contributions. The noise can roughly be divided in two groups, the noise inside the filter bandwidth and the contributions with higher frequencies. The out-of-band noise is mostly defined by the noise characteristic of the DCO, as it is high-pass filtered by the loop. For the in-band noise contributions the TDC quantization and the reference frequency are the major impact factors. Operating the circuit at the lower end of the specified reference frequency range increases the noise contributions due to jitter and quantization in the design. It also covers the lower corner for frequency acquisition timings.

A further noise source in fractional-N PLLs are spurious tones. They are caused by periodic operations in the control of the loop and especially severe

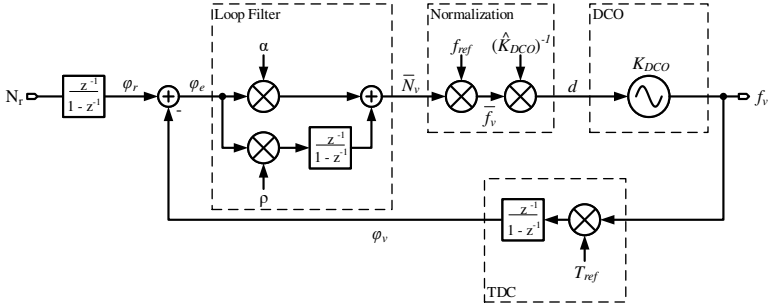


Figure 4.4: Z-Domain model of a fractional-N ADPLL.

for tuning settings close to integer values. Special care has to be taken to reduce the noise for these setting, which will be discussed in section 4.2.3.

To gain a better understanding of the phase noise contributions and filter characteristics, the ADPLL system will first be simulated and analyzed in the phase domain. These findings are then used in the implementation.

4.2.1.1 Phase-Domain Analysis

As a clocked, time-discrete system, the z-domain is well suited for modeling an ADPLL [MV07]. The blocks from the system overview introduced in figure 2.9 can be transformed to equivalent z-domain descriptions. The resulting system model is shown in figure 4.4. All components are linearized and a stead-state operation is assumed, meaning

$$f_v \rightarrow f_v^{ss} = N_r \cdot f_{ref} \,, \tag{4.1}$$

where N_r denotes the ratio between reference frequency and DCO output frequency similar to FCW in (2.14).

The DCO acts as a digital-to-frequency converter. Its transfer function is given by:

$$\frac{f_v(z)}{d(z)} = K_{DCO} \,. \tag{4.2}$$

The gain K_{DCO} converts the digital output tuning word d to a frequency f_v. Its unit is thus Hz/LSB. Since a steady-state is assumed, the expected frequency deviations are small so that a static linear relation is sufficient. This further allows to find an exact estimate $\hat{K}_{DCO} \approx K_{DCO}$ to normalize the DCO tuning to a unit gain as in (4.3):

$$\frac{f_v(z)}{\overline{f}_v(z)} = \frac{K_{DCO}}{\hat{K}_{DCO}} \approx 1 \,. \tag{4.3}$$

The control signals are processed in the phase-domain. Thus, a frequency-to-phase conversion is needed. It is implemented by the TDC which integrates the frequency to the instantaneous output phase. In the time-discrete model, clocked accumulation replaces the integration. The variable DCO output frequency is integrated over one reference period. The sampling process is modeled by an additional feedback delay:

$$\frac{\varphi(z)}{f(z)} = T_{ref} \cdot \frac{z^{-1}}{1 - z^{-1}} \,. \tag{4.4}$$

Similarly, the reference phase is computed by accumulation of the frequency ratio N_r.

The loop filter is modeled as a PI-element with the two coefficients α and ρ. The filter transfer function is

$$\frac{\overline{N}_v(z)}{\varphi_e(z)} = \alpha + \rho \cdot \frac{z^{-1}}{1 - z^{-1}} \,. \tag{4.5}$$

The closed-loop transfer function in the phase domain is thus given by (4.6). A similar relation is derived for the frequency signals (4.7):

$$\Phi_{cl}(z) \equiv \frac{\varphi_v(z)}{\varphi_r(z)} = \frac{\alpha(z-1) + \rho}{(z-1)^2 + \alpha(z-1) + \rho} \,, \tag{4.6}$$

$$H_{cl}(z) \equiv \frac{f_v(z)}{N_r(z)} = f_{ref} \cdot \Phi_{cl}(z) \,. \tag{4.7}$$

Depending on the choice for α and ρ the PLL type can be altered. Different PLL types are discerned by the number of poles at the origin of the loop transfer function. A first pole is inherently introduced by the frequency-to-phase conversion in the TDC block. If $\rho = 0$ is chosen, it is the only pole. The PI-element is effectively reduced to a proportional control element with

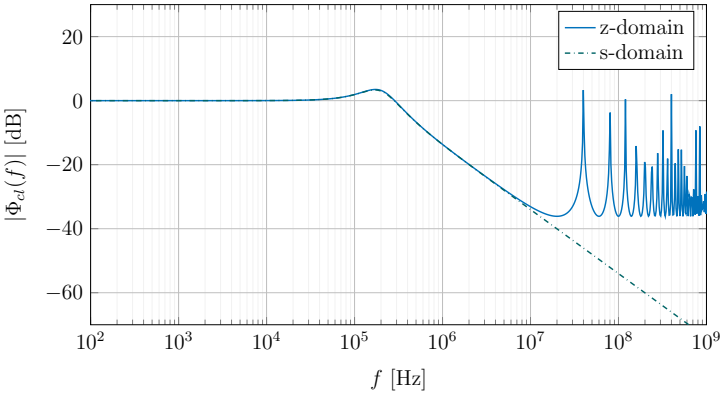


Figure 4.5: Comparison of z-domain phase-domain transfer function and s-domain approximation.

factor α. The PLL is then of type-I. For $\rho \neq 0$, the integral term of the loop filter adds a second pole resulting in a type-II PLL.

Modeling the PLL in the z-domain allows for an easy system description, but comes with a major drawback. The z-domain implies a fixed sampling rate, which in this case is the reference frequency f_{ref}. According to the Nyquist-criterion this causes aliasing in the spectrum around multiples of $f_{ref}/2$, as can be seen in figure 4.5. The ADPLL, especially the DCO and TDC, however, operate in different clock domains. To evaluate the noise impact on the system, all noise contributions need to be transformed to equivalent noise source in the reference clock domain.

The Laplace-domain approximation allows for an equivalent description of the system in a continuous-time domain. It can be generally applied if the loop-bandwidth of the PLL does not exceed 1/10 of f_{ref} [Kuz+19]. In this case the following approximation can be applied:

$$z = e^{s \cdot T_{ref}} \approx 1 + sT_{ref} \,. \tag{4.8}$$

Substituting (4.8) in the closed-loop transfer functions (4.6) and (4.7) yields:

$$\Phi_{cl}(s) = \frac{\varphi_v(s)}{\varphi_r(s)} = \frac{\alpha s f_{ref} + \rho f_{ref}^2}{s^2 + s\alpha f_{ref} + \rho f_{ref}^2} \text{ and} \tag{4.9}$$

$$H_{cl}(s) = \frac{f_v(s)}{N_r(s)} = f_{ref} \cdot \Phi_{cl}(s) \; . \tag{4.10}$$

Figure 4.5 compares the transfer functions for z- and s-domain models. As can be seen the behavior is identical for $f < 1/10 f_{ref}$.

4.2.1.2 Phase Noise Analysis

Different noise sources contribute to the output noise spectrum. Depending on the location where noise enters the system, its contribution is shaped differently by the loop filter characteristics. For analysis, the noise power density $\mathcal{L}_\eta(\Delta\omega)$ of each source needs to be derived. E.g. it can be computed from root-mean-square (RMS) time domain jitter σ_{t_j} with

$$\mathcal{L}_\eta = \frac{(2\pi\sigma_{t_j})^2}{f_{ref}} \; . \tag{4.11}$$

Detailed derivations can be found in [Sky12]. The resulting spectral noise $\mathcal{L}_\eta(\Delta\omega)$ is shaped by multiplication with the squared absolute value of the noise transfer function $H_\eta(\Delta\omega)$:

$$\mathcal{L}_{\eta,out}(\Delta\omega) = \mathcal{L}_\eta(\Delta\omega) \cdot \left|H_\eta(\Delta\omega)\right|^2 \; . \tag{4.12}$$

The noise transfer function $H_\eta(\Delta\omega)$ can be derived directly from the phase domain model, when considering where the noise enters the system.

Three major noise sources dominate the performance of the design. Reference phase noise enters the system at the reference frequency input. It exhibits the phase noise characteristics described in section 2.1.6.2. The same holds for the noise introduced by the DCO. Additionally, the DCO's tuning operation at discrete levels introduces a quantization error, which can be treated as an additional noise source. The quantization introduced by the limited time resolution of the TDC is the third major noise source. These sources are uncorrelated and are treated as additive contributions to the overall output noise spectrum.

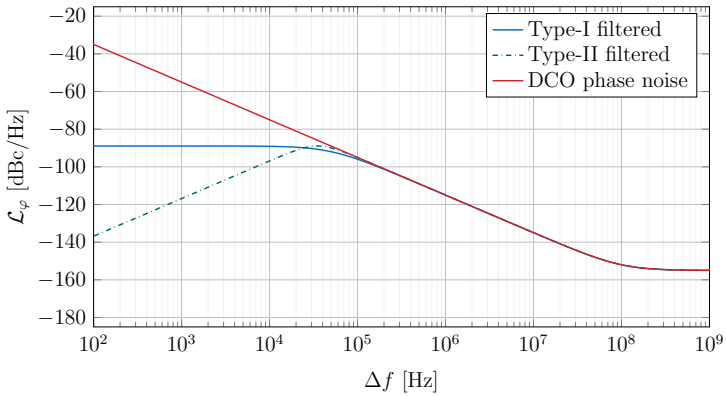


Figure 4.6: DCO phase noise in type-I and type-II PLL.

Reference Phase Noise Reference phase noise describes the external noise that enters the PLL together with its stable reference frequency. Since this frequency is often supplied by a crystal oscillator, the oscillator phase noise as introduced in section 2.1.6.2 is assumed. For simplicity only the broadband (f^0) and up-converted (f^{-2}) thermal noise are modeled.

DCO Phase Noise The DCO phase noise is modeled the same way as the reference noise. However, since the DCO noise enters the loop at the frequency output, its loop transfer function is different. From the s-domain model, the phase noise transfer function of the DCO can be identified as

$$\Phi_{\eta_{DCO}} = \frac{\varphi_v}{\eta_{DCO}} = \frac{s^2}{s^2 + \alpha f_{ref} s + \rho f_{ref}^2} \,. \tag{4.13}$$

Due to the high pass characteristic, the DCO noise dominates the ADPLL phase noise for the out-of-band region. To adhere to the specifications from table 4.1, the broadband noise must be smaller than −155 dBc and the SSD at offset frequency Δf of 1 MHz needs to meet (4.14):

$$\mathcal{L}_{DCO}(\Delta f)|_{\Delta f = 1\,\mathrm{MHz}} = -115\,\mathrm{dBc} \,. \tag{4.14}$$

The superposition of both noise contributions is shown in figure 4.6. The resulting phase noise at the PLL output for a type-I and type-II loop filter configuration is also depicted. As can be seen, the type-I PLL only provides

20 dB/dec attenuation in the stop band and thus, only flattens the phase noise to a constant level. Considering the up-converted flicker noise (f^3) would result in a continuing 10dB/dec raise. The type-II PLL, however, provides 40dB/dec attenuation due to the second zero in the origin. Consequently, even the higher order phase noise is attenuated at low frequencies.

DCO Quantization Noise In addition to the oscillator noise, the discrete control of the DCO adds another noise source: quantization. The frequency tuning is applied in discrete capacitor steps which control the output frequency. The resolution Δf_{res} is limited by the least significant bit (LSB) capacitance $DeltaC$ and can be calculated according to (2.27). The relevant step size utilized in this work is $\Delta f_{res} = 80\,\text{kHz}$. The frequency quantization error Δf_n is assumed to be uniformly distributed and thus follows the AWGN characteristics [SB06] with variance $\sigma^2_{\Delta f_n}$:

$$\sigma^2_{\Delta f_n} = \frac{(\Delta f_{res})^2}{12} \,. \tag{4.15}$$

In the frequency domain, the quantization noise SSD $\mathcal{L}(\Delta\omega)$ is defined by the integration of flat spectral frequency noise from DC to the Nyquist frequency $0.5 \cdot f_{ref}$:

$$\mathcal{L}_{DCO,qu}(\Delta f) = \frac{(\Delta f_{res})^2}{12 \cdot f_{ref}} \,. \tag{4.16}$$

Since the DCO is updated only once every reference period the noise needs to be folded with the filtering of a zero-order hold operation. It is then further filtered by the same transfer function as the DCO phase noise resulting in figure 4.7.

The impact of quantization can be improved by time averaging of fast dithered unit capacitor cells. A straightforward dithering scheme switches a single capacitor at the rate $f_{dth} > f_{ref}$ and reduces the SSD of the DCO quantization to

$$\mathcal{L}_{DCO,dth}(\Delta f) = \frac{(\Delta f_{res})^2}{12 \cdot f_{dth}} \,. \tag{4.17}$$

According to Staszewski and Balsara [SB06], low-frequency phase noise contributions are still too high for mobile systems and thus, noise shaping e.g. by a $\Sigma\Delta$-modulator is necessary. To do so, Δf_{LSB} is artificially increased by

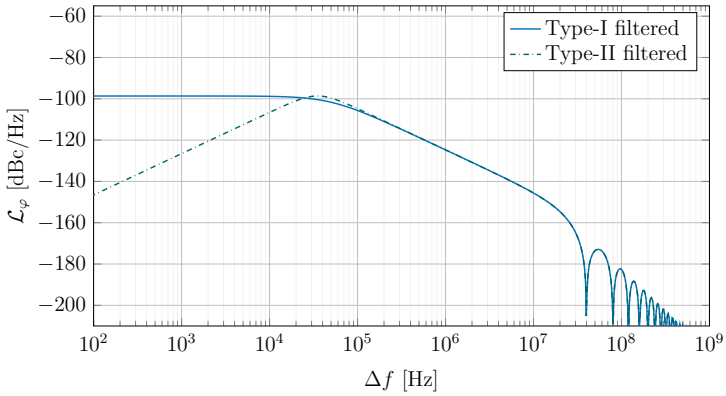


Figure 4.7: DCO quantization error shaped by type-I and type-II PLL.

W_F additional fractional bits so that the overall resolution is

$$\Delta f_{res,\Sigma\Delta} = \frac{\Delta f_{LSB}}{2^{W_F}} \,. \tag{4.18}$$

The noise transfer function of a n-th order ΣΔ-modulator in z-domain is given by

$$N(s) = \frac{N_{qu,\Sigma\Delta}(s)}{\eta_{qu,\Sigma\Delta}(s)} = (1 - z^{-1})^n \,, \tag{4.19}$$

which yields (4.20) for the SSD quantization phase noise following s-domain approximation:

$$\mathcal{L}_{qu,\Sigma\Delta}(\Delta f) = \frac{1}{12} \left(\frac{\Delta f_{res,\Sigma\Delta}}{\Delta f} \right)^2 \frac{1}{f_{dth}} \left(2 \cdot \sin\left(\frac{\pi \Delta f}{f_{dth}} \right) \right)^{2n} \,. \tag{4.20}$$

The quantized update noise signal with reduced quantization error in (4.17) as well as the high-pass filtered quantization error from the ΣΔ-modulator in (4.20) are shaped by the DCO transfer function. In figure 4.8, the resulting phase noise contribution is depicted in comparison to the non-dithered quantization error for a third order modulator with three bits fractional resolution. The overall quantization error is attenuated by approximately 30 dB at the cost of increased noise far off the carrier. It should be noted that the out-of-band noise floor is roughly −160 dBc/Hz covering the shifted high-frequency phase noise of the modulator.

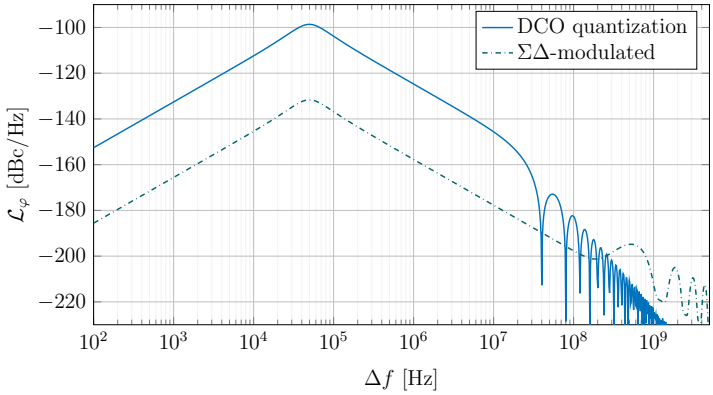


Figure 4.8: DCO quantization error and ΣΔ-modulated quantization.

TDC Quantization Noise Similar to the discrete capacitor values used in the DCO, the TDC resolution limits introduce additional phase noise. This noise is low-pass filtered by:

$$\Phi_{\eta_{TDC}} = \frac{\varphi_v}{\eta_{TDC}} = \frac{\alpha f_{ref} s + \rho f_{ref}^2}{s^2 + \alpha f_{ref} s + \rho f_{ref}^2} \,. \tag{4.21}$$

Again a uniformly distributed quantization noise with variance σ_{TDC}^2 is assumed. Its SSD $\mathcal{L}_{TDC,qu}$ is given by:

$$\mathcal{L}_{TDC,qu} = \frac{(2\pi \cdot \Delta t_{LSB} \cdot f_v)^2}{12 \cdot f_{ref}} \,. \tag{4.22}$$

Figure 4.9 shows the resulting phase noise output for $\Delta t_{LSB} = 5\,\text{ps}$. The plot shows that TDC noise is attenuated with 20 dB/dec in the low-pass stop-band for type-I and type-II PLLs. To reach the required in-band phase noise level from table 4.1 a TDC resolution limit $\Delta t_{LSB} < 3\,\text{ps}$ is needed.

4.2.2 Loop Filter Design

To attain concurrent demands in PLL phase noise, spur reduction and settling speed, an appropriate filter design is crucial. A type-I PLL allows for fast step-responses but can only guarantee a limited dynamic range due to its residual phase error. A type-II PLL does not have this limitation, since the

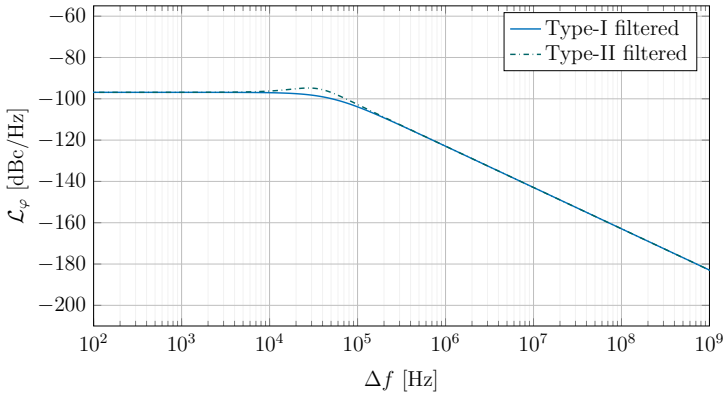


Figure 4.9: TDC quantization error shaped by type-I and type-II PLL.

pahse error diminishes during lock phase. Further it provides a steeper filter roll-off, which is favorable with regard to phase noise characteristics. E. g. a PLL with −40 dB/dec attenuation is able to suppress up-converted flicker noise. For efficient spur suppression even higher damping is amendable. This is easily implemented by cascading the main PI-Loop filter with additional inifinite impulse response (IIR) filter stages. The digital nature of the implementation allows to reconfigure the loop filter during operation. This enables the adoption of gear-shifting, where the loop filter is adjusted from a proportional filter during PVT and acquisition tuning to a higher order filter during tracking and lock.

4.2.3 Spur-Reduction Mechanisms

In an ADPLL spurs are introduced through regularly repeating patterns in the phase error and thus a modulation of the DCO. In a fractional PLL the fractional FCW and the limited TDC resolution will cause a slowly varying, periodic phase error pattern, giving rise to fractional spurs. The limited TDC resolution in combination with the targeted fine tuning step size results in multiple reference clock cycles yielding the same TDC measurement result while the DCO fractional phase slowly increases. This causes a periodic saw tooth shaped phase error seen in figure 4.10.¨ Especially for small fractional parts, the periodicity of this process is rather long causing spurs in the pass-band of the PLL. For a spur free operation the expected phase error as given

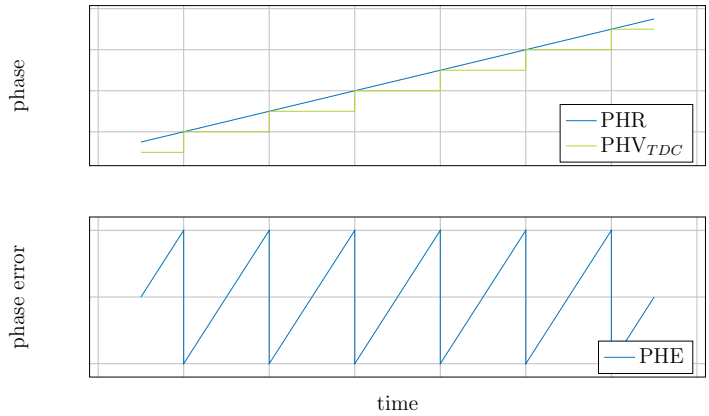


Figure 4.10: Ideal Phase error for FCW close to integer.

in (4.23) needs to be 0:

$$E\{phe\} = E\{phv - phr\} = 0\,. \tag{4.23}$$

Two main approaches are proposed in literature to mitigate frational spurs:

- predict and compensate the periodic phase error
- add a larger pseudo-random noise source to overshadow the effect and compensate the known noise source

Opteynde proposes to mitigate the spurs by quantifying the reference phase to the same step size as the TDC [Opt12]. Quantifying phr to the mid-level between two TDC quantization levels n and $n+1$ forces the PLL into a bang-bang mode as shown in figure 4.11. This, however, does not mitigate the spurs completely but only shifts them to a different, higher frequency depending on the chosen TDC step size as the probability of measuring a n or $n+1$ is not uniform over the whole step. Thus, the expected phase error is not zero. To achieve a linear output phase, the reference phase, needs to follow the error probability function of the jitter model for metastability [Kle90], instead of being compensated with a saw tooth:

$$p(1|0) = \frac{1}{2}\left(1 + erf\left(\frac{x-\mu}{\sqrt{2}\cdot\sigma}\right)\right)\,. \tag{4.24}$$

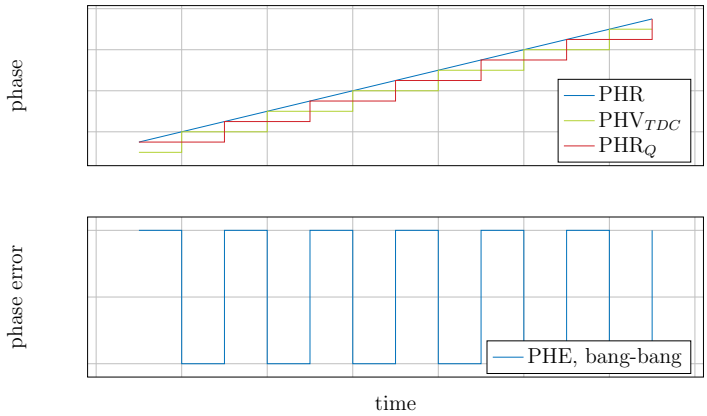


Figure 4.11: Phase error for FCW close to integer compensated with quantized *phr*.

Pre-distorting the reference phase *phr* to follow this probability results in an expected phase error $E\{phe\} = 0$.

$$0 = E\{phe\} = E\{phv - phr\} = E\{phv\} - phr \tag{4.25}$$

$$phr = E\{phv\} = n \cdot p(n) + (n+1) \cdot p(n+1) \tag{4.26}$$

$$= n \cdot (p(0|0) + p(1|0)) + 1 \cdot p(1|0) \tag{4.27}$$

$$= n + \frac{1}{2}\left(1 + erf\left(\frac{phr_{ideal} - phr_{quantized}}{\sqrt{2} \cdot \sigma_{TDC}}\right)\right) \tag{4.28}$$

As can be seen from figure 4.12, the reference phase is equal to the TDC measured value during the stable areas of TDC evaluation and the phase error diminishes to 0 unless the TDC detects a phase difference by at least 1 TDC step. This error is then passed along unweighted. If the phase discrepancy detection happens close to an expected TDC step, the phase error weight is reduced by the probability of the sampling event. To accurately calculate the probability function, the standard deviation of the jitter at the TDC input is needed, which can be estimated using a Goertzel filter and a least-mean-square (LMS) algorithm as shown in [PB11], where it is used to control the digital-to-time converter (DTC) setting in a DTC based ADPLL.

Alternatively the error probability function can be deduced from monitoring the phase error *phe*. Since the periodicity of the signal is known from the fractional part of the FCW and the TDC resolution, the phase error can be

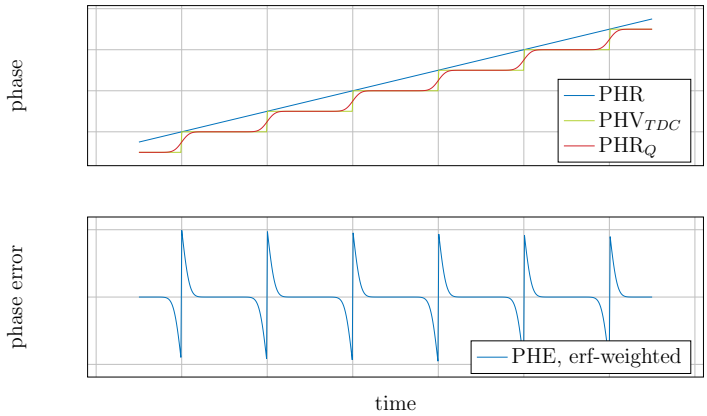


Figure 4.12: Phase error for FCW close to integer weighted according to (4.28).

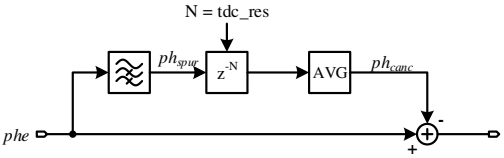


Figure 4.13: Schematic for extracting and compensating periodic phase error [HC16].

averaged over multiple periods to extract the needed transfer function. Ho and Chen propose a structure seen in figure 4.13 consisting of a high-pass filter and a successive IIR filter for averaging [HC16].

An example of the second approach to mitigate fractional spurs is proposed by Temporiti, Weltin-Wu, Baldi, Cusmai, and Svelto. A schematic of the operating principle is show in figure 4.14. By dithering the *FREF* signal at the TDC input, additional noise is added to the system. The added delay needs to scramble the *FREF* signal enough to ensure a white quantization noise within a TDC step size. Thus, ideally the delay steps should be prime with the TDC resolution as to avoid spurs at the common frequency components. Furthermore, if the delay range is chosen larger than a DCO period, the dithering can further be used to average TDC non-linearity.

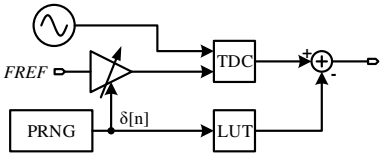


Figure 4.14: TDC with added pseudorandom fref dithering.

The added delay for each f_{ref} period is generated digitally using a pseudo-random linear feedback shift register (LFSR). Since the delay is known, it is then subtracted again after digitization by the TDC. The delay of the dithering line will vary with process supply voltage and temperature. Thus, to compensate with the right value the delay needs to be measured online. Temporiti, Weltin-Wu, Baldi, Cusmai, and Svelto propose to use the locked PLL and the TDC to measure the added delay [Tem+10]. In locked state, the fractional part of *phv* is expected to change by the fractional part of FCWs between two successive periods. By alternating the added dither between 0 and i and looking at the TDC output, the added delay time can be measured. Averaging over multiple measurements is needed to increase the accuracy.

4.2.4 System Implementation

The general signal flow and design implementation follow the recommendations given in [SB06]. It relies on the system layout presented in figure 4.15. The performance-critical components DCO and TDC are full-custom analog circuits, while the rest of the design is implemented as register transfer level (RTL) code using SystemVerilog. A more detailed view in the inner implementation of the analog components, as needed for modeling and simulation with the digital circuitry, will be given in chapter 5.2 together with the model development.

In the digital circuit fixed-point numbers are used to represent fractional values with a limited amount of digits. In contrast to floating point arithmetic, the decimal point is fixed by convention, resulting in a uniformly defined minimum resolution. It avoids the implementation of additional hardware required for handling variable floating-point real numbers.

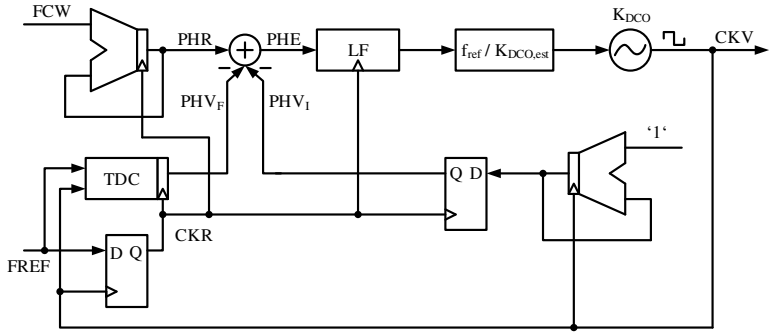


Figure 4.15: Block diagram of the proposed ADPLL design.

For a word with fixed size n, the integer range in unsigned representation is $[0, 2^n - 1]$. Fractional values for an m bits wide word are in the range $[2^{-m}, 0]$. This can be used to find an efficient trade-off for minimum sized design and maximum needed accuracy. The needed sizes for the FCW can be derived from specification and be used as a reference to derive other word lengths in the design:

$$n = \lceil \log_2(FCW_{max}) \rceil \,, \tag{4.29}$$

$$m = \left\lfloor \log_2 \left(\frac{\Delta f_{ch}}{f_{ref}} \right) \right\rfloor \,. \tag{4.30}$$

For $f_{ref} = 40\,\text{MHz}$, a minimum of 9 integer and 12 fractional bits are needed to achieve the necessary accuracy.

4.2.4.1 Phase Error Calculation and Loop Filter

The phase error PHE is the result of subtracting the variable phase, composed of integer count PHV_I and fractional count PHV_F, from the reference phase PHR:

$$\text{PHE} = \text{PHR} - \text{PHV}_I + \text{PHV}_F \,. \tag{4.31}$$

It is the input to the loop filter depicted in figure 4.16.

Three filter paths are implemented in parallel, each suitable for the require-

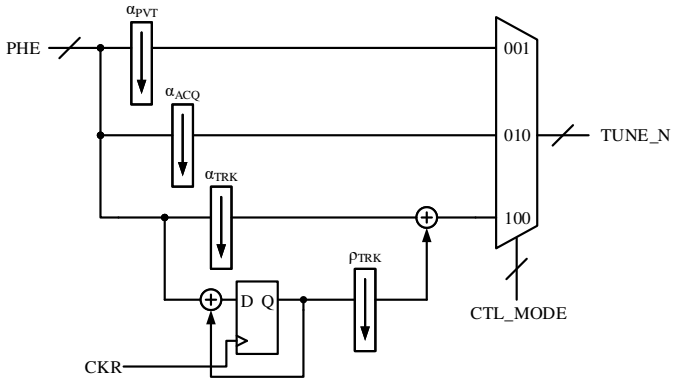


Figure 4.16: Loop filter schematic.

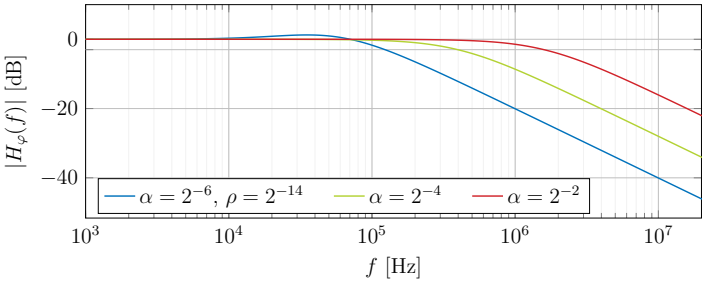


Figure 4.17: Mode dependent s-domain transfer functions.

ments of the loop control of the respective DCO bank. In PVT and acquisition (ACQ) mode, f_{BW} should be large to allow fast settling of the phase error φ_e. Thus, type-I PLL implementations with only proportional coefficients α_{PVT} and α_{ACQ} are chosen. This ensures fast decaying transients even for large phase error corrections. Instead of multiplication blocks, the filter operation is implemented with arithmetic right shifts. While this only allows a coarse selection of the bandwidths in powers of 2, it eases the design effort as shift operations are efficiently implemented in hardware. The tracking mode implements a type-II PLL with coefficients α_{TRK} and ρ_{TRK}. Choosing $\alpha_{PVT} = 2^{-2}$, $\alpha_{ACQ} = 2^{-4}$, $\alpha_{TRK} = 2^{-6}$ and $\rho_{TRK} = 2^{-14}$ results in the transfer functions as shown in figure 4.17. It ensures fast settling in PVT mode with $f_{BW,PVT} \approx 1.6\,\text{MHz}$ and successively approaches the desired tracking

bandwidth $f_{BW,TRK} \approx 100\,\text{kHz}$.

Limited word lengths can cause malfunctions in tracking (TRK) mode because wide range shift operations lead to loss of significant bits in the integrating path of the filter. By swapping the order of accumulation and shift operation, PHE is accumulated with fully available precision before shifting. This solution avoids extending word lengths and saves hardware effort at the cost of introducing a non-linearity if the filter coefficients are to be change online.

4.2.4.2 DCO Control

The loop filter output word size is the same size or larger than the FCW, which in general is larger than the available tuning bits of the DCO per bank. To maintain the loop gain and dynamic range of the loop, a gain normalization of the loop filter output is necessary.

An efficient way of implementing a coarse normalization is the selection of the appropriate bits from the OTW. From system considerations, we know that if a single unit capacitor in a DCO bank is altered, the output is increased or decreased by a linearized frequency step of width Δf. This relates to an LSB change in the tuning word of the corresponding filter output as in (4.32):

$$\begin{aligned} \Delta f_{mode} &= f_{ref} \cdot \Delta C_{mode} \\ &= \text{LSB}(\text{OTW}_{mode}) \,. \end{aligned} \tag{4.32}$$

It suggests that multiplication with f_{ref}/K_{DCO} can be approximately substituted by cutting the correct bits out of the OTW such that an LSB change of OTW equals a frequency step Δf of the currently active DCO capacitor bank. This procedure is schematically drawn in figure 4.18, in which OTW fractions of each mode are taken out of the fixed-point command word starting from the correct LSB position. Within each mode, an overlap of two LSB with the prior mode is implemented yielding the interlocking of consecutive tuning steps by design as proposed in figure 4.18. Tracking mode OTW is further divided into an integer and fractional portion which is used by a $\Delta\Sigma$-modulator to achieve finer frequency resolution.

Figure 4.19 shows the further processing of the selected TUNE_N words for PVT and acquisition mode. While their mode is active, the tuning word is passed to the oscillator. The highest bit is inverted, thus effectively shifting the tuning word from its signed region to an unsigned range. It is further offset by *MEM_DCO* to allow pre-tuning and safe startup of the DCO. This

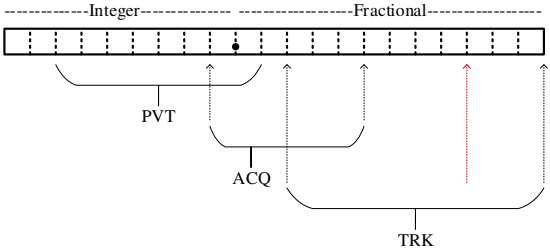


Figure 4.18: DCO gain normalization.

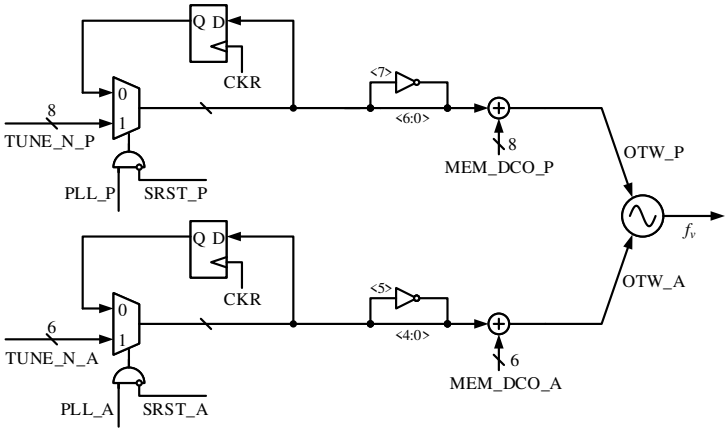


Figure 4.19: DCO control block diagram for PVT and ACQ mode.

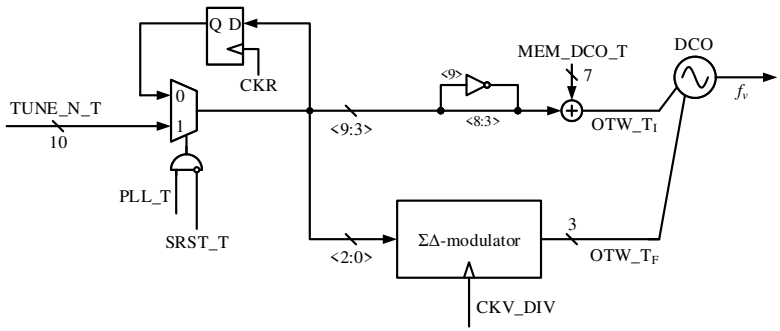


Figure 4.20: DCO control block for TRK mode.

static offset will be compensated by the loop. If the mode is no longer active, the last value will be held.

To achieve the specified fine resolution of 10 kHz, it is not sufficient to use the OTW directly to control the tuning bank. The lower limit of Δf in tracking mode is approximately 100 kHz, which is a decade larger than the required resolution. Thus, a dithering approach is needed. A ΣΔ-modulator is used to dither some capacitors multiple time during a reference period. The time averaged capacitance value will equal the fractional tuning word TUNE_N_T$_F$.

Figure 4.20 reveals the DCO control structure for tracking mode. The LF output TUNE_N_T is split into integer and fractional portion. With (4.32) rearranged to

$$\mathrm{LSB}(\mathrm{OTW_{TRK}}) = \frac{\Delta f_{TRK}}{f_{ref}}\,, \tag{4.33}$$

the OTW LSB of the integer valued tracking mode control word is determined as the −9-th LSB position of TUNE_N_T. The integer value packet is manipulated analogously to PVT and ACQ mode and the fractional part is modulated by the circuit in figure 4.21.

The ΣΔ-modulator structure is implemented as MASH-3 architecture [MC91]. It can be integrated in CMOS and minimizes fractional spurs. The modulator comprises three sequential adders and a combiner circuit consisting of 3-bit shift registers for each carry output of the adders. This structure constitutes

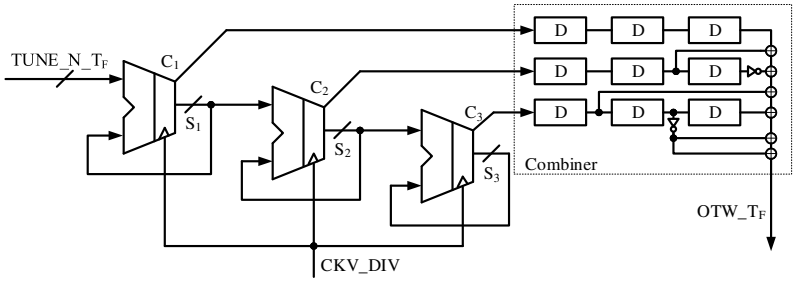


Figure 4.21: ΣΔ-modulator with combiner circuit for dithering of fractional tuning word.

the following transfer function:

$$H_{\Delta\Sigma}(z) = 1 + E_{q3}(z) \cdot (1 - z^{-1})^3 \,. \tag{4.34}$$

The original signal is not transformed from input to output. However, the quantization noise after the third stage output is high-pass filtered by $(1-z^{-1})^3$. The noise energy is shaped to higher frequencies by oversampling and increasing resolution with minimum design effort. It is suggested that the modulator order is at least three as otherwise, spurious emissions are too high overshadowing the increased resolution effect [SB06]. Even so, to ensure a spur-free spectrum the first stage accumulator needs to be initialized correctly [Näh07]. Ideally, the initial value should be an irrational number. In practical implementation a close enough approximation is sufficient. Nevertheless, this requires an adequate word width of the adders. Alternatively, to increase the randomness of the ΣΔ-modulator output, additional noise from an LFSR can be added to the input as LSBs.

The circuit is clocked on CKV_DIV, the DCO clock divided by 8. The modulator output is the sum of delayed and inverted versions of the carry bits [Ney10]:

$$\mathrm{OTW\_T_F} = C_1 D^3 + C_2(D^2 - D^3) + C_3(D - 2D^2 + D^3) \,. \tag{4.35}$$

The delay outputs could immediately modulate the tracking capacitors, depending on the internal DCO design, instead of passing a 3-bit binary control word to the DCO. In this case, the seven output bits are hard-wired to tracking capacitors implicitly summing up the bit values.

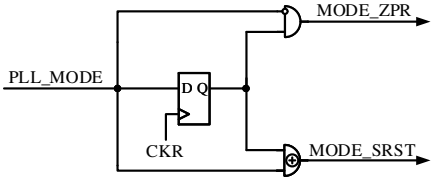


Figure 4.22: Mode control.

4.2.4.3 Mode Control and Zero Phase Reset

To calculate the phase error, the phase detector computes the difference between reference phase PHR and variable phase PHV. In normal operation, PHR is generated by accumulating the FCW:

$$\mathrm{PHR}[k] = \mathrm{FCW} + \mathrm{PHR}[k-1] \,. \tag{4.36}$$

At mode switching, the settled remaining phase error from the previous PLL mode needs to be reset in order for the LF to only control the excess phase around the new center frequency. The same goes for a software reset of the PLL. If a mode change or reset is indicated, the reference phase is thus loaded with

$$\mathrm{PHR}[k] = \mathrm{FCW} + \mathrm{PHV_I}[k-1] - \mathrm{PHV_F}[k-1] \,. \tag{4.37}$$

Inserting (4.37) in (4.31) yields:

$$\begin{aligned} \mathrm{PHE}[k] = \mathrm{FCW} &+ \mathrm{PHV_I}[k-1] - \mathrm{PHV_F}[k-1] && (4.38)\\ &- \mathrm{PHV_I}[k] \quad + \mathrm{PHV_F}[k] \,. && (4.39) \end{aligned}$$

Rearrangement and application of the expectation operator $E\{\cdot\}$ results in:

$$\begin{aligned} E\{\mathrm{PHE}[k]\} = \mathrm{FCW} &- E\{\mathrm{PHV_I}[k] - \mathrm{PHV_I}[k-1] && (4.40)\\ &- \mathrm{PHV_F}[k] \quad + \mathrm{PHV_F}[k-1]\} && (4.41)\\ &= 0 \,. && (4.42) \end{aligned}$$

A correct alignment of mode control and zero-phase-reset is critical for a glitch-free operation. The mode control is implemented as a one-hot shift register that shifts to the next state after a certain amount of reference cycles. The

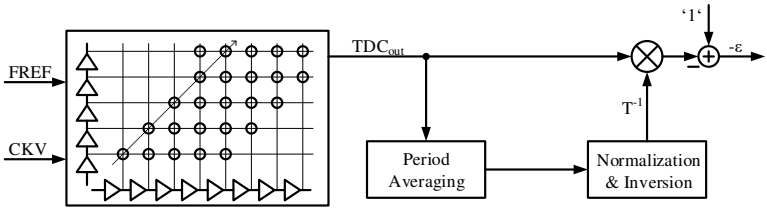


Figure 4.23: Block Diagram of TDC output processing.

schematic in figure 4.22 depicts the generation of zero-phase-reset signal ZPR and synchronous reset signal SRST of the DCO registers. If the ADPLL changes operating mode, PLL_MODE toggles from high to low for the previous mode bit and low to high for the new mode bit. Thus, MODE_ZPR of both modes is high for a single clock cycle and MODE_SRST of the succeeding mode is pulled high for one tact, resetting the corresponding DCO bank.

4.2.4.4 TDC Evaluation

The ADPLL features a 2D-Vernier TDC with a resolution of $\sim$ 3 ps. After conversion, it provides two values to the loop:

- a measurement of the last DCO period
- a fractional measurement of the time between last CKV and FREF edge

As shown in figure 4.23, the converter output needs further processing to compute the actual fractional phase error $-\varepsilon$. The measured DCO period is averaged over an arbitrary amount of samples. Choosing a multiple of two allows again for an efficient implementation in hardware, since the division by sample number can be replaced by an arithmetic shift. The result is then inverted before multiplying it with the fractional phase offset. The normalized fractional phase error in the range of $[0, 1]$ is then subtracted from 1.0 to yield the fixed-point fractional phase error $-\varepsilon = \mathrm{PHV}_F$.

4.2.4.5 Fractional Spur Cancelation

To implement the phase error compensation mechanism described in section 4.2.3, the reference phase PHR needs first to be quantized to the same

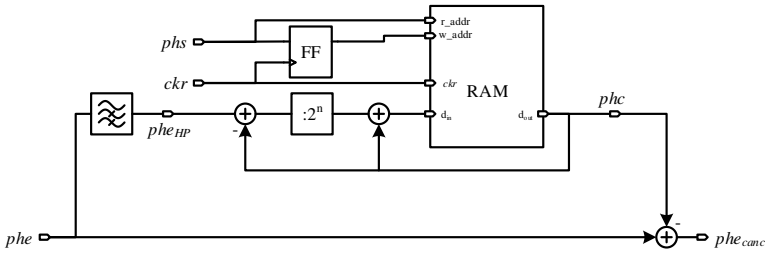


Figure 4.24: Implementation to estimate TDC metastability window from phase error.

step size as resolved by the TDC. This is achieved by supplying an additional input to the phase comparator that subtracts the expected ideal phase error sawtooth introduced in figure 4.10. It is calculated as

$$\mathrm{PHS} = \mathrm{PHR} \bmod \mathrm{TDC_{res}}\,. \tag{4.43}$$

$\mathrm{TDC_{res}}$ is provided as part of the TDC evaluation. Assuming equal TDC steps it is given by the output of "Normalization & Inversion" block of figure 4.23.

As explained before, the expected phase error $E\{\mathrm{PHE}\}$ is still not 0 but follows the error function. An estimate of it is generated by averaging multiple phase error periods of length $\mathrm{TDC_{res}}$. The implementation shown in figure 4.24 follows the recommended approach of Ho and Chen, but the hardware effort is greatly reduced. For one the periodicity to be covered is reduced from $size(\mathrm{PHV_F})$ to $size(\mathrm{TDC_{res}})$. Secondly the averaging filter can be shared for all sample bins. The accumulator values are stored in an on-chip random-access memory (RAM) and fetched once in each period for an update.

4.2.4.6 TDC Dithering

For comparison the approach from the second group of spur reduction mechanisms, introduced in section 4.2.3, is implemented. The reference clock input of the TDC is dithered to add additional noise and randomize the sampling point of the TDC, as depicted in figure 4.14. To achieve this, the *FREF* input is fed into a coarse 4-bit binary weighted delay line. The LSB of the delay is targeting 7 ps to be prime with the TDC resolution of 3 ps and to be able to cover more than one DCO period.

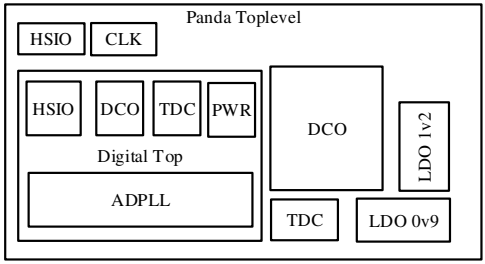


Figure 4.25: The circuits on the test chip.

The pseudo-random dithering values are created in digital domain using a LFSR. The 16 possible delay times are measured at start-up and saved in a look-up table. The compensation of the measured TDC values is introduced before the TDC evaluation block.

4.2.5 System Partitioning

The system described above in section 4.2.4 is realized in a 28 nm TSMC technology. Performance critical blocks, such as DCO, TDC, high-speed divider and output buffers, are full custom analog designs, while the loop behavior is synthesized from RTL code using digital standard cells. Figure 4.25 gives an overview of the relevant system partitions. Besides the main loop components, the chip features 2 low-noise LDOs to provide a stable and clean supply to the analog components. The digital circuitry is supplied directly from external. It consists of the PLL components as well as a register bank to configure the LDOs, manually control the analog components and allow to configure all loop parameters. The registers are controlled via an SPI interface. Furthermore, a high-speed-IO interface allows to monitor the TDC output online.

In figure 4.26, the circuit's implementation is shown with focus on the supply domains. The signal paths are shown in black while the supply paths are shown in red. The DCO is supplied with 1.2 V to provide a high output swing and isolate it from other circuits to prevent them from corrupting the oscillator phase. A second analog supply domain with 0.9 V is shared by the output buffers, dividers and TDC. The system analysis in section 5.2 will investigate

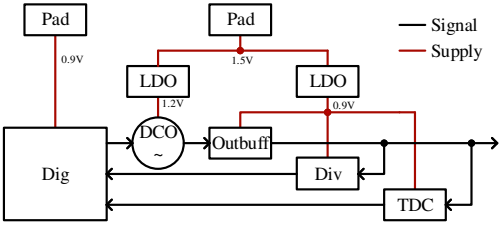


Figure 4.26: The ADPLL from a supply perspective.

the inter domain coupling of these blocks and isolate their effects on the system performance.

CHAPTER 5

SIMULATION RESULTS

5.1 ΔΣ-ADC

To verify the behavior of the ΔΣ-ADC introduced in section 4.1.1 on system level and investigate possible effects of supply modulation, the circuit is modeled using the framework introduced in chapter 3. The main signal path is modeled using the `AMS_Signal_Spectral` to allow inputs with arbitrary frequency components to be represented. The supply modulation is also modeled in the frequency domain, as will be explained in section 5.1.4.1. The models of the analog subblocks use a transfer function based approach as explained in section 3.3.2 to consider effects of a varying supply voltage where necessary. Since the supply net implementation provides a flexible combination of the available `AMS_Signal` types, models that do not take supply noise into account can fall back on a constant voltage implementation instead.

5.1.1 DAC Models

Both feedback and reference voltage DACs use the same model with different parameter sets as their behavior is similar. The transfer characteristics from supply input to the outputs have been determined with transistor-level node-based AC simulations. The resulting transfer functions are exported as .csv-files and during system simulation imported in the necessary C++ block objects as look-up tables. Linear interpolation is used to estimate the blocks response for inputs in between support points. The result from the supply path is then superposed with the nominal output voltage as additional time-varying DC component of the output spectrum. This implementation is visualized in figure 5.1.

The PSN transfer function for the reference voltage DAC to any of its outputs depends on the selected tuning word, as depicted in figure 5.2a for the p-output. Switching more current to one path lowers its impedance to the supply

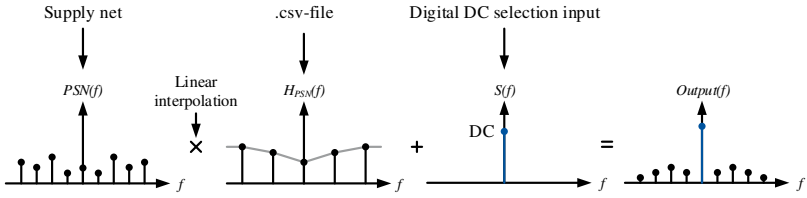


Figure 5.1: Spectrum processing in DAC models.

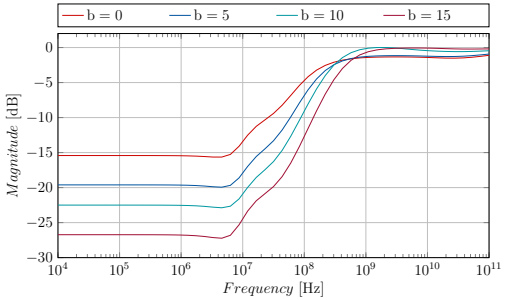
resulting in a decreased attenuation. The other output, of course, behaves reciprocal. It is inherent to the design that it necessarily has a different number of switched-on transistors in both differential output paths. This diminishes an advantage of the differential design in allowing the PSN to affect the differential ouput voltage as depicted in figure 5.2b.

Similarly, the influence of PSN on the differential current of the feedback DAC is of interest, as shown in figure 5.3. Same as above, the effect of PSN is dominated by parasitic capacitance at high frequencies. Furthermore, the behavior is determined by the *amp* and *tune* input values. It is to be noted, that a well-tuned output, where the absolute values of the currents of the p- and n-path are equal and the common-mode current is zero, does not necessarily reduce the influence of PSN. In the figure the well-tuned case is given for $tune = 8$. However, the current fluctuations due to PSN increase with the DC output current, regardless whether it is caused by higher *tune* or *amp* setting.

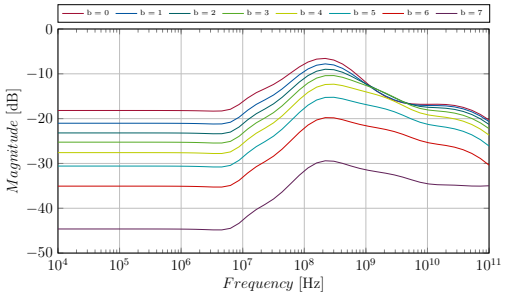
5.1.2 Comparator Model

The most sensitive part of the comparators is their first differential amplifier stage. Bae, Shim, Koo, Cho, Pak, and Kim [Bae+13] attribute PSN effects to an impedance difference on the gates of the differential pairs, resulting in a different impact of PSN on both differential paths. Schematic simulation of the circuit show a similar behavior. However, the influence of the PSN transfer function compared to the regular signal transfer function is at least 40 dB lower. The overall PSN effect on the comparator is thus negligible.

Since the sampling comparators are clocked, their outputs only need to be reevaluated once every period. That allows for an efficient and easy implementation of the comparison of two spectral inputs signals. When the sampling



(a) Transfer function of PSN to single-ended output.



(b) Differential transfer function.

Figure 5.2: PSN transfer function of the reference voltage DAC for various tuning settings.

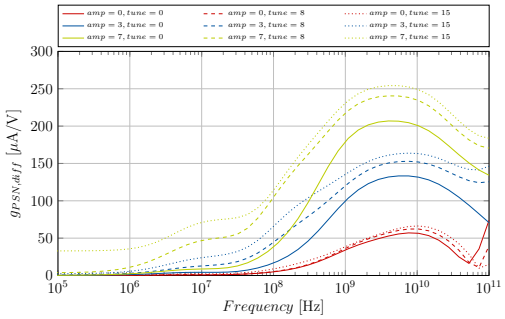


Figure 5.3: I_{diff} caused by PSN in feedback DAC.

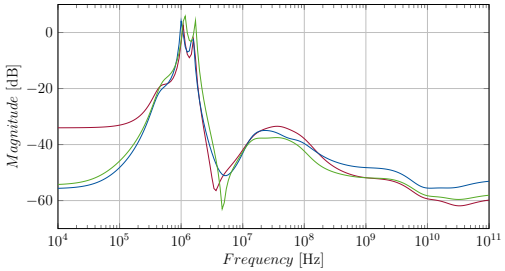


Figure 5.4: Exemplary transfer functions of PSN to the I-output of the loop filter considering mismatch from process variation.

process is triggered, the input Fourier series are evaluated at the current time stamp according to:

$$f(t_{sample}) = c_0 + \sum_{i=1}^{N} 2\arg(c_i)\cos(2\pi f_i t_{sample} + \phi(c_i)) \; . \tag{5.1}$$

The comparator output is then switched after comparing the sampled inputs $f(t_{sample})$.

5.1.3 Loop Filter Model

The filter consists of three stages with two differential integrators. The paths are cross-coupled to implement the quadrature transfer function. Furthermore, each path features a feedback path to optimize the zero placement and two capacitive feed-forward paths to improve stability and reduce power consumption [Ata+13]. To minimize path mismatch, all elements are tuneable, which results in a large amount of possible configurations. To reduce modeling effort, only one standard configuration is simulated.

The high PSRR of the operational amplifiers (OPAMPs) and the fully differential design show only a negligible influence of PSN on the filter's output in the nominal corner. However, statistical Monte Carlo simulations, taking process variation into account, reveal a much more severe influence. Figure 5.4 shows three exemplary transfer functions of PSN to the I-output.

A possible implementation of the filter transfer function would be to multiply each frequency component of the input with its transfer coefficient, as done

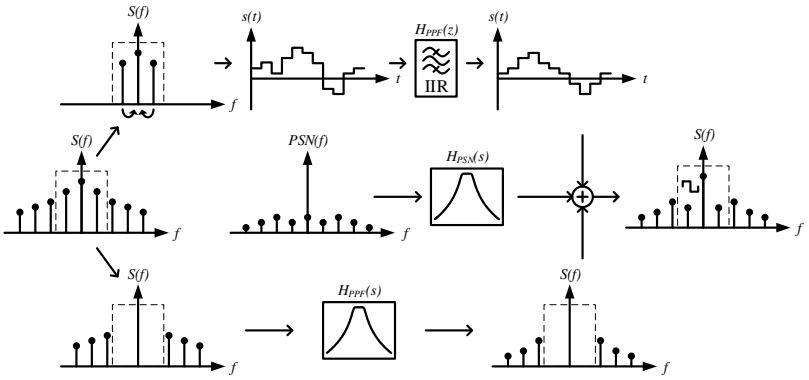


Figure 5.5: Spectrum processing in the poly-phase quadrature loop filter model.

for the DAC model. However, as described in section 3.2.3 the signal representation collects multiple similar tones in one frequency bin by describing them with a center frequency and a time-varying phasor. Depending on the modulation scheme of the received signals, the actual signal frequency does not necessarily equal the center frequency. While the introduced error is negligible in most cases this is not the case for a band-pass filter with steep edges. Therefore, the filter is implemented as shown in figure 5.5. The in-band signal is sampled and processed through an IIR filter following the approach from [Jun15]. The sampling frequency should equal the sampling rate of the overall ΔΣ-ADC to avoid the generation of unwanted intermodulation products. The output signal is then included as a time-varying DC component of the spectrum.

The noise spectrum at out-of-band frequencies is assumed to be virtually static between two sampling events. It is multiplied with a static transfer function in the frequency domain and added to the output spectrum.

5.1.4 System Evaluation

Using the implemented models, a testbench consisting of the analog blocks together with the digital post-processing and calibration circuits is implemented as shown in figure 5.6.

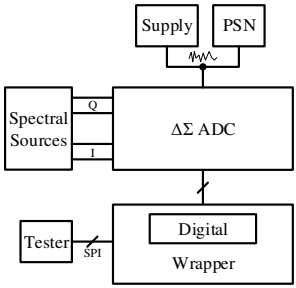


Figure 5.6: ΔΣ-ADC simulation testbench.

White noise on the supply is considered in the following evaluation. Its generation is described in section 5.1.4.1. It is applied to the circuit with different noise levels and the effect is investigated for single-bit and multi-bit mode.

5.1.4.1 Noise Generation

A supply source has been implemented to generate discrete Gaussian white noise. It creates a spectrum from a single fundamental frequency and a given number of harmonics. Each component has a normally distributed amplitude and uniformly distributed phase. The standard deviation is configurable. After each period the spectrum is randomized again using a Mersenne Twister [MN98]. Figure 5.7 shows the generated noise signal during simulation sampled at 96 MHz with a single fundamental frequency of 1 MHz and 100 harmonics. Each component has a normally distributed amplitude with a standard deviation of 1 mV.

5.1.4.2 Single-bit mode

For comparison the setup is first simulated without any PSN present. The results of an event-driven simulation of the ADC in single-bit mode are shown in figure 5.8. The inputs Q_{in} and I_{in} are two sinusoids with a phase difference of $\frac{\pi}{2}$, a differential voltage of 800 mV and a frequency of 1 MHz. The ADC's sampling rate is 96 MHz. Q_{fb} and I_{fb} are the feedback currents from the internal DACs to the loop filter. The filter outputs $Q_{LF,out}$ and $I_{LF,out}$ are fed into the internal ADC comparators. The delta-sigma encoded ADC output

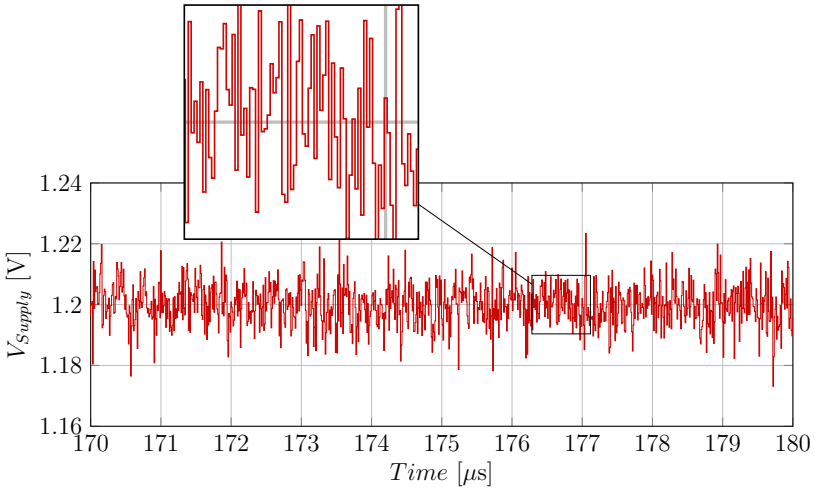


Figure 5.7: White noise used in ADC simulation.

signal, $Q_{out,dig}$ and $I_{out,dig}$, are the inputs to the digital post-processing, where they are down-sampled using digital filtering. Without noise, the feed-back loop is stable and shows the expected behavior. Figure 5.10 plots the fast Fourier transform (FFT) of the ADC outputs. The characteristic noise shaping of the ΔΣ-modulation shapes the quantization noise away from the signal peak at 1 MHz and thereby improves the SNR.

Figure 5.9 shows the same simulation with white noise as described in section 5.1.4.1. The supply voltage is also plotted. As can be seen, the output signal of the filter is significantly disturbed. The noise causes a phase shift between the two quadrature paths which leads to an instability the feedback DACs can not compensate. The result is a saturation of the ΔΣ-ADC that can be seen in the long periods during which the digital output signals do not switch.

Figure 5.11 depicts the output spectrum for the simulation. As can be seen around 1 MHz the noise floor is elevated. Furthermore, the second signal at 2 MHz appears. This is caused by the supply noise propagation through the non-ideal filter which does not suppress the second harmonic anymore.

The SNR values in table 5.1 are obtained by performing an FFT of the output signals, filtering the signal with the decimation filter transfer function of the demodulator and separating the signal peak at the input frequency from

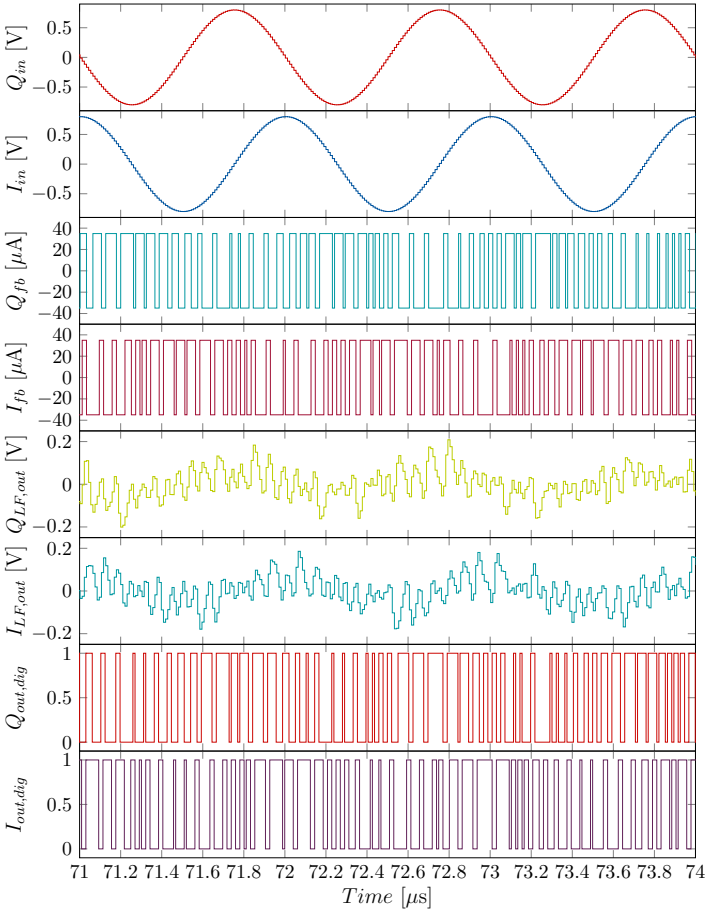


Figure 5.8: ADC simulation results in single-bit mode without PSN.

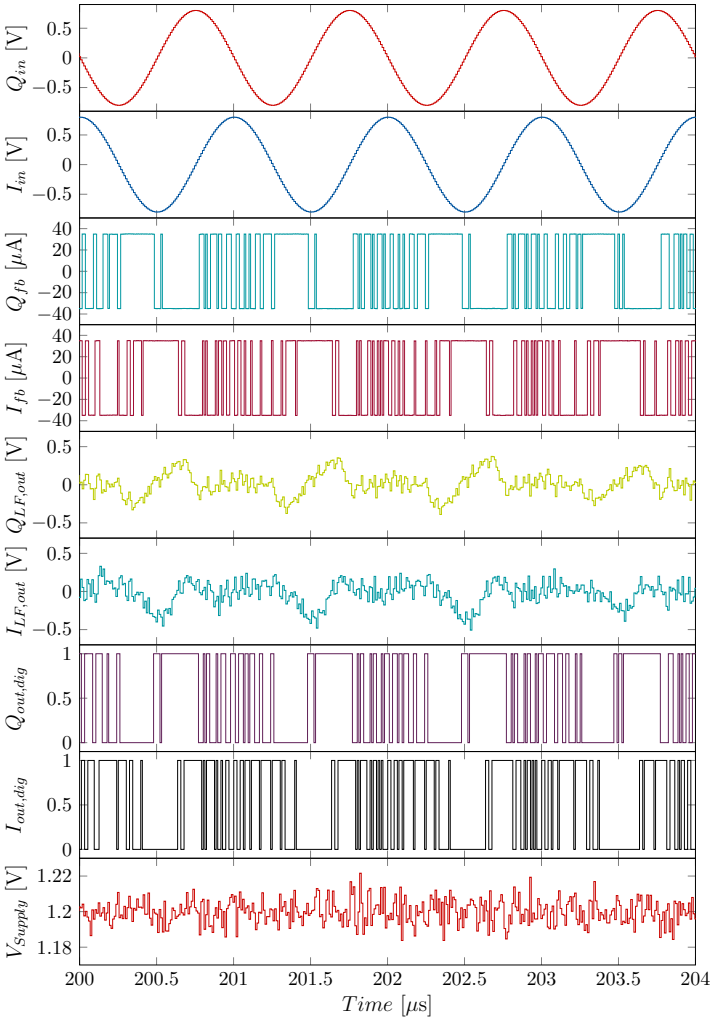


Figure 5.9: ADC simulation results in single-bit mode with PSN.

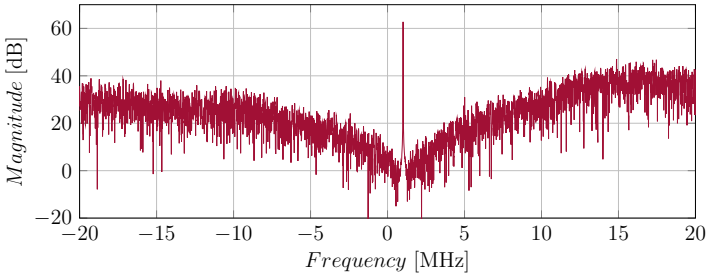


Figure 5.10: FFT of ADC simulation results in single-bit mode without PSN.

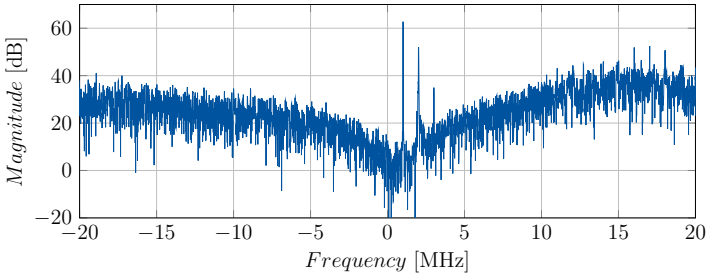


Figure 5.11: FFT of ADC simulation results in single-bit mode with white PSN as described in sec. 5.1.4.1.

σ per harmonic [µV]	SNR [dB]
0	42.6
250	40.7
500	39.1
750	37.8
1000	34.4

Table 5.1: SNR with white noise as PSN in single-bit mode.

the noise integrated over the bandwidth. The performance deterioration is clearly visible as the SNR decreases with increasing PSN levels, implemented as different standard deviations σ of the noise source harmonics.

5.1.4.3 Multi-bit mode

Results for multi-bit mode simulations, with the ADC configured as a 4-bit tracking quantizer, are shown in figure 5.12 without noise and in figure 5.13 with the same noise level as for the single bit mode. For one, even though the DAC can compensate for the error, the tracking quantizer can not follow the signal with added noise fast enough. This is further aggravated due to the implementation choice, that the sign is determined independently of the other comparators using the third single-ended comparator. Skipping one or more bits while changing the sign thereby worsens the error. For example, instead of going from 10 to 7 due to skipping two bits, the output jumps from 10 to 5 due to the inverting sign causing high jumps in the digital output signal and thus the feed-back voltage.

The comparison of the output spectrum with and without noise shown in figure 5.14 and 5.15 respectively also clearly illustrates the influence of noise. While the spectrum without noise shows only uneven harmonics with an alternating sign, as expected from a quadrature signal, the spectrum with noise shows additional harmonics at uneven and even multiples on both sides of the spectrum. Similar to the single-bit simulations, the harmonics at $2f_{in}$ are especially significant, not least because it is very close to signal. Additionally, the overall noise level is considerably higher compared to the simulation results without noise.

The SNR results for different noise levels are listed in table 5.2. Comparing the results to the single-bit mode simulations, it is notable, that in multi-bit

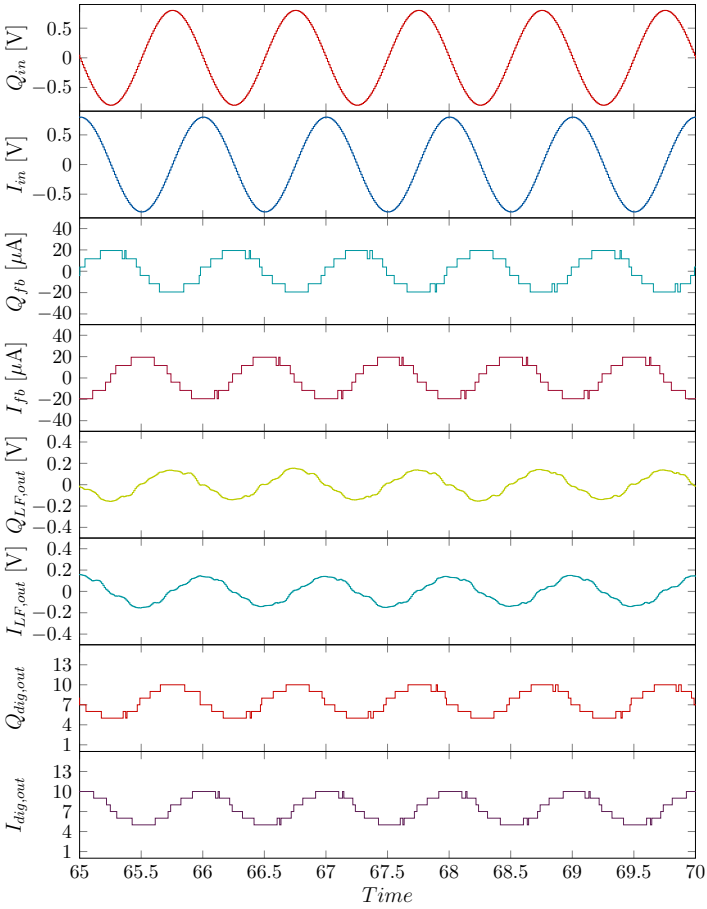


Figure 5.12: AMS simulation filter signals in multi-bit mode without PSN.

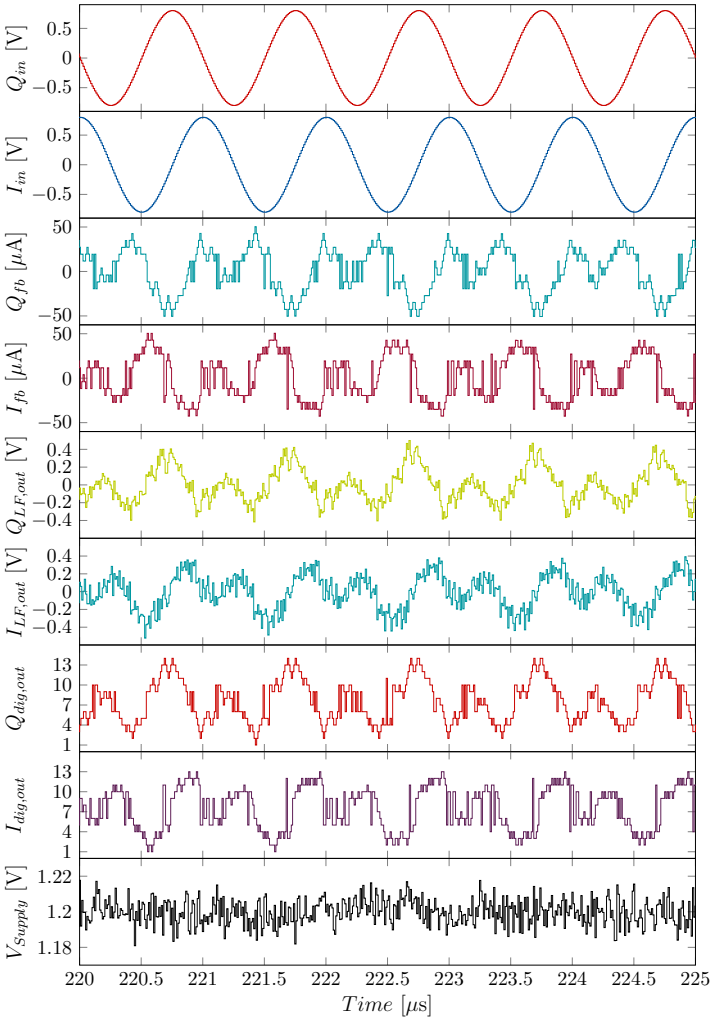


Figure 5.13: AMS simulation filter signals in multi-bit mode with PSN.

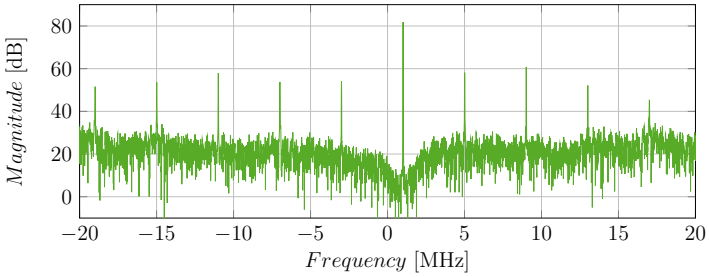


Figure 5.14: FFT of the ADC's output in multi-bit mode without PSN.

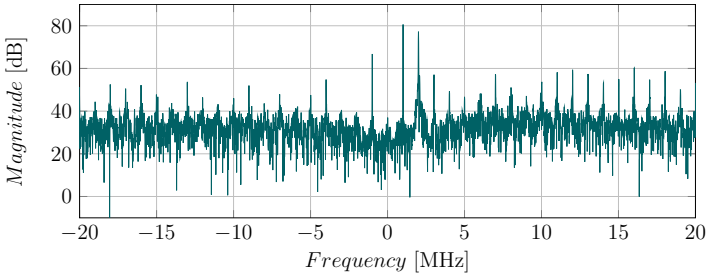


Figure 5.15: FFT of the ADC's output in multi-bit mode with PSN.

σ per harmonic [µV]	SNR [dB]
0	56.0
250	45.5
500	40.8
750	35.6
1000	33.7

Table 5.2: SNR with white noise as PSN in multi-bit mode.

operation the SNR is considerably better without noise, but also deteriorates more severely with increasing noise.

5.2 All-Digital PLL

To simulate the ADPLL as a complete system, analog components are replaced by functional models. They provide a pin-equivalent description of the circuit behavior, where major performance characteristics are configurable via parameters. Thus, an early system design is enabled by using assumptions for these parameters while they can be updated when accurate results from analog simulations are available. The parameters presented here are extracted from schematic simulation of the developed designs.

To consider the effects of system partitioning the models incorporate behavior for supply variation. To be able to act as aggressor in the system, the current drawn by the subcircuit from the supply is modeled. On the victim side, timing variation due to a changing supply voltage is considered. The α-model introduced in section 3.3.2 is used to model this effect. Translating the accumulated current on the supply to a voltage ripple is incorporated in the LDO model. All circuits are modeled in time-domain to be completely compatible with the digital RTL simulation.

5.2.1 LDO Modeling

The LDO implementation on chip is depicted in figure 5.16. It has been published in [Wan+20]. A PMOS pass device is used to control the LDO current. The output voltage V_{out} is fed back directly to an error Amplifier, where it is compared to a reference voltage V_{ref}. Instead of the typical approach to use a down-divided value for feedback, a programmable reference voltage

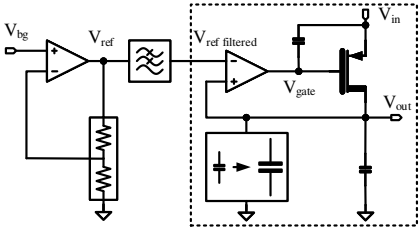


Figure 5.16: The LDO's implementation on the chip.

is used to control the output voltage. This minimizes the noise sources at the output. The reference voltage is generated from a bandgap voltage and low-pass filtered afterwards to suppress as much noise as possible. A g_m based current mode capacitance multiplier is used to reduce the capacitor size on-chip [BCB18].

5.2.1.1 Model Development

The LDO model only considers the main subcomponents inside the dotted box. It recreates the toplevel hierarchy by replacing the sub-blocks with macro models.

The two capacitors, gate capacitance and output capacitance, use the same model, which integrates the current into the device to an output voltage following $V_{cap} = \frac{1}{C} \int I_{cap} dt$. The algorithm integrates the current over discrete steps (5.2). The step size is determined dynamically either by the next external event or a fixed maximum time step that depends on the maximum allowed voltage tolerance V_{tol}.

$$V_{cap,n} = \frac{1}{C} I_{cap,n-1} \Delta t + V_{cap,n-1} \,, \tag{5.2}$$

$$\Delta t = t_n - t_{n-1} \,, \tag{5.3}$$

$$t_{step} = \frac{V_{tol} C}{I_n} \tag{5.4}$$

For the output capacitance consisting of a fixed capacitor and the virtual capacitance from the cap multiplier, an equivalent output capacitance modeled as two leaky capacitors in series is implemented.

The error amplifier (EA) acts as voltage controlled current source with simple linear gain and limiting minimum and maximum currents, resulting in the

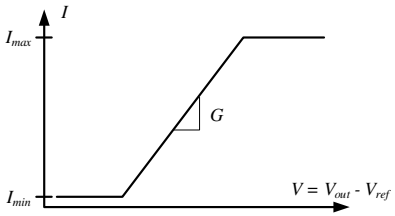


Figure 5.17: The modeled EA's voltage current relation.

Table 5.3: EA model parameters.

Parameter	G[mS]	I_{max}[mA]	I_{min}[mA]
Value	13	6.426	-1.132

transfer function shown in figure 5.17. The values derived from schematic simulation are given in table 5.3

The pass device transistor is implemented as the standard three-region transistor model with equations:

$$I_d = \begin{cases} 0 & V_{GS} < V_{th} \\ \frac{k}{2}(V_{GS} - V_{th})^2 & V_{GS} - V_{th} < V_{DS} \\ k((V_{GS} - V_{th})V_{DS} - \frac{V_{DS}^2}{2}) & V_{GS} - V_{th} > V_{DS} \end{cases} . \tag{5.5}$$

5.2.1.2 Model evaluation

The developed model is evaluated by comparison to the results from transient schematic simulation. Load regulation behavior is considered, as it provides an important characteristic for analyzing the noise behavior of the circuit. It determines the impact of a load current modulation on the supply voltage. The testbench setup is schematically drawn in figure 5.18. A current step of $I_{out} = 10\,\text{mA}$ is applied.

The resulting voltage response of model and schematic are shown in figure 5.19 The model shows a higher overshoot but an similar settling time, compared to the schematic simulation. This could be caused by a deviation between the gain of the error amplifier model and circuit implementation.

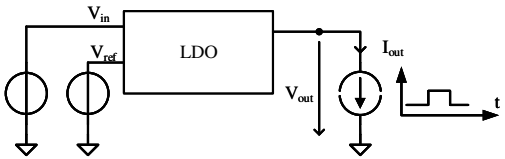


Figure 5.18: The LDO's testbench.

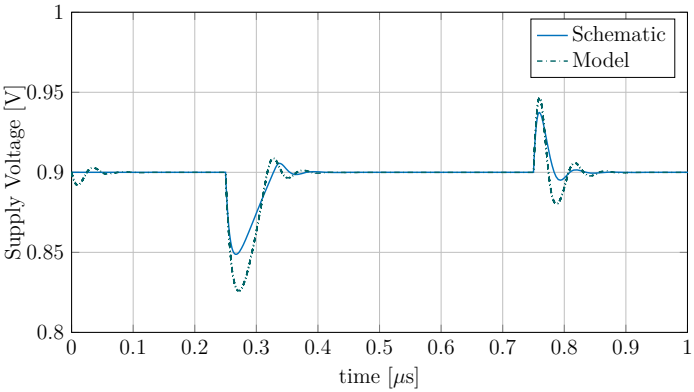


Figure 5.19: The LDO's step responses of schematic and model compared.

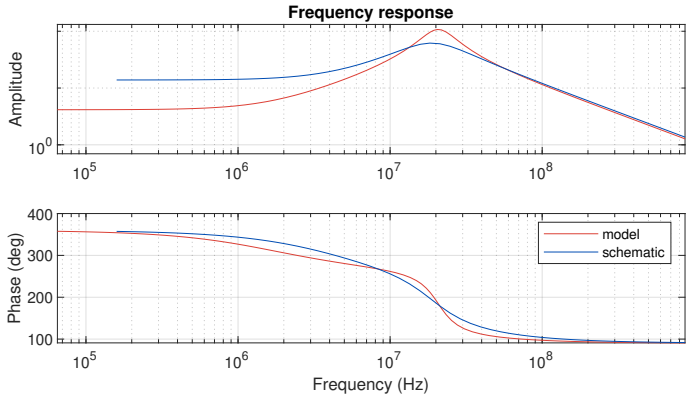


Figure 5.20: The frequency response of LDO model and schematic.

Looking at the frequency response of the circuit (figure 5.20), it can be seen that the behavior for high frequencies ($f > 10\,\text{MHz}$) matches very well.

5.2.2 DCO Model

The DCO used in the ADPLL is depicted in figure 5.21. Contrary to the idealized design shown in figure 2.12, the capacitor bank is split in three separately tunable banks with different tuning step size to facilitate a large design space. It covers a frequency range from 9 GHz to 11 GHz by using 256 PVT tuning steps, 64 acquisition steps and 128 tracking steps. As already mentioned in section 4.2.4.2, additional tracking capacitors are used for modulation by a $\Sigma\Delta$-modulator. The tuning ranges by smaller banks cover at least 4 LSBs of the larger bank to facilitate for process variation.

As can be seen the DCO is connected to two supply domains. The main 1.2 V domain supplies the DCO core, while the capacitor bank control is connected to the 0.9 V domain to allow an easy connection to the digital signals.

5.2.2.1 Model Development

The DCO needs to generate an output signal with tunable frequency. All phase information of such a signal is included in the zero crossings of the

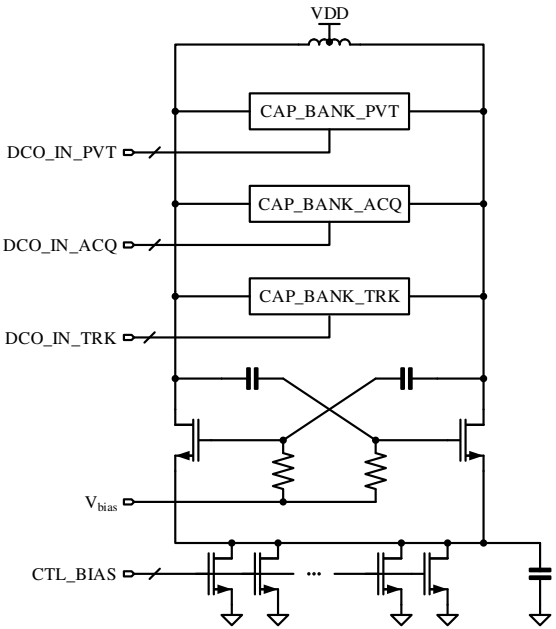
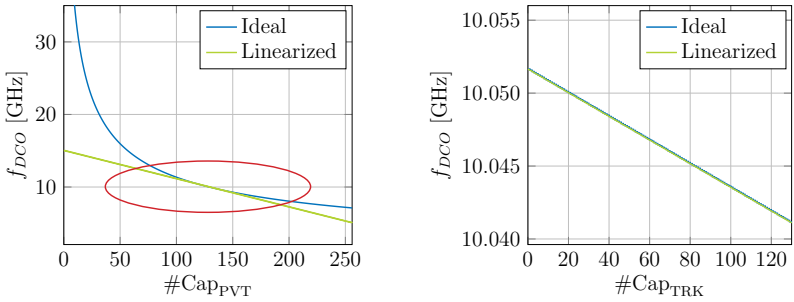


Figure 5.21: Block diagram of DCO with different capacitor banks.



(a) PVT mode.

(b) TRK mode.

Figure 5.22: Ideal and linearized frequency tuning in relation to ΔC.

Table 5.4: Capacitor unit-cell model parameters.

Cell	PVT_{on}	PVT_{off}	ACQ_{on}	ACQ_{off}	TRA_{on}	TRA_{off}
$C[fF]$	5.094	2.792	1.961	1.902	1.487	1.487
$\alpha C[aF/V]$	1.596	-47.72	0.028	-3.441	0.015	-0.004

output. Thus, a digital signal with equivalent jitter is used to model the DCO output.

Figure 5.22 plots the frequency of an ideal LC-tank in contrast to a linearized model. As can be seen, while the linearization is suitable for small deviation such as in the tracking bank, it does not cover the whole PVT bank accurately. Thus, the model calculates the oscillator frequency directly summing up all capacitor contributions and using (2.22). The switchable capacitor cells are controlled using transistors with gates connected to the 0.9 V analog domain. Thus the capacitance value varies with fluctuations on the supply. Since it varies differently with the on/off state of the cell, each cell is modeled separately. The characterizing parameters are summarized in table 5.4. The variation of the total capacitance over supply voltage for different usecase settings is shown in figure 5.23. The value of an acquisition LSB is also ploted for reference. As capacitance variation is lower for switched on cells, the influence is smaller for high settings, while for a typical usecase at 10 GHz center frequency the deviation for 1 % supply variation accumulates to more than 1 ACQ LSB.

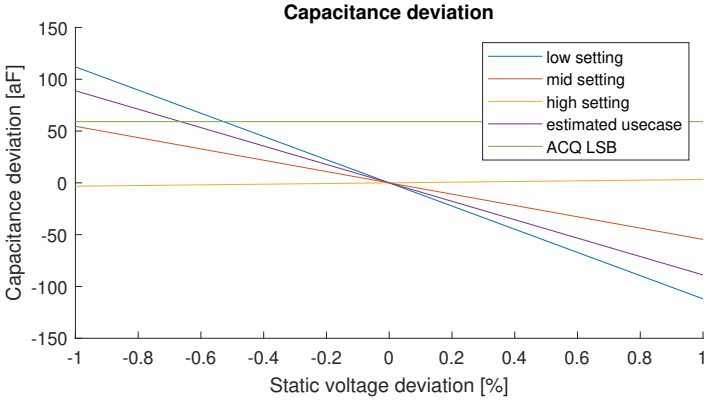


Figure 5.23: The capacitor banks with supply modulation.

The influence of the DCO core from the 1.2 V domain on itself has also been investigated. Figure 5.24 plots the expected voltage and current variations, as extracted from schematic simulation of the DCO's core together with the LDO. It can be seen that the supply voltage ripple, with $< 6\,\mu V$, is small. Thus, only a static noise model considering the white noise floor and wander noise for the DCO is implemented. The noise levels are extracted from schematic simulations at $-155\,dBc/Hz$ for the noise floor and $-120\,dBc/Hz$ at 1 MHz offset frequency for the wander noise. These noise source are transformed to equivalent jitter contributions, following the derivations in section 2.1.6.3. The resulting phase noise spectrum for the DCO is plotted in Figure 5.25.

5.2.2.2 Model Evaluation

The verification testbench for evaluating the capacitor bank modeling is depicted in Figure 5.26. The main performance criterion is the output phase noise, which is calculated from zero-crossings of the output clock signal. The phase noise of the model and schematic simulation is compared for two scenarios, one with a modulated 0.9 V supply and one without modulation. The modulating test signal V_{ac} is chosen to be similar to the expected modulation of the output dividers, as will be shown in section 5.2.3. Its mathematical

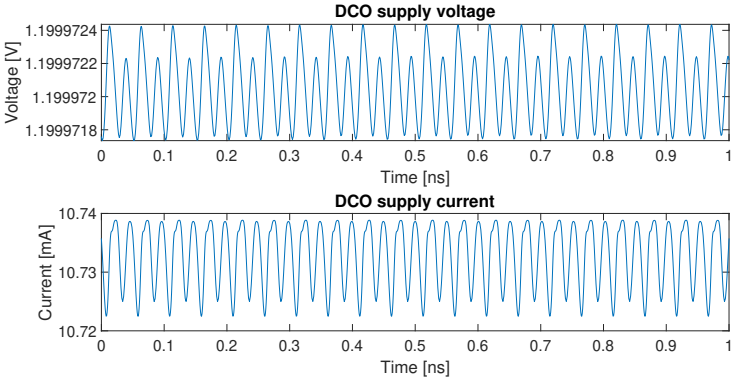


Figure 5.24: The DCO's supply voltage and current.

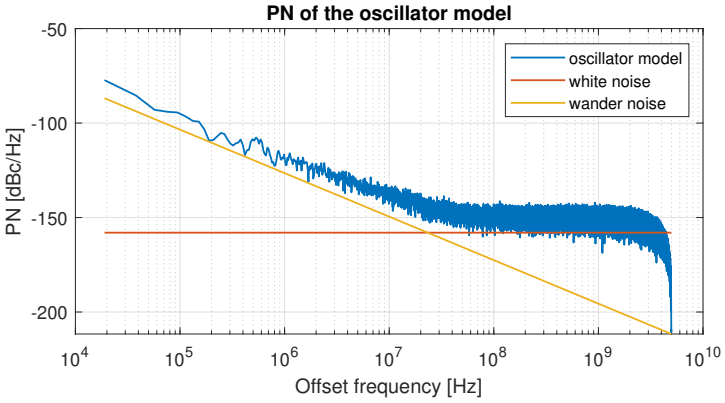


Figure 5.25: The DCO's output spectrum compared to the parameters.

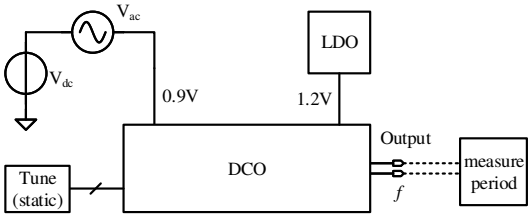


Figure 5.26: Testbench for the DCO model vs. schematic simulations.

expression is shown in (5.6):

$$V_{ac} = -0.5\,\text{mV} \cdot (sin(\omega/2 \cdot t) + sin(\omega/4 \cdot t)) \ , \tag{5.6}$$

$$\omega = 2\pi \cdot 10\,\text{GHz} \ . \tag{5.7}$$

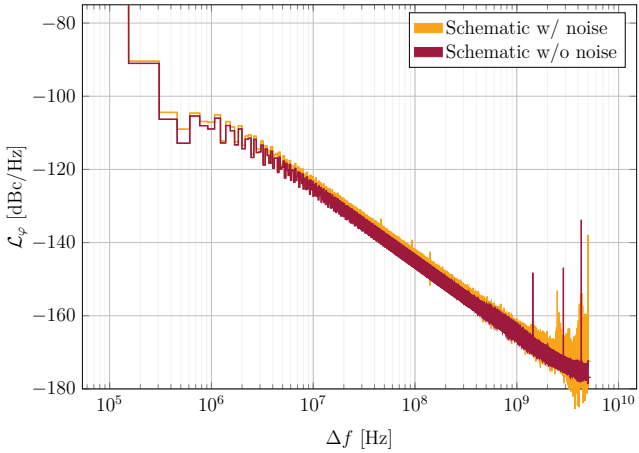
The schematic simulations are performed without transient noise to be able to observe effects which would otherwise be overshadowed by the noise floor. For the same reason, the white phase noise in the DCO model is disabled. The resulting output phase noise spectra are shown in Figure 5.27. As can be seen, the induced spurs at 5 GHz and 2.5 GHz are captured accurately, although their level is close to the expected noise floor.

5.2.2.3 Time Rescaling

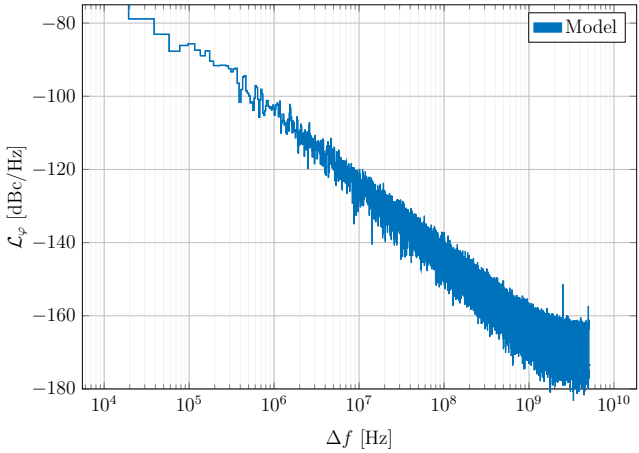
As explained in section 5.2.2 and section 2.1.6.3, the static DCO phase noise is modeled as additional jitter components on the center period. For a DCO with center frequency of 10 GHz, noise floor at −155 dBc/Hz and up-converted thermal noise of −115 dBc/Hz at 1 MHz offset, this results in RMS jitter contributions of $\sigma_{\Delta t} = 28.302\,\text{fs}$ and $\sigma_{\Delta T} = 1.7783\,\text{fs}$, respectively. The time resolution feasible in digital simulations is defined by the HDL standards, e.g. [IEE18]. The smallest possible delay is defined as 1 fs, which results in a quantization noise from rounding in a similar range as the jitter of the DCO:

$$\sigma_{\Delta T,q} = \frac{t_q}{\sqrt{12}} = 0.287\,\text{fs} \ . \tag{5.8}$$

Figure 5.28 shows an ADPLL model with additional noise source η_q due to the limited timing resolution of the simulator on the right. Depending on whether the DCO period duration for phase noise evaluation is saved within the DCO



(a) DCO phase noise from schematic simulation.



(b) DCO phase noise from model.

Figure 5.27: The DCO's output spectrum compared to the schematic's.

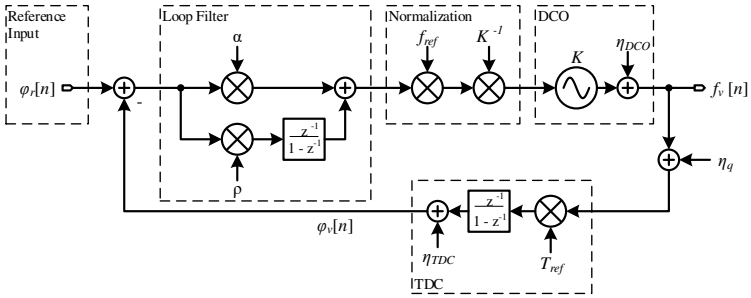


Figure 5.28: A z-domain model of an ADPLL with quantization noise from simulator resolution.

model – with full precision – or calculated externally in the testbench by looking at the switching events – with simulator resolution –, the quantization noise is shaped differently by the loop. In the first case, the noise is low-pass filtered in the same manner as the TDC noise. For the second option, it is high-pass filtered together with the DCO noise contribution. The resulting over all phase noise is plotted for both cases in figure 5.29.

To overcome this limitation and increase the effective resolution, it is proposed to introduce a scaling factor T_{scale} to increase the simulation time resolution in comparison to the system time, as already published in advance in [Mei+20]:

$$[1\,\mathrm{s}]_{sys} \equiv [1\,\mathrm{s} \cdot T_{scale}]_{sim} \ . \tag{5.9}$$

As the simulation frequencies are now scaled by $1/T_{scale}$, the jitter also needs to be scaled by T_{scale} for the noise contributions to stay equivalent:

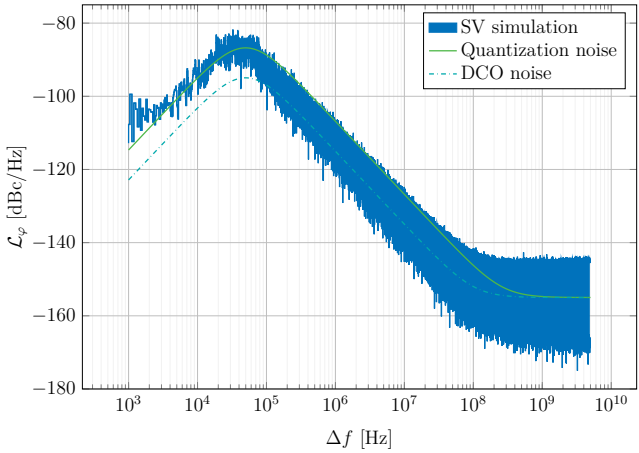
$$\sigma_{sim} = T_{scale} \cdot \sigma_{sys} \ . \tag{5.10}$$

Substituting (2.41) and (2.43) in (5.10), the spectral noise density of the thermal noise converts to:

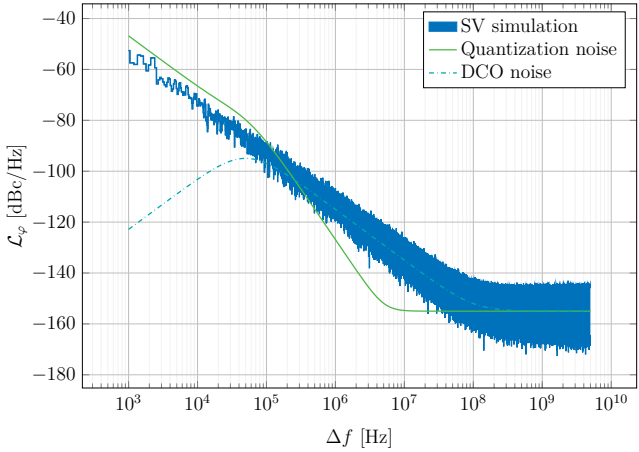
$$\mathcal{L}_{therm,sim} = \mathcal{L}_{therm,sys} \cdot T_{scale} \ , \tag{5.11}$$

$$\mathcal{L}_{upconv,sim} = \mathcal{L}_{upconv,sys} \cdot \frac{1}{T_{scale}} \ . \tag{5.12}$$

The quantization on the simulator is fixed, thus, effectively reducing on the



(a) Phase noise of free-running oscillator model calculated from transition timestamps.



(b) Phase noise of free-running oscillator model calculated from noisy frequency variable.

Figure 5.29: Phase noise simulations with highlighted effect of time resolution quantization.

```
var real t_scale = 1e3;

module DCO (input [LEN_OTW-1:0] OTW, output CKV);
[...]
  always @(CKV) begin
    [...]
    #(t_scale * half_period_n) CKV = !CKV;
  end
endmodule
```

Listing 5.1: SystemVerilog implementation of time-domain rescaling.

system scale:

$$\sigma_{\Delta T,q,sys} = \sigma_{\Delta T,q,sim} \cdot \frac{1}{T_{scale}}, \tag{5.13}$$

$$\mathcal{L}_{q,sys}[\text{dB}] = \mathcal{L}_{q,sim}[\text{dB}] - 10\log_{10}(T_{scale}). \tag{5.14}$$

The rescaling is implemented consistently in SystemVerilog by introduction a global variable **t_scale**. Thus, instead of rescaling all delays individually, scaling happens by multiplying all delay statements in models and blocks with the scaling factor. Adjusting the time base variable in the standard cell characterizations gives the same results for synthesized netlists. A possible implementation of the DCO model is shown in listing 5.1. The effects of DCO phase noise on the ADPLL after rescaling by **t_scale** = 10^3 are depicted in Figure 5.30. The quantization noise from simulator is reduced by 30 dB compared to Figure 5.29 and the resulting overall phase noise is dominated by the DCO noise contribution as expected.

5.2.3 Buffer and Divider

Due to the high switching activities, the output buffer and high speed dividers are expected to be a main source for SSN. Thus, the model needs to accurately reflect the timing and current demands of the circuit. The structure implemented on-chip is visualized in figure 5.31. It consists of a sine-to-square buffer followed by a chain of buffers and dividers. The down-divided outputs are fed into delay lines to align the phase of all four clock outputs. All instances are implemented differentially.

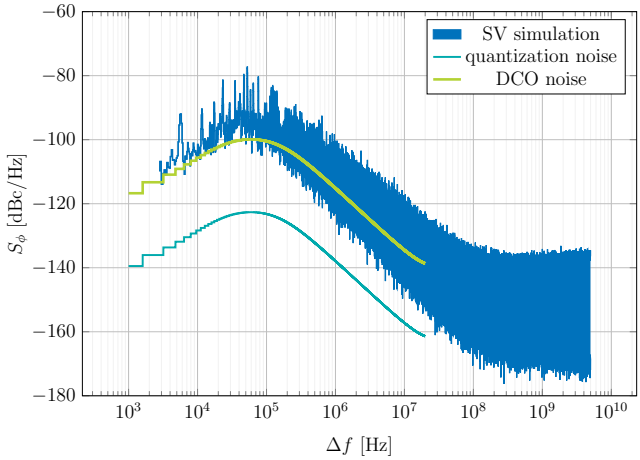


Figure 5.30: Effect of DCO phase noise on ADPLL output after rescaling.

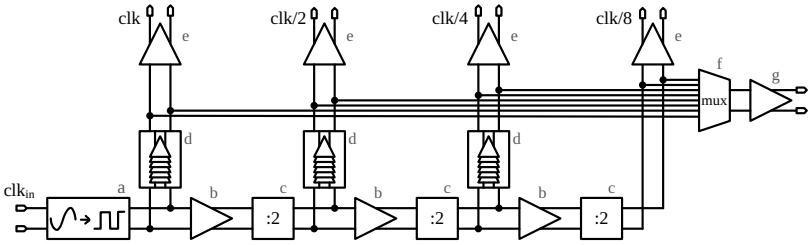


Figure 5.31: The output buffer and divider as circuit.

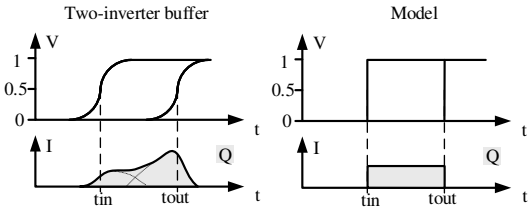


Figure 5.32: Buffer's current and voltage waveform.

5.2.3.1 Model Development

The circuit model is split into functional models for inversion, division and multiplexing as well as a variable standard buffer model to accommodate for the supply dependent delay and the aggregated current consumption of the functional blocks before. The buffer model can be parameterized using a static transport delay, the charge consumed by on switching event and an alpha factor for describing a linear delay variation dependent on the supply voltage.

Modeling the supply current waveform requires a trade-off between accuracy, simplicity and simulation speed. Normally, the currents drawn for rising and falling transitions are not the same, but since the whole structure is differential, the current for each event can be assumed equal. To further simplify the necessary computations, a zero-order approximation of the current waveform is assumed. The current is approximated by a rectangle whose integral equals the charge drawn by one switching impulse (see figure 5.32):

$$I_{DD} = \begin{cases} Q/(t_{out} - t_{in}) & \text{for } In \neq Out \\ 0 & \text{otherwise} \end{cases} . \tag{5.15}$$

Table 5.5 lists the parameters for modeling the buffer as indicated in figure 5.31.

5.2.3.2 Model Evaluation

To verify model vs. schematic (MvS) an ideal 10 GHz oscillator is connected as input source. The LDO model introduced in section 5.2.1 is used to provide the

Table 5.5: Output buffer and divider model parameters.

Cell	h	e	b	c	d	f	g	a
α[ps/V]	-2.543	8.998	7.182	1.248	6.327	17.67	5.948	28.74
d[ps]	15.33	49.69	16.17	11.99	2.271	17.02	58.81	77.34
q[µA · ps]	1.412	6.996	6.13	6.532	2.337	11.73	8.165	21.27

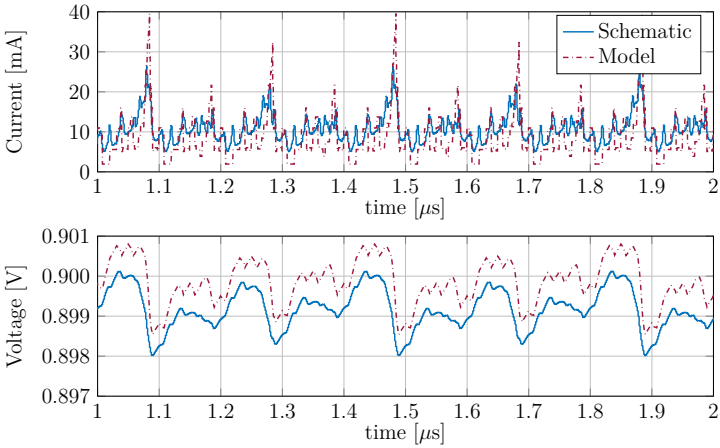


Figure 5.33: Transient current and voltage plot of the output buffer's supply.

supply. To estimate the model's performance, the transient supply current and voltage as well as the phase noise of the output buffer outputs are evaluated.

Figure 5.33 shows the simulated supply voltage and current for the setup. By visual inspection, it can be verified that the main characteristics are captured by the model. The low level abstraction allows for a coarse approximation of the supply currents to be sufficient since the sum of multiple buffer currents with the correct timing still give a good estimate of the total power consumption. The supply voltage waveform also follows the results from schematic simulation, although with an offset of ≈ 0.7 mV. This offset is caused by a 2 mA deviation of the mean current drawn from the buffers. Table 5.6 lists the main performance criteria of the MvS testcase. It can be seen that although there is a larger deviation on the currents due to neglected dummy capacitance and simplified waveforms, the voltage deviation is considerable smaller. Also, as explained in section 5.2.1.2, the LDO model is only accurate

Table 5.6: Comparison of the model's interaction.

	V_{mean}	V_{min}	V_{max}	σ_V	I_{mean}	I_{min}	I_{max}	σ_I
Schematic	0.8992	0.9001	0.8980	0.000497	10.54	26.53	4.98	3.53
Model	0.8999	0.9008	0.8985	0.000555	8.58	39.56	1.94	5.54

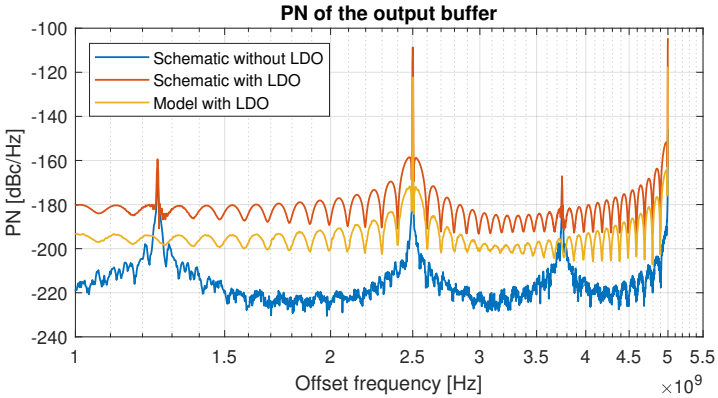


Figure 5.34: Phase noise of the output buffer compared.

for frequencies above 10 MHz but since only high frequencies are present in this usecase, the model shows a good accuracy.

As a second metric, the output buffer's phase noise is compared in Figure 5.34. First, by looking at the results for simulation of the schematic without LDO self-induced spurs at 1.25 GHz, 3.75 GHz and 5 GHz can be observed. They are comparably small with a level below −160 dBc/Hz. This is lower than the expected noise level from the oscillator and can thus be neglected. More important are the spurs at 2.5 GHz and 5 GHz introduced by usage of the LDO as power supply. They occur at 1/2 and 1/4 of the input frequency and are induced by SSN, which is well captured by the models.

5.2.4 TDC Modeling

The 2-D-Vernier-TDC implemention is shown in figure 5.35. It consists of

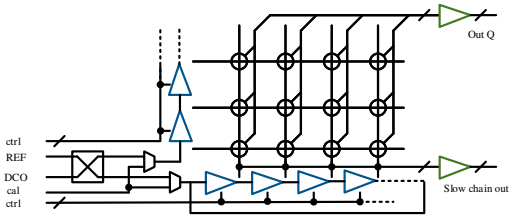


Figure 5.35: The TDC's implementation.

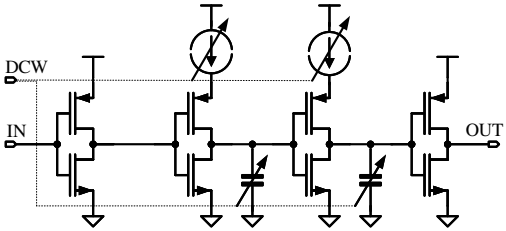


Figure 5.36: The delay cell.

two delay lines: the slow line with 13 delay elements and a nominal delay of 30 ps each, and the fast line with 11 delay elements and 33 ps nominal delay. A schematic of the delay cells is depicted in figure 5.36. The delay is tuneable by the discrete control word (DCW) input controlling the available current and the load capacitance.

In between the delay chains, the TDC is composed of a matrix of RS-flip-flops comparing the outputs of both delay lines and forming the 44 bit output thermometer code (Out Q). Prior to the delay chains a control block is implemented, consisting of a pulse steering block and various multiplexers to enable proper use and online calibration in between measurements.

5.2.4.1 Model Development

Similar to the model development in section 5.2.3, the dynamic supply current and accurate cell delays are considered the most important model characteristics. Thus, the same approach as for modeling buffers and dividers is chosen. To model the transport delay and power consumption of buffers and flip-flops, the previously introduced model is reused with different parameter setting

Table 5.7: Flip-flop model parameters.

Cell	Buffer SC	Buffer Mat
d [ps]	13.9	11.76
α [ps/V]	22.82	11.39
q [mA*ps]	4.72	1.57

Table 5.8: Delay cell model parameters.

parameters	C	K	V_{th}	G	H
values	0.0419	0.002637	0.2734	16.49	7.771

as given in table 5.7. Only for the unit delay cells used in the delay chains, the model needs to be extended to also cover the programmability of the cell. The propagation delay t_Δ and charge Q consumed by the cell mainly depend on the load capacitance $C_L(DCW_1)$ and available current I:

$$t_\Delta = 2 \cdot \frac{C_L(\mathrm{DCW}_1) \cdot V_{DD}}{2 \cdot I}\,, \tag{5.16}$$

$$I = K_n \frac{W(\mathrm{DCW}_2)}{L}(V_{DD} - V_{th})^2\,. \tag{5.17}$$

Substituting (5.17) in (5.16) and grouping constants together leads to :

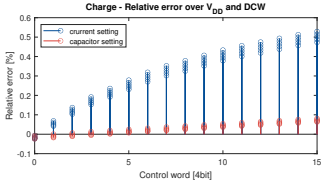
$$\begin{aligned} t_\Delta &= \frac{(C_L \cdot \mathrm{DCW}_1 + C_0) \cdot V_{DD}}{(K \cdot \mathrm{DCW}_2 + K_0)(V_{DD} - V_{th})^2} \\ &= H \cdot \frac{(\mathrm{DCW} \cdot C + 1) \cdot V_{DD}}{(\mathrm{DCW}_2 \cdot K + 1)(V_{DD} - V_{th})^2}\,, \end{aligned} \tag{5.18}$$

$$Q = G \cdot (\mathrm{DCW}_1 \cdot C + 1) \cdot V_{DD}\,. \tag{5.19}$$

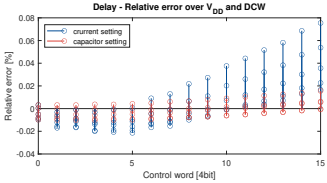
The parameters derived from schematic simulations and used for the models are found in table 5.8.

5.2.4.2 Model Evaluation

To analyze the model performance, the modeled delay and charge of the delay cell is compared to results from schematic simulation. Figure 5.37 shows the relative error of both parameters for different DCW values. As can be



(a) Charge modelling error of the delay cell.



(b) Delay modelling error of the delay cell.

Figure 5.37: Relative error of delay cell parameters.

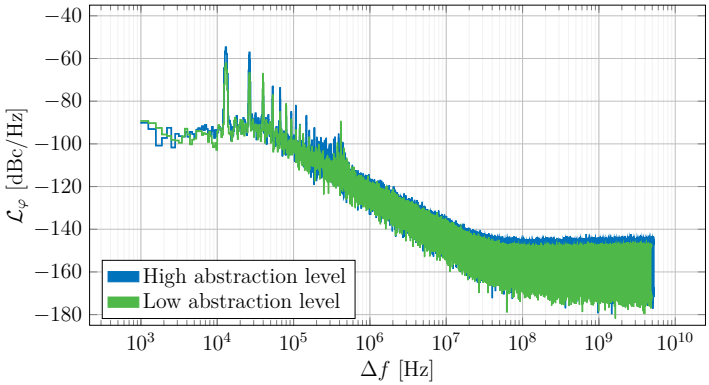


Figure 5.38: The phase noise spectrum without noise sources.

seen, the error is smaller than 0.1 % for all cases except the charge variation caused by tuning the available current. Since (5.19) is not dependent on the current, the model does not capture this change due to varying cross-currents. Nevertheless, the current variation is small compared to the change in load capacitance, so that the absolute error can be neglected.

5.2.5 System-Level Evaluation

5.2.5.1 Model Performance

The phase noise results for a simulation with close-to-integer FCW setting without any supply modulation effects are show in figure 5.38. The resulting

fractional spurs around 15 kHz offset and its multiples are clearly seen. The simulation result is compared to a high-level pure digital behavioral model that has no supply pins modeled. As can be seen, the resulting phase noise matches as expected. Slight differences in spur level can be accounted for by differences in TDC delay cells. While they are ideally matched for the high-level model, the low abstraction level models uses the values extracted from schematic simulation.

A performance comparison has been done using Xcelium's "-profile" option. The results are highlighted in figure 5.39. The penalty introduced due to the lower abstraction level used in the supply models is about a factor of 8. Further, the simulation performance for various use-cases considering supply modulation are plotted. As can be expected, a further degradation in simulation speed can be observed due to the additional events that need to be processed.

With the supply as the most active net in the circuit, further optimization of its processing promises huge performance benefits. As the supply net only connects to one voltage driver, the LDO, and multiple current loads, a simplification of the net resolution function is possible. Instead of the electrical equivalent net modeling as described in section 3.2.2, a striped down version, named IVnet is used. It provides all loads with the supply voltage generated by the driving source while the load currents are simply summed up. The translation of the currents to a voltage drop on the net are shifted into the LDO model and only updated whenever the LDO voltage is newly calculated. Compared to the full EEnet implementation, this achieves a performance gain by a factor of approximately 12.

5.2.5.2 Supply Coupling Path Analysis

To analyze possible cross-coupling paths over supply lines the effects of the involved sub-circuits are investigated individually. First only the divider and output buffer stage is considered as an aggressor load. Looking at the results in figure 5.40, it can be seen that the divider hardly affects the other circuits. Only the additional spurs at 2.5 GHz and 5 GHz already explained in section 5.2.3 are added.

However, by looking at figure 5.41, which also considers the TDC supply current, several changes in the output spectrum can be identified. First, the fractional spurs, as seen in the green curve, are ~10 dBc/Hz larger, compared to figure 5.38. This indicates additional non-linearity in the TDC induced by

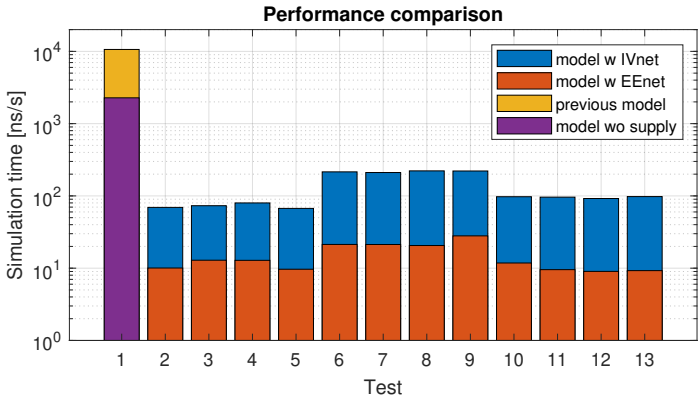


Figure 5.39: Performance of supply net implementations compared.

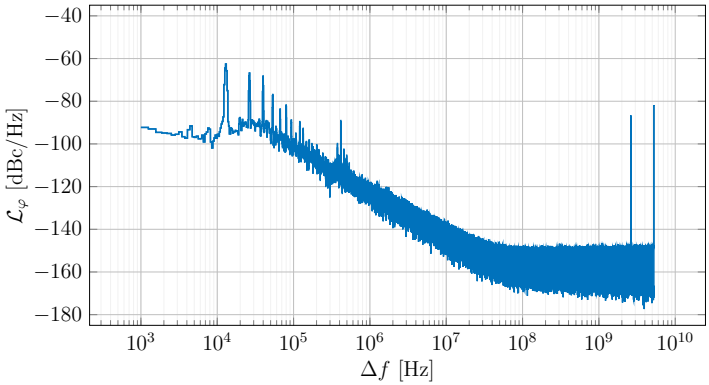


Figure 5.40: Output phase noise considering variation of supply voltage caused by the divider.

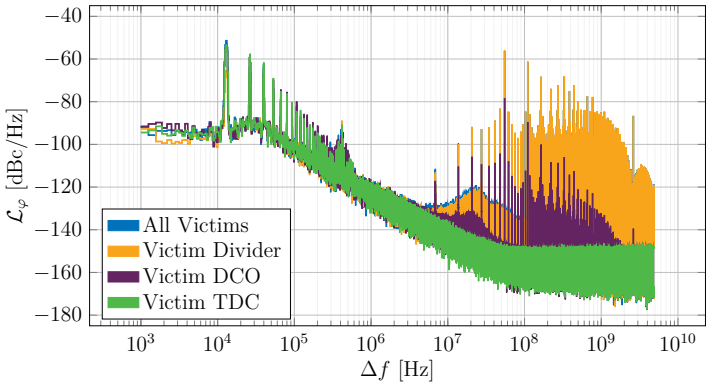


Figure 5.41: Output phase noise considering TDC and divider as current loads.

its own current consumption. Secondly, the overall phase noise in the region starting at 10 MHz is elevated. This can clearly be attributed to the divider by comparing the plots for TDC (green), DCO (purple) and divider (orange). Last, additional spurs at 6.8 MHz and multiples occur due to modulation of the DCO and dividers by the TDC. Due to the TDC's periodical recalibration its input activity also contains other frequency components than that caused by DCO and reference clock. The input spectrum of the TDC is plotted in figure 5.42. As can be seen, it already contains the spur at 6.8 MHz, which can then propagate over the supply to DCO and divider.

5.2.5.3 Path Sensitivity Analysis

The previous analysis only considered a lumped supply net, where all subcircuits are directly connected to one node. While this approach provides a smaller netlist, it is not able to address the effects of local decoupling capacitors placed close to the relevant designs. To investigate how much local decoupling can dampen the crosstalk the supply net is extended to a star topology with series resistance in between subcircuit and LDO and an additional local capacitance on the circuits side. The testbench setup is shown in figure 5.43.

Multiple simulations, sweeping resistance and capacitance value are run. Figures 5.44a and 5.44b visualize the effect of varying decoupling capacitance of TDC and divider, respectively. As can be seen, the fractional spurs at low frequencies are largely unaffected by the decoupling. A good decoupling on

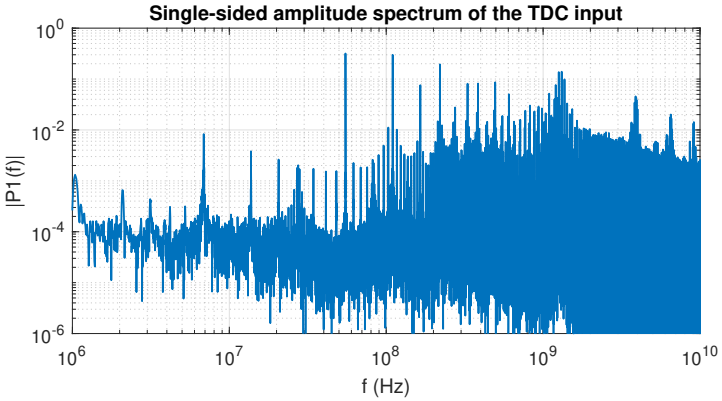


Figure 5.42: Spectrum of the TDC input signal.

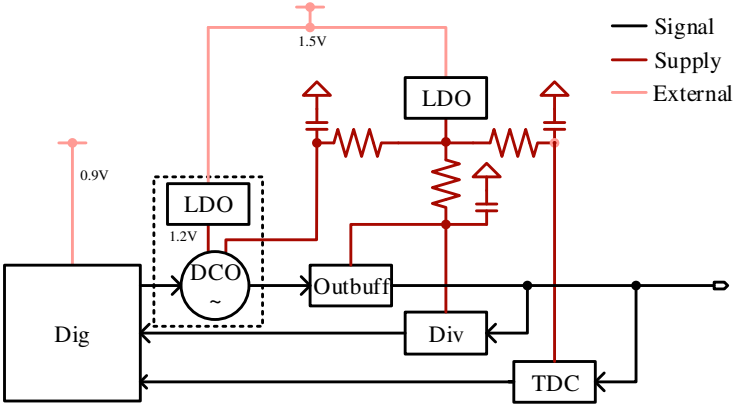
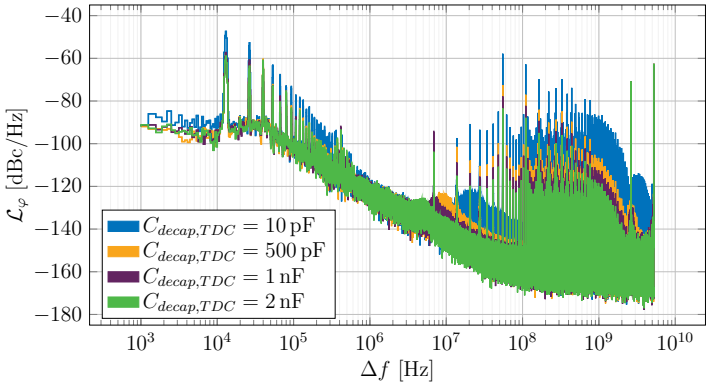
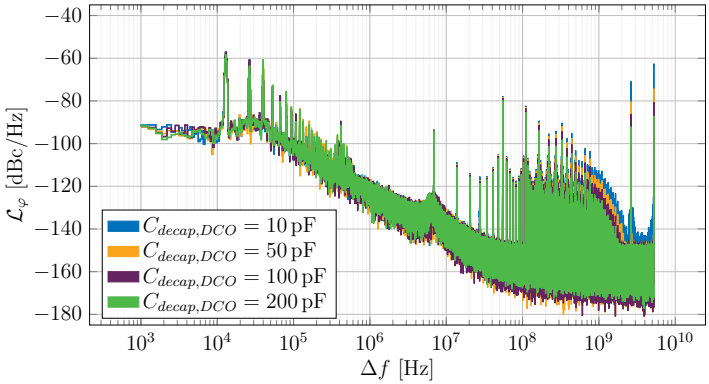


Figure 5.43: Configuration to analyze decoupling effectiveness of ADPLL.



(a) TDC decoupling sweep, with $C_{decap,DCO} = 10\,\text{pF}$.



(b) Divider decoupling sweep, with $C_{decap,TDC} = 1\,\text{nF}$.

Figure 5.44: Output phase noise simulation of the ADPLL for different decoupling parameters.

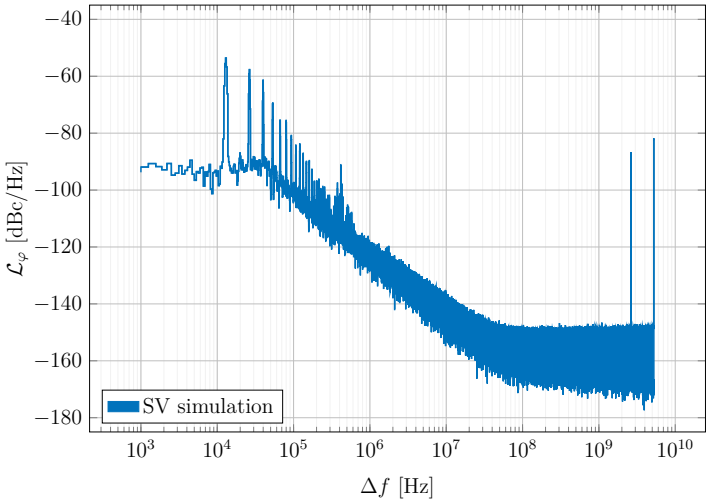


Figure 5.45: Output phase noise of the ADPLL with separate LDOs for TDC and DCO/Divider.

TDC side can reduce them to the level without any supply voltage (VDD) modulation present. Nevertheless, they need to be addressed separately by the spur reduction mechanisms and will be discussed in the following section 5.2.5.4. The TDC decoupling is also able to reduce the elevated noise floor in the output phase spectrum. The spurs introduced due to the TDC input spectrum, however, are still present event with large decoupling capacitor of 2 nF. Similarly, a decoupling on the divider side reduces its spurs in the DCO output phase spectrum but does not affect any spurs coming from the TDC side.

To eliminate the spurs and noise introduced by the TDC to the LDO, another feasible solution is to provide an own LDO for each block. Simulation results of the output phase noise are shown in figure 5.45. As expected the output spectrum resembles the results from the simulation considering the TDC as only victim with the additional spurs at $f_{DCO}/2$ and $f_{DCO}/4$ from the divider.

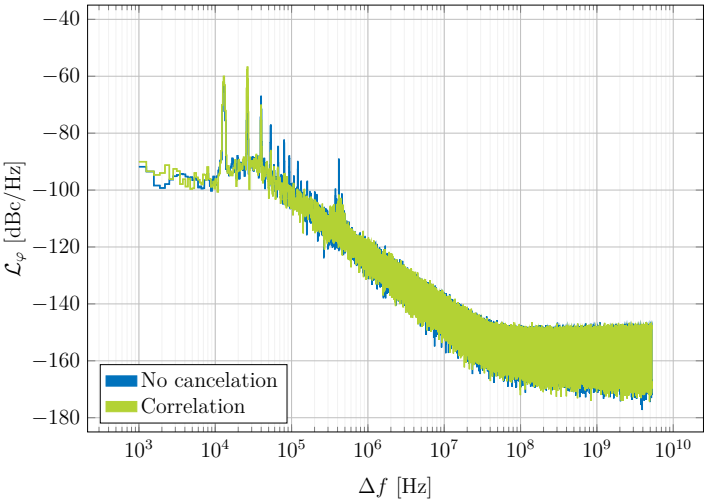


Figure 5.46: Output phase noise for spur cancellation through correlation with ideal VDD.

5.2.5.4 Robustness of Spur Cancellation Mechanisms

Separating the analog circuits in their own supply domain can reduce spurs due to crosstalk drastically. Nevertheless, the fractional spurs remain since they are caused inherently by the operating mode of the PLL, when used with a close-to-integer setting.

The simulation results for spur reduction using the correlation algorithm introduced in section 4.2.4.5 with ideal supply conditions are shown in figure 5.46. A good suppression for higher frequency spurs is seen. Only the lowest frequencies corresponding to one LSB offset from integer FCW frequency with 2 additional harmonics remain. These are probably caused by the fact that the TDC step size does not align with the DCO period. Thus, the last step has different size than the previous ones and is not canceled properly, causing the remaining periodic phase disturbance. Further some noise around 400 kHz is visible. It aligns with the time needed to sweep on TDC step with this fraction FCW setting.

The effectiveness of the spur reduction depends on the accuracy of the estimated metastability probability function of the flip-flops in the TDC, which is gained by averaging the periodic phase error over multiple periods. Results

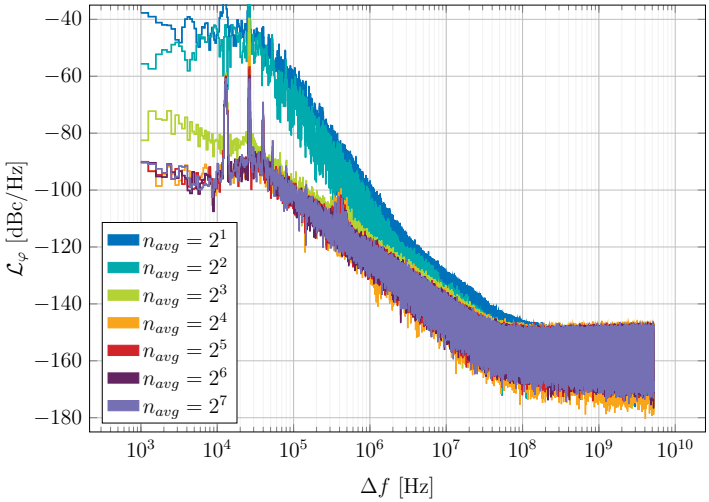


Figure 5.47: Spur cancellation comparison for different averaging lengths.

for different averaging lengths are shown in figure 5.47 As can be seen, for too small n_{avg} the algorithm only adds additional noise from the previous periods to the phase error, while averaging longer than 2^4 periods does only offer limited improvement.

Running the same simulation with a modulated supply shows reappearing fractional spurs, as seen in figure 5.48. The varying supply adds non-linearity to the TDC that is not considered by the algorithm. The TDC step sizes vary from step to step so that the periodic approximation fails. Thus, for this algorithm to work, a highly linear TDC is necessary.

Dithering the TDC reference input is the second approach to randomize the periodic phase error disturbances. Simulation results are shown in figure 5.49. The dithering and correction in digital afterwards is able to reduce spurs by approximately 15 dB. The lowest fractional spur, however, does not diminish completely. Further, spur reduction needs a stronger scrambling of the TDC input, which can be achieved by a delay line with more output taps. As can be seen a small new spur is added around 150 kHz, which corresponds to the sweep time of $1/T_{LSB,dither}$. Some variation due to mismatch is of advantage here, as it spreads the spur energy in the spectrum.

Simulation with VDD modulation shows a similar picture seen in figure 5.50. While the absolute values are a bit higher, a good suppression is still achieved.

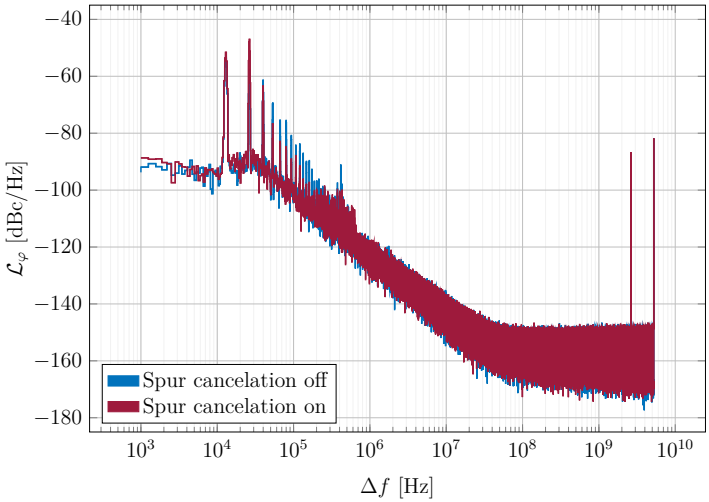


Figure 5.48: Spur cancellation effectiveness with supply modulation.

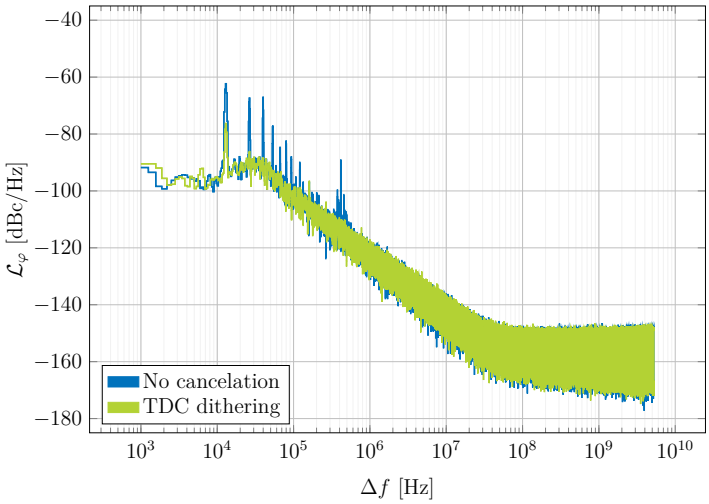


Figure 5.49: Output phase noise for spur cancellation through TDC input dithering with ideal VDD.

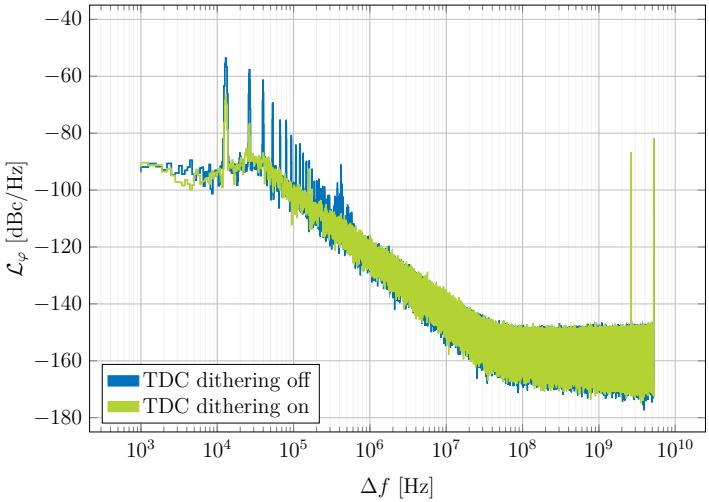


Figure 5.50: Output phase noise for spur cancellation through TDC input dithering with modulated VDD.

CHAPTER 6

CONCLUSION

6.1 Summary

The high demand for low-power electronics has favored SoC designs in small technology nodes. Digital-centric designs do have an advantage here, as they scale better with the shrinking feature size than their analog counterparts. However, the growing complexity of the systems, with ever-increasing interaction between subsystems, in addition to new challenges and growing importance of low-level error sources on the overall system performance drives the pre-tapeout verification effort.

This thesis address this increasing verification and design effort to ensure first-time-right production. An efficient simulation method for co-simulation of analog, RF circuits and digital system parts is needed. This is addressed by ensuring a use-case tailored signal representation in a fast, true event-driven simulation and modeling environment. A generic and extensible modeling framework is developed, based on SystemVerilog and its DPI to C++. While SystemVerilog ensures a good integration in state-of-the-art design flows, the C++ connection allows implementing signal description and processing using existing optimized libraries.

A suitable subset of different signal types needed in complex RF SoCs is developed and applied to verify a multi-band, multi-standard transceiver chip. The developed framework enables the systematic investigation on low-level effects of supply modulation in the $\Delta\Sigma$-ADC subsystem. Thus, it is possible to find root causes for performance degradation and spurs of the demodulated spectrum in pre-tapeout simulation by considering the complete mixed-signal system. These were otherwise only discovered in the lab during measurements requiring a re-spin to fix them in a product.

The simulation methods and parameterized model building approach is further extended to aid the design process of an ADPLL. Applied in a top-down procedure, the parameterized models for analog subblocks allow to be refined

during development process as more information becomes available. Thus, this thesis is able to define a suitable power domain partitioning to ensure low phase noise and spur level PLL operation.

The developed fractional ADPLL inherently produces fractional spurs for close-to-integer frequency setting. While a multitude of approaches to limit them exist in literature, two approaches are investigated further. Based on this, a digital only phase error prediction and correction algorithm is newly developed and compared to a reference path dithering approach. Both implementations promise comparable performance in a noise-free simulation environment. However, considering power supply noise induced non-linearity in the fractional TDC measurement discloses a weakness in the purely digital approach. It cannot compensate for the varying step size induced by the non-linearity. The dithering method shows a more robust behavior in such a noisy environment.

6.2 Outlook and Open Research Topics

The application to the pre-tapeout verification of a complex multi-band, multi-standard transceiver as well as the development of an ADPLL subcircuit highlight the feasibility and large potential of a versatile simulation framework and the modeling approaches developed in this thesis.

However, the work also shows that the integration in the design process can be further optimized to speed up the model development and move the verification effort to an eralier design stage. Two possible practices come to mind. Either, the library of available blocks with standard operations needs to be systematically extended to allow a faster model build up by combining and connecting them to build pin equivalent representation for analog schematic views. The other way forward focuses on a more fully and versatile signal API including operations on the signal class to be available in SystemVerilog. This allows the verification engineer to stay in their current, familiar environment and eases integration in industry workflows. While the first approach promises an overall faster model development process once truly established, the second allows to quickly benefit from the advanced signal modeling capabilities.

Further, the developed ADPLL needs validation on silicon. The investigations done in this thesis are based on simulations from schematics, partly considering parasitic effects. The system-level investigation enabled by the developed

methods already provides a deep insight in expected performance. Nevertheless, the full picture is only available in the final hardware, but fabrication and measurement was beyond the scope of this thesis. Especially, the performance of spur canceling algorithms needs to be further investigated. Extending the pure digital approach to monitoring and incorporating non-linearity of the fractional TDC measurement promises a versatile and low-cost solution to boost overall PLL performance.

BIBLIOGRAPHY

[Acu+90] E.L. Acuna, J.P. Dervenis, A.J. Pagones, F.L. Yang, and R.A. Saleh. "Simulation techniques for mixed analog/digital circuits". In: *IEEE Journal of Solid-State Circuits* 25.2 (1990), pp. 353–363. DOI: 10.1109/4.52156.

[Ala14] Morteza Alavi. "All-Digital I/Q RF-DAC". PhD thesis. TU Delft, 2014.

[All02] Douglas R. Allen Phillip E.; Holberg. *CMOS Analog Circuit Design*. Oxford University Press, 2002.

[Ata+13] A. Atac, L. Liao, Y. Wang, M. Schleyer, Y. Zhang, R. Wunderlich, and S. Heinen. "A 1.7mW quadrature bandpass Delta Sigma ADC with 1MHz BW and 60dB DR at 1MHz IF". In: *2013 IEEE International Symposium on Circuits and Systems (ISCAS2013)*. May 2013, pp. 1039–1042. DOI: 10.1109/ISCAS.2013.6572027.

[Bae+13] B. Bae, Y. Shim, K. Koo, J. Cho, J. S. Pak, and J. Kim. "Modeling and Measurement of Power Supply Noise Effects on an Analog-to-Digital Converter Based on a Chip-PCB Hierarchical Power Distribution Network Analysis". In: *IEEE Transactions on Electromagnetic Compatibility* 55.6 (Dec. 2013), pp. 1260–1270. ISSN: 0018-9375. DOI: 10.1109/TEMC.2013.2250506.

[Bar57] R. G. Baron. "The Vernier Time-Measuring Technique". In: *Proceedings of the IRE* 45.1 (Jan. 1957), pp. 21–30. ISSN: 0096-8390. DOI: 10.1109/JRPROC.1957.278252.

[Ber12] Janick Bergeron. *Writing testbenches: functional verification of HDL models*. Springer Science & Business Media, 2012.

[BCB18] Gabriel Bonteanu, Arcadie Cracan, and Radu Gabriel Bozomitu. "A Study on the Gm Based Current Mode Capacitance Multipliers Implementation". In: *2018 IEEE 24th International Symposium for Design and Technology in Electronic Packaging*. IEEE. 2018, pp. 205–208.

[Che09] Jesse E. Chen. "A modeling methodology for verifying functionality of a wireless chip". In: *2009 IEEE Behavioral Modeling and Simulation Workshop*. 2009, pp. 96–101. DOI: 10.1109/BMAS.2009.5338880.

[CC11] Pedro Miguel Cruz and Nuno Borges Carvalho. "Enhanced architecture to increase the dynamic range of SDR receivers". In: *2011 IEEE Radio and Wireless Symposium*. 2011, pp. 331–334. DOI: 10.1109/RWS.2011.5725481.

[Dav10] Jonathan B. David. "Radio receiver mixer model for event-driven simulators to support functional verification of RF-SOC wireless links". In: *2010 IEEE International Behavioral Modeling and Simulation Workshop*. 2010, pp. 42–47. DOI: 10.1109/BMAS.2010.6156596.

[EDA14] EDA-Stds. "Verilog-AMS Language Reference Manual". In: *Eda-Stds. Org* (2014).

[FM98] Dan FitzPatrick and Ira Miller. *Analog behavioral modeling with the Verilog-A language*. Springer Science & Business Media, 1998.

[FO00] P. Frey and D. O'Riordan. "Verilog-AMS: Mixed-signal simulation and cross domain connect modules". In: *Proceedings 2000 IEEE/ACM International Workshop on Behavioral Modeling and Simulation*. 2000, pp. 103–108. DOI: 10.1109/BMAS.2000.888372.

[Gib+16] Gian Piero Gibiino, Gustavo Avolio, Dominique M. M.-P. Schreurs, Alberto Santarelli, and Fabio Filicori. "A Three-Port Nonlinear Dynamic Behavioral Model for Supply-Modulated RF PAs". In: *IEEE Transactions on Microwave Theory and Techniques* 64.1 (2016), pp. 133–147. DOI: 10.1109/TMTT.2015.2504467.

[Gra+18] R. Granja, M. Santos, J. Guilherme, and N. Horta. "11.7b Time-To-Digital Converter with 0.82ps resolution in 130nm CMOS Technology". In: *2018 14th Conference on Ph.D. Research in Microelectronics and Electronics (PRIME)*. July 2018, pp. 29–32. DOI: 10.1109/PRIME.2018.8430374. URL: https://ieeexplore.ieee.org/document/8430374.

[HC16] Cheng-Ru Ho and Mike Shuo-Wei Chen. "A Digital PLL With Feedforward Multi-Tone Spur Cancellation Scheme Achieving lt;–73 dBc Fractional Spur and lt;–110 dBc Reference Spur in 65 nm CMOS". In: *IEEE Journal of Solid-State Circuits* 51.12 (2016), pp. 3216–3230. DOI: 10.1109/JSSC.2016.2596770.

[IEE08] IEEE. *IEEE Standard Definitions of Physical Quantities for Fundamental Frequency and Time Metrology—Random Instabilities*. Tech. rep. IEEE, 2008. DOI: 10.1109/IEEESTD.2008.4797525. URL: http://ieeexplore.ieee.org/document/4797525/ (visited on 02/16/2020).

[IEE13] IEEE. "IEEE Standard for SystemVerilog–Unified Hardware Design, Specification, and Verification Language". In: *IEEE Std 1800-2012 (Revision of IEEE Std 1800-2009)* (2013), pp. 1–1315. DOI: 10.1109/IEEESTD.2013.6469140.

[IEE01] IEEE. "IEEE Standard Verilog Hardware Description Language". In: *IEEE Std 1364-2001* (2001), pp. 1–792. DOI: 10.1109/IEEESTD.2001.93352.

[IEE99] IEEE. "IEEE Standard VHDL Analog and Mixed-Signal Extensions". In: *IEEE Std 1076.1-1999* (1999), pp. 1–314. DOI: 10.1109/IEEESTD.1999.90578.

[IEE09] IEEE. "IEEE Standard VHDL Language Reference Manual". In: *IEEE Std 1076-2008 (Revision of IEEE Std 1076-2002)* (2009), pp. 1–640. DOI: 10.1109/IEEESTD.2009.4772740.

[IEE18] IEEE. "IEEE Std 1800-2017 (Revision of IEEE Std 1800-2012) IEEE Standard for SystemVerilog—Unified Hardware Design, Specification, and Verification Language". en. In: (2018), p. 1315.

[Ier+08] Marcus van Ierssel, Hisakatsu Yamaguchi, Ali Sheikholeslami, Hirotaka Tamura, and William W. Walker. "Event-Driven Modeling of CDR Jitter Induced by Power-Supply Noise, Finite Decision-Circuit Bandwidth, and Channel ISI". In: *IEEE Transactions on Circuits and Systems I: Regular Papers* 55.5 (2008), pp. 1306–1315. DOI: 10.1109/TCSI.2008.916454.

[Jan+12] Ji-Eun Jang, Myeong-Jae Park, Dongyun Lee, and Jaeha Kim. "True event-driven simulation of analog/mixed-signal behaviors in SystemVerilog: A decision-feedback equalizing (DFE) receiver example". In: *Proceedings of the IEEE 2012 Custom Integrated Circuits Conference*. 2012, pp. 1–4. DOI: 10.1109/CICC.2012.6330558.

[Jun15] D. Juneja. "On Event Driven Modeling of Continuous Time Systems". In: *2015 28th International Conference on VLSI Design*. Jan. 2015, pp. 198–203. DOI: 10.1109/VLSID.2015.39.

[Kes12] Frank Kesel. *Modellierung von digitalen Systemen mit SystemC*. Oldenbourg Wissenschaftsverlag, 2012.

[Kes] Walt Kester. *Converting Oscillator Phase Noise to Time Jitter*. URL: https://www.analog.com/media/en/training-seminars/tutorials/MT-008.pdf (visited on 02/11/2020).

[Kim+10] Hyung-Jung Kim, Jin-Up Kim, Jae-Hyung Kim, Hongmei Wang, and In-Sung Lee. "The Design Method and Performance Analysis of RF Subsampling Frontend for SDR/CR Receivers". In: *IEEE Transactions on Industrial Electronics* 57.5 (2010), pp. 1518–1525. DOI: 10.1109/TIE.2009.2033491.

[KLD05] J.W. Kim, Y.-C. Lu, and R.W. Dutton. "Modeling and simulation of jitter in phase-locked loops due to substrate noise". In: *Proceedings of the 2005 IEEE International Behavioral Modeling and Simulation Workshop*. 2005, pp. 25–30. DOI: 10.1109/BMAS.2005.1518182.

[Kle90] L. Kleeman. "The jitter model for metastability and its application to redundant synchronizers". In: *IEEE Transactions on Computers* 39.7 (July 1990), pp. 930–942. ISSN: 1557-9956. DOI: 10.1109/12.55694.

[Kun97] Kundert. "Simulation methods for RF integrated circuits". In: *1997 Proceedings of IEEE International Conference on Computer Aided Design (ICCAD)*. 1997, pp. 752–765. DOI: 10.1109/ICCAD.1997.643622.

[Kun15] Ken Kundert. *Predicting the Phase Noise and Jitter of PLL-Based Frequency Synthesizers*. en. Oct. 2015. URL: https://designers-guide.org/analysis/PLLnoise+jitter.pdf (visited on 02/16/2020).

[Kun06] Ken Kundert. *The Designer's Guide to SPICE and SPECTRE®*. Springer Science & Business Media, 2006.

[Kuz+19] Nikolay Kuznetsov, Marat Yuldashev, Renat Yuldashev, Mikhail Blagov, Elena Kudryashova, Olga Kuznetsova, and Timur Mokaev. "Charge pump phase-locked loop with phase-frequency detector: closed form mathematical model". In: *arXiv:1901.01468 [eess]* (Mar. 2019). URL: http://arxiv.org/abs/1901.01468 (visited on 03/17/2020).

[Kwo+11] H. Kwon, J. Lee, J. Sim, and H. Park. "A high-gain wide-input-range time amplifier with an open-loop architecture and a gain equal to current bias ratio". In: *IEEE Asian Solid-State Circuits Conference 2011*. Nov. 2011, pp. 325–328. DOI: 10.1109/ASSCC.2011.6123579.

[Lam05] William K Lam. *Hardware Design Verification: Simulation and Formal Method-Based Approaches (Prentice Hall Modern Semiconductor Design Series)*. Prentice Hall PTR, 2005.

[LA08] M. Lee and A. A. Abidi. "A 9 b, 1.25 ps Resolution Coarse–Fine Time-to-Digital Converter in 90 nm CMOS that Amplifies a Time Residue". In: *IEEE Journal of Solid-State Circuits* 43.4 (Apr. 2008), pp. 769–777. ISSN: 0018-9200.

[Lie+14] Barend van Liempd, Jonathan Borremans, Ewout Martens, Sungwoo Cha, Hans Suys, Bob Verbruggen, and Jan Craninckx. "A 0.9 V 0.4–6 GHz Harmonic Recombination SDR Receiver in 28 nm CMOS With HR3/HR5 and IIP2 Calibration". In: *IEEE Journal of Solid-State Circuits* 49.8 (2014), pp. 1815–1826. DOI: 10.1109/JSSC.2014.2321148.

[MN98] Makoto Matsumoto and Takuji Nishimura. "Mersenne twister: a 623-dimensionally equidistributed uniform pseudo-random number generator". In: *ACM Transactions on Modeling and Computer Simulation (TOMACS)* 8.1 (1998), pp. 3–30.

[Mei+20] J. Meier, F. Maul, M. Scholl, C. Beyerstedt, R. Wunderlich, and S. Heinen. "Simulating Phase Noise in Multi-Gigahertz High Precision All-Digital Phase-Locked Loops". In: *27th IEEE International Conference on Electronics, Circuits and Systems (ICECS)*. 2020, pp. 1–4. DOI: 10.1109/ICECS49266.2020.9294980.

[Mei+19a] Jonas Meier, Christoph Beyerstedt, Fabian Speicher, Florian Menke, Ralf Wunderlich, and Stefan Heinen. "Event - Driven Modeling of Cross-Coupling in Phase-Locked Loops Through Supply Paths". In: *Kleinheubach Conference*. 2019, pp. 1–4.

[Mei+19b] Jonas Meier, Fabian Speicher, Christoph Beyerstedt, Tobias Saalfeld, Gregor Boronowsky, Ralf Wunderlich, and Stefan Heinen. "Modeling Power Supply Noise Effects for System-Level Simulation of ΔΣ -ADCs". In: *16th International Conference on Synthesis, Modeling, Analysis and Simulation Methods and Applications to Circuit Design (SMACD)*. 2019, pp. 265–268. DOI: 10.1109/SMACD.2019.8795288.

[MV07] Stefan Mendel and Christian Vogel. "A z-domain model and analysis of phase-domain all-digital phase-locked loops". In: *Norchip 2007*. Aalborg, Denmark: IEEE, Nov. 2007, pp. 1–6. ISBN: 978-1-4244-1516-8 978-1-4244-1517-5. DOI: 10.1109/NORCHP.2007.4481030. URL: http://ieeexplore.ieee.org/document/4481030/ (visited on 02/11/2020).

[MC91] B. Miller and R.J. Conley. "A multiple modulator fractional divider". In: *IEEE Transactions on Instrumentation and Measurement* 40.3 (June 1991), pp. 578–583. ISSN: 00189456. DOI: 10.1109/19.87022. URL: http://ieeexplore.ieee.org/document/87022/ (visited on 03/27/2020).

[MHN16] Bastian Mohr, Stefan Heinen, and Renato Negra. *System design and implementation of digital centric multi-standard transmitters*. Tech. rep. Lehrstuhl für Integrierte Analogschaltungen und Institut für Halbleitertechnik, 2016.

[Näh07] Felix Nähte. "Analysis and design of a signal-delta-modulator for RF-applications". MA thesis. Fakultät Technik und Informatik der Hochschule für Angewandte Wissenschaften Hamburg, Nov. 22, 2007.

[Ney10] Andreas August Neyer. *Nanoscale SoC-kompatibler FM-Radio Transmitter auf Basis einer vollständigen digitalen PLL*. 2010.

[OL07] Jens-Rainer Ohm and Hans Dieter Lüke. *Signalübertragung*. 10th ed. Springer, 2007.

[Opt12] Frank Opteynde. "A 40nm CMOS all-digital fractional-N synthesizer without requiring calibration". In: *2012 IEEE International Solid-State Circuits Conference*. Feb. 2012, pp. 346–347. DOI: 10.1109/ISSCC.2012.6177039.

[OG06] Maurits Ortmanns and Friedel Gerfers. *Continuous-Time Sigma-Delta A/D Conversion: Fundamentals, Performance Limits and Robust Implementations*. Springer, 2006.

[PB11] Nenad Pavlovic and Jos Bergervoet. "A 5.3GHz digital-to-time-converter-based fractional-N all-digital PLL". In: *2011 IEEE International Solid-State Circuits Conference*. 2011, pp. 54–56. DOI: 10.1109/ISSCC.2011.5746216.

[PC03] Jose Carlos Pedro and Nuno Borges Carvalho. *Intermodulation distortion in microwave and wireless circuits*. Artech House, 2003.

[RTL09] P. Raikwar, P. Trivedi, and J. D. Lal. "Effect of Timing Jitter on Sigma Delta ADC for SDR Mobile Receiver". In: *2009 Second International Conference on Emerging Trends in Engineering Technology*. Dec. 2009, pp. 1097–1103. DOI: 10.1109/ICETET.2009.87.

[Raz01] B. Razavi. *Design of Analog CMOS Integrated Circuits*. 2001.

[Riv+07] F. Rivet, Y. Deval, J.B. Begueret, D. Dallet, and D. Belot. "A Disruptive Software-Defined Radio Receiver Architecture Based on Sampled Analog Signal Processing". In: *2007 IEEE Radio Frequency Integrated Circuits (RFIC) Symposium*. 2007, pp. 197–200. DOI: 10.1109/RFIC.2007.380864.

[Saa+16] T. Saalfeld, A. Atac, L. Liao, R. Wunderlich, and S. Heinen. "A 2.3mW quadrature bandpass continuous-time ΔΣ modulator with reconfigurable quantizer". In: *12th Conference on Ph.D. Research in Microelectronics and Electronics (PRIME)*. June 2016, pp. 1–4. DOI: 10.1109/PRIME.2016.7519526.

[ST04] Richard Schreier and Gabor C. Themes. *Understanding Delta-Sigma Data Converters*. IEEE Press / Wiley, 2004.

[Seg+04] E. Segev, S. Goldshlager, H. Miller, O. Shua, O. Sher, and S. Greenberg. "Evaluating and comparing simulation verification vs. formal verification approach on block level design". In: *Proceedings of the 2004 11th IEEE International Conference on Electronics, Circuits and Systems, 2004. ICECS 2004.* 2004, pp. 515–518. DOI: 10.1109/ICECS.2004.1399731.

[Shi10] C.-J. Richard Shi. "Mixed-signal system-on-chip verification using a recursively-verifying-modeling (RVM) methodology". In: *Proceedings of 2010 IEEE International Symposium on Circuits and Systems*. 2010, pp. 1432–1435. DOI: 10.1109/ISCAS.2010.5537313.

[Sky12] Skyworks Solutions, Inc. *Integrated Phase Noise- AN256*. 2012. URL: https://www.silabs.com/documents/public/application-notes/AN256.pdf (visited on 03/19/2020).

[Spe08] Chris Spear. *SystemVerilog for verification: a guide to learning the testbench language features*. Springer Science & Business Media, 2008.

[Spe+18] F. Speicher, J. Meier, C. Beyerstedt, R. Wunderlich, and S. Heinen. "Advanced Modeling Methodology for Expedient RF SoC Verification and Performance Estimation". In: *2018 15th International Conference on Synthesis, Modeling, Analysis and Simulation Methods and Applications to Circuit Design (SMACD)*. July 2018, pp. 221–224. DOI: 10.1109/SMACD.2018.8434866.

[SB05] R.B. Staszewski and P.T. Balsara. "Phase-domain all-digital phase-locked loop". In: *IEEE Transactions on Circuits and Systems II: Express Briefs* 52.3 (Mar. 2005), pp. 159–163. ISSN: 1057-7130. DOI: 10.1109/TCSII.2004.842067. URL: http://ieeexplore.ieee.org/document/1406208/ (visited on 03/03/2020).

[SFB05] R.B. Staszewski, C. Fernando, and P.T. Balsara. "Event-driven Simulation and modeling of phase noise of an RF oscillator". In: *IEEE Transactions on Circuits and Systems I: Regular Papers* 52.4 (Apr. 2005), pp. 723–733. ISSN: 1057-7122. DOI: 10.1109/TCSI.2005.844236. URL: http://ieeexplore.ieee.org/document/1417066/ (visited on 02/11/2020).

[Sta+03] R.B. Staszewski, D. Leipold, K. Muhammad, and P.T. Balsara. "Digitally controlled oscillator (DCO)-based architecture for RF frequency synthesis in a deep-submicrometer CMOS process". In: *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing* 50.11 (Nov. 2003), pp. 815–828. ISSN: 1057-7130. DOI: 10.1109/TCSII.2003.819128. URL: http://ieeexplore.ieee.org/document/1246359/ (visited on 02/11/2020).

[SA11] Robert Bogdan Staszewski and Morteza S. Alavi. "Digital I/Q RF transmitter using time-division duplexing". In: *2011 IEEE International Symposium on Radio-Frequency Integration Technology*. 2011, pp. 165–168. DOI: 10.1109/RFIT.2011.6141744.

[SB06] Robert Bogdan Staszewski and Poras T. Balsara. *All-Digital Frequency Synthesizer in Deep-Submicron CMOS: Staszewski/All-Digital Frequency Synthesizer in Deep-Submicron CMOS*. Hoboken, NJ, USA: John Wiley & Sons, Inc., Aug. 2006. ISBN: 978-0-470-04195-6 978-0-471-77255-2. DOI: 10.1002/0470041951. URL: http://doi.wiley.com/10.1002/0470041951 (visited on 02/11/2020).

[Ste11] Sebastian Steinhorst. "Formal verification methodologies for nonlinear analog circuits". PhD thesis. Ph. D. dissertation, Goethe-University of Frankfurt am Main, 2011.

[SH08] Sebastian Steinhorst and Lars Hedrich. "Model Checking of Analog Systems using an Analog Specification Language". In: *2008 Design, Automation and Test in Europe*. 2008, pp. 324–329. DOI: 10.1109/DATE.2008.4484700.

[Sto13] Josef Stoer. *Einführung in die numerische Mathematik I: unter Berücksichtigung von Vorlesungen von FL Bauer*. Vol. 105. Springer-Verlag, 2013.

[SDF06] Stuart Sutherland, Simon Davidmann, and Peter Flake. *SystemVerilog for Design Second Edition: A Guide to Using SystemVerilog for Hardware Design and Modeling*. Springer Science & Business Media, 2006.

[Tem+10] Enrico Temporiti, Colin Weltin-Wu, Daniele Baldi, Marco Cusmai, and Francesco Svelto. "A 3.5 GHz Wideband ADPLL With Fractional Spur Suppression Through TDC Dithering and Feedforward Compensation". In: *IEEE Journal of Solid-State Circuits* 45.12 (2010), pp. 2723–2736. DOI: 10.1109/JSSC.2010.2077370.

[VLC10a] L. Vercesi, A. Liscidini, and R. Castello. "Two-Dimensions Vernier Time-to-Digital Converter". In: *IEEE Journal of Solid-State Circuits* 45.8 (Aug. 2010), pp. 1504–1512. ISSN: 0018-9200. DOI: 10.1109/JSSC.2010.2047435. URL: https://ieeexplore.ieee.org/document/5518485.

[VLC10b] Luca Vercesi, Antonio Liscidini, and Rinaldo Castello. "Two-Dimensions Vernier Time-to-Digital Converter". In: *IEEE Journal of Solid-State Circuits* 45.8 (Aug. 2010), pp. 1504–1512. ISSN: 0018-9200. DOI: 10.1109/JSSC.2010.2047435. URL: http://ieeexplore.ieee.org/document/5518485/ (visited on 02/11/2020).

[VOM15] Ron Vogelsong, Ahmed Hussein Osman, and Moustafa Mohamed. "Practical RNM with SystemVerilog". In: *CDNLive 2015*. 2015.

[Vos78] Richard F. Voss. "''1/f noise'' in music: Music from 1/f noise". In: *The Journal of the Acoustical Society of America* 63.1 (1978), p. 258. ISSN: 00014966. DOI: 10.1121/1.381721. URL: http://scitation.aip.org/content/asa/journal/jasa/63/1/10.1121/1.381721 (visited on 03/30/2020).

[WD17] H. Wang and F. F. Dai. "A 14-Bit, 1-ps resolution, two-step ring and 2D Vernier TDC in 130nm CMOS technology". In: *ESSCIRC 2017 - 43rd IEEE European Solid State Circuits Conference*. Sept. 2017, pp. 143–146. DOI: 10.1109/ESSCIRC.2017.8094546. URL: https://ieeexplore.ieee.org/document/8094546.

[WDW17] H. Wang, F. F. Dai, and H. Wang. "A 330μW 1.25ps 400fs-INL vernier time-to-digital converter with 2D reconfigurable spiral arbiter array and 2nd-order $\Delta\Sigma$ linearization". In: *2017 IEEE Custom Integrated Circuits Conference (CICC)*. Apr. 2017, pp. 1–4. DOI: 10.1109/CICC.2017.7993635. URL: https://ieeexplore.ieee.org/document/7993635.

[Wan+20] Lantao Wang, Marc Fassbender, Markus Scholl, Jonas Meier, Ralf Wunderlich, and Stefan Heinen. "A Low-noise Low-Dropout Regulator Using a 28-nm Technology". In: *2020 27th IEEE International Conference on Electronics, Circuits and Systems (ICECS)*. 2020, pp. 1–4. DOI: 10.1109/ICECS49266.2020.9294787.

[WM13] Xiaoqing Wang and Aaron Martin. "On-die supply-inducecd jitter behavioral modeling". In: *IEEE 22nd Conference on Electrical Performance of Electronic Packaging and Systems*. 2013, pp. 147–150. DOI: 10.1109/EPEPS.2013.6703486.

[WA14] Yifan Wang and Universitätsprofessor Dr-Ing Gerd Ascheid. *A hierarchical modeling and virtual prototyping methodology for functional verification of RF mixed-signal SoCs*. Verlag Dr. Hut, 2014.

[War09] Colin Warwick. "In a Nutshell: How SPICE works". In: *IEEE EMC Soc. Newslett* 22 (2009).

[WK08] Tingjun Wen and Tad Kwasniewski. "Phase Noise Simulation and Modeling of ADPLL by SystemVerilog". In: *2008 IEEE International Behavioral Modeling and Simulation Workshop*. San Jose, CA, USA: IEEE, Sept. 2008, pp. 29–34. ISBN: 978-1-4244-2896-0. DOI: 10.1109/BMAS.2008.4751235. URL: http://ieeexplore.ieee.org/document/4751235/ (visited on 02/11/2020).

[Xu+02] Jianjun Xu, M.C.E. Yagoub, Runtao Ding, and Qi-Jun Zhang. "Neural-based dynamic modeling of nonlinear microwave circuits". In: *IEEE Transactions on Microwave Theory and Techniques* 50.12 (2002), pp. 2769–2780. DOI: 10.1109/TMTT.2002.805192.

[Ypm95] Tjalling J Ypma. "Historical development of the Newton–Raphson method". In: *SIAM review* 37.4 (1995), pp. 531–551.

[YCS19] Huan Yu, Hemanth Chalamalasetty, and Madhavan Swaminathan. "Modeling of Voltage-Controlled Oscillators Including I/O Behavior Using Augmented Neural Networks". In: *IEEE Access* 7 (2019), pp. 38973–38982. DOI: 10.1109/ACCESS.2019.2905136.

[Yu+19] Huan Yu, Jaemin Shin, Tim Michalka, Mourad Larbi, and Madhavan Swaminathan. "Behavioral Modeling of Pre-emphasis Drivers Including Power Supply Noise Using Neural Networks". In: *IEEE 10th Latin American Symposium on Circuits Systems (LASCAS)*. 2019, pp. 37–40. DOI: 10.1109/LASCAS.2019.8667589.

[Zim11] Niklas Zimmermann. *Design and implementation of a broadband RF-DAC transmitter for wireless communications*. Verlag Dr. Hut, 2011.

[Zwo+95] M. Zwolinski, C. Garagate, Z. Mrcarica, T.J. Kazmierski, and A.D. Brown. "Anatomy of a simulation backplane". English. In: *IEE Proceedings - Computers and Digital Techniques* 142 (6 Nov. 1995), 377–385(8). ISSN: 1350-2387. URL: https://digital-library.theiet.org/content/journals/10.1049/ip-cdt_19952128.

APPENDIX A

SV SOURCE CODE

A.1 RXADC_DAC Module

```
`ifdef SIMULATION
import ams_spectral_dpi_pkg::*;
import ams_spectral_sv_user_pkg::*;
`endif

module RX_ADC_DAC_2bit ( OUTN, OUTP, AVDD, AVSS, CTRLIN, IREF, CAL );

`ifndef SIMULATION
typedef real signal_type;
typedef real signal_net;
`endif

//parameters
parameter bit probe_enable = 1;
string instance_name = $sformatf("%m");
parameter string digital_domain_in = "no_domain";
parameter real digital_level_in = 1.2;
parameter string digital_domain_out = "no_domain";
parameter real digital_level_out = 1.2;
parameter real IREF_value = 1.5e-6;
parameter real IREF_tolerance = 1.0;
parameter string supply_domain_AVDD = "VDD_A";
parameter real supply_level_AVDD = 1.2;
parameter string supply_domain_AVSS = "GND";
parameter real supply_level_AVSS = 0.0;
parameter bit psn_influence = 1;
parameter string psn_file = "./parameter_files/AIXRF_DAC_PSN.csv";
parameter string dc_out_file = "./parameter_files/AIXRF_DAC_DC.csv";

//in and output port declaration
input signal_net IREF; //type:ibias
input signal_net AVSS; //type:supply
output signal_net OUTP; //type:spectral
output signal_net OUTN; //type:spectral
input signal_net CTRLIN[2:0]; //type:logic
input signal_net CAL[3:0]; //type:logic
input signal_net AVDD; //type:supply

bit trigger_probe = 0;
bit triggered = 0;

//declaration of variables
chandle instance_pointer;
```

```
44  bit instance_initialized;
45
46  bit pin_IREF_initialized = 0;
47  chandle IREF_port_pointer = null;
48  chandle IREF_net_pointer = null;
49  chandle IREF_transitem_pointer = null;
50  signal_type IREF_s;
51  real IREF_iprobe;
52  assign IREF_s = IREF;
53
54  bit pin_AVDD_initialized = 0;
55  chandle AVDD_port_pointer = null;
56  chandle AVDD_net_pointer = null;
57  chandle AVDD_transitem_pointer = null;
58  signal_type AVDD_s;
59  real supply_probe_AVDD;
60  assign AVDD_s = AVDD;
61
62  bit pin_OUTP_initialized = 0;
63  chandle OUTP_port_pointer = null;
64  chandle OUTP_net_pointer = null;
65  chandle OUTP_transitem_pointer = null;
66  signal_type OUTP_s;
67  real OUTP_iprobe;
68  assign OUTP = OUTP_s;
69
70  bit pin_OUTN_initialized = 0;
71  chandle OUTN_port_pointer = null;
72  chandle OUTN_net_pointer = null;
73  chandle OUTN_transitem_pointer = null;
74  signal_type OUTN_s;
75  real OUTN_iprobe;
76  assign OUTN = OUTN_s;
77
78  bit pin_CTRLIN_initialized = 0;
79  chandle CTRLIN_port_pointer[2:0] =  '{null,null,null};
80  chandle CTRLIN_net_pointer[2:0] =  '{null,null,null};
81  chandle CTRLIN_transitem_pointer[2:0] =  '{null,null,null};
82  signal_type CTRLIN_s[2:0];
83  logic CTRLIN_probe[2:0];
84  assign CTRLIN_s = CTRLIN;
85
86  bit pin_CAL_initialized = 0;
87  chandle CAL_port_pointer[3:0] = '{null,null,null,null};
88  chandle CAL_net_pointer[3:0] = '{null,null,null,null};
89  chandle CAL_transitem_pointer[3:0] = '{null,null,null,null};
90  signal_type CAL_s[3:0];
91  logic CAL_probe[3:0];
92  assign CAL_s = CAL;
93
94  bit pin_AVSS_initialized = 0;
95  chandle AVSS_port_pointer = null;
96  chandle AVSS_net_pointer = null;
97  chandle AVSS_transitem_pointer = null;
98  signal_type AVSS_s;
99  real supply_probe_AVSS;
100 assign AVSS_s = AVSS;
101
102 string s_bit_position;
103
104 time t_clk;
```

```
105 `ifdef SIMULATION
106 initial begin
107
108   t_clk = 1s/96e6;
109   AMS_Base_set_global_loglevel("none");
110   instance_pointer = AMS_new("AMS_DAC_PSN");
111   AMS_Base_set_loglevel(instance_pointer,"fatal");
112   AMS_Object_set_name(instance_pointer, instance_name);
113   AMS_DAC_PSN_set_parameters(instance_pointer, psn_file, dc_out_file, "AixRF_ADC_DAC");
114   AMS_Block_Spectral_set_psn_influence(instance_pointer, psn_influence);
115   //if (display_flag == 1) begin $display("BLOCK:\tModule AMS_ADC_DAC %s created with pointer
      ↪  %d",instance_name,instance_pointer); end
116
117   //supply
118   AVDD_port_pointer =  AMS_new("AMS_Port_Supply");
119   AMS_Object_set_name(AVDD_port_pointer,"AVDD");
120   AMS_Port_set_direction_input(AVDD_port_pointer);
121   AMS_Port_Logic_set_parameters(AVDD_port_pointer, digital_domain_in,digital_level_in,1,0); // replace
      ↪  1,0 with bit_width,bit_position
122   AMS_Port_Supply_set_supply_domain(AVDD_port_pointer,supply_domain_AVDD);
123   AMS_Port_Supply_set_supply_level(AVDD_port_pointer,supply_level_AVDD);
124   AMS_Block_add_port(instance_pointer,AVDD_port_pointer);
125
126   AVSS_port_pointer =  AMS_new("AMS_Port_Supply");
127   AMS_Object_set_name(AVSS_port_pointer,"AVSS");
128   AMS_Port_set_direction_input(AVSS_port_pointer);
129   AMS_Port_Logic_set_parameters(AVSS_port_pointer, digital_domain_in,digital_level_in,1,0); // replace
      ↪  1,0 with bit_width,bit_position
130   AMS_Port_Supply_set_supply_domain(AVSS_port_pointer,supply_domain_AVSS);
131   AMS_Port_Supply_set_supply_level(AVSS_port_pointer,supply_level_AVSS);
132   AMS_Block_add_port(instance_pointer,AVSS_port_pointer);
133
134   //logic inputs
135   foreach (CTRLIN[i]) begin
136     CTRLIN_port_pointer[i] =  AMS_new("AMS_Port_Logic");
137     s_bit_position.itoa(i);
138     AMS_Object_set_name(CTRLIN_port_pointer[i],{"CTRLIN_", s_bit_position});
139     AMS_Port_set_direction_input(CTRLIN_port_pointer[i]);
140     AMS_Port_Logic_set_parameters(CTRLIN_port_pointer[i], digital_domain_in,digital_level_in,3,i); //
      ↪  replace 1,0 with bit_width,bit_position
141     AMS_Block_add_port(instance_pointer,CTRLIN_port_pointer[i]);
142   end
143
144   foreach( CAL[i] ) begin
145     CAL_port_pointer[i] =  AMS_new("AMS_Port_Logic");
146     s_bit_position.itoa(i);
147     AMS_Object_set_name(CAL_port_pointer[i],{"CAL_",s_bit_position});
148     AMS_Port_set_direction_input(CAL_port_pointer[i]);
149     AMS_Port_Logic_set_parameters(CAL_port_pointer[i], digital_domain_in,digital_level_in,4,i);
150     AMS_Block_add_port(instance_pointer,CAL_port_pointer[i]);
151   end
152
153   //bias inputs
154   IREF_port_pointer =  AMS_new("AMS_Port_Bias");
155   AMS_Object_set_name(IREF_port_pointer,"IREF");
156   AMS_Port_set_direction_input(IREF_port_pointer);
157   AMS_Port_Bias_set_bias_parameters(IREF_port_pointer,"current",IREF_value,IREF_tolerance);
158   AMS_Block_add_port(instance_pointer,IREF_port_pointer);
159
160   //spectral outputs
161   OUTP_port_pointer =  AMS_new("AMS_Port_Spectral");
```

```
162     AMS_Object_set_name(OUTP_port_pointer,"OUTP");
163     AMS_Port_set_direction_output(OUTP_port_pointer);
164     OUTP_transitem_pointer =  AMS_new("AMS_TransactionItem");
165     OUTP_s =  AMS_TransactionItem_get_id_by_ptr(OUTP_transitem_pointer);
166     AMS_TransactionItem_set_port_ptr(OUTP_transitem_pointer,OUTP_port_pointer);
167     AMS_Block_add_port(instance_pointer,OUTP_port_pointer);
168
169     OUTN_port_pointer =  AMS_new("AMS_Port_Spectral");
170     AMS_Object_set_name(OUTN_port_pointer,"OUTN");
171     AMS_Port_set_direction_output(OUTN_port_pointer);
172     OUTN_transitem_pointer =  AMS_new("AMS_TransactionItem");
173     OUTN_s =  AMS_TransactionItem_get_id_by_ptr(OUTN_transitem_pointer);
174     AMS_TransactionItem_set_port_ptr(OUTN_transitem_pointer,OUTN_port_pointer);
175     AMS_Block_add_port(instance_pointer,OUTN_port_pointer);
176
177     ////////////////// prepare block and ports ////////////////////;
178     AMS_Object_prepare(instance_pointer);
179   end
180
181
182   always @(AVDD_s iff pin_AVDD_initialized == 0) begin
183
184     if (AVDD_transitem_pointer == null) begin
185       AVDD_transitem_pointer = AMS_TransactionItem_get_ptr_by_id(AVDD_s);
186       if (AVDD_net_pointer == null) begin
187         AVDD_net_pointer = AMS_TransactionItem_get_net_ptr(AVDD_transitem_pointer);
188         AMS_Port_set_net(AVDD_port_pointer,AVDD_net_pointer);
189         AMS_Net_add_port(AVDD_net_pointer,AVDD_port_pointer);
190       end
191       if (AVDD_net_pointer != null) begin
192         pin_AVDD_initialized = 1;
193       end
194     end
195   end
196
197   always @(AVSS_s iff pin_AVSS_initialized == 0) begin
198
199     if (AVSS_transitem_pointer == null) begin
200       AVSS_transitem_pointer = AMS_TransactionItem_get_ptr_by_id(AVSS_s);
201       if (AVSS_net_pointer == null) begin
202         AVSS_net_pointer = AMS_TransactionItem_get_net_ptr(AVSS_transitem_pointer);
203         AMS_Port_set_net(AVSS_port_pointer,AVSS_net_pointer);
204         AMS_Net_add_port(AVSS_net_pointer,AVSS_port_pointer);
205       end
206       if (AVSS_net_pointer != null) begin
207         pin_AVSS_initialized = 1;
208       end
209     end
210   end
211
212   always @(IREF_s iff pin_IREF_initialized == 0) begin
213
214     if (IREF_transitem_pointer == null) begin
215       IREF_transitem_pointer = AMS_TransactionItem_get_ptr_by_id(IREF_s);
216       if (IREF_net_pointer == null) begin
217         IREF_net_pointer = AMS_TransactionItem_get_net_ptr(IREF_transitem_pointer);
218         AMS_Port_set_net(IREF_port_pointer,IREF_net_pointer);
219         AMS_Net_add_port(IREF_net_pointer,IREF_port_pointer);
220       end
221       if (IREF_net_pointer != null) begin
222         pin_IREF_initialized = 1;
```

```
223       end
224     end
225   end
226
227   always @(CTRLIN_s) begin
228
229     foreach (CTRLIN[i]) begin
230       if (CTRLIN_transitem_pointer[i] == null) begin
231         CTRLIN_transitem_pointer[i] = AMS_TransactionItem_get_ptr_by_id(CTRLIN[i]);
232         if (CTRLIN_net_pointer[i] == null) begin
233           CTRLIN_net_pointer[i] = AMS_TransactionItem_get_net_ptr(CTRLIN_transitem_pointer[i]);
234           AMS_Port_set_net(CTRLIN_port_pointer[i],CTRLIN_net_pointer[i]);
235           AMS_Net_add_port(CTRLIN_net_pointer[i],CTRLIN_port_pointer[i]);
236         end
237       end
238     pin_CTRLIN_initialized = 1;
239     end
240
241   end
242
243   always @(CAL_s) begin
244
245     foreach (CAL[i]) begin
246       if (CAL_transitem_pointer[i] == null) begin
247         CAL_transitem_pointer[i] = AMS_TransactionItem_get_ptr_by_id(CAL[i]);
248         if (CAL_net_pointer[i] == null) begin
249           CAL_net_pointer[i] = AMS_TransactionItem_get_net_ptr(CAL_transitem_pointer[i]);
250           AMS_Port_set_net(CAL_port_pointer[i],CAL_net_pointer[i]);
251           AMS_Net_add_port(CAL_net_pointer[i],CAL_port_pointer[i]);
252         end
253       end
254     pin_CAL_initialized = 1;
255     end
256
257   end
258
259   always @(IREF,AVDD,CTRLIN_s,CAL_s,AVSS) begin
260
261     if (display_flag == 1) begin $display("ModuleRX_ADC_DAC %s entered",instance_name); end
262     if (instance_initialized == 0) begin
263
264       if (OUTP_net_pointer == null) begin
265         OUTP_net_pointer = AMS_TransactionItem_get_net_ptr(OUTP_transitem_pointer);
266         if (OUTP_net_pointer != null) begin
267           AMS_Port_set_net(OUTP_port_pointer,OUTP_net_pointer);
268           pin_OUTP_initialized = 1;
269         end
270         else begin
271           pin_OUTP_initialized = 0;
272         end
273       end
274
275       if (OUTN_net_pointer == null) begin
276         OUTN_net_pointer = AMS_TransactionItem_get_net_ptr(OUTN_transitem_pointer);
277         if (OUTN_net_pointer != null) begin
278           AMS_Port_set_net(OUTN_port_pointer,OUTN_net_pointer);
279           pin_OUTN_initialized = 1;
280         end
281         else begin
282           pin_OUTN_initialized = 0;
283         end
```

```
284       end
285
286       if ((pin_IREF_initialized == 1) && (pin_AVDD_initialized == 1) && (pin_CAL_initialized == 1) &&
          ↪  (pin_OUTP_initialized == 1) && (pin_OUTN_initialized == 1) && (pin_CTRLIN_initialized == 1) &&
          ↪  (pin_AVSS_initialized == 1) && (AMS_Object_get_initialized(instance_pointer)==0)) begin
287         //AMS_Block_set_supply(instance_pointer,supply_pointer_avdd,supply_pointer_AVSS); // change here
            ↪  to correct supply pointers
288         AMS_Object_initialize(instance_pointer);
289         instance_initialized = AMS_Object_get_initialized(instance_pointer);
290         if (display_flag == 1 && instance_initialized == 1) begin $display("RX_ADC_DAC_2bit %s
            ↪  initialized!",instance_name); end
291
292
293       end
294     end
295     else begin
296
297       if (display_flag == 1) begin $display("RX_ADC_DAC_2bit %s activity!",instance_name); end
298       AMS_Block_evaluate(instance_pointer,$realtime*timescale);
299       trigger_probe <= ~ trigger_probe;
300
301       //OUTP_s = OUTP_s * -1;
302       //OUTN_s = OUTN_s * -1;
303
304       OUTP_s =  AMS_Port_get_transaction_id_number(OUTP_port_pointer);
305       OUTN_s =  AMS_Port_get_transaction_id_number(OUTN_port_pointer);
306       if (display_flag == 1) begin $display("RX_ADC_DAC_2bit %s evaluated!",instance_name); end
307       if (display_flag == 1) begin $display("RX_ADC_DAC_2bit realtime: %f = %f",$realtime,$time); end
308     end
309   end
310
311
312   always @(trigger_probe iff probe_enable == 1) begin
313
314     ////////////probes to see signal values in waveform window//////////////////////
315     IREF_iprobe = AMS_Port_Bias_get_current(IREF_port_pointer);
316
317     foreach (CTRLIN[i]) begin
318       CTRLIN_probe[i] = AMS_Port_Logic_get_logic_value(CTRLIN_port_pointer[i]);
319     end
320
321     foreach (CAL[i]) begin
322       CAL_probe[i] = AMS_Port_Logic_get_logic_value(CAL_port_pointer[i]);
323     end
324
325     supply_probe_AVDD = AMS_Port_Supply_get_supply_voltage(AVDD_port_pointer);
326     supply_probe_AVSS = AMS_Port_Supply_get_supply_voltage(AVSS_port_pointer);
327   end
328
329
330   always begin
331
332     #(t_clk)
333     OUTP_iprobe = AMS_Port_Spectral_get_voltage(OUTP_port_pointer, $realtime*timescale);
334     OUTN_iprobe = AMS_Port_Spectral_get_voltage(OUTN_port_pointer, $realtime*timescale);
335
336   end
337   `endif
338   endmodule
339
```

APPENDIX B

CURRICULUM VITAE

Name	Jonas Meier
Academic Degree	Master of Science (M. Sc.)
Date of Birth	18. May 1991
Place of Birth	Hamburg, Germany

Professional Experience

since 09/2021	ADPLL Design Engineer, *NXP Semiconductors GmbH*, Hamburg, Germany
12/2016 – 05/2021	Research Assistant / PhD Candidate, Chair of Integrated Analog Circuits and RF Systems (IAS), *RWTH Aachen University*, Aachen, Germany
10/2015 – 02/2016	Internship Design-for-Test Department, *Dialog Semiconductor*, Kirchheim (Teck), Germany
11/2013 – 09/2015	Student Job, Chair for Electrical Engineering and Computer Systems (EECS), *RWTH Aachen University*, Aachen, Germany
04/2012 – 12/2012	Student Job, Chair of Medical Engineering (mediTEC), *RWTH Aachen University*, Aachen, Germany

Education

10/2010 – 09/2013	Master of Science: Electrical Engineering, Information Technology and Computer Engineering, *RWTH Aachen University*, Aachen, Germany
10/2010 – 09/2013	Bachelor of Science: Electrical Engineering, Information Technology and Computer Engineering, *RWTH Aachen University*, Aachen, Germany
09/2001 – 07/2009	High School Diploma (A level) *Gymnasium Christianeum*, Hamburg, Germany

APPENDIX C

PUBLICATIONS

C.1 Peer-reviewed Journal Papers

C. Beyerstedt, J. Meier, F. Speicher, R. Wunderlich, and S. Heinen. "Analysis of the frequency conversion of spurious tones in frequency dividers and development of an event-driven model for system simulations". In: *Advances in Radio Science* 17 (2019), pp. 101–107. DOI: 10.5194/ars-17-101-2019. URL: https://ars.copernicus.org/articles/17/101/2019/

C.2 Peer-reviewed Conference Papers

Lantao Wang, Jonas Meier, Ralf Wunderlich, and Stefan Heinen. "A Comparative Study of Switchable Capacitor Structures for LC Oscillators in a 28-nm Technology". In: *2021 28th IEEE International Conference on Electronics, Circuits, and Systems (ICECS)*. 2021, pp. 1–4. DOI: 10.1109/ICECS53924.2021.9665608

J. Meier, F. Menke, L. Wang, T. Lauber, R. Wunderlich, and S. Heinen. "Modeling Power Supply Noise in RF SoCs". In: *International Conference on SMACD and 16th Conference on PRIME*. 2021, pp. 1–6

Lantao Wang, Adrian Arnold, Jonas Meier, Markus Scholl, Ralf Wunderlich, and Stefan Heinen. "A 55 MHz Integrated Crystal Oscillator with Chirp Injection Using a 28-nm Technology". In: *SMACD / PRIME 2021; International Conference on SMACD and 16th Conference on PRIME*. 2021, pp. 1–4

Christoph Beyerstedt, Jonas Meier, Fabian Speicher, Ralf Wunderlich, and Stefan Heinen. "An Event-Driven System-Level Noise Analysis Methodology for RF Systems". In: *2021 Design, Automation Test in Europe Conference Exhibition (DATE)*. 2021, pp. 380–385. DOI: 10.23919/DATE51398.2021.9474077

J. Meier, F. Maul, M. Scholl, C. Beyerstedt, R. Wunderlich, and S. Heinen. "Simulating Phase Noise in Multi-Gigahertz High Precision All-Digital Phase-Locked Loops". In: *27th IEEE International Conference on Electronics, Circuits and Systems (ICECS)*. 2020, pp. 1–4. DOI: 10.1109/ICECS49266.2020.9294980

C. Beyerstedt, J. Meier, F. Speicher, M. Scholl, D. Blase, R. Wunderlich, and S. Heinen. "A Fast and Accurate True Event-Driven Phase Locked Loop Model". In: *27th IEEE International Conference on Electronics, Circuits and Systems (ICECS)*. 2020, pp. 1–4. DOI: 10.1109/ICECS49266.2020.9294942

L. Wang, M. Fassbender, M. Scholl, J. Meier, R. Wunderlich, and S. Heinen. "A Low-Noise Low-Dropout Regulator Using a 28-nm Technology". In: *27th IEEE International Conference on Electronics, Circuits and Systems (ICECS)*. 2020, pp. 1–4. DOI: 10.1109/ICECS49266.2020.9294787

M. Scholl, T. Saalfeld, C. Beyerstedt, F. Speicher, J. Meier, M. Hanhart, L. Rolff, V. Bonehi, M. Schrey, R. Wunderlich, and S. Heinen. "A 32 MHz Crystal Oscillator with Fast Start-Up Using Dithered Injection and Negative Resistance Boost". In: *45th European Solid State Circuits Conference (ESSCIRC)*. 2019, pp. 49–52. DOI: 10.1109/ESSCIRC.2019.8902894

J. Meier, C. Beyerstedt, F. Speicher, F. Menke, R. Wunderlich, and S. Heinen. "Event-Driven Modeling of Cross-Coupling in Phase-Locked Loops Through Supply Paths". In: *2019 Kleinheubach Conference*. 2019, pp. 1–4

C. Beyerstedt, F. Speicher, J. Meier, R. Wunderlich, and S. Heinen. "Baseband Equivalent Modeling Approach for Analog Linear Transfer Functions in Event-driven Simulations". In: *16th International Conference on Synthesis, Modeling, Analysis and Simulation Methods and Applications to Circuit Design (SMACD)*. 2019, pp. 281–284. DOI: 10.1109/SMACD.2019.8795301

J. Meier, F. Speicher, C. Beyerstedt, T. Saalfeld, G. Boronowsky, R. Wunderlich, and S. Heinen. "Modeling Power Supply Noise Effects for System-Level Simulation of ΔΣ -ADCs". In: *16th International Conference on Synthesis, Modeling, Analysis and Simulation Methods and Applications to Circuit Design (SMACD)*. 2019, pp. 265–268. DOI: 10.1109/SMACD.2019.8795288

F. Speicher, J. Meier, C. Beyerstedt, R. Wunderlich, and S. Heinen. "Advanced Modeling Methodology for Expedient RF SoC Verification and

Performance Estimation". In: *15th International Conference on Synthesis, Modeling, Analysis and Simulation Methods and Applications to Circuit Design (SMACD)*. 2018, pp. 221–224. DOI: 10.1109/SMACD.2018.8434866

F. Speicher, J. Meier, S. Aghaie, R. Wunderlich, and S. Heinen. "AMS Verification Methodology Regarding Supply Modulation in RF SoCs Induced by Digital Standard Cells". In: *Design, Automation and Test in Europe Conference (DATE)*. 2018, pp. 633–636. DOI: 10.23919/DATE.2018.8342087