CORA Cork Open Research Archive
Cartlann Taighde Oscailte Chorcaí

Title	Analog mixed signal IC design for magnetic tracking in surgery: low area clocking for CTΔΣM ADCs for in-vivo sensing
Authors	Ferro, Alessandro
Publication date	2024
Original Citation	Ferro, A. 2024. Analog mixed signal IC design for magnetic tracking in surgery: low area clocking for CTΔΣM ADCs for in-vivo sensing. MSc Thesis, University College Cork.
Type of publication	Masters thesis (Research)
Rights	© 2024, Alessandro Ferro. - https://creativecommons.org/licenses/by-sa/4.0/
Download date	2026-11-06 19:29:10
Item downloaded from	https://hdl.handle.net/10468/16466



UCC
University College Cork, Ireland
Coláiste na hOllscoile Corcaigh





University College of Cork

School of Engineering & Tyndall National Institute

Analog Mixed Signal IC Design for Magnetic Tracking in Surgery

Low area clocking for CTΔΣM ADCs for in-vivo sensing

Supervisor: *Prof. Dr.* Pádraig Cantillon-Murphy

Co-Supervisor: *Dr.* Daniel O'Hare

Student: Alessandro Ferro [122108521]

academic year 2023/24

Abstract

The application for this work is an *in-vivo sensor system* for capturing **magnetic signals** used for tracking surgical instruments such as catheters in the body during **minimally invasive surgical procedures**. With the ascent of **image-guided interventions**, the precision in determining instrument pose has become paramount. The proposed system leverages *low-frequency electromagnetic fields* for magnetic tracking, ensuring non-ionising, line-of-sight-free measurements with *millimetre-scale accuracy*.

The miniaturisation demands of **electromagnetic sensors**, crucial to this endeavour, necessitate dimensions less than 0.5 mm in diameter, with the entire system fitting within an area of 1 mm x 0.5 mm. Continuous Time Delta Sigma (CT$\Delta\Sigma$M) ADCs emerge as a promising solution given their power efficiency and reduced requirements on peripheral circuits. These ADCs boast advantages like built-in *anti-alias filtering* and simplified input buffer requirements. However, their susceptibility to **clock jitter**, especially in single-bit versions, presents challenges.

This research introduces an original MATLAB and Simulink model for the CT$\Delta\Sigma$M. The model initially accounted for clock jitter employing a *white noise spectrum*. However, subsequent analysis unearthed a profound limitation: while **FIR filters** within the CT$\Delta\Sigma$M reduced periodic jitter's impact, they were vulnerable to non-white noise spectrums. Although present in CT$\Delta\Sigma$M literature, this significant observation was previously uncharted for the signal bandwidth applications akin to this work.

To address this, the research delves into creating a comprehensive Phase noise **spectrum model** using MATLAB. This model elucidates the profound limitations imposed on the ADC's in-band noise by the clock source's phase noise spectrum, a revelation that reshaped our understanding of the CT$\Delta\Sigma$M's constraints. Originally, a detailed schematic design for the clock source was undertaken with the primary objective of optimising *area* and *power consumption*, relying on a *Resistor Capacitor based clock source* for the Frequency-Locked Loop (FLL). The prevalent belief was that the typical phase noise spectrum of clock sources would have a negligible influence on the CT$\Delta\Sigma$M ADC. However, post-chip measurement analysis dispelled this notion. This research discovered that the clock source's phase noise spectrum profoundly affected the CT$\Delta\Sigma$M ADC's performance, highlighting an overlooked yet critical interplay.

Declaration

I hereby declare that the dissertation entitled "**Analog Mixed Signal IC Design for Magnetic Tracking in Surgery: Low Area Clocking for CT$\Delta\Sigma$ ADCs for In-Vivo Sensing**", being submitted to the **School of Engineering at University College Cork**, is an original piece of work conducted in fulfilment of the requirements for the award of the Degree of Master of Engineering Science. This work was entirely conducted at **MCCI** (Microelectronic Circuits Centre Ireland), a department of **Tyndall National Institute**, under the supervision of Professor *Pádraig Cantillon-Murphy* and Dr. *Daniel O'Hare*.

I affirm that the research and findings presented in this thesis are my own and contain no material which, to a substantial extent, has been submitted to any other university or institution for the award of any degree or diploma, except where due acknowledgement is explicitly made within the text.



Alessandro Ferro

Acknowledgment

I want to express my most profound appreciation to all those who allowed me to complete this thesis. Special gratitude is given to our project manager, **Professor Pádraig Cantillon-Murphy**, whose contribution in stimulating suggestions and encouragement helped me to coordinate my project, especially in writing this thesis.

I thank **Dr. Daniel O'Hare** for his continuous support during my study and research, patience, motivation, and immense knowledge. His guidance helped me in all the research and writing of this thesis.

I would also like to thank *Mikhail Gaidukov*, *Aleksandr Sidun*, and *Manish Srivastava* for their insightful help, comments and suggestions. My sincere thanks also go to **Tyndall National Institute** and the **MCCI team**.

I would also like to thank my parents and friends for their understanding, endless patience, and encouragement when it was most required. I thank my grandparents for providing me with a healthy and tranquil environment to complete this work.

Finally, my deepest thanks to *Anđela Vuković*. I cannot overlook her pivotal role in my life during this Master's. In a country that often felt gloomy, Anđela provided a beacon of warmth. It was always comforting to know that we could count on each other, being far from our families. Her presence and support were invaluable; I will always be grateful to her.

In closing, I am reminded of two quotes from Seneca[1]. The first, '*Ita fac, mi Lucili,* ***vindica te tibi***'[2], indeed '***Non est*** *ad astra* ***mollis*** *e terris* ***via***'[3].

[1]These sentences translate to 'Do this way, my Lucilius, reclaim yourself', indeed 'there is no easy way from the Earth to the stars'.

[2]Seneca, Letters to Lucilius, Letter 94.

[3]Seneca, Hercules Furens, Line 437.

Contents

Abstract . . . . . **i**

Declaration . . . . . **ii**

Acknowledgment . . . . . **iii**

Contents . . . . . **vii**

List of Figures . . . . . **xi**

List of Scripts . . . . . **xii**

List of Tables . . . . . **xiii**

Glossary and Abbreviations . . . . . **xiv**

I Magnetic Tracking in Modern Medical Interventions **1**

1.1 Anser EMT project 2
1.1.1 Validation and Performance 3
1.2 Scope and Structure of the Thesis 4
1.2.1 Challenges and Innovations 4
1.3 Top-Level Implementation Overview 4
1.3.1 Preliminary System Modules 4
1.3.2 ADC Modeling and Clock Source Design 4
1.3.3 Power Management and Signal Transmission 5
1.3.4 I^2C Communication Protocol 5
1.4 Thesis Breakdown 6
1.4.1 Supplementary Insights: Appendices 7

II Continuous-time ΔΣ ADC model design 9
2.1 Proposed Modulator Architecture . . . 9
2.2 Clock Jitter Model . . . 12
2.2.1 Simulink Model . . . 13
2.3 Initial Modelling and Results . . . 15
2.3.1 Determining the Optimal OBG . . . 15
2.3.2 Noise and Loop-filter Transfer Function . . . 18
2.3.3 Signal Transfer Function . . . 21
2.3.4 Input Signal Analysis . . . 22
2.3.5 Ideal model with Clock Jitter . . . 23
2.4 FIR Filter implementation . . . 28
2.4.1 DAC and FIR DAC time response . . . 28
2.4.2 Feedforward Coefficient Scaling . . . 30
2.4.3 Incorporation of Time Delay . . . 35
2.4.4 Feedforward coefficient adjustment . . . 36
2.4.5 Compensation Path . . . 37
2.4.6 CTΔΣM Topology A . . . 41
2.4.7 Moving the Compensation Path . . . 43
2.4.8 CTΔΣM Topology B . . . 46
2.5 CTΔΣM Model results . . . 49
2.5.1 Dynamic Range Scaling . . . 49
2.6 $L(s)$: Towards Schematic . . . 55
2.6.1 Divergence from the Model . . . 57
2.6.2 Advanced Model . . . 57
2.7 Summary of Chapter II . . . 61

III Phase Noise Profile Model 63
3.1 First model characterisation . . . 63
3.2 Advanced model characterisation . . . 67
3.3 How Phase Noise Affects CTΔΣM's SNR . . . 68
3.3.1 Phase Noise profile . . . 69
3.3.2 Generating the Phase Noise . . . 72
3.3.3 Frequency Shifting of Phase Noise . . . 75
3.3.4 In-Band Noise Computation . . . 78
3.4 Phase Noise Analysis in the Discrete-Time Domain . . . 84
3.4.1 Phase Noise into the Discrete-Time Domain . . . 84
3.4.2 Verifying the Frequency Deviation . . . 87
3.5 Summary of Chapter III . . . 94

IV FLL as the Clock Source for CTΔΣM 97
4.1 Frequency Reference choise . . . 99
4.2 VCO Design and Analysis . . . 101
4.3 Feedback Mechanism . . . 107
4.3.1 Components and Configuration . . . 110
4.3.2 Determining the Ideal V_{fb} . . . 110
4.4 Jitter Analysis of the Feedback . . . 112
4.4.1 Ring oscillator output . . . 113
4.4.2 Feedback frequency response . . . 116
4.4.3 Feedback Noise . . . 120
4.5 Final FLL Structure . . . 121
4.5.1 Voltage reference . . . 121
4.5.2 Amplifier . . . 123
4.5.3 Final VCO . . . 124
4.5.4 Final Model for the Clock Source . . . 131
4.5.5 Full Model Results . . . 134
4.6 Summary of Chapter IV . . . 136

V Final Results and Conclusions 139
5.1 CTΔΣM with different VCOs in the FLL . . . 140
5.2 Cadence: Phase noise clock model . . . 143
5.3 Anser EMT Results . . . 144
5.3.1 CTΔΣM and Integrated Clock Results . . . 146
5.4 Conclusions and Future Work . . . 151

Appendix 155

A Discrete Fourier transform . . . 155
A.1 FT, DTFT and DFT . . . 155
A.2 Fast Fourier transform FFT . . . 157
A.2.1 From DFT to FFT . . . 157
A.3 Window function . . . 157
A.3.1 NENBW and ENBW . . . 158
A.4 Periodogram . . . 163
A.5 FFT using a Window function . . . 164
A.6 Conclusion and Summary of FFT using a Window Function . . . 165

B Configuring Cadence Virtuoso . . . 167
B.1 Output Spectrum in ADC . . . 167

B.1.1 Clipping the Signal . . . 168
B.1.2 Sampling the Clipped Signal . . . 168
B.1.3 Spectrum Evaluation . . . 169
B.2 White Noise Clock model . . . 170
B.3 Non-White Clock Noise . . . 170
B.3.1 Possible solutions . . . 173
B.4 Conclusion . . . 173

Index of References . . . 175

List of Figures

1.1 Electromagnetic navigational bronchoscopy and robotic-assisted thoracic surgery [1]. 2
1.2 Anser EMT structure [4]. 3
1.3 Top Level Implementation of the IC. 5
2.4 A Block diagram of the conventional 2^{nd} order CT$\Delta\Sigma$M. . . . 10
2.5 Block diagram of a generic single-loop (CT$\Delta\Sigma$M) [8], [9]. . . . 12
2.6 Additive error: equivalent DAC model [8], [9]. 13
2.7 Additive error: Simulink model [8], [9]. 14
2.8 Additive error: DAC output waveform comparison [8], [9]. . . 15
2.9 Flowchart of the Matlab code for generating the OBG. 17
2.10 Comparison between transfer function with and without zeros optimisation. 19
2.11 Loop filter comparison between discrete and continuous time. . 20
2.12 Illustration of the ADC's signal transfer function. 22
2.13 ADC's output spectra. 24
2.14 Magnified view of the ADC's output spectra. 25
2.15 SNR versus Periodic jitter performaces. 25
2.16 Measured `RMS` in-band noise versus rms periodic jitter. 26
2.17 Noise Transfer Function Verification. 27
2.18 Schematic Illustration of the Correct FIR Filter Application. . 30
2.19 Simplified CIFF structure. 31
2.20 Open-loop structure with FIR-DAC. 32
2.21 Preliminary CT$\Delta\Sigma$M configuration with FIR-DAC. 35
2.22 (a) Ideal and (b) FIR-DAC open loop response. 38
2.23 Amplitude difference between the ideal and FIR-DAC open loop responses. 39
2.24 Complete open loop response with compensation path. 40
2.25 Restored open loop response with compensation path. 40
2.26 CT$\Delta\Sigma$M with *fast-loop* compensation. 41
2.27 Open-loop responses of the first integrator output. 43
2.28 Amplitude difference between open-loop responses of the first integrator. 44

2.29 Complete block diagram of the second open-loop responses. . . 45
2.30 Restoration of the second open-loop response. 45
2.31 CT$\Delta\Sigma$M configuration with compensation between integrators. 46
2.32 CT$\Delta\Sigma$M CIFF topologies. 47
2.33 SNR improvement using the compensation path 49
2.34 CT$\Delta\Sigma$M - (B) with dynamic range scaling. 50
2.35 Dynamic range scaling waveforms (continued on the next page) 53
2.35 Dynamic range scaling waveforms (continued) 54
2.36 $L(s)$ single-ended implementation. 55
2.37 Refined $L(s)$ single-ended implementation. 57
2.38 Complete $L(s)$ model. 58
3.39 Generic IIR filter structure [17] 64
3.40 Conversion of a Laplace transform into a $\mathcal{Z}$-transform 65
3.41 Flowchart of the Conversion into a $\mathcal{Z}$-transform. 66
3.42 Phase noise profile (Single-Sided). 70
3.43 Flowchart of the Matlab code for generating the phase noise profile. 71
3.44 Phase Noise profile (Double-Sided). 73
3.45 Phase Noise generated - positive frequencies. 74
3.46 Phase Noise generated - full spectrum. 75
3.47 Phase Noise Spectrum Shifted to ADC Input Frequency. . . . 76
3.48 Phase Noise Spectrum Shifted. 77
3.49 Flowchart of the Matlab code for performing the frequency shift operation. 77
3.50 Phase Noise shifted by the input frequency of the ADC. 80
3.51 Phase Noise convoluted with the NTF. 81
3.52 Total IBN until the CT$\Delta\Sigma$M bandwidth of 20kHz. 82
3.53 ADC output spectrum with internal clock. 83
3.54 Conversion in Time Domain. 85
3.55 Flowchart of the Matlab code for converting frequency deviation to phase noise in the time domain. 86
3.56 Conversion in Time Domain within the range $[-\pi, \pi]$. 88
3.57 Original and Noisy Signal in Time Domain. 88
3.58 PSD of signal $x[n]$ and $x_{\Phi_{DS}}[n]$. 89
3.59 Flowchart of the Matlab code for modulating a signal with phase noise. 91
3.60 Phase Noise measured 92
3.61 Phase Noise measured with levels. 93
4.62 FLL Schematic 98
4.63 Fundamental configuration of a PLL. 100
4.64 Architectural overview of an FLL [26]. 101

4.65 Schematic Representation of a Five-Stage VCO Designed in Cadence. . . . 104
4.66 VCO transient simulation. . . . 105
4.67 Varactor used in the design. . . . 105
4.68 Output frequency responses of VCOs with different loads. . . . 106
4.69 Feedback architecture. . . . 108
4.70 Feedback time response 109
4.71 Feedback schematic in Candence, as described in Figure 4.69. 111
4.72 Specific Cadence's schematic utilized for the simulation of the ideal feedback voltage. . . . 112
4.73 VCO and feedback output frequency in relation to V_C. . . . 113
4.74 VCO's Phase Noise calculated 116
4.75 VCO's output spectrum 117
4.76 FLL's Feedback DFT analysis. . . . 118
4.76 FLL's Feedback DFT analysis. . . . 119
4.77 Resistor Divider Schematic on Cadence. . . . 122
4.78 Amplifier schematic used in the FLL 123
4.79 Time analysis of OTA Input Voltage and Output Current for Transconductance g_m Evaluation. . . . 124
4.80 Three stages VCO design in Cadence. . . . 126
4.81 Comparison of FLL spectra 127
4.82 Comparison of FLL's σ_p 128
4.83 PVT Analysis of VCO Frequency Variation 129
4.84 Capacitive load with trimming circuit 130
4.85 Clock source noise transfer functions 132
4.86 Expected standard deviation of absolute jitter σ_a and periodic jitter σ_p. . . . 133
4.87 Complete Clock schematic, the FLL, in Cadence. . . . 134
4.88 Power distribution in the FLL. . . . 135
5.89 Schematic of the Top-Level testing setup in Cadence. . . . 140
5.90 Verilog-A flowchart to send data to the workspace. . . . 141
5.91 Simulated CTΔΣM's PSD comparison. . . . 143
5.92 CTΔΣM's SNR comparison of Figure 5.91 144
5.93 Test setup for the CTΔΣM and on-chip magnetic sensor. . . . 145
5.94 Magnitude plot showing the performance of the CTΔΣM with magnetic sensor with the time-varying 8-tone (20-34 kHz) magnetic field. . . . 146
5.95 Detailed view of the PSD from Figure 5.94 147
5.96 Detailed view of the testing setup at Tyndall National Institute. 147
5.97 Oscilloscope clock's spectra comparison. . . . 148
5.98 Clock PSD from Figure 5.97. . . . 149

5.99 Internal's clock PSD from Figure 5.97 150
5.100CT$\Delta\Sigma$M's IBN PSD plot. 151
1 Comparison of FT and DTFT Magnitudes 156
2 Comparison of DFT and FFT Magnitudes 156
3 NENBW example 159
4 NENBW formula for a window function in MATLAB 160
5 Calculating the NENBW for a Hanning window of length $N = 2^{16}$. 160
6 ENBW formula for a window function in MATLAB 161
7 ENBW for a Hannin window of length $N = 2^{16}$ 161
1 White Noise Jitter model 171
2 Phase Noise Jitter model 172

List of Scripts

2.1 OGB for a chosen transfer function 17
3.2 Conversion into a $\mathcal{Z}$-transform 66
3.3 Phase Noise profile (Single-Sided). 71
3.4 Phase Noise profile (Double-Sided). 72
3.5 Generating Phase Noise. 74
3.6 Frequency Shift Operation. 78
3.7 Converting Frequency Deviation to Time Domain. 86
3.8 Modulating Signal with Phase Noise. 91
3.9 Measuring Phase Noise Profile. 94
5.10 Verilog-A code for TWS_VerilogA 141

List of Tables

1.1 System-level specifications for key blocks in the Anser EMT project. 6
2.2 Topology A coefficients 42
2.3 Coefficient comparison for topologies A and B 48
2.4 Coefficient comparison for B topology 54
4.5 CTΔΣM Model Specifications for Topologies A and B 97
4.6 Key Design Constraints for Clock Source 97
4.7 Notation utilised in the VCO design discussion. 102
4.8 Component values and configurations. 110
4.9 Correlation between V_C, f_{out}, and $\bar{V}_{\text{fb}}$. 111
4.10 PMOS differential pair 124
4.11 Three stages ring oscillator 125
4.12 Code for trimming circuit 131
4.13 Summary of Key Parameters. 134
4.14 Component values and configurations. 135

Glossary and Abbreviations

ADC	Analog to digital converter.
AFE	Analog-frontend.
BIBO	Bounded Input, Bounded Output Stability.
BW	Bandwidth.
Chip Area	The physical space taken up by a component or set of components on an integrated circuit (IC).
CIFF	Cascade Integrator for feed-forward.
CT$\Delta\Sigma$M	Continuous-time delta sigma modulator.
DAC	Digital to analogue converter.
DEM	Dynamic Element Matching.
EMT	Electromagnetic tracking.
FIR	Finite impulse response filter.
FIR-DAC	Finite impulse response digital-to-analogue converter.
fs	Fast corner for nMOS and slow corner for pMOS in semiconductor process.
ff	Fastest corner for semiconductor process.
FFT	Fast Fourier Transform.
FLL	Frequency Locked Loop clock source.
GHz	Gigahertz, a unit of frequency denoting one billion cycles per second.
IC	Integrated Circuit.
IGS	Image-guided surgery.
LC Tank	An LC circuit, also called a resonant, tank, or tuned circuit.
LDO	Low DropOut regulator, a type of linear voltage regulator.

L	Loop filter.
LVDS	Low-Voltage Differential Signaling, a digital signaling system that can run at high speeds over inexpensive twisted-pair copper cables.
M-taps	The number of taps in a filter, typically in an FIR filter.
MHz	Megahertz, a unit of frequency denoting one million cycles per second.
NTF	Noise Transfer Function.
NRZ	Non-Return-to-Zero (binary signal format).
$\mathcal{O}(\cdot)$	A mathematical notation called 'Big O' is used to describe the limiting behaviour of a function.
$o(\cdot)$	A mathematical notation called 'Little o' is used to describe the limiting behaviour of a function as it approaches zero.
OBG	Out-of-Band Gain.
Op. Amp	Operational amplifier.
$R_{\mathbf{eq}}$	Equivalent resistance.
PEX	Parasitic Extraction, a process in IC design to model and analyze parasitic elements that affect circuit performance.
Phase Comparator	An electronic device that compares the phase of two input frequencies.
PLL	Phase Locked Loop.
PVT	Process, Voltage, and Temperature variations in semiconductor manufacturing.
$R_{\mathbf{off}}$	Resistance when a transistor is in the off state.
PSD	Power Spectral Density.
SAR ADC	Successive Approximation Register Analog-to-Digital Converter.
Simulink	A MATLAB-based graphical programming environment.

ss	Slow corner for semiconductor process.
SNDR	Signal-to-Noise-and-Distortion Ratio.
SNR	Signal-to-Noise Ratio.
sf	Slow corner for nMOS and fast corner for pMOS in semiconductor process.
SQNR	Signal-to-Quantization Noise Ratio.
STF	Signal Transfer Function.
tt	Typical corner for semiconductor process.
VCO	Voltage-Controlled Oscillator.
$V_{\mathbf{ref}}$	Reference voltage.
$\mathcal{N}$	Normal Gaussian distribution, represented by $\sim \mathcal{N}(0, \sigma)$.
σ	Standard deviation, the square root of variance.

Chapter I:
Magnetic Tracking in Modern Medical Interventions

MAGNETIC tracking has emerged as an indispensable tool in image-guided interventions. Guiding surgical instruments with remarkable precision during **minimally invasive procedures**. The allure of magnetic tracking lies in its accuracy and capacity to reduce the perilous exposure to radiation, thereby safeguarding both the patient and the medical professionals.

Electromagnetic tracking has shown significant potential, especially when integrated with robotic systems in procedures like electromagnetic navigational bronchoscopy. This technology, similar to the global positioning system, enables precise localization and marking of thoracic tumours, enhancing the accuracy of minimally invasive procedures [1]. A groundbreaking development by researchers at Tyndall National Institute and Microelectronics Circuits Centre Ireland (MCCI), based at University College Cork (UCC), has further revolutionised surgical navigation. Researchers have developed the first sensor-on-a-chip for magnetic tracking in surgery and other image-guided interventions. This advancement is a significant step forward, moving away from reliance on harmful radiation imaging (x-rays) towards a safer, more precise approach to navigating medical instruments within the body. Magnetic tracking uses low-frequency magnetic fields to precisely detect the position of tiny sensors inside the patient. However, existing sensors are complex to manufacture, expensive, and extremely delicate. The new sensor developed at Tyndall overcomes these challenges by using standard silicon chip technology, resulting in a simplified and cost-effective manufacturing process. This innovation not only enhances the accuracy of tracking (less than a millimetre) but also reduces the cost, making the technology accessible for widespread clinical use. Preliminary results, published in the IEEE Transactions on Biomedical Circuits journal [2], [3], demonstrate the use of the chip for tracking instruments inside the lungs. This is particularly important for targeting and treating diseases like lung cancer, which is a leading cause of global cancer incidence and mortality. The integration of this sensor with flexible circuits makes assembly quick and reliable, further contributing to its practical application in clinical settings. Professor *Pádraig Cantillon-Murphy*, who led the research team, highlights that this development represents the culmination of 10 years of progress in magnetic tracking technology at Tyndall and UCC. The team looks forward to translating this technology

into clinical applications, where it can make a significant difference in patient outcomes. Figure 1.1 illustrates the application of this technology in thoracic surgery.

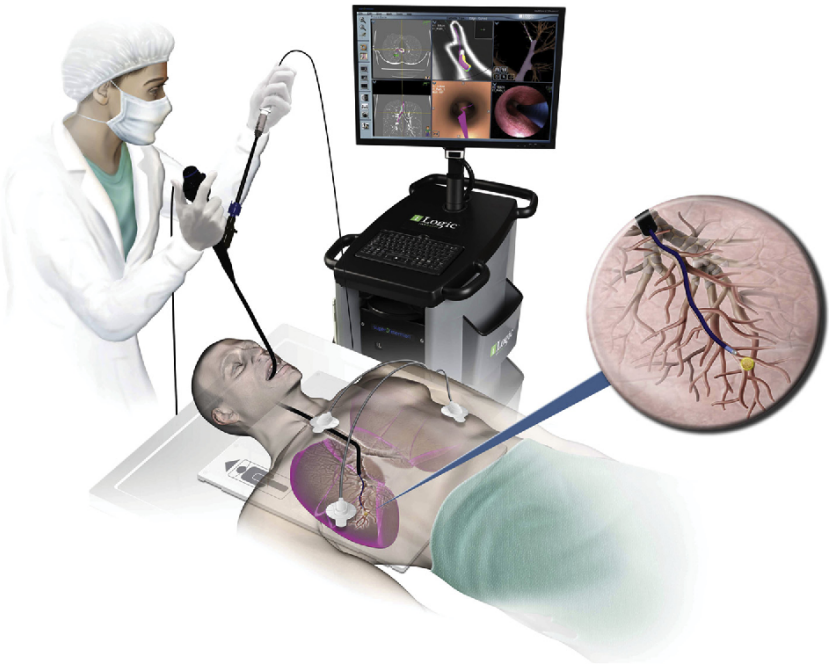


Figure 1.1: Electromagnetic navigational bronchoscopy and robotic-assisted thoracic surgery [1].

1.1 Anser EMT project

OLD Among the trailblazers in this domain is the **Anser EMT** project, an open hardware platform for electromagnetic tracking developed at Tyndall & UCC. The project is methodically segmented into three core components:

- **Transmitter:** The primary source emitting the electromagnetic signals.
- **Receiver:** The IC component is tasked with accurately receiving the transmitted signals.
- **Solver:** A crucial element for computing and interpreting the signals, enabling precise tracking.

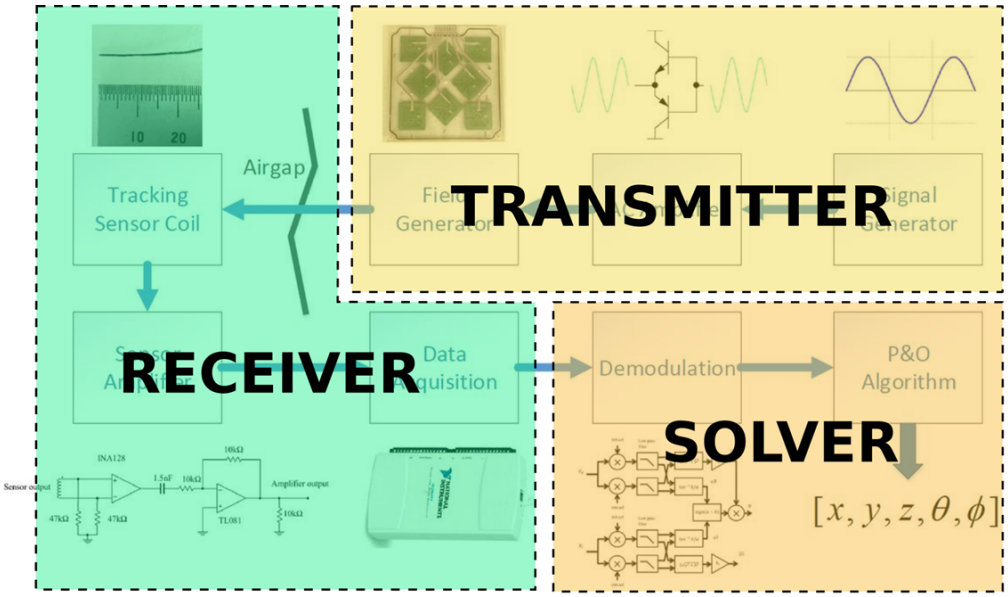


Figure 1.2: Anser EMT structure [4].

As an *open-source initiative*, Anser EMT offers an affordable yet sophisticated solution, especially pertinent for research endeavours. One of its cardinal strengths lies in its compatibility with commercial EMT sensors and its seamless integration with medical imaging toolkits such as **MITK** and **3D Slicer**. Further details and resources related to Anser EMT can be accessed at the following link of the project: https://www.opensourceimaging.org/project/anser-emt/.

1.1.1 Validation and Performance

The efficacy of Anser EMT is not just a theoretical postulation. Objective evaluations benchmarking it against other commercial tracking platforms have been undertaken. As highlighted in a peer-reviewed study:

> "Positional accuracy of 1.14mm and angular rotation accuracy of [Formula: see text] are reported. [...] By providing an *open-source platform* for research investigations, we believe that novel and collaborative approaches can overcome the limitations of current EMT technology."

Such validations underscore the potential and reliability of Anser EMT, positioning it as a potent tool in the ever-evolving landscape of image-guided interventions [4].

1.2 Scope and Structure of the Thesis

This research primarily focuses on the *receiver component* of the magnetic tracking system (see Figure 1.2). This integrated circuit (IC) captures electromagnetic signals from the transmitter, a matrix of coils generating the magnetic field. After amplification, these signals indicate position, processed further by the solver.

1.2.1 Challenges and Innovations

Initially, the system included the IC chip with its on-chip **coil** and a front-end low-noise **amplifier**. A challenge was the low amplitude of the analog signal at the output, which was comparable to the system's thermal noise. To address this, a **Continuous-Time Delta Sigma (CT$\Delta\Sigma$M) modulator** was introduced. A custom model for the ADC was developed for the Anser system.
ADCs require a *low-jitter* clock source. The strategy was to integrate an internal clock source for the chip, prioritising *low-noise* and *compact design*. However, challenges arose from the ADC's architecture, notably its intolerance to a **non-white phase noise spectrum**. This will be detailed in subsequent sections.

1.3 Top-Level Implementation Overview

The design and development of the IC chip for the Anser EMT project involved collaborative efforts. Figure 1.3 provides an overview of the IC chip's top-level implementation.

1.3.1 Preliminary System Modules

The components in green in Figure 1.3 represent foundational modules, including the inductor and a front-end amplifier. These components, designed for *low noise*, play crucial roles in signal acquisition and amplification.

1.3.2 ADC Modeling and Clock Source Design

The segments in red in Figure 1.3 focus on the ADC. Given the subdued amplitude of the analog signal and the system's thermal noise requirements, a CT$\Delta\Sigma$ ADC was integrated. A tailored MATLAB and Simulink model for this ADC was developed for the Anser project. An integrated clock source

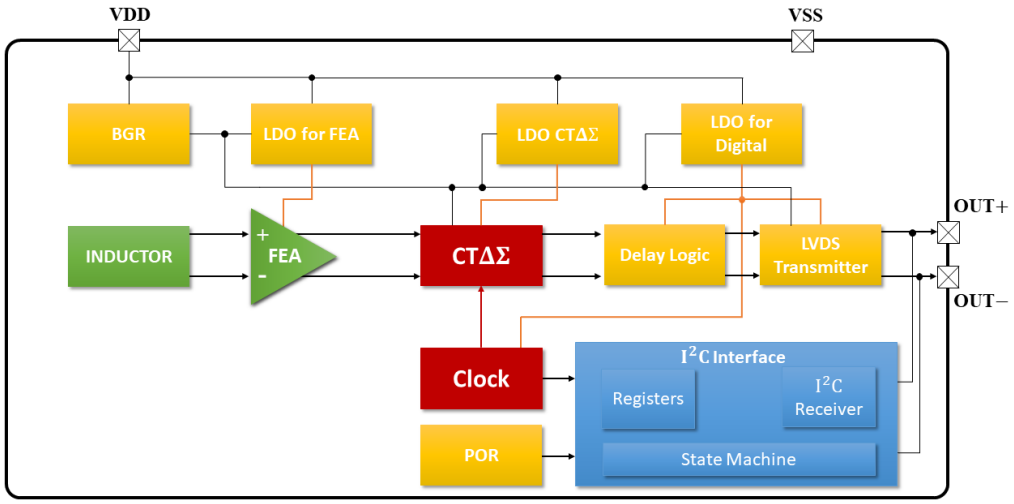


Figure 1.3: Top Level Implementation of the IC.

was also designed to emphasise *low-noise* characteristics. The challenges linked to the ADC's specific design will be discussed in Chapter II.

1.3.3 Power Management and Signal Transmission

The components in yellow in Figure 1.3 pertain to power management. Another team member developed a bandgap reference, foundational for the system's Low Dropout Regulators (LDOs). Three LDOs were integrated for the Front-End Amplifier (FEA), the ADC, and the digital domain, ensuring stability against switching noise. A distribution network and an LVDS were also included for efficient signal transmission.

1.3.4 I^2C Communication Protocol

The I^2C protocol, depicted in blue in Figure 1.3, is pivotal for chip interfacing and communication. This bidirectional two-wire interface enhances the tunability and flexibility of the chip. Within the Anser EMT project, it plays a crucial role in data transfer and ensures cohesive interfacing between different IC modules.

1.4 Thesis Breakdown

The system-level specifications of the Anser EMT project include high positional accuracy and resolution, low latency in signal processing to ensure real-time tracking, robustness against electromagnetic interference, and a compact, power-efficient hardware design. The performance requirements for the key blocks, highlighted in **red** in Figure 1.2, are particularly stringent. These blocks include the 2nd order continuous-time sigma-delta modulator and the clock source, which are critical for achieving the desired system-level specifications. The 2nd order CT$\Delta\Sigma$M was chosen for its ability to provide *high resolution* and *low noise performance*, essential for *accurate magnetic tracking* [2]. Its second-order architecture offers a good balance between complexity and performance, meeting the stringent requirements of the Anser EMT project [3]. The following specifications are essential for these blocks, see Table 1.1.

Specification	**Value**
Technology	65nm TSMC
Supply Voltage	1.2V
Signal Bandwidth	20kHz
ADC modulator	2nd-order CT$\Delta\Sigma$M
SNR	$\geq$ 75dB

Table 1.1: System-level specifications for key blocks in the Anser EMT project.

This thesis is organised to offer a thorough understanding of the research:

- **Chapter II: CT$\Delta\Sigma$M Model Creation** - A deep dive into the CT$\Delta\Sigma$ Modulator, discussing its principles, benefits, and challenges. The custom ADC model for Anser, a highlight of this research, is elaborated upon in Table 1.1. This chapter also details the *equation discovery* and implementation of the **CIFF architecture** [3] and the choice of a 10.24MHz sampling frequency, which were critical to meeting the specified SNR requirements.

- **Chapter III: Phase Noise Spectrum Model** - The focus here is on modelling phase noise using MATLAB. The chapter justifies the impact

of the phase noise spectrum on the CT$\Delta\Sigma$M, making it crucial for understanding the modulator's behaviour under real-world conditions, which require precise signal processing and low noise performance.

- **Chapter IV: Clock Source Circuit Design** - This section illuminates the nuances of the clock source design, its significance, and the hurdles faced during its integration with the ADC. The clock source, designed to meet the 10.24MHz sampling frequency and high SNR requirements, plays a critical role in the overall system performance.

- **Chapter V: Chip Results** - A showcase of the research's tangible outcomes, drawing parallels between theoretical expectations and empirical findings and offering a critical performance assessment of the developed sensor-on-a-chip technology.

1.4.1 Supplementary Insights: Appendices

Complementing the primary content:

- **Appendix A:** A detailed spectral density analysis focusing on the FFT algorithm. It highlights methodologies to compute power spectral density optimally, addressing challenges like spectrum leakage and windowing.

- **Appendix B:** A primer on 'Cadence Virtuoso', underscoring precision's role in frequency analysis and offering best practices.

This thesis aims to bridge the gap between intricate technical details and their broader implications in magnetic tracking for medical applications.

Clarification on Figure Conventions: All thesis figures depicting a PSD utilise a vertical magnitude scale expressed in *dB, normalised*, it can be referred as scaled [5]. This convention arises from dividing the central bin value of the input signal, effectively scaling the chart to 0 dB. Such a method allows for direct observation of the quantisation noise floor without additional scale adjustments. More details about PSD can be found in Appendix A.

References

[1] S. Christie, "Electromagnetic navigational bronchoscopy and robotic-assisted thoracic surgery," *AORN J*, vol. 99, no. 6, pp. 750–763, 2014. DOI: 10.1016/j.aorn.2013.09.016 (Cited on pages 1, 2).

[2] M. Srivastava, K. O'Donoghue, A. Sidun, *et al.*, "3d position tracking using on-chip magnetic sensing in image-guided navigation bronchoscopy," *IEEE Transactions on Biomedical Circuits and Systems*, pp. 1–17, 2024. DOI: 10.1109/TBCAS.2024.3384016 (Cited on pages 1, 6).

[3] M. Srivastava, A. Ferro, A. Sidun, *et al.*, "A small-area 2nd-order adder-less continuous-time $\Delta\Sigma$ modulator with pulse shaping fir dac for magnetic sensing," *IEEE Open Journal of Circuits and Systems*, vol. 5, pp. 42–54, 2024. DOI: 10.1109/OJCAS.2024.3378653 (Cited on pages 1, 6).

[4] H. Jaeger, A. Franz, K. donoghue, *et al.*, "Anser emt the first open-source electromagnetic tracking platform for image-guided interventions," *International Journal of Computer Assisted Radiology and Surgery*, vol. 12, Mar. 2017. DOI: 10.1007/s11548-017-1568-7 (Cited on pages 3, 144, 145).

[5] F. Harris, "On the use of windows for harmonic analysis with the discrete fourier transform," *Proceedings of the IEEE*, vol. 66, no. 1, pp. 51–83, 1978. DOI: 10.1109/PROC.1978.10837 (Cited on page 7).

Chapter II:
Continuous-time $\Delta\Sigma$ ADC model design

In recent advancements in *in-vivo* applications, especially in the realm of **Image-Guided Surgery (IGS)**, precision and miniaturisation have become paramount [6]. The exploration of diminutive anatomical structures demands sensor systems capable of sub-1-mm manoeuvring. Central to this is the **Analog-to-Digital Converter (ADC)**, instrumental in digitising sensor signals and ensuring precise navigation during medical interventions. Yet, peripheral components, including input buffers, power supply regulators, and clock generators, are known to be power-intensive and occupy significant space. This has led to the exploration of ADC designs that are both compact and energy-efficient.

While **Continuous-Time Delta-Sigma Modulators (CT$\Delta\Sigma$M)** have been considered for their area requirements, they tend to consume more power than alternative designs such as SAR ADCs.

Embracing a *single-bit* CT$\Delta\Sigma$M simplifies the architecture, especially concerning the **Digital-to-Analog Converter (DAC)**, avoiding the complexities of calibration or Dynamic Element Matching (DEM) procedures. However, this design approach has challenges, mainly due to its susceptibility to clock jitter. Addressing these challenges mandates comprehensive mathematical modelling, efficiently achieved using tools like **Matlab**/**Simulink**.

This research delves into formulating a mathematical model to simulate the behaviour of a 2^{nd}-order CT$\Delta\Sigma$M. This initiative aims to provide insights into challenges and potential solutions in CT$\Delta\Sigma$M system design.

Subsequent sections elucidate the system design, encompassing the mathematical model and simulation results, laying the groundwork for in-depth discussions on the proposed modulator architecture. The thesis offers a detailed perspective on the system design, highlighting the mathematical model and its simulation outcomes.

2.1 Proposed Modulator Architecture

The proposed modulator is anticipated to be a pivotal component in Image-Guided Surgery (IGS) and robotic surgery applications. Its performance is inherently bounded by noise factors from the input sensor to the analog front end. To attain the desired system precision, a modulator signal-to-

noise ratio (**SNR**) in the range of **70-80 dB** is essential. The integration of a Continuous-Time Delta-Sigma modulator CTΔΣM, featuring a bandwidth (**BW**) of **20 kHz**, is crucial to this endeavour.

Figurw 2.4 showcases the block diagram of the **Cascade of Integrators in Feedforward** (CIFF) 2nd-order CTΔΣM [7]. The feedback pathway, driven by gain g, incorporates a set of complex conjugate poles within the loop-filter transfer function $L(s)$. This operates as a low-pass inverse Chebyshev filter, typified as a high-pass filter, leading to zeros in the noise transfer function (NTF). Rather than being localized at DC, the strategic positioning of NTF zeros throughout the signal band bolsters the dynamic range scaling of model coefficients, targeting an *out-of-band gain* (**OBG**) of roughly 1.5 in magnitude (not in dB) [8].

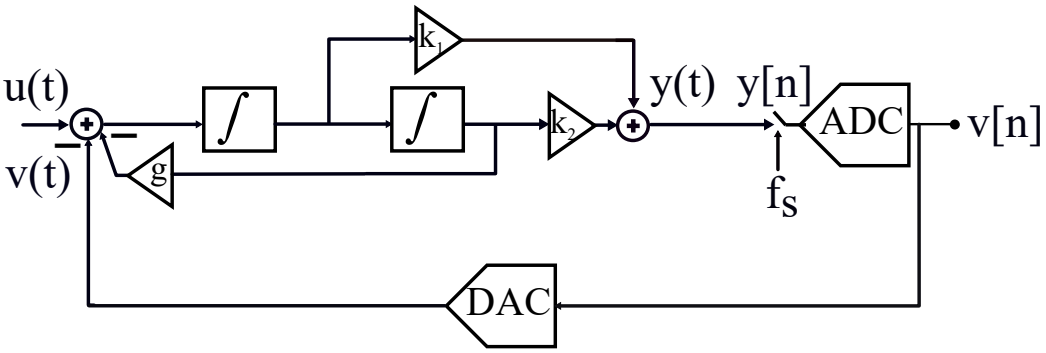


Figure 2.4: A Block diagram of the conventional 2^{nd} order CTΔΣM.

The *second-order design* aligns with the requirements of the **Anser EMT project**. Elevating the **ADC's order**—which implies augmenting the **loop filter's order** and thereby integrating a series of n integrators and k_i coefficients for a generic order n—shapes the quantisation noise profile outside the **CTΔΣM low-pass filter's** bandwidth. Specifically, the **NTF** characterises the quantisation noise as a high-pass filter with a gradient of $20\,\mathrm{dB/dec} \cdot n$. Therefore, a gradient of $40\,\mathrm{dB/dec}$ is deemed adequate. However, further elevation in order introduces intricacy. One primary challenge is ensuring the **ADC's BIBO stability**, a fundamental property where every bounded input guarantees a bounded output. Ensuring BIBO stability for a higher CTΔΣM order necessitates more coefficients in the transfer function and demands additional amplifiers for the **IC realisation**. This translates to increased area, heightened power consumption, and supplementary system noise, underscoring the prudence of the second-order design choice.

The transfer functions of the CTΔΣM, particularly the Signal Transfer Func-

tion (STF(s)), Noise Transfer Function (NTF(s)), and Loop Filter (L(s)), are defined as follows [8]:

$$STF(s) = \frac{L(s)}{1 + L(s)}\,, \tag{2.1}$$

$$NTF(s) = \frac{1}{1 + L(s)}\,, \tag{2.2}$$

$$L(s) = \frac{k_2 f_s^2 + k_1 f_s s}{g f_s^2 + s^2}\,. \tag{2.3}$$

The loop-filter transfer functions are characterised by:

- k_1 and k_2: the feedforward coefficients;
- g: the gain associated with the complex conjugates poles in equation 2.3 and Figure 2.4;
- Equation 2.3 simplifies the representation of the integrator's open-loop transfer function. To accurately represent the block diagram, replace the integral symbol $\int$ with $\boldsymbol{H}_{\text{Integrators}}(\boldsymbol{s}) = \frac{\boldsymbol{f_s}}{\boldsymbol{s}}$, where f_s is the clock sampling frequency. It is essential to scale the integrator to the chosen f_s frequency to ensure the loop filter operates at the desired sampling frequency.

The system converts an analog input signal $u(t)$ into a digital output $v[n]$, facilitated by the **1-bit quantiser** embedded within its structure. This **ADC**, functioning as a **comparator** due to its *single-bit nature*, converts the analog output of the **loop filter**. In this context, $L(s)$ is mathematically represented as

$$L(s) = \frac{Y(s)}{U(s)} \tag{2.4}$$

corresponds to the ratio between the continuous-time loop filter output $y(t)$ and the signal input in the **Laplace domain**. Subsequently, the discrete signal $v[n]$ is reverted to an analog format through a **1-bit DAC**, facilitating the implementation of **feedback mechanisms** and enabling accurate subtraction of signals, $u(t)$ and $v(t)$.

As illustrated in Fig. 2.5, both the **ADC** and **DAC** are clocked synchronously, ideally operating at $T_s = \frac{1}{f_s}$. However, deviations in the timing of the clock edges, depicted in Fig. 2.5, introduce errors termed as **jitter**. In the context of the CT$\Delta\Sigma$ architecture, the *ADC-induced error* is negligible as it undergoes shaping by the modulator's **NTF**, thereby not influencing the in-band

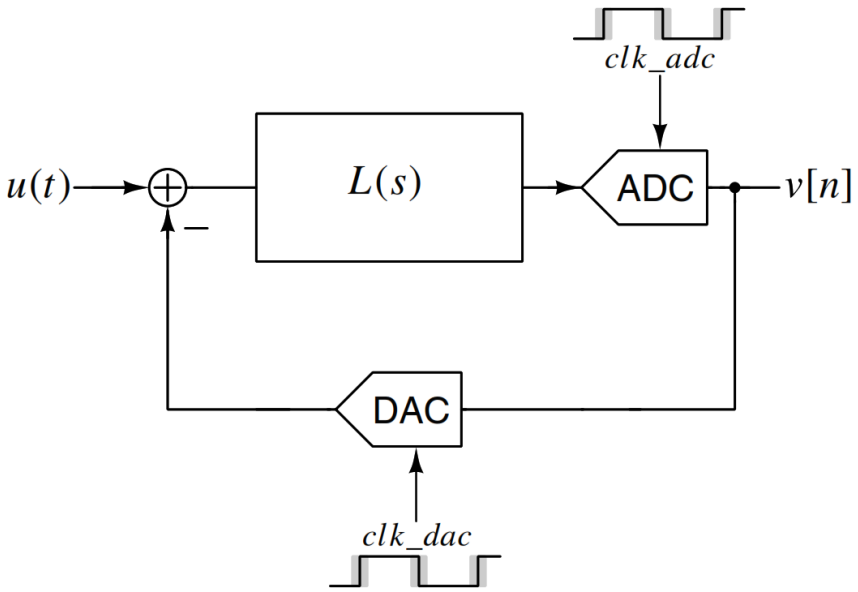


Figure 2.5: Block diagram of a generic single-loop (CTΔΣM) [8], [9].

spectrum. Conversely, the *DAC-induced error* is applied at the loop filter's input $L(s)$, generating uncertainty in the integrated values and instigating errors in the final output $v[n]$. This noise possesses an *in-band* content, proportional to the input signal $u(t)$, which cannot be mitigated by **NTF**, resulting in a degradation of the in-band **SNR**.

2.2 Clock Jitter Model

Within the CTΔΣM dynamics, it is imperative to determine an *equivalent error sequence* at the **DAC** input. This determination becomes particularly pivotal when the system operates at frequencies much higher than the input signal frequency, denoted as $f_{u(t)} < f_s$. In the given context, $f_{u(t)}$ is limited to a 20 kHz bandwidth, leading to an increment in f_s, as depicted by the subsequent equation [8]:

$$f_s = \text{BW} \cdot 2 \cdot \text{OSR} \tag{2.5}$$

where OSR represents the oversampling ratio. This sequence is construed as an error pulse, characterized by a width of T_s and a height of $v[n] - v[n-1]$, further complicated by a random timing error, $\Delta t[n]$, a function of n^{th} clock cycle, a distinct trait of jitter noise.

Figure 2.6 illustrates the real DAC's representation as an ideal DAC com-

plemented with a **discrete-time additive error** [9] at its input. The error, denoted by $e_j[n]$, is defined as:

$$e_j[n] = (v[n] - v[n-1])\frac{\Delta t[n]}{T_s} \tag{2.6}$$

Consequently, the additive error spectrum mirrors that of a pulse characterized by a width T_s and a height $e_j[n]$. It is worth noting that employing a

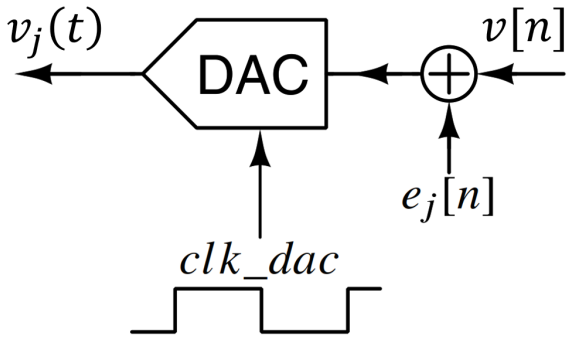


Figure 2.6: Additive error: equivalent *DAC* model [8], [9].

Non-return-to-zero DAC (*NRZ DAC*) can marginally diminish jitter sensitivity, as the clock-jitter is prominent only during the clock *edges* where data transitions occur.

2.2.1 Simulink Model

Building on the detailed periodic jitter noise model, a corresponding Matlab/Simulink block is constructed based on the principles presented in equation (2.6). Two key distinctions characterise this adaptation:

First, the term $\frac{\Delta t[n]}{T_s}$ is symbolized using a random white Gaussian source, as illustrated in:

$$\Delta t \sim \mathcal{N}\left(0, \sigma^2_{\text{Jitter}}\right) \tag{2.7}$$

Here, the variance, $Var(\Delta t) = \sigma^2_{\text{Jitter}}$ [10], originates from the *white spectrum jitter* definition. Given the positive half of the Jitter PSD, the squared standard deviation becomes:

$$\sigma^2_{\text{Jitter}} = 2\int_0^{f_s/2} S(f)df \tag{2.8}$$

With $S(f)$ being constant in frequency and having a value of:

$$S(f) = \frac{\sigma^2_{\text{Jitter}}}{f_s} \tag{2.9}$$

As highlighted in Figure 2.7, the RMS JITTER block works with a sampling period of T_s. Using its standard deviation, σ_{Jitter}, a unique jitter noise magnitude in pico seconds is set for each CT$\Delta\Sigma$M simulation. The white noise output, however, needs scaling by a factor of $\sqrt{2}$ to ensure the correct `RMS` jitter value is injected into the DAC pulses [11]. This leads to:

$$e_j[n] = (v[n] - v[n-1])\frac{\Delta t[n]}{\sqrt{\mathbf{2}}T_s} \tag{2.10}$$

Equation 2.10 can also be represented by an equivalent Normal Gaussian distribution with a variance of $\frac{\sigma^2_{\text{Jitter}}}{2}$, implying a standard deviation of $\frac{\sigma_{\text{Jitter}}}{\sqrt{2}}$.

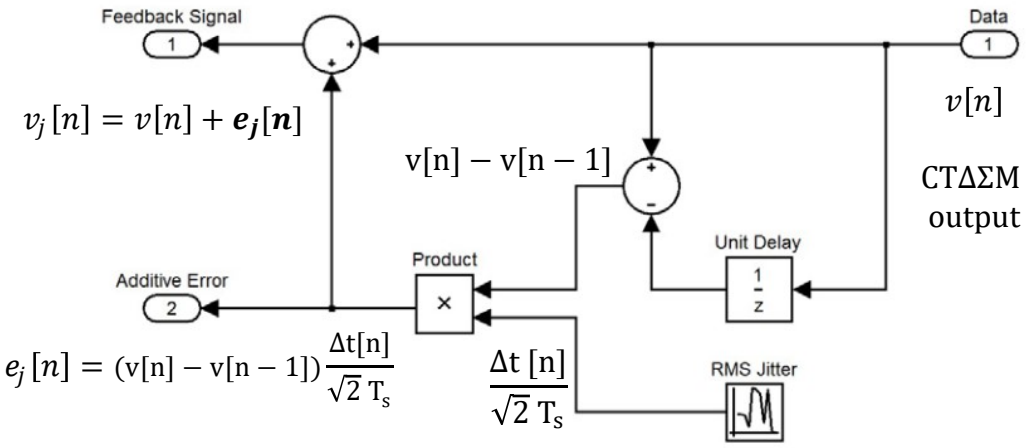


Figure 2.7: Additive error: Simulink model [8], [9].

Second, the model for the equivalent DAC characterises jitter errors as *amplitude variations*. This approach simplifies the simulation by bypassing the complexities associated with a variable step-size simulation. Instead, a fixed-step simulation is employed, ensuring efficiency and predictability. The choice to express jitter effects through amplitude variations rather than edge variations stems from the emphasis on the **DAC pulse area** [12]. The DAC pulse area becomes crucial given the presence of *integrators* in the loop filter. This strategy accurately represents the errors in feedback pulses for each sampling interval, as depicted in Figure 2.8. This figure contrasts various DAC waveforms, elucidating the NRZ DAC model behaviour.

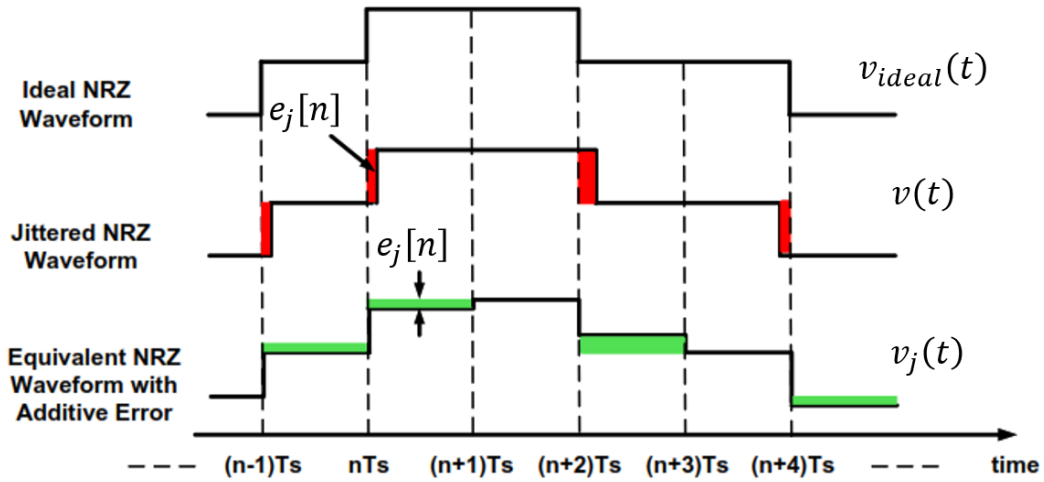


Figure 2.8: Additive error: *DAC* output waveform comparison [8], [9].

2.3 Initial Modelling and Results

To achieve an **ADC SNR** exceeding 70 dB, an **oversampling ratio** (OSR) of 256 was selected, as delineated in equation 2.5. Given a signal bandwidth (BW) of 20 kHz, the resultant sampling frequency (f_s) is 10.24 MHz, yielding a sampling period (T_s) of approximately 97.66 ns. Moreover, the optimal value for a 2$^{\text{nd}}$-order CT$\Delta\Sigma$M *Out-of-Band Gain (OBG)* is constrained to be less than or equal to 1.5 in magnitude (not in dB) [8]. To optimize the SNR, it is imperative to approach this limit as closely as possible; consequently, an **OBG** of **1.4947** was utilized.

The fundamental design specifications derived from these considerations are as follows:

- **Oversampling Ratio:** OSR = 256
- **Bandwidth:** BW = 20 kHz
- **Sampling Frequency:** $f_s = 10.24$ MHz
- **Sampling Time:** $T_s = 97.6562$ ns

2.3.1 Determining the Optimal OBG

The optimal **Out-of-Band Gain (OBG)** is pivotal in enhancing the ADC's SNR. This task is somewhat complex in a Continuous-Time Delta-Sigma modulator scenario due to the integration of $\mathcal{Z}$-domain analysis, albeit a

continuous-time system. The preference for $\mathcal{Z}$-domain stems from the ADC output's discrete nature and the advantages retained from a continuous-time filter.

The function `ntfMaker`[4] operates as detailed below:

- Defines the sample time, Ts, as f_s reciprocal, marking the initial transition from continuous to discrete time domain.
- Sets the value of a to $\sqrt{2}$ for the selected `InvChev` (Inverse Chebyshev) filter type, dictating the distance between the filter design's zero and the cutoff frequency.
- Computes the stopband edge, `stB_E`, utilizing the equation $\frac{BW \cdot a}{f_s/2}$, aiding in the high-pass filter construction with the `cheby2` function.
- Derives the transfer function's numerator and denominator through the `zp2tf` MATLAB function, forming the $H(z^{-1})$ transfer function in the preferred $\mathcal{Z}$-domain.
- Calculates the crucial **Hinf** value, the coefficient of z at infinity, by extracting the numerator's first coefficient, instrumental in $\mathcal{Z}$-domain NTF scaling for optimal OBG attainment.
- Formulates the NTF by dividing all numerator coefficients by **Hinf**, facilitating the transition to a Laplace domain Loop Filter per the ADC's continuous attributes.
- Returns the calculated values of a (alpha), **Hinf**, and the NTF, setting the stage for subsequent design phases, including the transition to a continuous-time Laplace domain Loop Filter.

Consequently, the OBG is deduced as **Hinf**'s reciprocal, equating to approximately **1.4947** in magnitude (not in dB), playing a significant role in modulating the modulator's characteristics and augmenting the ADC's SNR.

The flowchart describing the Matlab code (see Figure 2.9) and the Matlab code itself are presented below, see Script 2.1.

[4]The `ntfMaker` function is an original creation of this research and is not a part of the Sigma Delta Toolbox [13].

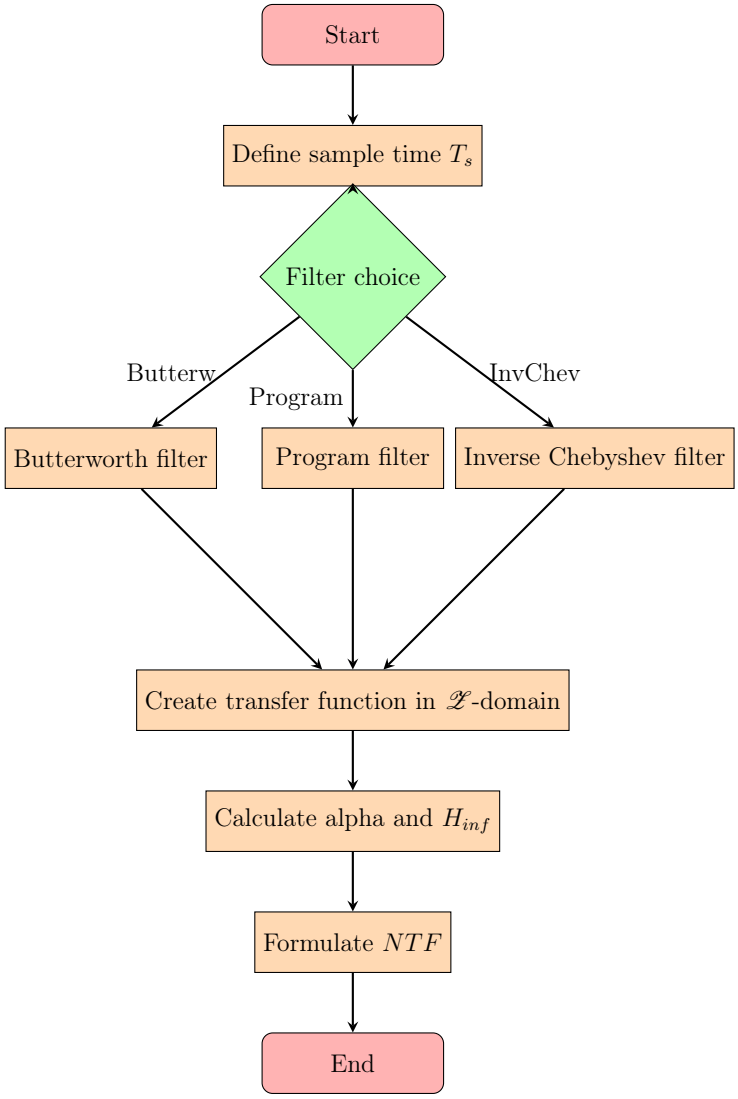


Figure 2.9: Flowchart of the Matlab code for generating the OBG.

Script 2.1: OGB for a chosen transfer function

```
function [alpha, Hinf, NTF] = ntfMaker(BW, x, stB_A, fs, order, choice)
% This function generates the NTF for a given filter type and parameters.

% Define the sample time as the reciprocal of the sampling frequency
Ts = 1/fs;

% Choose the filter type and calculate necessary parameters
switch choice
    case 'Butterw'
        a = BW*100;  % Define the cutoff frequency parameter for Butterworth
            filter
        [zH, pH, kH] = butter(order, a/(fs/2), 'high');  % Construct the
```

```
            high-pass filter
12
13      case 'InvChev'
14          a = sqrt(3)/2*x;  % Define the cutoff frequency parameter for
                Inverse Chebyshev filter
15          stB_E = (BW*a^-1)/(fs/2);  % Calculate the stopband edge
16          [zH, pH, kH] = cheby2(order, stB_A, stB_E, 'high');  % Construct the
                high-pass filter
17
18      case 'Program'
19          a = sqrt(2);  % Define the cutoff frequency parameter for the custom
                program filter
20          stB_E = (BW*a)/(fs/2);  % Calculate the stopband edge
21          [zH, pH, kH] = cheby2(order, stB_A, stB_E, 'high');  % Construct the
                high-pass filter
22
23      otherwise
24          error('Invalid choice. Please choose between Butterw, InvChev, or
                Program.');
25  end
26
27  % Get the numerator and denominator of the transfer function
28  [numH, denH] = zp2tf(zH, pH, kH);
29
30  % Create the transfer function in the Z-domain
31  H = tf(numH, denH, Ts, 'Variable','z^-1');
32
33  % Calculate alpha and Hinf for OBG optimization
34  alpha = a;
35  Hinf = numH(1);
36
37  % Formulate the NTF using the calculated Hinf value
38  numNTF = numH/Hinf;
39  denNTF = denH;
40
41  % Create the NTF transfer function
42  NTF = tf(numNTF, denNTF, Ts, 'Variable','z^-1');
43  end
```

2.3.2 Noise and Loop-filter Transfer Function

This section delves into the characterization of the inverse Chebyshev filter, focusing on maintaining an OBG of ≈1.49, as highlighted in the previous subsection. Initially, the zeros of the **NTF** were set at the bandwidth (BW), enabling high-pass type noise shaping. The Sigma-Delta toolbox [13] was employed to find the optimal zero positions for the defined 2nd order, OSR, and OBG.

The subsequent step identifies the zeros and poles of the NTF, highlighting that the zeros of the NTF correspond to the poles of $L(s)$. Initially placed at a 20kHz bandwidth, these were adjusted to **14.7kHz** using the Sigma-Delta toolbox. The comparison of the initial and optimized **NTFs** is presented below:

$$NTF(z) = \frac{1 - 2z + z^2}{1 - 1.225z + 0.4508z^2} \tag{2.11}$$

$$NTF_{\text{optimized}}(z) = \frac{1 - 2z + z^2}{z^2 - 1.219z + 0.4477} \tag{2.12}$$

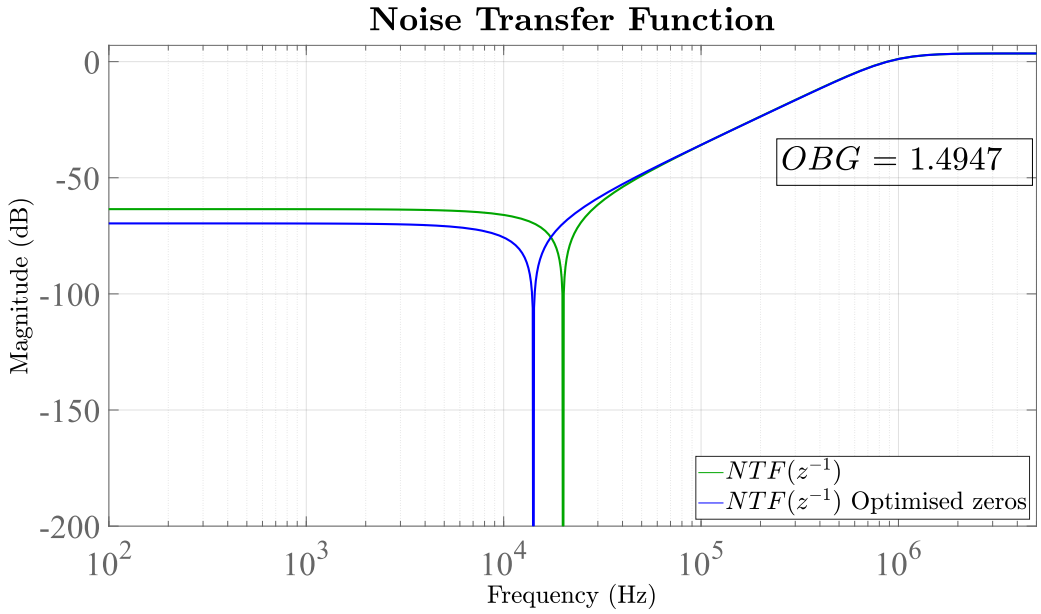


Figure 2.10: Comparison between transfer function with and without zeros optimisation.

The next phase involves retracing from the NTF to define $L(s)$, as referenced in equation 2.3. This necessitates the conversion of the loop filter from the discrete to the continuous domain using the **forward differencing** method [7], $s = \frac{z-1}{T_s}$, culminating in:

$$L(s) = \frac{0.0008646s + 3038}{1.266 \times 10^{-10}s^2 + 1} \tag{2.13}$$

The structure of the loop filter, illustrated in Figure 2.4, informs the initial formulation of the transfer function $L(s)$, as detailed in equation 2.3:

$$L(s) = \frac{k_2}{g} \frac{1 + \frac{k_1}{k_2 f_s} s}{1 + \frac{s^2}{g f_s^2}} \text{ with values } \begin{cases} k_1 & = 0.6667 \\ k_2 & = 0.2287 \\ g & = 7.5300 \cdot 10^{-5} \\ f_s & = 10.24 \cdot 10^6 \end{cases} \tag{2.14}$$

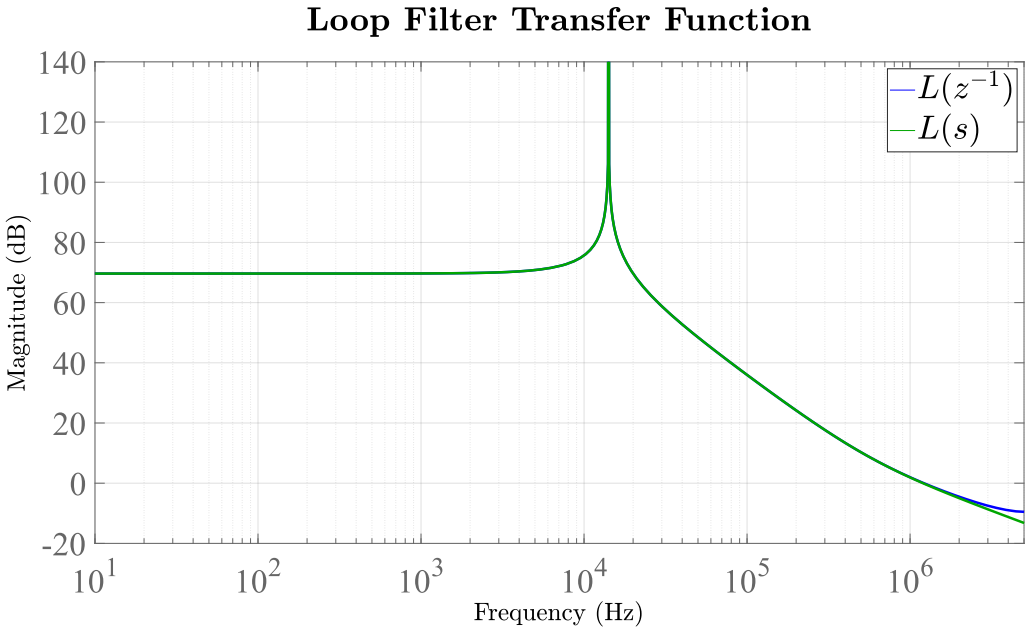


Figure 2.11: Loop filter comparison between discrete and continuous time.

Our objective is to rewrite this function in the canonical form:

$$L(s) = K\frac{1+s\tau}{1+\frac{2\xi}{\omega_n}s+\frac{s^2}{\omega_n^2}} \tag{2.15}$$

We identify the parameters K, τ, ξ, and ω_n as follows:

$$\begin{cases} K &= \frac{k_2}{g} \\ \tau &= \frac{k_1}{k_2 f_s} \\ \xi &= 0 \\ \omega_n &= \sqrt{f_s^2 g}. \end{cases} \tag{2.16}$$

Following this procedure, we derive the **NTF** as a function of the coefficients k_1, k_2, g, and f_s:

$$NTF(s) = \frac{1}{1+L(s)} = \frac{f_s^2 g + s^2}{f_s^2 g + f_s^2 k_2 + f_s k_1 s + s^2} \tag{2.17}$$

The goal is to represent the **NTF** in the canonical form:

$$NTF(s) = K_2\frac{1+2\xi_1\frac{s}{\omega_{1_n}}+\left(\frac{s}{\omega_{1_n}}\right)^2}{1+2\xi_2\frac{s}{\omega_{2_n}}+\left(\frac{s}{\omega_{2_n}}\right)^2} \tag{2.18}$$

The parameters are identified as:

$$\begin{cases} \xi_1 & = 0, \\ \omega_{1_n}^2 & = g f_s^2, \\ \omega_{2_n}^2 & = f_s^2(g + k_2), \\ K_2 & = \frac{g+k_2}{g}. \end{cases} \tag{2.19}$$

By substituting these parameters into the canonical form, we obtain the **NTF** as:

$$NTF(s) = \frac{g + k_2}{g} \frac{1 + \dfrac{s^2}{g f_s^2}}{1 + \dfrac{2}{\sqrt{f_s^2(g + k_2)}} s + \dfrac{s^2}{f_s^2(g + k_2)}} \tag{2.20}$$

Finally, it is essential to note that the *poles* of L are the **zeros** of the **NTF**, establishing a vital correlation in the analysis of the noise transfer function. To further elucidate, we can calculate the poles directly from the coefficients k_1, k_2, g, and f_s as follows:

$$s_{\text{poles L(s)}} = s_{\text{zeros NTF(s)}} = \pm j\sqrt{g} f_s, \quad f_{\text{poles or zeros}} = \frac{|\pm j\sqrt{g} f_s|}{2\pi} \tag{2.21}$$

2.3.3 Signal Transfer Function

The objective of the ADC's signal transfer function (STF) is to digitise the input signal $u(t)$ without amplification, maintaining a 0 dB magnitude and a 90° phase margin to ensure stability. As illustrated in equation 2.3, the canonical form reveals a zero and two poles. The NTF's complex conjugate zeros and the poles of $L(s)$, represented by $\left(1 + \frac{s^2}{g f_s^2}\right)$, nullify each other, simplifying to:

$$STF(s) = NTF(s) \cdot L(s) = \frac{\dfrac{(g + k_2)k_2}{g} \cdot \left(1 + \dfrac{k_1}{k_2 f_s} s\right)}{\left(1 + \dfrac{2}{\sqrt{f_s^2(g + k_2)}} s + \dfrac{s^2}{f_s^2(g + k_2)}\right)} \tag{2.22}$$

The **poles** and zeros of $STF(s)$ are defined as:

$$s_{\text{poles STF}} = \left\{\pm j\sqrt{g + k_2} f_s\right\} \tag{2.23}$$

$$f_{\text{poles STF}} = \frac{|s_{\text{poles STF}}|}{2\pi} \tag{2.24}$$

Instead, the **zeros** of $STF(s)$ are:

$$s_{\text{zeros STF}} = \left\{ -\frac{k_2 f_s}{k_1} \right\} \tag{2.25}$$

$$f_{\text{zeros STF}} = \frac{|s_{\text{zeros STF}}|}{2\pi} \tag{2.26}$$

Figure 2.12 illustrates a noticeable peaking around 0.9MHz, resulting from the CIFF loop filter architecture implementation, which induces an overshoot near 1MHz. However, given the 20kHz signal bandwidth, this will not affect the performance. Since no phase variation exists, the phase margin remains stable at values greater than 90° degrees.

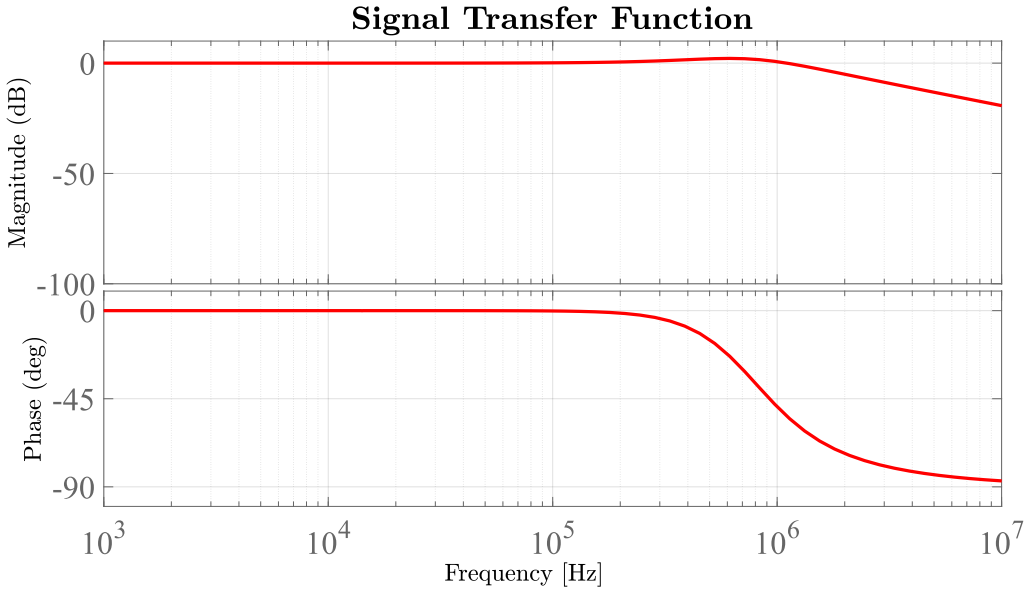


Figure 2.12: Illustration of the ADC's signal transfer function.

2.3.4 Input Signal Analysis

In line with the Anser EMT specifications, the parameters for the input signal $u(t)$ have been defined as:

- **Amplitude:** 125 mV (zero-peak).
- **Tone Frequency:** 17.65 kHz.

A test tone with a frequency close to the bandwidth (BW) is employed to quantify the ADC model, the Signal to Quantisation Noise performance, and thus evaluate the ADC's performance limits.

Additionally, the input signal frequency is determined by the spectral resolution required for analysis. The *output spectrum* of the ADC model, necessary for calculating the SNR, is obtained using FFT, which implies circular convolution. The computation requires a frequency resolution or FFT bin (Δf), defined as $\Delta f = \frac{f_s}{N}$ (f_s refers to equation 2.5). It depends on the number of signal points N, which represent the Simulink **simulation time** and needs to be a power of two to prevent error due to the FFT algorithm used.

The input signal frequency $f_{u(t)}$ must be an **integer** and prime multiple k of Δf, constrained by $f_{u(t)} \leq \text{BW}$, where k is a natural number. Therefore, avoiding *spectrum* leakage ensures FFT algorithm efficiency. More details are available in Appendix A.

$$f_{u(t)} = k \cdot \Delta f \leq \text{BW}, \quad \text{where } k \in \mathbb{N} \tag{2.27}$$

2.3.5 Ideal model with Clock Jitter

This subsection evaluates the performance of the ADC model. Figure 2.13 illustrates three distinct plots corresponding to the output spectra from various Simulink model simulations with different standard deviations for the *periodic jitter*, $\sigma_{p_{\text{jitter}}}$:

- The black plot represents the output spectrum of the model with a perfect clock source, where no jitter is included, thus $\sigma_{p_{\text{jitter}}} = 0$ ps. In this case, the spectrum precisely aligns with the NTF outlined in Figure 2.10.

- The red plot illustrates a scenario with $\sigma_{p_{\text{jitter}}} = 10$ ps, where the *periodic jitter* begins to elevate the in-band noise by increasing the quantization noise level. The effects of the NTF complex conjugates are null at 14.7 kHz.

- The last blue plot showcases the worst-case scenario where the substantial *periodic jitter* causes a significant rise in the noise floor, altering the high-pass quantization noise shaping beyond 20 kHz, indicating an incorrect NTF, thereby highlighting the severity of jitter in CT$\Delta\Sigma$M.

The output spectrum of the quantiser is significantly affected by the **clock jitter** modulation in the DAC pulses. This influence deteriorates the SNR

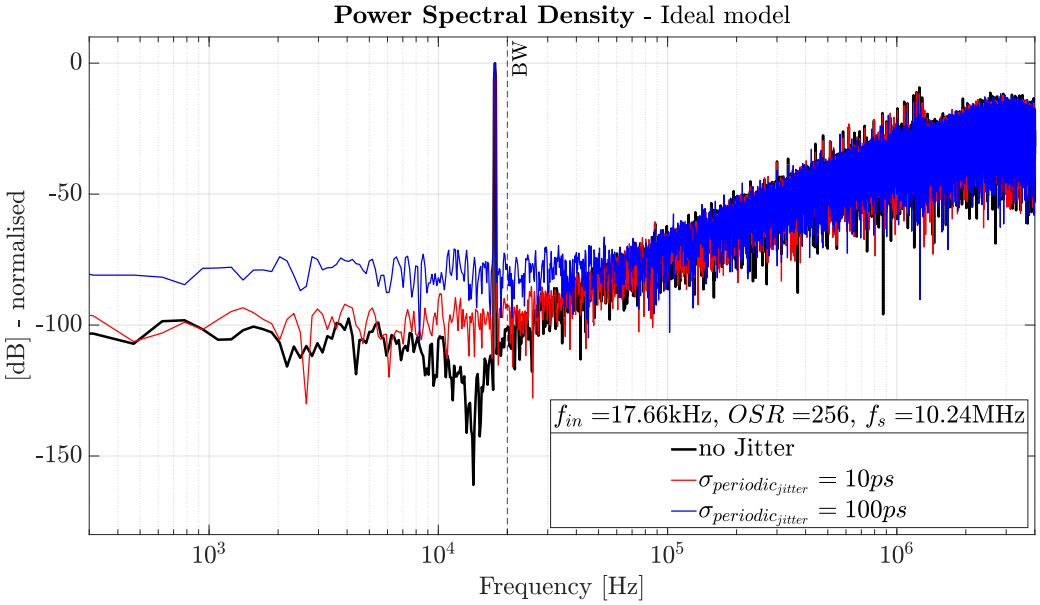


Figure 2.13: ADC's output spectra.

by amplifying the **in-band noise** within the BW. Essentially, the in-band quantisation noise, which the designed NTF uniformly suppresses, experiences a rise, as depicted in Figure 2.15. This phenomenon is attributed to the **white spectrum** *periodic jitter* of the model (refer to Figure 2.6 and equation 2.6).

A detailed examination of the plot reveals an elevation in the FFT bins with the integration of *periodic jitter* into the model. Despite this, the positions of the input signal bins remain unchanged since the input signal $u(t)$ remains constant. The simulation adopted a resolution of $N = 2^{16}$ throughout the duration, yielding a Δf of 156.25 Hz for a rectangular window. However, a **Hanning window** is utilized here, distributing the input signal power across three bins and establishing an equivalent noise bandwidth of 234.38 Hz, representing the frequency resolution for each bin in figures 2.13 and 2.14.

The CT$\Delta\Sigma$M exhibits sensitivity to **clock jitter**, as depicted in Figure 2.13. This factor results in a reduced in-band signal-to-noise ratio (SNR), documented in Figure 2.15, delineating the SNR response in 10 ps incremental steps. Initially, the system maintains a **85 dB SNR** under an ideal clock condition, exceeding the Anser EMT SNR benchmark of 70-80 dB. However, it diminishes to **57 dB SNR** when a 100 ps standard deviation of *periodic jitter* is introduced to the clock source, as described in equation 2.10.

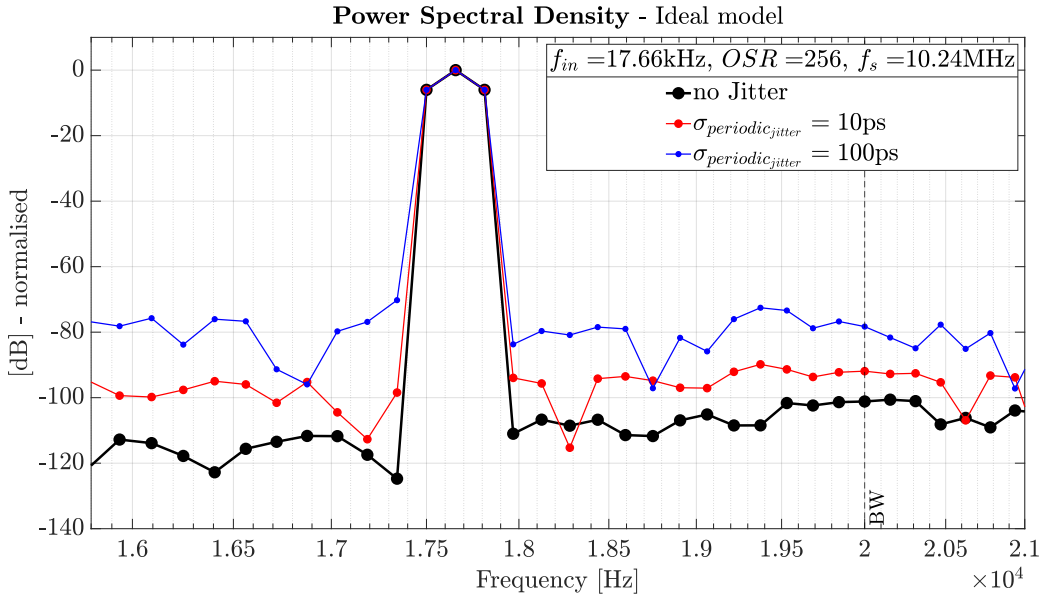


Figure 2.14: Magnified view of the ADC's output spectra.

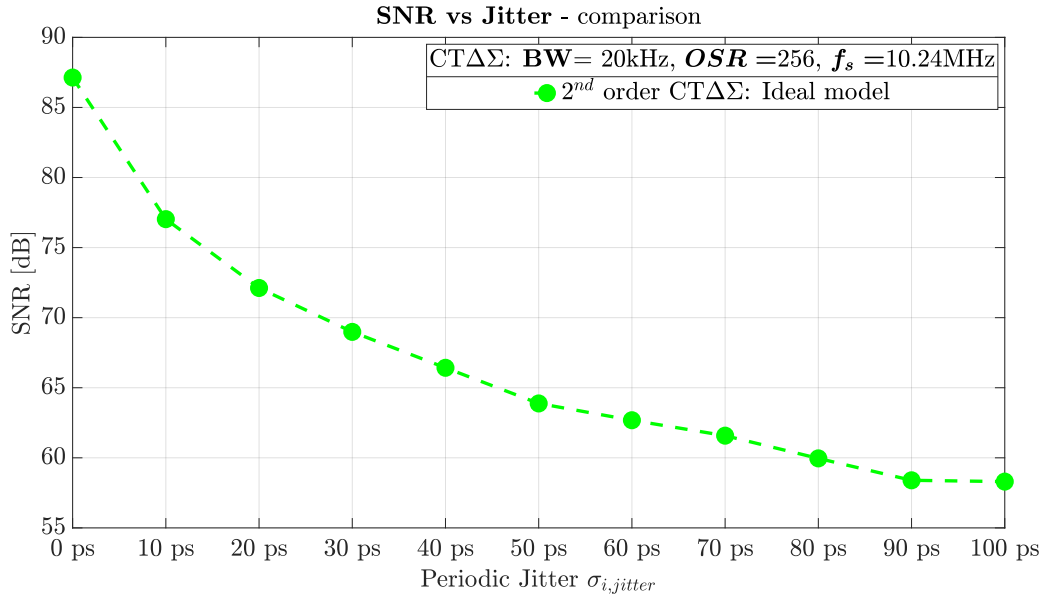


Figure 2.15: SNR versus Periodic jitter performaces.

The increase in quantization noise due to the *periodic jitter* noise applied to the DAC can be quantified and anticipated. Before delving further, it is essential to understand the quantization noise variance inherent in the system. Given the **1-bit** CTΔΣM and the project requirement peak-to-peak voltage

(V_{pp}) of 1.2V, it is possible to calculate the quantization step size (Δ) using $\Delta = \frac{V_{pp}}{M}$, where $M = 2^{\text{bit}}$ denotes the number of quantization levels. This yields a quantization noise variance of $\sigma^2_{quant.} = 0.03$, determined through the formula $\sigma^2_{quant.} = \left(\frac{\Delta}{\sqrt{12}}\right)^2$.

With this understanding, we can proceed to analyze the equation 2.28, which, while aligned with equations(2.6) and(2.7), proposes a Gaussian standard deviation due to the randomness of the process. Nevertheless, it **does not** account for the *jitter spectrum*. This equation, illustrated in Figure 2.16, is pertinent only to a *white spectrum model* [9], [14], [15]:

$$IBN(e^{j\omega}) = \left(\frac{\sigma_{i_{\text{jitter}}}}{T_s}\right)^2 \frac{\Delta^2}{12\pi \text{ OSR}} \int_0^{\pi} \left|NTF(e^{j\omega})(1 - e^{-j\omega})\right|^2 d\omega \qquad (2.28)$$

However, this equation lacks completeness and is confined to the scope of a *white spectrum model*. For a more nuanced and comprehensive analysis of the IBN, refer to Section 3.4, equation(3.85), which elucidates how the **phase noise spectrum** of a generic clock source significantly curtails the performance of the CT$\Delta\Sigma$ modulator.

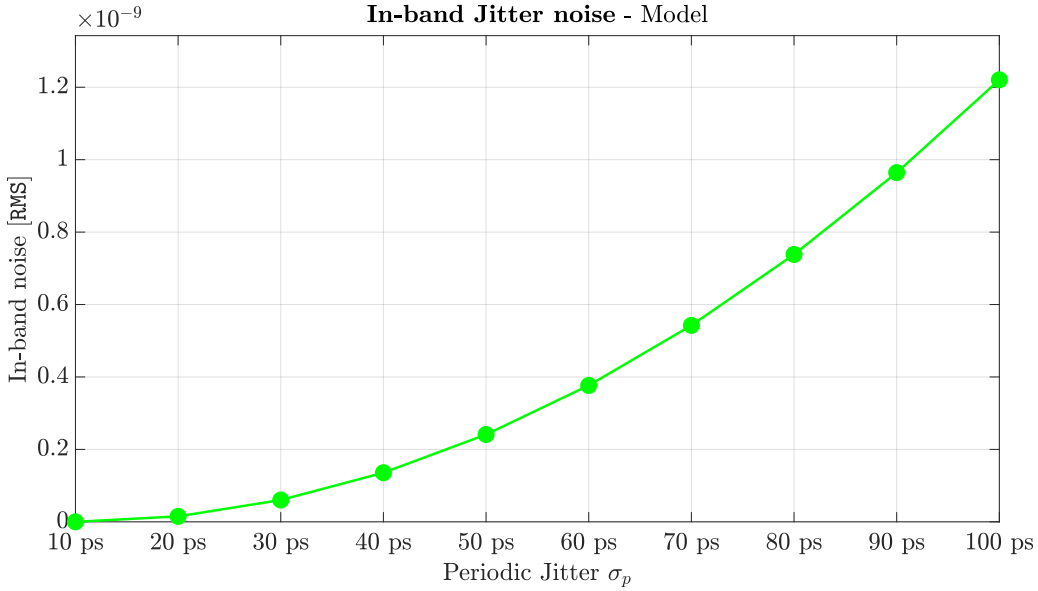


Figure 2.16: Measured RMS in-band noise versus rms periodic jitter.

NTF Verification through Simulation

The validity of the Simulink model **simulations** waveforms generated can be confirmed by evaluating the NTF and, consequently, the overall performance of the CTΔΣM. Utilizing the block diagram depicted in Figure 2.4, the signals $v[n]$, representing the ADC discrete output, and $y[n]$, the discretised continuous time loop filter output, are analyzed. The *power spectral density* (PSD) of $v[n]$ and of the difference $v[n] - y[n]$ are computed. Subsequently, the division of these PSDs yields the following equation:

$$|NTF_{\text{calculated}}(e^{j\omega})|^2 = 10\log\left(\frac{PSD(v[n])}{PSD(v[n]-y[n])}\right) \tag{2.29}$$

The quantization noise appears to be shaped as anticipated. However, the confirmation of the NTF (shape and OBG) through the PSD visualization is challenged by the inherent noise in the PSD, originating from the stochastic nature of the quantization error. This error is explicitly available during simulations, facilitating a noise-free verification of the NTF through the division of PSDs, as demonstrated in equation 2.29.

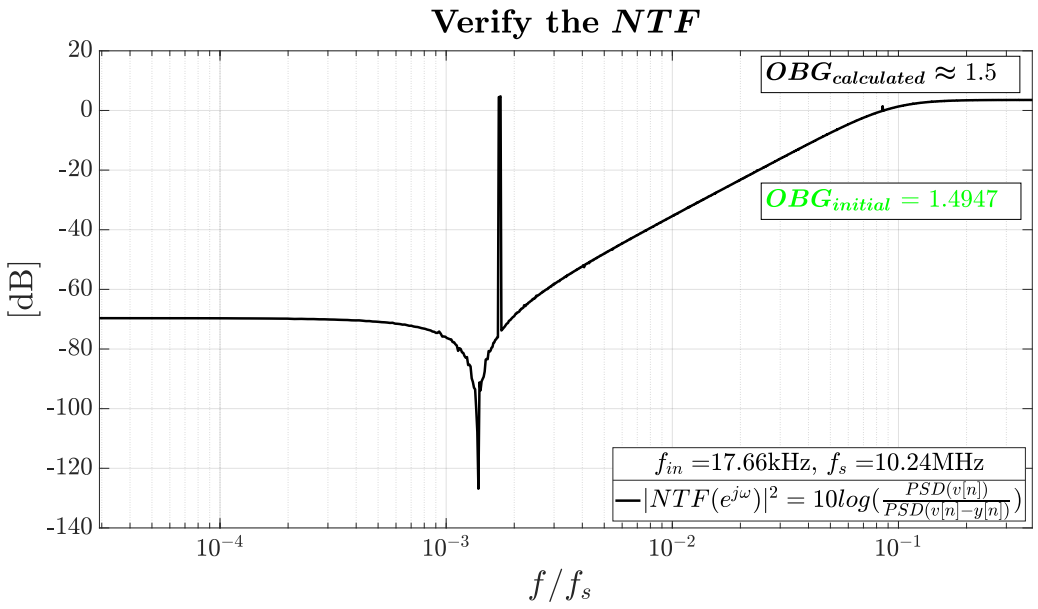


Figure 2.17: Noise Transfer Function Verification.

Figure 2.17 reveals a spike at the input signal power frequency, 17.66kHz,a consequence of the residual input signal in the division $\frac{PSD(v[n])}{PSD(v[n]-y[n])}$. Despite this, the outcome is accurate, aligning with the designed NTF in Figure 2.10

and yielding an OBG[5] close to 1.5. Given the initial OBG[5] of 1.4947, the validation is deemed successful.

2.4 FIR Filter implementation

To mitigate the SNR degradation posed by **white spectrum clock jitter**, incorporating a low-pass *M-tap* **FIR filter** in the feedback loop, specifically preceding the equivalent *DAC*, serves as a viable strategy. Hereafter, this combined entity will be referred to as the FIR-DAC [9], [14], [15], characterised by the filter formula $F(z^{-1})$. This modification in the discrete-time section of the CT$\Delta\Sigma$M modulator necessitates a systematic recalibration of various coefficients to retain the efficacy of the quantization noise shaping, a critical requirement for successfully implementing the Anser EMT. The recalibration encompasses:

- Adjusting the feedforward coefficients; the initial values k_1 and k_2 (refer to Figure 2.4) will transition to new estimations, denoted as $\hat{k}_1$ and $\hat{k}_2$.
- Including an FIR-DAC warrants the integration of a secondary FIR-DAC, functioning as a *Compensation Path*. This element, characterized by the transfer function $F_{\text{comp.}}(z^{-1})$, is instrumental in preserving the desired characteristics of the NTF, while leveraging the jitter resistance imparted by the primary FIR-DAC.
- The implementation of the compensation path necessitates the introduction of a generalised **delay** component within the feedback loop, denominated as t_{Delay}, which corresponds to a multiple of the sampling period T_s.

2.4.1 DAC and FIR DAC time response

Understanding the dynamics of feedforward coefficient scaling within the loop filter, structured based on the CIFF framework, requires a detailed examination of the time response of a **1-bit** *non-return-to-zero* (NRZ) DAC. Let us denote the time response as $p(t)$ and the chosen sampling period as T_s. The **NRZ DAC** can be mathematically represented as:

$$p(t) = \begin{cases} 1 & 0 \leq t < T_s \\ 0 & \text{elsewhere} \end{cases} \tag{2.30}$$

[5]The *out-of-band gain* is expressed in magnitude and not dB [9].

The insertion of an FIR filter ahead of the NRZ DAC alters the time response, adding a summing or averaging process to the DAC pulse response to attenuate the impact of *periodic jitter* on the DAC. This concept stems from the stochastic nature of jitter, which introduces errors at the transition points of the clock signal, thus influencing the DAC response. Averaging is a strategy to **minimise** this variance, converging towards the true value over a theoretically infinite number of iterations. The FIR filter facilitates this process by sampling historical data and applying weighted coefficients, akin to averaging a random variable. The transfer function defines the filter:

$$F(z^{-1}) = \sum_{i=0}^{M-1} \alpha_i z^{-i} \tag{2.31}$$

Here, $F(z^{-1})$ denotes the $\mathcal{Z}$-domain transfer function of the FIR filter, where α_i represents the filter coefficients, M is the **filter order**, and z^{-1} signifies a unit delay in the $\mathcal{Z}$-domain. It's important to note that the summation extends from zero to the order minus one, given that the first coefficient, α_0, is always multiplied by z^0. To effectively mitigate the influence of periodic jitter, the transfer function presented in equation 2.31 evolves as follows:

$$F(z^{-1}) = \sum_{i=0}^{M-1} \frac{\mathbf{1}}{\boldsymbol{M}} z^{-i} = \frac{\mathbf{1}}{\boldsymbol{M}} \sum_{i=0}^{M-1} z^{-i} \tag{2.32}$$

Adopting uniform coefficients, equivalent to the reciprocal of the filter order, facilitates the effective **averaging** of the DAC input signal $v[n]$, thereby mitigating the jitter influences on $v(t)$. The prescribed sequence for this procedure is delineated below:

1. Acquire the discrete ADC output $v[n]$;
2. Implement the FIR filter;
3. Incorporate the white spectrum jitter model, detailed in Figure 2.7;
4. Utilize an ideal Simulink block representing the 1-bit NRZ DAC;
5. Generate the precise analog output $v(t)$.

Strict adherence to this sequence is vital to preserving the **integrity** of the clock jitter model. This approach necessitates treating the ideal NRZ DAC and its preceding additive model as singular entities, which echoes real-world implementations. Refer to Figure 2.18 for a schematic depiction.

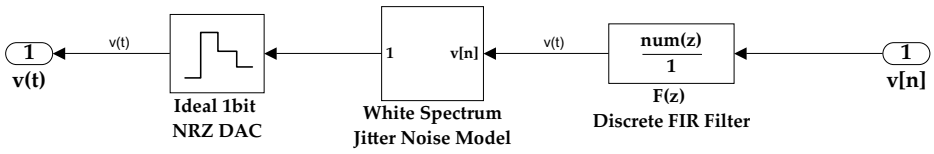


Figure 2.18: Schematic Illustration of the Correct FIR Filter Application.

The **FIR-DAC**'s time response diverges from equation 2.30. Given the averaging delineated in transfer function 2.32, it can be characterized as:

$$q(t) = \begin{cases} \frac{1}{M} & 0 \leq t < T_s \\ \frac{1}{M} & T_s \leq t < 2T_s \\ \frac{1}{M} & 2T_s \leq t < 3T_s \\ \vdots & \\ \frac{1}{M} & (M-1)T_s \leq t < MT_s \\ 0 & \text{elsewhere} \end{cases} = \begin{cases} \frac{1}{M} & 0 \leq t < MT_s \\ 0 & \text{elsewhere} \end{cases} \tag{2.33}$$

The time responses $p(t)$ and $q(t)$, delineated in equations 2.30 and 2.33 respectively, are pivotal in defining the new feedforward coefficients $\hat{k}_1$ and $\hat{k}_2$.

2.4.2 Feedforward Coefficient Scaling

To accurately scale the coefficients k_1 and k_2, essential components of the CIFF architecture within the loop filter $L(s)$, a detailed analysis of the *open-loop pulse response* of $L(s)$ is necessary. This involves evaluating the **time response** of the outputs of each integrator, $x_1(t)$ and $x_2(t)$. Initially, analyse the CIFF structure without applying feedback g to maintain simplicity, as shown in Figure 2.19.

Utilising the Dirac delta function, $\delta(t)$, to represent the pulse and considering the responses $p(t)$ and $q(t)$ defined in Equations 2.30 and 2.33 respectively,

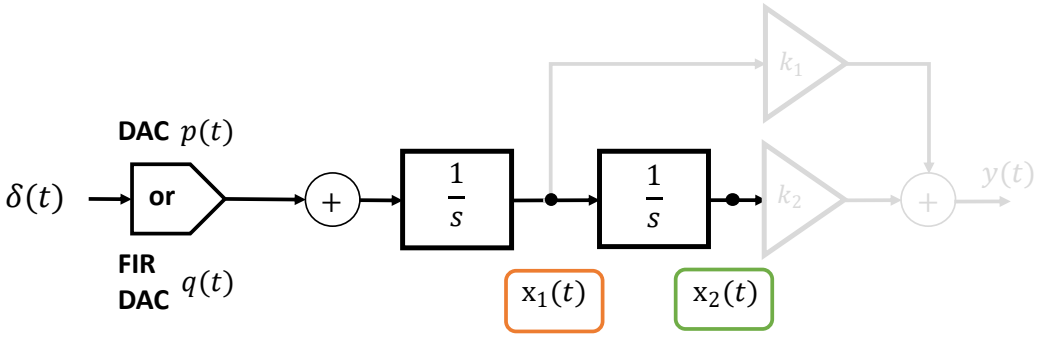


Figure 2.19: Simplified CIFF structure.

the *time response* for each integrator can be formulated as:

$$\text{NRZ DAC:}$$
$$\begin{cases} x_{1_{p(t)}}(t) &= p(t) * \int \delta(t)\,dt = p(t) * u_0(t) \\ x_{2_{p(t)}}(t) &= p(t) * \int \left(\int \delta(t)\,dt\right) d\tau = p(t) * tu_0(t) \end{cases} \tag{2.34}$$
$$\text{FIR-DAC:}$$
$$\begin{cases} x_{1_{q(t)}}(t) &= q(t) * \int \delta(t)\,dt = q(t) * u_0(t) \\ x_{2_{q(t)}}(t) &= q(t) * \int \left(\int \delta(t)\,dt\right) d\tau = q(t) * tu_0(t) \end{cases}$$

The integrator chain within the CIFF transforms the initial pulse response into the **step response** $u_0(t)$, derived from $\int \delta(t)dt$. The subsequent integrator, i.e. $\int u_0(\tau)d\tau$, further modifies this response into a **ramp function** $tu_0(t)$.

Incorporating the loop-filter feedback g, which integrates complex conjugate poles into $L(s)$, complicates the system, making time-domain open-loop analysis challenging. Hence, recalling a property of the inverse Laplace transform proves beneficial, facilitating direct derivation of the system's time response. The transfer functions for the first and second integrators are given by $X_1(s)/U(s) = s/(g + s^2)$ and $X_2(s)/U(s) = 1/(g + s^2)$, respectively. The inverse Laplace transformation of these functions would yield periodic time-domain responses, making them impossible to use. Yet, the CIFF output is constructed as a linear combination of the responses from both integrators. Using the model's linear time-invariant nature [8], which means $Y(s) = k_1X_1(s) + k_2X_2(s)$, decoupling the contributions from each integrator

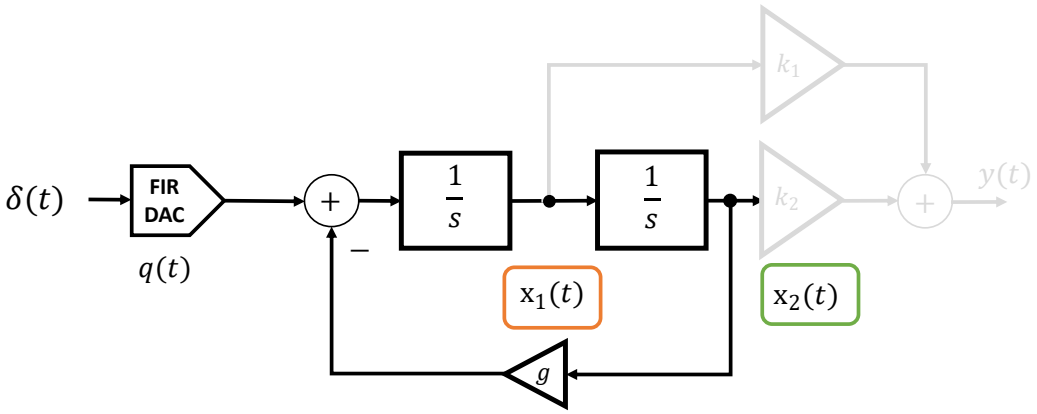


Figure 2.20: Open-loop structure with FIR-DAC.

response is feasible [7], [8]. This separation simplifies the equations to:

$$\begin{cases} X_1(s) = \dfrac{1}{s+g} \\ \\ X_2(s) = \dfrac{1}{s(s+g)} \end{cases} \tag{2.35}$$

Consequently, the system's time responses are given by:

NRZ DAC:

$$\begin{cases} x_{1_{p(t)}}(t) = p(t) * \mathscr{L}^{-1}\left\{\dfrac{1}{s+g}\right\} = p(t) * \dfrac{e^{-gt}}{g} \\ \\ x_{2_{p(t)}}(t) = p(t) * \mathscr{L}^{-1}\left\{\dfrac{1}{s(s+g)}\right\} = p(t) * \dfrac{1-e^{-gt}}{g} \end{cases} \tag{2.36}$$

FIR-DAC:

$$\begin{cases} x_{1_{q(t)}}(t) = q(t) * \mathscr{L}^{-1}\left\{\dfrac{1}{s+g}\right\} = q(t) * \dfrac{e^{-gt}}{g} \\ \\ x_{2_{q(t)}}(t) = q(t) * \mathscr{L}^{-1}\left\{\dfrac{1}{s(s+g)}\right\} = q(t) * \dfrac{1-e^{-gt}}{g} \end{cases}$$

This approach simplifies the convolution operation in the time domain, enabling straightforward computation of the final results, using Equations 2.30

and 2.33:

$$\text{NRZ DAC:}$$
$$\begin{cases} x_{1_{p(t)}}(t) = p(t) * \dfrac{e^{-gt}}{g} = \dfrac{e^{g}-1}{g}e^{-gt} \\ \\ x_{2_{p(t)}}(t) = p(t) * \left(1 - \dfrac{e^{-gt}}{g}\right) = \dfrac{1}{g} - \dfrac{e^{g}-1}{g^2}e^{-gt} \end{cases} \tag{2.37}$$
$$\text{FIR-DAC:}$$
$$\begin{cases} x_{1_{q(t)}}(t) = q(t) * \dfrac{e^{-gt}}{g} = \dfrac{e^{Mg}-1}{Mg}e^{-gt} \\ \\ x_{2_{q(t)}}(t) = q(t) * \left(1 - \dfrac{e^{-gt}}{g}\right) = \dfrac{1}{g} - \dfrac{e^{Mg}-1}{Mg^2}e^{-gt} \end{cases}$$

Equation 2.37 presents the four distinct solutions for open-loop responses, both with and without the implementation of an FIR-DAC, as functions of the filter transfer function's order M (as defined in equation 2.32) and the loop filter feedback g. These solutions are unique to this specific loop filter architecture. They cannot be applied elsewhere without re-evaluating the *open-loop pulse response* response for the new loop filter implementation, as equation 2.35 would no longer be applicable.

Now, considering the CIFF structure, shown in Figure 2.4 and 2.20, the output of the loop filter can be evaluated as follows:

$$\begin{cases} y_{p(t)}(t) = k_1 x_{1_{p(t)}}(t) + k_2 x_{2_{p(t)}}(t) \\ \\ y_{q(t)} = \hat{\mathbf{k}}_{\mathbf{1}} x_{1_{q(t)}}(t) + \hat{\mathbf{k}}_{\mathbf{2}} x_{2_{q(t)}}(t) \end{cases} \tag{2.38}$$

Here, $y_{p(t)}$ symbolizes the ideal response with the original coefficients $k_1 = 0.6667$ and $k_2 = 0.2287$ (as defined in equation 2.14), while $y_{q(t)}$ represents the response with the new coefficients $\hat{k}_1$ and $\hat{k}_2$ to be determined. However, equation 2.37 features an exponential component, e^{-gt}, which complicates the comparison of the two time-domain loop filter outputs. Thus, a solution for the system is sought:

$$\boldsymbol{y_{p(t)} = y_{q(t)}} \tag{2.39}$$

Equation (2.39) ensures that the FIR-DAC will not affect the feedforward coefficients, facilitating the determination of $\hat{\mathbf{k_1}}$ and $\hat{\mathbf{k_2}}$ as functions of $f(k_1, k_2, g, M)$. Consequently, a first-order approximation is utilized to maintain the system solution, adjusting through Taylor expansion. For simplicity, $y_{p(t)}$ is considered, although the analysis applies equivalently to $y_{q(t)}$:

$$\begin{cases} y_{q(t)} = \left(k_1 \dfrac{e^g - 1}{g} + k_2 \dfrac{1 - e^g}{g^2} \right) e^{-gt} + \dfrac{k_2}{g}, \\ \mathbf{e^{-gt} \approx 1 - gt} + \mathcal{O}(t^2). \end{cases} \tag{2.40}$$

Here, the term $\mathcal{O}(t^2)$ denotes the order of magnitude of the approximation error, indicating that terms of order t^2 and higher are neglected. This simplification provides a tractable expression while maintaining a reasonable approximation accuracy for small values of t.

Combining the above expressions yields:

$$\begin{cases} y_{p(t)}(t) \approx \left(k_1 \left(\dfrac{e^g - 1}{g} \right) + k_2 \left(\dfrac{1 - e^g}{g^2} \right) \right) (1 - gt) + \dfrac{k_2}{g}, \\ y_{q(t)}(t) \approx \left(\hat{\mathbf{k}}_\mathbf{1} \left(\dfrac{e^{Mg} - 1}{Mg} \right) + \hat{\mathbf{k}}_\mathbf{2} \left(\dfrac{1 - e^{Mg}}{Mg^2} \right) \right) (1 - gt) + \dfrac{\hat{\mathbf{k}}_\mathbf{2}}{g}, \\ y_{p(t)}(t) = y_{q(t)}(t). \end{cases} \tag{2.41}$$

Consequently, the **final solution**, derived from the existence of two unknowns and two equations, leads to a unique solution, as facilitated by the approximation in equation 2.40:

$$\begin{cases} \hat{\mathbf{k}}_\mathbf{2} = k_2 \\ \hat{\mathbf{k}}_\mathbf{1} = k_1 \left(\dfrac{M(e^g - 1)}{e^{Mg} - 1} \right) + k_2 \left[\dfrac{1}{g} \left(1 - \dfrac{M(e^g - 1)}{e^{Mg} - 1} \right) \right] \end{cases} \tag{2.42}$$

It is affirmed that equation 2.42 holds since the CIFF structure invariably maintains the **last coefficient** in the chain of integrators (in this case, the second) **constant** [9], [14], [15] for every order of the CT$\Delta\Sigma$M.

2.4.3 Incorporation of Time Delay

Upon the introduction of the FIR-DAC, as detailed in equation 2.32, a challenge arises in addressing the DAC jitter. By invoking the **forward differencing** method, defined by $s = \frac{z-1}{T_s}$, an initial signal transfer function (STF) is formulated:

$$STF(s)_{1,\text{Incomplete}} = \frac{L(s)}{1 + L(s)F(sT_s + 1)} \neq \frac{L(s)}{1 + L(s)} \tag{2.43}$$

As deduced from equation 2.43, the BIBO stability is now **compromised**. To remedy the STF and NTF, two crucial modifications are suggested:

1. Introduce a *time delay*, expressed as $z^{-\frac{t_{\text{Delay}}}{T_s}}$. Through the forward differencing method, this becomes $(\mathbf{sT_s + 1})^{-\frac{\mathbf{t_{\text{Delay}}}}{\mathbf{T_s}}}$.
2. Construct a Compensation Path, to be elaborated in section 2.4.5.

This adjustment, visualized in Figure 2.21, leads to an evolving STF:

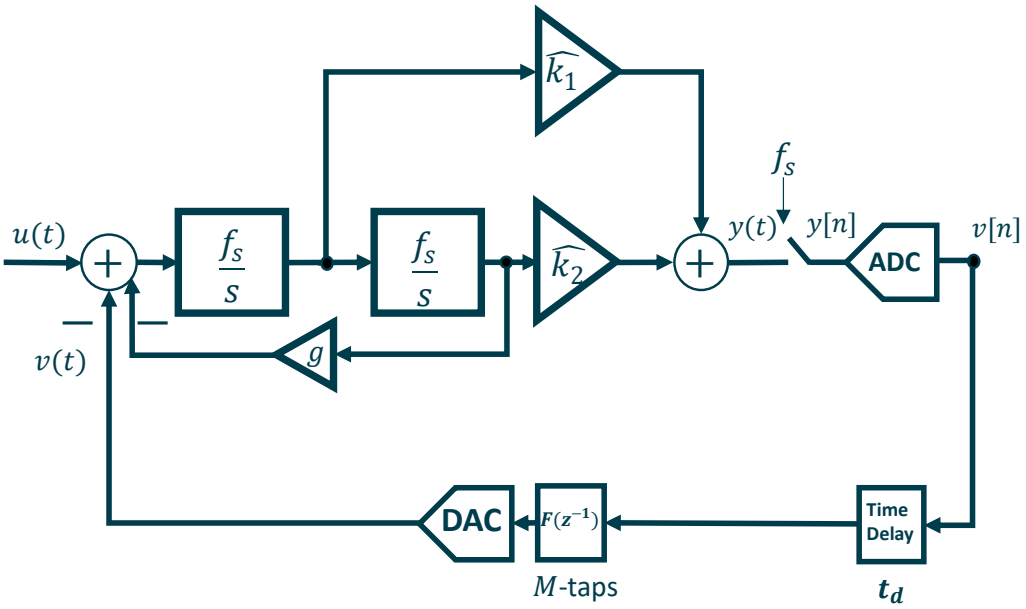


Figure 2.21: Preliminary CTΔΣM configuration with FIR-DAC.

The derived yet incomplete STF is given by:

$$STF(s)_{2,\text{Incomplete}} = \frac{L(s)}{1 + L(s)F(sT_s + 1)\left((\mathbf{sT_s + 1})^{-\frac{\mathbf{t_{\text{Delay}}}}{\mathbf{T_s}}}\right)} \tag{2.44}$$

Benefits of Time Delay

Incorporating a *time delay* not only addresses a system requisite but also strategically augments system performance [7]. Key benefits include:

- *System Stability*: The time delay amplifies stability, countering **meta-stability** issues and preventing undesired oscillations in the delta-sigma ADC feedback loop [15].

- *Filter Latency Compensation*: The delay counteracts latency from filters, such as the FIR, ensuring feedback and input signal alignment and bolstering system efficacy [11].

- *Noise-Shaping Performance Enhancement*: The delay refines the noise transfer function, shifting noise outside the desired band and enhancing noise-shaping performance [14].

Notably, an optimal time delay satisfies $t_{\text{delay}} \geq T_s$. Simulink simulations use a **fixed-step simulation** setting in this scenario. This choice maximizes the clock jitter model's efficiency, which alters the *amplitude* instead of signal edges, as indicated in equation 2.10.

2.4.4 Feedforward coefficient adjustment

Given that the *time delay block* **alters** the feedback, adjustments to the feedforward coefficients are necessary to compensate for this and preserve the desired NTF, as illustrated in equation 2.42. Given the characteristics of the CIFF, the new values can be deduced using the formula [8], [9]:

$$\begin{cases} k_{1_{\text{Delay}}} = k_1 + k_2 \dfrac{t_{\text{Delay}}}{T_s} \\ k_{2_{\text{Delay}}} = k_2 \end{cases} \tag{2.45}$$

Note that t_{Delay} is effective for any chosen time, yet utilizing an integer multiple of T_s proves more beneficial for simulation purposes. In the simulation, $\mathbf{t}_{\text{Delay}}$ is **selected** to equal $\mathbf{T_s}$, thereby securing the *meta-stability*.

Ultimately, the **final** feedforward coefficient, influenced by the ideal coefficients k_1 and k_2, the loop filter feedback g, the FIR-DAC order M, and the general time delay t_{Delay}, can be represented in two equivalent forms, emphasising the effect of the time delay on the coefficients. This is delineated

as:

$$\begin{cases} \hat{\mathbf{k_2}} = k_{2_{\text{Delay}}} \\ \hat{\mathbf{k_1}} = k_{1_{\text{Delay}}} \left(\dfrac{M(e^g - 1)}{e^{Mg} - 1} \right) + k_{2_{\text{Delay}}} \left[\dfrac{1}{g} \left(1 - \dfrac{M(e^g - 1)}{e^{Mg} - 1} \right) \right] \end{cases} \tag{2.46}$$

Or equivalently, taking into account the time delay:

$$\begin{cases} \hat{\mathbf{k_2}} = k_2 \\ \hat{\mathbf{k_1}} = \left(k_1 + k_2 \dfrac{t_{\text{Delay}}}{T_s} \right) \left(\dfrac{M(e^g - 1)}{e^{Mg} - 1} \right) + k_2 \left[\dfrac{1}{g} \left(1 - \dfrac{M(e^g - 1)}{e^{Mg} - 1} \right) \right] \end{cases} \tag{2.47}$$

Once more, it is reaffirmed that the proposition delineated in equation 2.42 extends to equation 2.47, underlining that the CIFF structure consistently preserves the **last coefficient** as a **constant** through every order of the CTΔΣM, even when accounting for the introduced time delay.

2.4.5 Compensation Path

This section discusses the compensation path, an essential component in the CTΔΣM feedback system. This path will *solve* and complete the system characterised by integrating FIR-DAC and time delay, key to significant improvements in system robustness against jitter.

The preliminary step in this analysis involves the integration of FIR-DAC and time delay, which unfortunately results in an incomplete system, as elucidated in equation 2.44. This arrangement, however, paves the way for verifying the accuracy of coefficients $\hat{\mathbf{k_1}}$ and $\hat{\mathbf{k_2}}$, as indicated in equation 2.47, through the comparative analysis of *open-loop pulse responses* [8], [9]. The pivotal stage in this process is the critical analysis of the open-loop filter time response $y(t)$, juxtaposed with the modified system.

Figure 2.22 (a) initiates the discourse by delineating the ideal *open-loop pulse response*, with a particular emphasis on the loop filter time response $y_{\text{Ideal}}(t)$, depicted in green. In Figure 2.22 (b), a detailed display of the response with the time-delay cell, **M-order** FIR-DAC, and the adjusted feedforward coefficient is presented, with the loop filter time response $\hat{\mathbf{y}}(\mathbf{t})$ marked distinctly in black. This illustration aids in affirming the validation of coefficients $\hat{\mathbf{k_1}}$

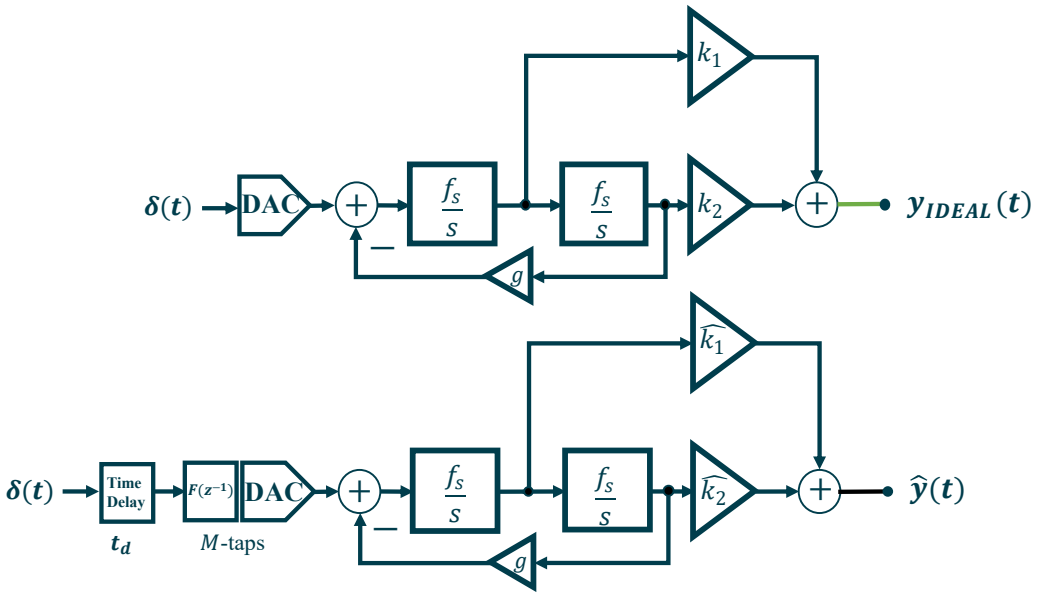


Figure 2.22: (a) Ideal and (b) FIR-DAC open loop response.

and $\hat{\mathbf{k}}_\mathbf{2}$, as evidenced by the following relationship:

$$y_{\text{Ideal}}(t) = \hat{\mathbf{y}}(\mathbf{t}), \quad \text{for } t \geq (M+1)T_s \tag{2.48}$$

The dynamics encapsulated in equation 2.48 can also be portrayed as the two responses $y_{\text{Ideal}}(t)$ and $\hat{\mathbf{y}}(\mathbf{t})$ must *coincide* for $\frac{t}{T_s} \geq M+1$, providing the **mathematical proof** for the correct coefficient scaling. This connection is further illustrated in Figure 2.23, which accentuates the persistent **amplitude differece** between the two *open-loop pulse responses*, adhering to equation 2.48, as delineated below:

$$|y_{\text{Ideal}}(nT_s) - \hat{\mathbf{y}}(\mathbf{nT_s})| > 0 \quad \text{for} \begin{cases} 0 \leq \dfrac{t}{T_s} \leq M \\ \\ \text{or equivalently} \\ \\ 1 \leq n \leq M, \quad \text{where } n \in \mathbb{N} \end{cases} \tag{2.49}$$

Equation 2.49 is shown in Figure 2.23 as the blue stem plot, and **corresponds** to the second FIR-DAC coefficients, alternatively termed as the *compensation path*, characterized by the transfer function $F_{\text{Comp.Path}}(z^{-1})$. This rationale advocates the sampling of time responses $y_{\text{Ideal}}(t)$ and $\hat{\mathbf{y}}(\mathbf{t})$ at

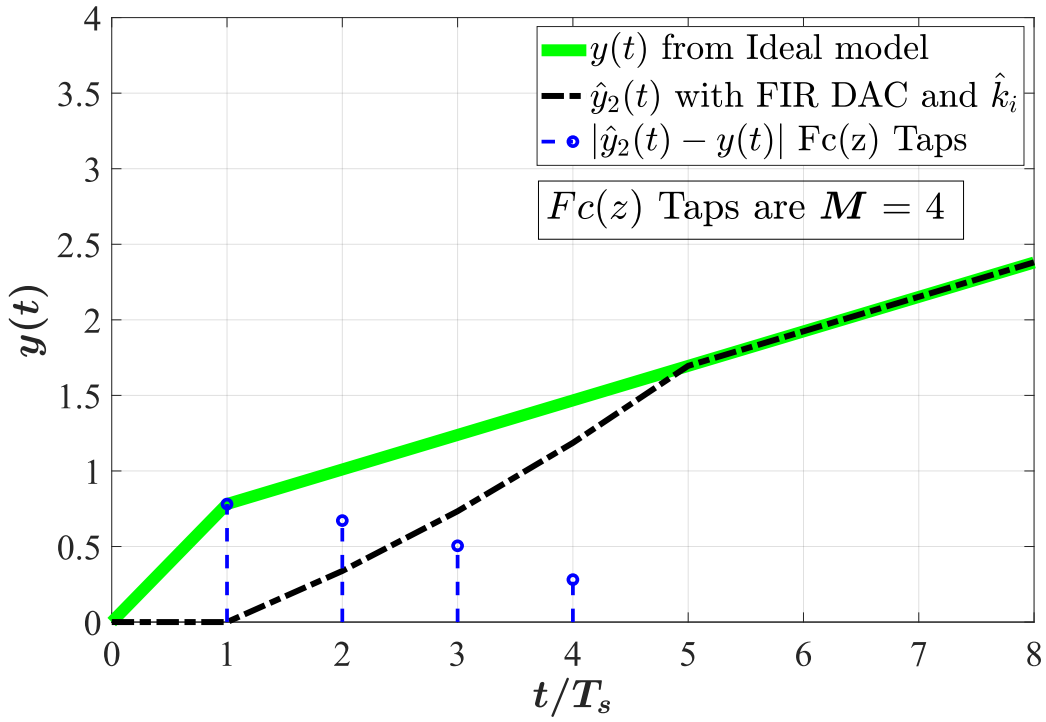


Figure 2.23: Amplitude difference between the ideal and FIR-DAC open loop responses.

intervals of $\frac{t}{T_s}$. Furthermore, equation 2.48, and consequently equation 2.49, infer that the FIR-DAC utilized in the CT$\Delta\Sigma$M feedback to average the jitter clock and the second FIR filter employed to restore the STF share the **same order M**.

Delving into the *open-loop pulse response* alleviates the complexities of scrutinizing a hybrid transfer function comprised of STF and NTF, a confluence of continuous-time and discrete elements. This approach lays the mathematical foundation necessary for optimizing the architecture to minimise the **impact** of clock **jitter** on the ADC. Post verification of equation 2.48 with the compensation path is M coefficients for the FIR-DAC, the model's complexity can be escalated, as depicted in Figure 2.24.

A critical insight obtained from Figure 2.23 is the comparative analysis between $y_{\text{Ideal}}(t)$ and $\mathbf{y}(\hat{\mathbf{t}})$, mathematically showing that $F_{\text{Comp. Path}}(z^{-1})$ should be integrated **solely** at the loop filter L **output**, where its time response has been analysed. This inclusion manifests as a **positive** feedforward influence exclusively for the *open-loop pulse response*, as illustrated in Figure 2.24, transitioning to a **negative** attribute in the closed-loop system.

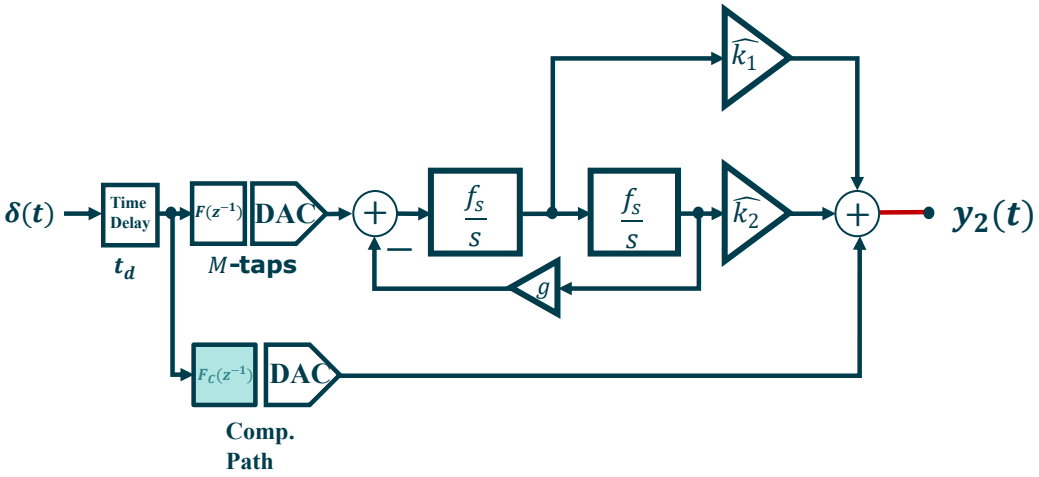


Figure 2.24: Complete open loop response with compensation path.

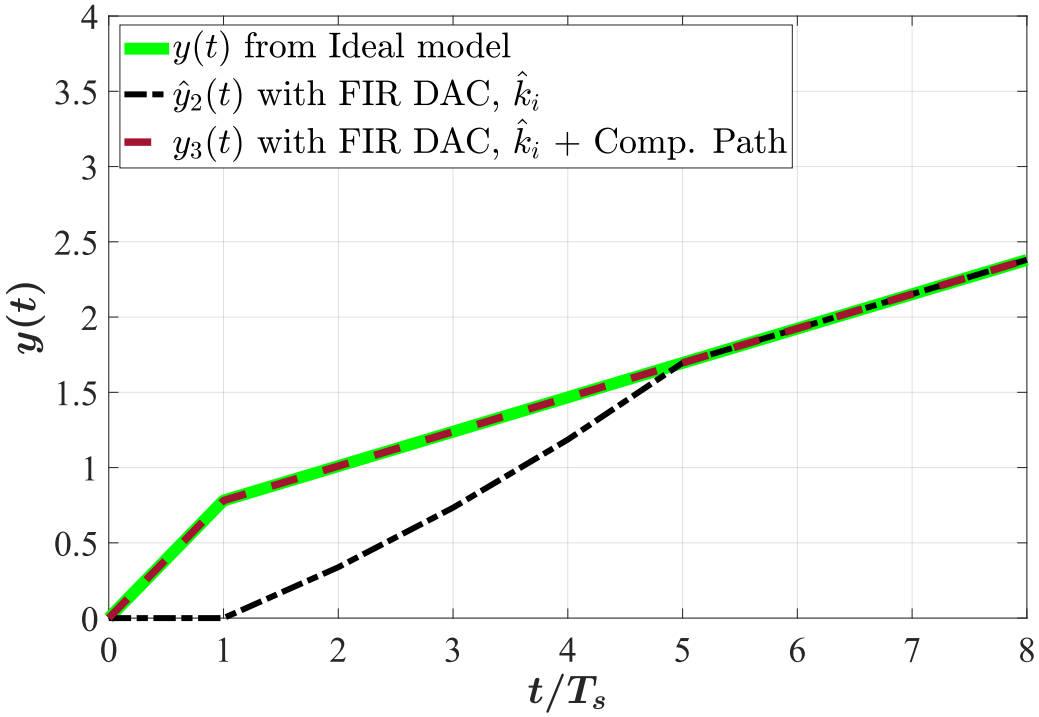


Figure 2.25: Restored open loop response with compensation path.

Figure 2.25, showcasing the red plot $y_3(t)$ as a counterpart to $y_2(t)$ in Figure 2.24, elucidates how the second FIR filter, applied at the loop filter output as positive feedback during the open loop analysis, **restores** the system response precisely to the **ideal topology**, yet imbuing the design with

enhanced robustness against jitter.
Specifically, the synergy between the M-order FIR-DAC and its accompanying compensation filter path amplifies the CTΔΣM SNR as delineated below [8]:

$$SNR_{\text{Improvement}}[\text{dB}] = 20\log_{10}(M) \tag{2.50}$$

Figure 2.23 portrays a scenario with a quartet of taps ($\mathbf{M = 4}$), facilitating an SNR enhancement of **12 dB**. This augmentation suffices in achieving the Anser EMT 70-80dB SNR target, accounting for the SNR degradation observable in the ideal model as exhibited in Figure 2.15.

2.4.6 CTΔΣM Topology A

Figure 2.26 presents the CIFF 2nd-order topology for a CTΔΣM. This design demonstrates robustness against **white spectrum jitter** with applying the FIR-DAC in the feedback loop. A second FIR filter, a compensation path, is integrated with the quantizer. Termed the *fast loop*, this path introduces a second **negative feedback** at the output of $L(s)$, located adjacent to the 1-bit quantizer or comparator.

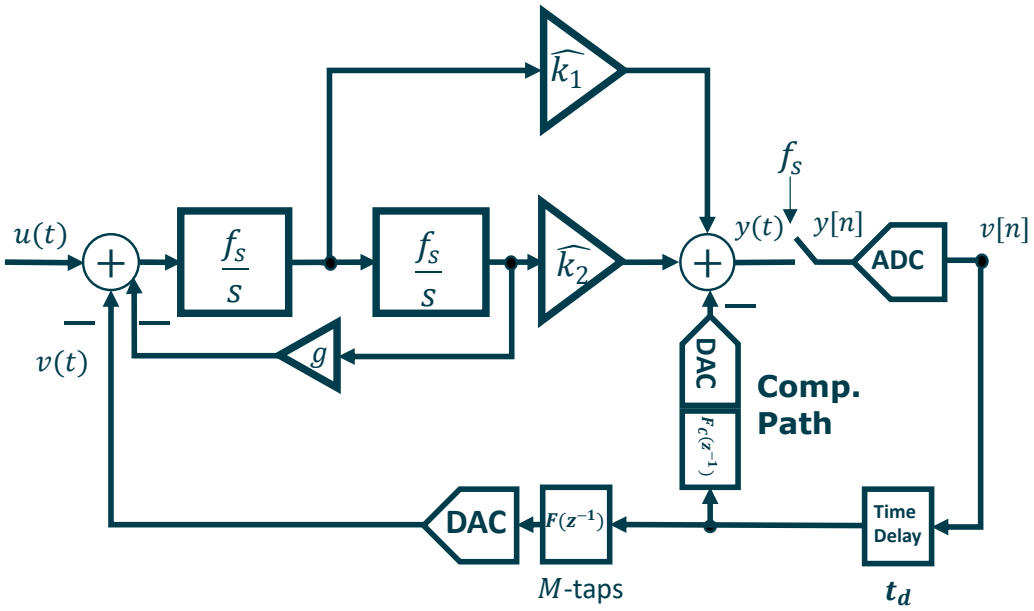


Figure 2.26: CTΔΣM with *fast-loop* compensation.

At this juncture, it is feasible to evaluate all the system coefficients as functions of a *time delay*, **equivalent** to T_s, and a variable M-order for the

FIR-DAC, including all the corresponding *compensation path* coefficients for the second FIR filter. Due to the chosen fixed time delay, it is important to note that all compensation paths retain the same primary coefficient, valued at **0.781**.

Table 2.2 shows how the values of the complex conjugate poles, represented by g, and the second feedforward coefficient, k_2, remain constant for a 2nd order CTΔΣM, per CIFF theory. The primary variable to scale is the *first* feedforward coefficient in $L(s)$, k_1, alongside the compensation path, which is determined based on the chosen M-order to enhance the SNR.

Ideal Model	**FIR-DAC 4-taps**: $F(z^{-1})$	**FIR-DAC 5-taps**: $F(z^{-1})$	**FIR-DAC 6-taps**: $F(z^{-1})$
	$\frac{1}{4}\sum_{i=0}^{3} z^{-i}$	$\frac{1}{5}\sum_{i=0}^{4} z^{-i}$	$\frac{1}{6}\sum_{i=0}^{5} z^{-i}$

Loop Filter Coefficients $L(s)$

$k_1 = 0.667$	$k_1 = 1.2384$	$k_1 = 1.3527$	$k_1 = 1.4671$
$k_2 = 0.229$	$k_2 = 0.229$	$k_2 = 0.229$	$k_2 = 0.229$
$g = 7.53 \cdot 10^{-5}$	$g = 7.53 \cdot 10^{-5}$	$g = 7.53 \cdot 10^{-5}$	$g = 7.53 \cdot 10^{-5}$

Compensation Path Filter: $F_{\text{Comp. Path}}(z^{-1})$

	0.781	+	**0.781**	+	**0.781**	+
	$0.6715z^{-1}$	+	$0.7163z^{-1}$	+	$0.7461z^{-1}$	+
	$0.5048z^{-2}$	+	$0.6058z^{-2}$	+	$0.6731z^{-2}$	+
	$0.2808z^{-3}$		$0.4494z^{-3}$	+	$0.5618z^{-3}$	+
			$0.2473z^{-4}$		$0.4124z^{-4}$	+
					$0.2249z^{-5}$	

Table 2.2: Topology A coefficients

The CTΔΣM topology A is recognized as a standard reference in literature [8]. However, tailoring it to meet the specific requirements of the Anser EMT project, particularly regarding BW and SNR, presented a significant challenge. The loop filter architecture was derived from the discrete-time rep-

resentation of the NTF in equation 2.12 and the final equation for the feedforward coefficients in equation 2.47, ensuring it aligns with all project parameters. Modifications to the topology can be pursued without compromising SNR's resilience to jitter, as defined in equation 2.50. Such modifications, however, will require adjustments to the coefficients of the $F_{\text{Comp. Path}}(z^{-1})$ filter, details of which will be provided in subsequent sections.

2.4.7 Moving the Compensation Path

The CIFF architecture allows for repositioning the compensation path at the **second integrator's** input [8], [9], [7]. This change simplifies system analysis, aligning with the **logical framework** of Figures 2.22 and 2.24. The supporting *open-loop pulse response* is detailed in Figures 2.23 and 2.25. Additionally, this modification enhances the system's structural and operational efficiency.

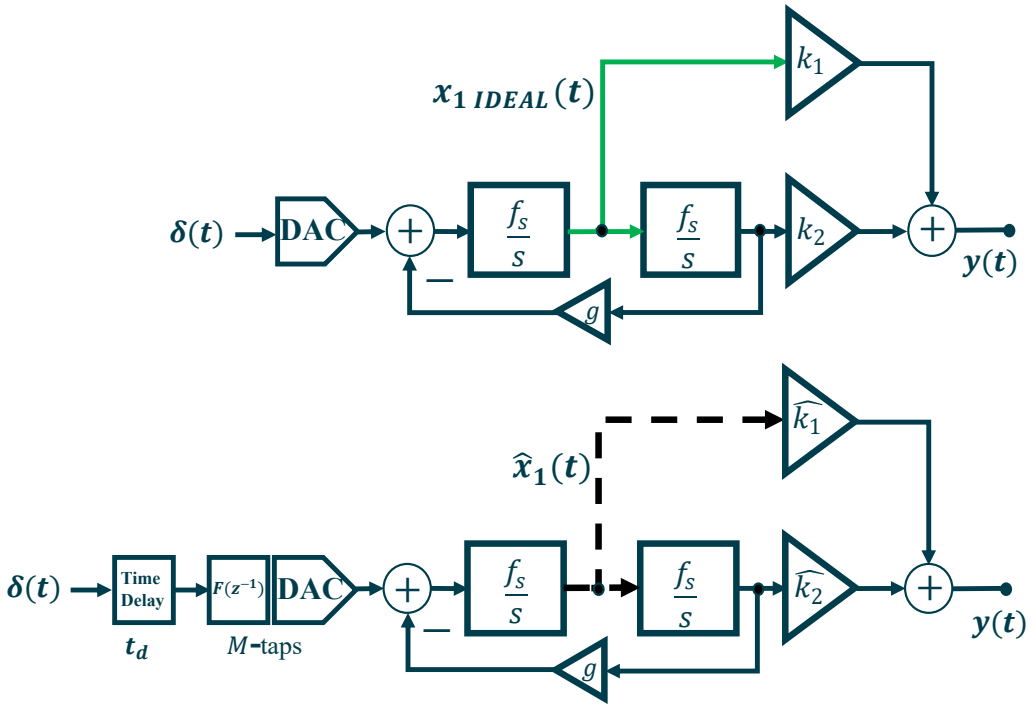


Figure 2.27: Open-loop responses of the first integrator output.

Specifically, this change eliminates the need for a *third adder stage*, an operational amplifier, previously essential for the summation node in topology A (Figure 2.26). As a result, schematic implementation challenges are reduced.

This adjustment also optimizes **power consumption** and **chip area**, crucial for the Anser EMT project.

An essential aspect of this change is measuring the time-domain open-loop pulse response at the first integrator's output, denoted as $x_1(t)$. This response aims to introduce compensation at the beginning of the second integrator. As Figure 2.28 demonstrates, the system exhibits a **step** response due to a *single integration* of the Dirac delta function $\delta(t)$.

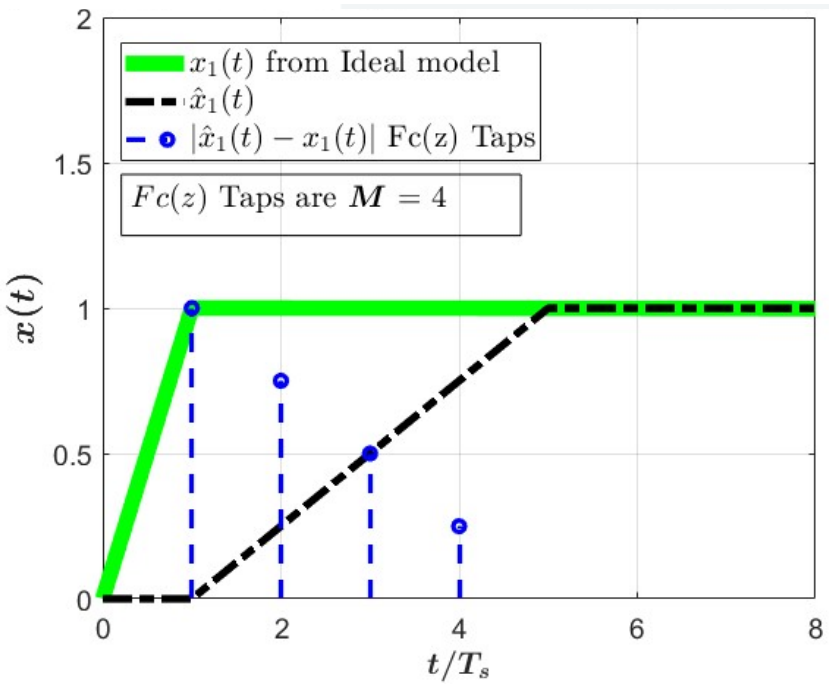


Figure 2.28: Amplitude difference between open-loop responses of the first integrator.

The approach is mathematically consistent with equations 2.48 and 2.49, but applied to the current signal under consideration.

$$x_{1,\,\text{Ideal}}(t) = \hat{\mathbf{x}}_\mathbf{1}(\mathbf{t}), \quad \text{for}\, t \geq (M+1)T_s \tag{2.51}$$

Therefore, $x_{1,\,\text{Ideal}}(t)$ and $\hat{\mathbf{x}}_\mathbf{1}(\mathbf{t})$ coincide for $\frac{t}{T_s} \geq (M+1)$. The *coefficients* of the compensation path are then determined by evaluating the **amplitude difference** in the open-loop pulse responses at integer multiples of the sam-

pling period T_s.

$$|x_{1,\text{Ideal}}(nT_s) - \hat{\mathbf{x}}_\mathbf{1}(\mathbf{nT_s})| > 0 \quad \text{for} \begin{cases} 0 \leq \frac{t}{T_s} \leq M \\ \\ 1 \leq n \leq M, \quad \text{where } n \in \mathbb{N} \end{cases} \tag{2.52}$$

Figure 2.28 visualizes equation 2.52 through the blue discrete plot defined as $|\hat{x}_1(t) - x_1(t)|$. The FIR-DAC order is maintained at four ($\mathbf{M = 4}$).

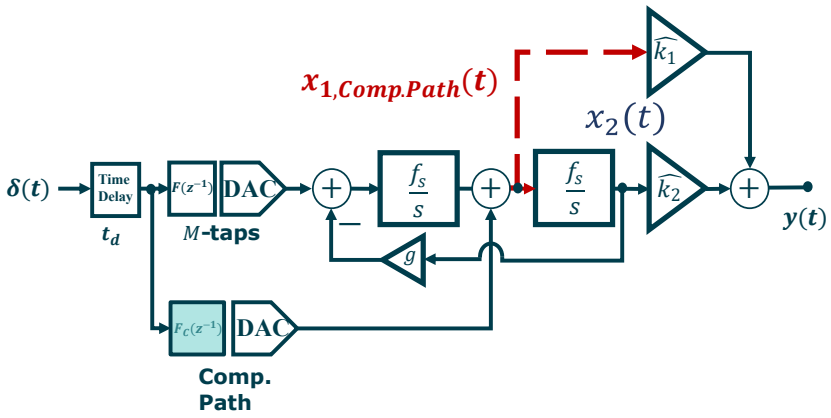


Figure 2.29: Complete block diagram of the second open-loop responses.

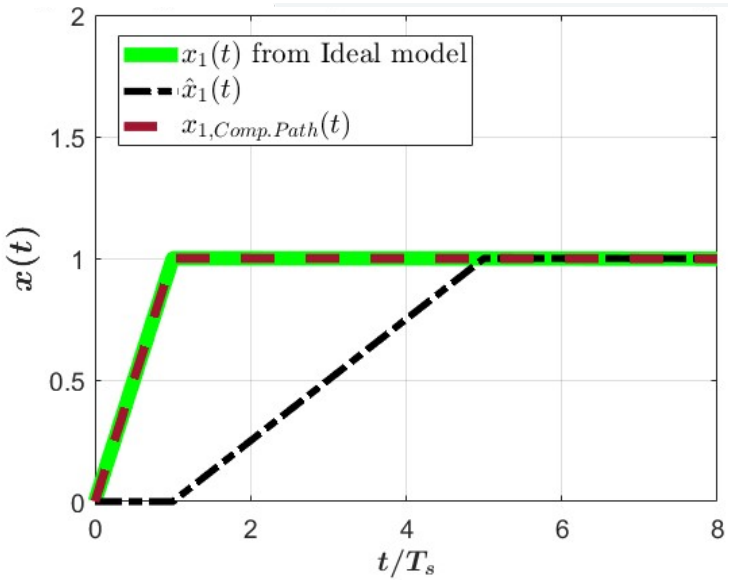


Figure 2.30: Restoration of the second open-loop response.

After evaluating the $F_{\text{Comp. Path}}(z^{-1})$, a second FIR filter is integrated as **positive** feedback. This addition aims to restore the ideal response $x_{1,\text{Ideal}}(t)$ as shown in Figure 2.30. The open-loop pulse response is represented by the red signal $x_{1,\text{ Comp.Path}}(t)$ in Figure 2.29. The overlap of $x_{1,\text{ Comp.Path}}(t)$ with $x_1(t)$ (green plot) indicates that the first integrator's output can be accurately restored while preserving the SNR improvements.

It is crucial to note that the compensation path must be applied as feedback (positive for open loop and negative for closed loop) as defined in Figure 2.29. The resulting **summation signal** should be scaled by the feedforward coefficient $\hat{k}_1$ if this condition is not met—specifically, if $\hat{k}_1$ amplifies the signal at the integrator's output before summation by the $F_{\text{Comp. Path}}(z^{-1})$—the system will not accurately restore the STF, leading to a change in the NTF and a reduction in SNR.

2.4.8 CTΔΣM Topology B

Through mathematical analysis of the CTΔΣM designs, it was found that the compensating second FIR filter can be optimally positioned between the two integrators of the loop filter $L(s)$. This adjustment obviates the need for a third operational amplifier and allows the second negative feedback to be implemented using the loop filter's transfer function. Topologies A and B are mathematically **equivalent** and can be used *interchangeably* since they have the same signal and noise transfer functions.

Figure 2.31 depicts topology B:

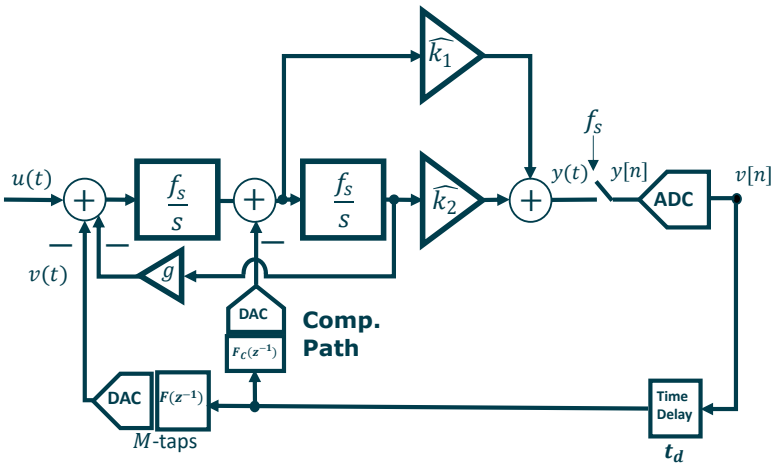


Figure 2.31: CTΔΣM configuration with compensation between integrators.

Figure 2.32 presents a comparison of the two CIFF topologies. Both topologies utilize **identical** loop filter coefficients: $\widehat{k}_1$, $\widehat{k}_2$, and g. Furthermore, they employ the *same* FIR-DAC in the primary feedback, given by $F(z^{-1}) = \frac{1}{4}(1 + z^{-1} + z^{-2} + z^{-3})$, for jitter averaging. However, they differ in their compensation paths' filter coefficients.

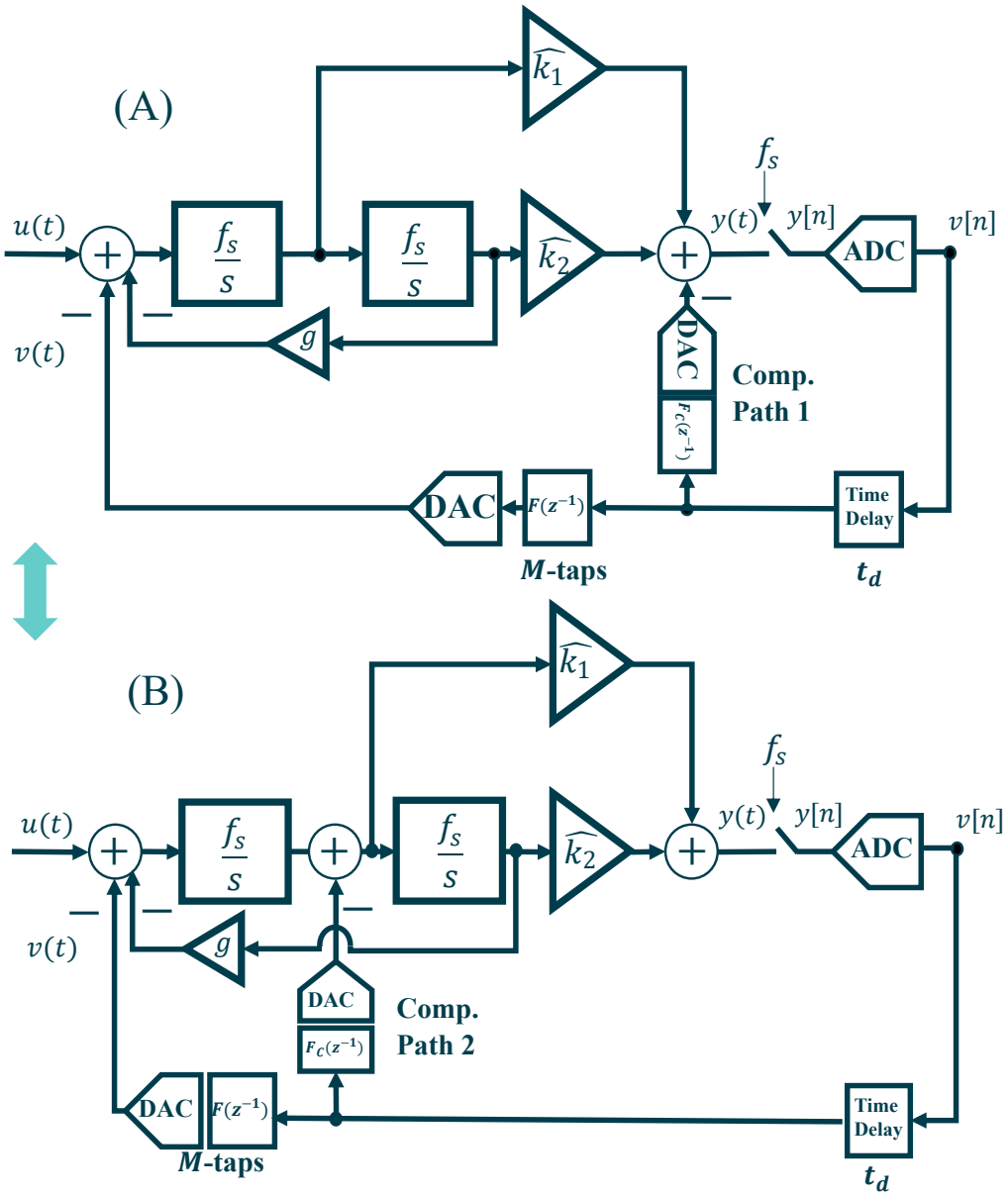


Figure 2.32: CTΔΣM CIFF topologies.

Table 2.3 shows the coefficient comparison for topologies A and B. The emphasis is on the 2[nd]-order 1-bit CTΔΣM employing a CIFF loop filter architecture, as expounded in preceding sections. The FIR filter order is consistently maintained at $\mathbf{M = 4}$ with a *time delay* of T_s.

Ideal Model	**CTΔΣ – (A)**	**CTΔΣ – (B)**
	FIR DAC 4-taps: $F(z^{-1})$	
	$\frac{1}{4}\sum_{i=0}^{3} z^{-i}$	
	Loop filter coefficients: $L(s)$	
$k_1 = 0.667$	$\widehat{k}_1 = 1.2384$	
$k_2 = 0.229$	$\widehat{k}_2 = 0.229$	
$g = 7.53 \cdot 10^{-5}$	$g = 7.53 \cdot 10^{-5}$	
	Compensation Path: $F_{\text{Comp.Path}}(z^{-1})$	
	$0.781 + 0.6715z^{-1} + 0.5048z^{-2} + 0.2808z^{-3}$	$1.000 + 0.7499z^{-1} + 0.4998z^{-2} + 0.2496z^{-3}$

Table 2.3: Coefficient comparison for topologies A and B

This configuration is congruent with a *functional* Simulink model comprising:

- Schematic realization as illustrated in Figure 2.32.
- DAC subjected to white spectrum clock periodic jitter, as characterized in Figure 2.18.
- A 1-bit quantizer's comparator.

The requisite blocks can be sourced from the standard Simulink library or through additional add-ons in the official MATLAB repository.

2.5 CTΔΣM Model results

The CTΔΣM is sensitive to white spectrum clock jitter, as shown in Fig. 2.15, leading to a significant in-band Signal-to-Noise Ratio (SNR) degradation. The FIR filter approach has been employed to address this issue. By implementing fourth-order FIR filters, as defined in equation 2.50, a **12 dB** SNR *improvement* is achieved, mitigating the in-band noise caused by the white spectrum periodic jitter model (refer to equation 2.28).

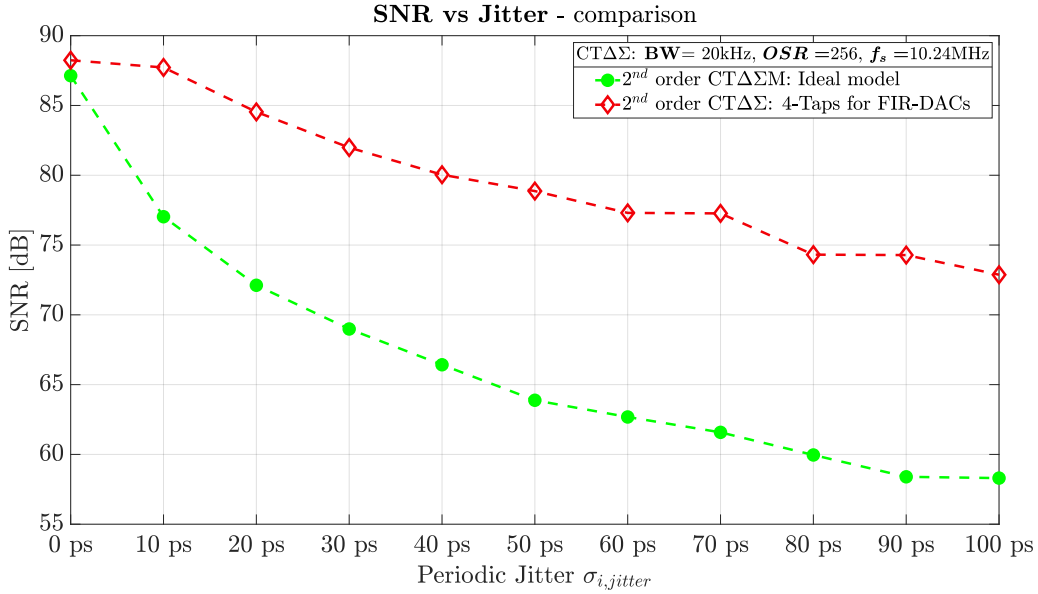


Figure 2.33: SNR improvement using the compensation path

Fig. 2.33 displays two plots: the green represents the *ideal model* from Fig. 2.15, and the red represents both A and B topologies. As these topologies are mathematically equivalent (refer to Table 2.3 for coefficients), their SNR performances **concide**.

2.5.1 Dynamic Range Scaling

The A and B topologies can be further refined to more closely match the schematic implementation, reducing discrepancies between the model and real-world implementation. An essential step is to **define** a maximum output

voltage for the time responses of the first and second integrators, $x_1(t)$ and $x_2(t)$. For the Anser Project, this maximum voltage is set to:

$$V_{x_i(t)} = V_{\text{Desired}} = \pm 300\text{mV} \tag{2.53}$$

Using first-order inverse proportionality, coefficients a and b are established to scale the integrators, with a defined maximum positive voltage:

$$\begin{cases} V_{x_1(t)} = \dfrac{\max\{x_1(t)\}}{\mathbf{a}} \\ \\ V_{x_2(t)} = \dfrac{\max\{x_2(t)\}}{\mathbf{b}} \end{cases} \qquad \text{where } \mathbf{a}, \mathbf{b} \in \mathbb{R}^+ \tag{2.54}$$

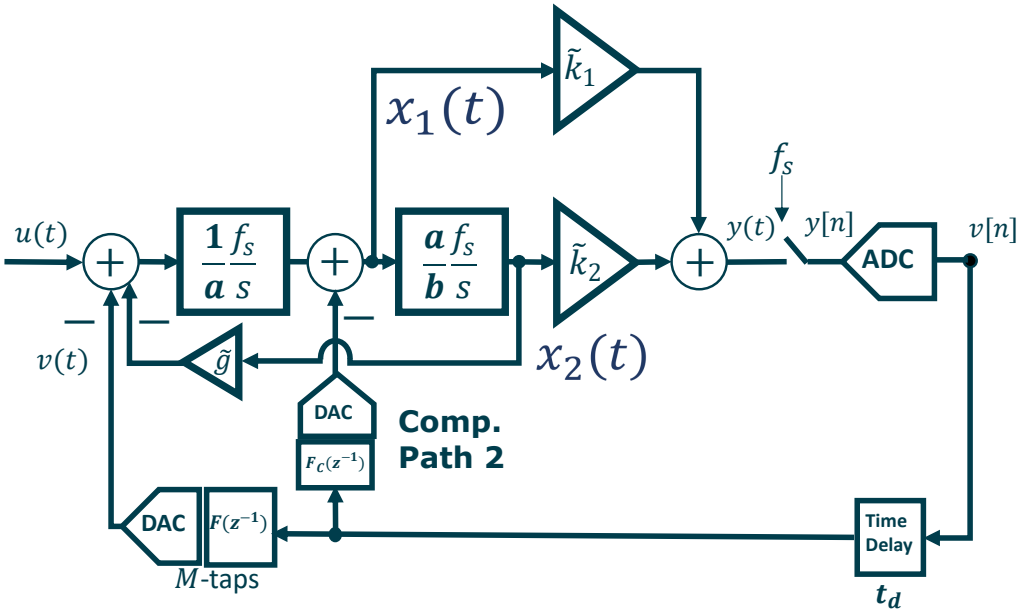


Figure 2.34: CTΔΣM - (B) with dynamic range scaling.

Simulation results provide the scalar coefficients $\mathbf{a}$ and $\mathbf{b}$ as:

$$\begin{cases} \mathbf{a} \triangleq \dfrac{\max\{x_1(t)\}}{V_{x_1(t)}} \\ \\ \mathbf{b} \triangleq \dfrac{\max\{x_2(t)\}}{V_{x_2(t)}} \end{cases} \qquad \text{where } \mathbf{a}, \mathbf{b} \in \mathbb{R}^+ \tag{2.55}$$

Adjusting the integrators requires loop filter modifications. The first integrator is divided by **a**, and the second integrator's input is multiplied by the same factor, ensuring no alterations to the loop filter's direct path. The input to the second integrator is also scaled by **b**. To maintain the positioning of poles and zeros, as indicated in equation 2.14, the transfer function $L(s)$ must be modified. The revised loop filter is:

$$\tilde{L}(s) = \frac{\tilde{k}_2}{\tilde{g}} \frac{1 + \dfrac{\mathbf{b}\tilde{k}_1}{\mathbf{a}\tilde{k}_2 f_s} s}{1 + \dfrac{\mathbf{b}}{\tilde{g} f_s^2} s^2} \equiv L(s) \iff \begin{cases} \tilde{k}_1 & = a \cdot \hat{k}_1 \\ \tilde{k}_2 & = b \cdot \hat{k}_2 \\ \tilde{g} & = b \cdot g \end{cases} \tag{2.56}$$

The modified feedforward coefficients $\tilde{k}_1$, $\tilde{k}_2$, and the complex conjugate $\tilde{g}$ shift the poles and zeros as follows: The **poles** of $\tilde{L}(s)$ are:

$$s_{\text{poles}} = \left\{ \pm j \sqrt{\frac{\tilde{g} f_s^2}{\mathbf{b}}} \right\} \tag{2.57}$$

$$f_{\text{poles}} = \frac{|s_{\text{poles}}|}{2\pi} \tag{2.58}$$

The **zeros** of $\tilde{L}(s)$ are:

$$s_{\text{zeros}} = \left\{ -\frac{\mathbf{b}\tilde{k}_1}{\mathbf{a}\tilde{k}_2 f_s} \right\} \tag{2.59}$$

$$f_{\text{zeros}} = \frac{|s_{\text{zeros}}|}{2\pi} \tag{2.60}$$

By scaling $\tilde{k}_1$, $\tilde{k}_2$, and $\tilde{g}$ in the manner shown in equation 2.56, the loop filter, and consequently the NTF and STF, remain unchanged.

Model's Adjustment Procedure

Modifying loop filter coefficients changes the system's *open-loop pulse response*, necessitating new coefficients for the FIR filter compensation path. The adjustment process is outlined below:

1. Extract k_1, k_2, and g from the NTF.
2. Express $\hat{k}_1$ and $\hat{k}_2$ using a linear combination of g, t_{Delay}, and the M-taps FIR filter.
3. Examine the Open-loop pulse response to determine the coefficients for $F_{\text{Comp.Path}}(z^{-1})$.

4. Adjust the integrator outputs to achieve the target values $V_{1,\text{Desired}}$ and $V_{2,\text{Desired}}$:

$$\begin{cases} \mathbf{a} = \dfrac{\max\{x_1(t)\}}{V_{1,\text{Desired}}} \\ \\ \mathbf{b} = \dfrac{\max\{x_2(t)\}}{V_{2,\text{Desired}}} \end{cases}$$

5. Revise the still-evolving **model** to:

$$\begin{cases} \tilde{k}_1 = a \cdot \hat{k}_1, \\ \tilde{k}_2 = b \cdot \hat{k}_2, \\ \tilde{g} = b \cdot g \end{cases}$$

6. **Re-assess** the Open-loop pulse response to obtain coefficients for the modified $F_{\text{Comp.Path}}(z^{-1})$.

With these adjustments, the NTF functions correctly within the closed-loop architecture, maintaining the ideal model's poles and zeros. Additionally, it exhibits improved SNR robustness (refer to equation 2.50) against white spectrum periodic jitter (as detailed in 2.10) and produces the desired integrator output waveforms.

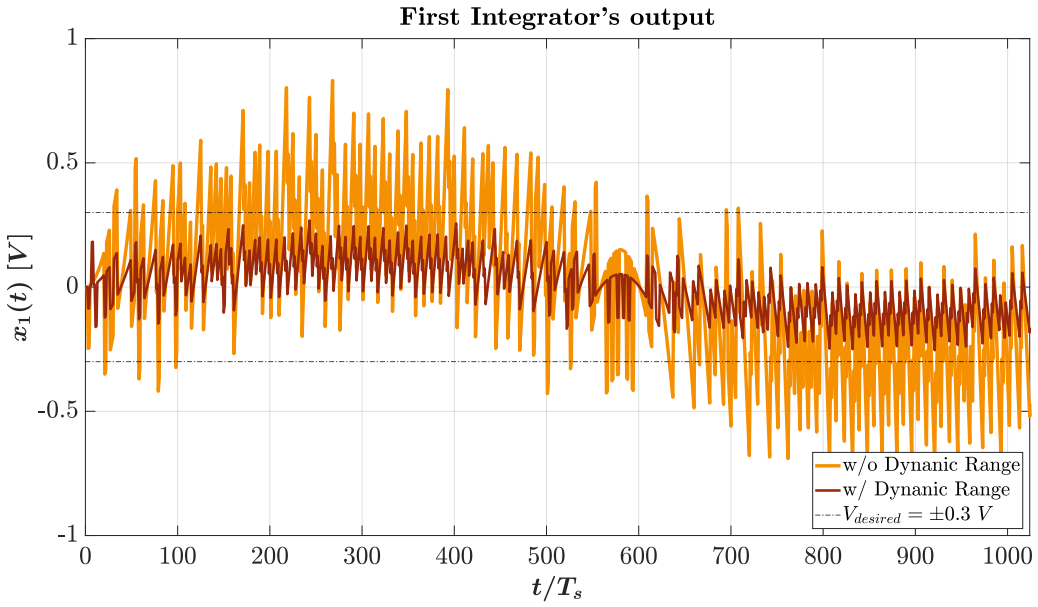
Simulation Waveforms

The CTΔΣM's model facilitates waveform data retrieval using MATLAB. This data can be stored in workspace variables generated by Simulink. The process enables straightforward identification of peak values of each integrator's closed-loop time responses, aligning with project requirements. The methodology is depicted in Figures 2.35 and 2.35. For ensuring the **accuracy** of linear coefficient scaling, the loop filter's continuous-time output must be verified to be constrained within [8], [7]:

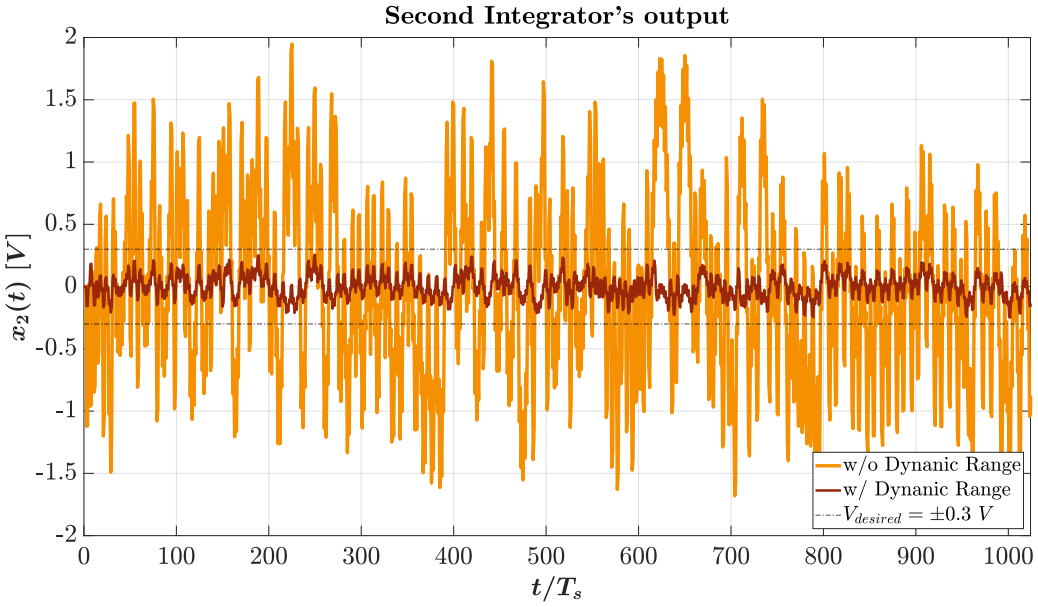
$$y(t) \in [-M, M] \qquad \text{where } M = \text{CT}\Delta\Sigma\text{M's number of bits} \tag{2.61}$$

As confirmed in Figure 2.35c, the response is consistent with the ideal model and lies within the -1V to 1V range, given the model's usage of a 1-bit ADC.

With all requirements met, table 2.4 presents the new coefficients $\tilde{k}_1$, $\tilde{k}_2$, $\tilde{g}$, **a**, and **b**, as depicted in the block diagram 2.34. These coefficients represent a fully operational CTΔΣM with dynamic range scaling for the B topology, using $M = 4$-taps, a *time delay* of T_s, and a V_{Desired} of 300mV.

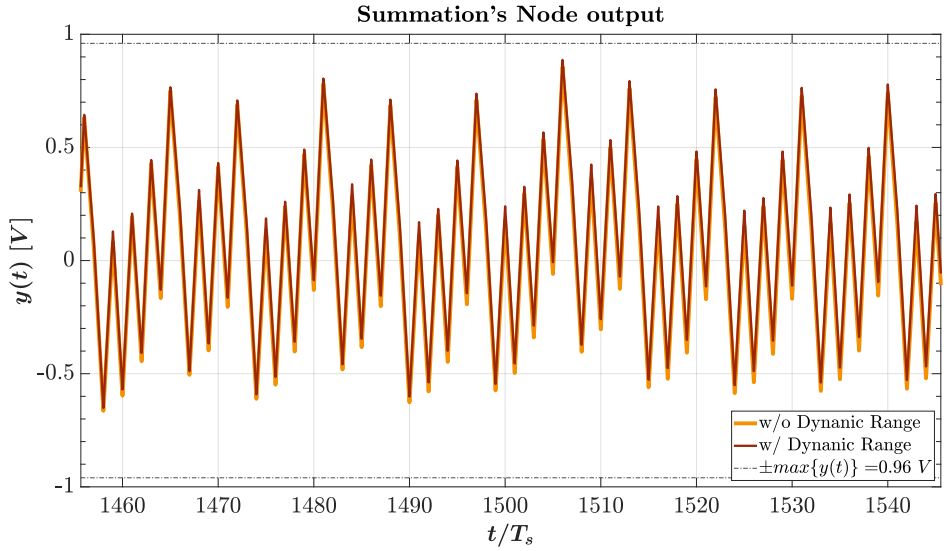


(a) First integrator's output $x_1(t)$



(b) Second integrator's output $x_2(t)$

Figure 2.35: Dynamic range scaling waveforms (continued on the next page)



(c) Loop filter output $y(t)$

Figure 2.35: Dynamic range scaling waveforms (continued)

Ideal Model	**CTΔΣ – (B)**	**CTΔΣ – (B) scaled**
	FIR DAC 4-taps: $F(z^{-1})$	
	$\frac{1}{4}\sum_{i=0}^{3} z^{-i}$	
Loop filter coefficients: $L(s)$		
$k_1 = 0.67,\ k_2 = 0.23$	$\widehat{k}_1 = 1.24,\ \widehat{k}_2 = 0.23$	$\tilde{k}_1 = 3.53,\ \tilde{k}_2 = 1.67$
$\mathbf{a} = 1,\ \mathbf{b} = 1$	$\mathbf{a} = 1,\ \mathbf{b} = 1$	$\mathbf{a} = 2.86,\ \mathbf{b} = 1.11$
$g = 7.53 \cdot 10^{-5}$	$g = 7.53 \cdot 10^{-5}$	$g = 5.55 \cdot 10^{-4}$
	Compensation Path: $F_{\text{Comp.Path}}(z^{-1})$	
	$1.00 + 0.75z^{-1} + 0.50z^{-2} + 0.25z^{-3}$	$0.35 + 0.26z^{-1} + 0.28z^{-2} + 0.09z^{-3}$

Table 2.4: Coefficient comparison for B topology

2.6 $L(s)$: Towards Schematic

Topology B provides an advantage with its adder-less implementation in the loop's filter design. In the Anser-EMT project, L was realized using the *biquad-cell* structure, where the feedforward path employs capacitors instead of resistors. The structure in Figure 2.36 corresponds to the following

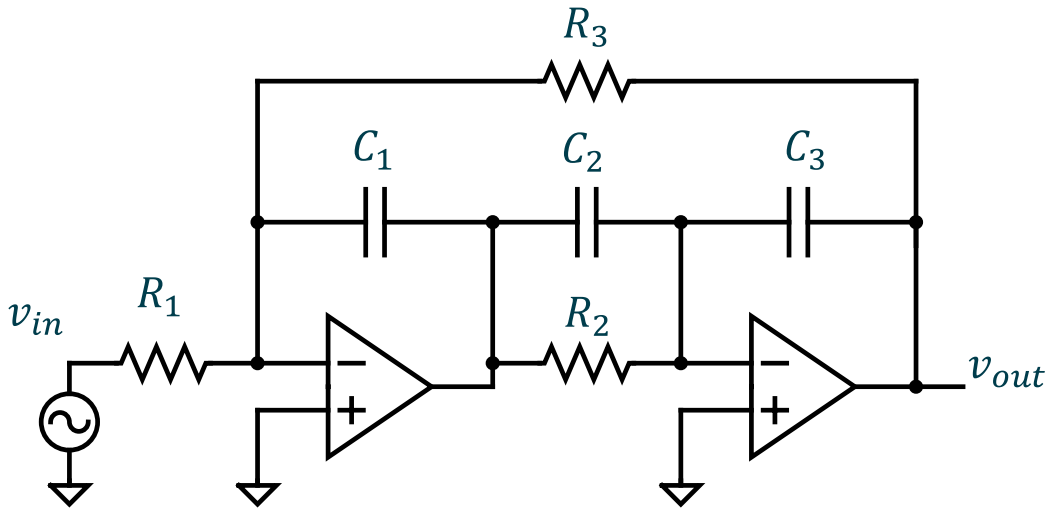


Figure 2.36: $L(s)$ single-ended implementation.

transfer function, assuming ideal op. amp. behaviour:

$$L_{\text{Ideal Op.Amp}}(s) = -\frac{R_3}{R_1} \cdot \frac{1 + C_2 R_2 s}{1 + C_2 R_2 s - C_1 C_3 R_2 R_3 s^2} \tag{2.62}$$

This equation, 2.62, is already in its canonical form, which highlights *dc gain*, poles, and zeros as:

$$\begin{cases} K = k_{\text{dc gain}} = -\dfrac{R_3}{R_1} \\ \dfrac{1}{\tau} = \omega_{\text{zeros}} = \dfrac{1}{R_2 C_2} \\ \omega_{n_2}^2 = -\dfrac{1}{R_2 R_3 C_1 C_2} \\ \xi = -\dfrac{C_2}{2 C_1 C_3 R_3 \omega_n} \end{cases} \tag{2.63}$$

Equation 2.62 is in its canonical form, highlighting the *dc gain*, poles, and zeros:

$$\begin{cases} K = k_{\text{dc gain}} = -\dfrac{R_3}{R_1} \\ \dfrac{1}{\tau} = \omega_{\text{zeros}} = \dfrac{1}{R_2 C_2} \\ \omega_n^2 = -\dfrac{1}{R_2 R_3 C_1 C_2} \\ \xi = -\dfrac{C_2}{2 C_1 C_3 R_3 \omega_n} \end{cases} \tag{2.64}$$

A *discrepancy* emerges when comparing equation 2.62 with the **ideal model** equation 2.15:

$$\begin{array}{cc} \text{Ideal model} & \text{System of Equations} \\ \begin{cases} \dfrac{k_2}{g} = k_{\text{dc gain}} \\ \dfrac{k_2}{k_1} f_s = \omega_{\text{zeros}} \\ f_s^2 g = \omega_n^2 \\ 0 = \xi \end{cases} \implies & \begin{cases} \dfrac{k_2}{g} = -\dfrac{R_3}{R_1} \\ \dfrac{k_2}{k_1} f_s = \dfrac{1}{R_2 C_2} \\ f_s^2 g = -\dfrac{1}{R_2 R_3 C_1 C_2} \\ 0 = -\dfrac{C_2}{2 R_3 C_1 C_3 \omega_n^2} \end{cases} \end{array} \tag{2.65}$$

$$Solutions : \nexists (g, k_1, k_2), \forall (R_i, C_i) \in \mathbb{R}^+$$

The indication suggest that the system's space lacks algebraic solutions, pointing to a misalignment with the ideal scenario. Nonetheless, by considering the pair (R_2, C_2) as variables, specifically represented in the denominator as $R_2 C_2 s$, one can derive a **consistent** system solution:

$$\begin{cases} \dfrac{k_2}{g} = k_{\text{dc gain}} \\ \dfrac{k_2}{k_1} f_s = \omega_{\text{zeros}} \\ f_s^2 g = \omega_n^2 \end{cases} \implies \begin{cases} k_1 = -\dfrac{1}{C_1 C_2 R_2 R_3 f_s^2} \\ k_2 = \dfrac{C_2}{C_1 C_2 R_1 f_s} \\ g = \dfrac{1}{C_1 C_2 R_1 R_2 f_s^2} \end{cases}$$

$$Solution : \exists (g, k_1, k_2), \forall (R_2 C_2) \in \mathbb{R}^+ \tag{2.66}$$

For any positive real value of the product C_2R_2, the system conditions are satisfied, yielding a solution set (g, k_1, k_2). However, it's pivotal to acknowledge that the schematic will not align perfectly with the ideal $L(s)$. This misalignment is largely attributed to the architecture not employing complex conjugate poles. Furthermore, equation 2.62 assumes using an *ideal op. amp.* with infinite bandwidth and gain. Delving deeper into a final solution is essential.

2.6.1 Divergence from the Model

Even when a solution is discerned, it is imperative to recognise that it remains fundamentally *numerical* rather than algebraically exact. The schematic's distinct characteristic of not solely relying on complex conjugate poles invariably modifies L, influencing both the NTF and the STF. Such a deviation in the NTF directly impacts the ADC's noise characteristics, altering its OGB and potentially compromising the SNR performance. Moreover, if the transfer function of the schematic, as depicted in Figure 2.36, can be demonstrated to closely resemble that of the ideal loop filter, as shown in Figure 2.11, it strengthens the assurance of the ADC's optimal functioning. Nonetheless, this highlights the importance of thorough calibration and validation to ensure the system's optimal operation and circumvent potential discrepancies.

2.6.2 Advanced Model

To enhance the model's precision, the *operational amplifier* should not be considered as an ideal component, thereby removing the "virtual ground" simplification.

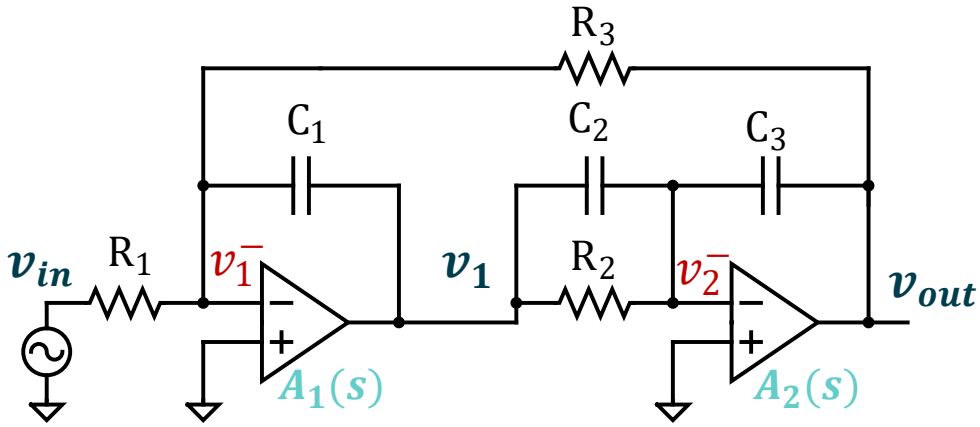


Figure 2.37: Refined $L(s)$ single-ended implementation.

This approach results in a differential voltage at the input that's not zero and a gain that isn't infinite, as depicted in Figure 2.37. The relationship between the differential voltages and the output is represented by:

$$A_{\text{Op.Amp}}(s) = \frac{V_{\text{out,Op.Amp}}(s)}{\left[V_i^+(s) - V_i^-(s)\right]} \qquad \text{where} \begin{cases} V_i^+(s) - V_i^-(s) \neq 0 \\ A_{\text{Op.Amp}}(s) \neq +\infty \end{cases} \tag{2.67}$$

Figure 2.37, therefore, introduces two transfer functions, $A_1(s)$ and $A_2(s)$, that need to be factored into the comprehensive evaluation of the loop filter $L(s)$. This version encapsulates the poles and zeros inherent to a practical amplifier implementation.

A block diagram can be formulated to encompass all the negative feedback paths detailed in the schematic. This includes one main feedback path and one for each of the two integrators, summing up to three paths. Although this topology complicates the transfer function evaluation, it offers the completeness of modelling precision, as illustrated in Figure 2.38.

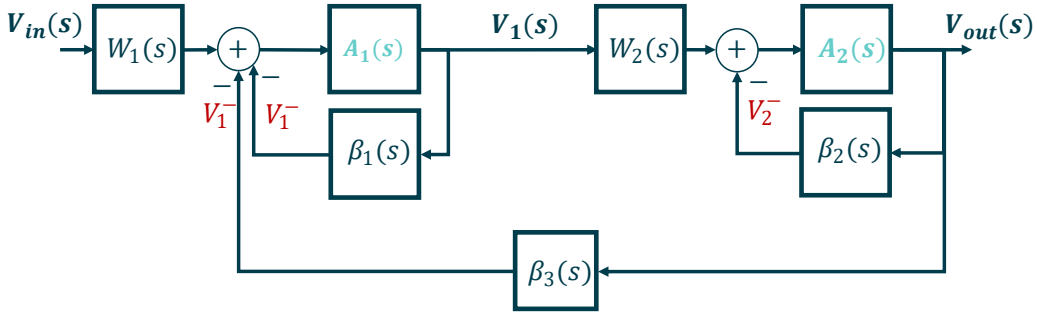


Figure 2.38: Complete $L(s)$ model.

In this configuration, each integrator is modelled as a negative feedback structure, $\beta_{1,2}(s)$, because capacitive feedback is essential for integrator creation. Moreover, Figure 2.38 comprises a direct path, $A_{1,2}(s)$, and a shared negative feedback, $\beta_3(s)$, linking the output of the second integrator to the first integrator's input. Two additional transfer functions, $W_1(s)$ and $W_2(s)$, represent the input and the interlink between the integrators, respectively.

By leveraging the *superposition principle*, each transfer function is deduced. This process involves evaluating the contributions of individual blocks, grounding either the inputs or outputs, depending on their placement in the block

diagram. Referring to Figure 2.38, we can define each parameter as follows:

$$\begin{cases} \beta_1 = \left.\dfrac{V_1^-}{V_1}\right|_{\substack{V_{\text{in}}=0V \\ V_{\text{out}}=0V}} \\ \beta_2 = \left.\dfrac{V_2^-}{V_{\text{out}}}\right|_{\substack{V_{\text{in}}=0V \\ V_1=0V}} \\ \beta_3 = \left.\dfrac{V_1^-}{V_{\text{out}}}\right|_{\substack{V_{\text{in}}=0V \\ V_1=0V}} \\ W_1 = \left.\dfrac{V_1^-}{V_{\text{in}}}\right|_{\substack{V_1=0V \\ V_{\text{out}}=0V}} \\ W_2 = \left.\dfrac{V_2^-}{V_1}\right|_{\substack{V_{\text{in}}=0V \\ V_{\text{out}}=0V}} \end{cases} \tag{2.68}$$

From the above, the expressions are solved to yield:

$$\begin{cases} \beta_1(s) = \dfrac{C_1R_1R_3s}{(R_1+R_3)\left(\dfrac{C_1R_1R_3s}{R_1+R_3}+1\right)} \\ \beta_2(s) = \dfrac{C_3R_2s}{R_2s(C_2+C_3)+1} \\ \beta_3(s) = \dfrac{R_1}{R_1+R_3(C_1R_1s+1)} \\ W_1(s) = -\dfrac{V_1^-}{R_3+R_1(C_1R_3s+1)} \\ W_2(s) = -\dfrac{C_2R_2s+1}{R_2s(C_2+C_3)+1} \end{cases} \tag{2.69}$$

While Equations 2.69 are not presented in canonical form, they quantify each block's contribution within the circuits defined in Figures 2.36 and 2.37 for all resistors and capacitors. By integrating these contributions, the **complete** loop filter transfer function is obtained:

$$L(s) = \frac{W_1(s)A_1(s)W_2(s)A_2(s)}{[1+A_1(s)\beta_1(s)]\,[1+A_2(s)\beta_2(s)] + A_1(s)A_2(s)W_2(s)\beta_3(s)} \tag{2.70}$$

Substituting the values from Equations 2.69 into Equation 2.70, the comprehensive loop filter transfer function is derived. By mapping the loop filter coefficients into a first-order zero and a second-order pole transfer function according to Equation 2.15, the following is obtained:

$$\begin{cases} k_{\text{dc}} = \dfrac{A_1(s)A_2(s)R_3}{R_1 + R_3A_1(s)A_2(s)R_1} \\ \\ \omega_{\text{zero}} = \dfrac{1}{R_2C_2} \\ \\ \omega_n = \sqrt{\dfrac{R_1 + R_3 - A_1(s)A_2(s)R_1}{C_1R_1R_2R_3(A_1(s)+1)(C_2 + C_3 + A_2(s)C_3)}} \\ \\ \xi = \dfrac{\begin{array}{l} C_1R_1R_3 + C_2R_1R_2 + C_3R_1R_2 + C_2R_2R_3 + C_3R_2R_3 + \\ + A_1(s)C_1R_1R_3 + A_2(s)C_3R_1R_2 + A_2(s)C_3R_2R_3 + \\ - A_1(s)A_2(s)C_2R_1R_2 \end{array}}{2C_1R_1R_2R_3(A_1(s)+1)(C_2 + C_3 + A_2(s)C_3)\omega_n} \\ \\ \omega_r = \omega_n\sqrt{1 - 2\xi^2} \qquad \exists\forall|\xi| \leqslant \dfrac{1}{\sqrt{2}} \end{cases} \tag{2.71}$$

Equation 2.71 provides *all* the loop filter information, the specification for the resonating pulsation ω_r and its existence rule, since a second-order poles denominator is present. To finalize the loop filter design, the transfer functions $A_{1,2}(s)$ for the two amplifiers need **replacement**. A designer might employ a *first-order dominant pole* approximation to quantify the amplifier's static gain—denoted as A_o—and its associated unity gain frequency:

$$\begin{cases} A_i(s) = A_{o_i} \cdot \dfrac{1}{1 + \dfrac{s}{\omega_{pole}}} \\ \\ \omega_{\text{unity}} = \omega_{\text{pole}}\sqrt{A_{o_i}^2 - 1} \end{cases} \tag{2.72}$$

Replacing Equation 2.72 into 2.71 will provide the final **complete** description of the loop filter schematic transfer function without approximation or error. The margin of precision depends on the level of precision of the two operational amplifier desired by the designer.

2.7 Summary of Chapter II

This chapter provides an in-depth exploration of the **Continuous-Time Delta-Sigma (CT$\Delta\Sigma$) Modulator** design, highlighting its significance in **Image-Guided Surgery (IGS)** applications due to its potential for high precision and miniaturisation. The primary focus is on developing a 2^{nd}-order CT$\Delta\Sigma$ modulator, with an emphasis on addressing challenges such as **clock jitter** and achieving an optimal **Signal-to-Noise Ratio (SNR)**.

Key points discussed include:

- The choice of a *single-bit* CT$\Delta\Sigma$M simplifies the architecture and avoids complexities related to calibration or Dynamic Element Matching (DEM) procedures.
- The mathematical modelling of the CT$\Delta\Sigma$ modulator using Matlab/Simulink, including the derivation of the **STF**, **NTF**, and the Loop Filter $L(s)$.
- Analysis of the modulator's **performance limitations** due to clock jitter and the implementation of a jitter model in Simulink.
- Introduction of an **FIR filter** in the feedback loop to mitigate the effects of clock jitter, ensuring improved stability and noise performance.
- Detailed discussion on the calculation and adjustment of feedforward coefficients in the presence of an FIR-DAC to maintain the system's efficacy.

The chapter concludes with a comprehensive examination of simulation results, verifying the performance of the proposed CT$\Delta\Sigma$ modulator design. The findings highlight the modulator's ability to achieve the desired SNR and stability, thus laying a solid foundation for its application in IGS.

The subsequent chapter will introduce the **Phase Noise Profile Model**, expanding on the noise analysis and its implications on the CT$\Delta\Sigma$ modulator's performance in practical applications.

References

[6] S. Ahmed and V. Kakkar, "Modeling and simulation of an eight-bit auto-configurable successive approximation register analog-to-digital converter for cardiac and neural implants," *SIMULATION*, vol. 94, no. 1, pp. 11–29, 2018. DOI: 10.1177/0037549717716537 (Cited on page 9).

[7] M. Ortmanns and F. Gerfers, *Continuous-Time Sigma-Delta A/D Conversion: Fundamentals, Performance Limits and Robust Implementations*. Springer Berlin Heidelberg, 2006 (Cited on pages 10, 19, 32, 36, 43, 52).

[8] R. Schreier, S. Pavan, and G. C. Temes, *Understanding Delta-Sigma Data Converters*. John Wiley & Sons, Inc., 2017 (Cited on pages 10–15, 31, 32, 36, 37, 41–43, 52).

[9] K. Reddy and S. Pavan, "Fundamental limitations of continuous-time delta-sigma modulators due to clock jitter," *IEEE Trans. Circuits Syst. I*, vol. 54, no. 10, pp. 2184–2194, 2007. DOI: 10.1109/TCSI.2007.905648 (Cited on pages 12–15, 26, 28, 34, 36, 37, 43, 97, 98).

[10] P. Malcovati, S. Brigati, F. Francesconi, F. Maloberti, P. Cusinato, and A. Baschirotto, "Behavioral modeling of switched-capacitor sigma-delta modulators," *IEEE Trans. Circuits Syst. I*, vol. 50, no. 3, pp. 352–364, 2003. DOI: 10.1109/TCSI.2003.808892 (Cited on page 13).

[11] R. Saad, S. Hoyos, and S. Palermo, "Analysis and modeling of clock-jitter effects in delta-sigma modulators," in *MATLAB - A Fundamental Tool for Scientific Computing and Engineering Applications - Volume 1*, V. Katsikis, Ed., InTech, 2012 (Cited on pages 14, 36).

[12] T. Nandi, K. Boominathan, and S. Pavan, "Continuous-time $\Delta\Sigma$ modulators with improved linearity and reduced clock jitter sensitivity using the switched-capacitor return-to-zero dac," *IEEE J. Solid-State Circuits*, vol. 48, no. 8, 2013. DOI: 10.1109/JSSC.2013.2259012 (Cited on page 14).

[13] MathWorks. "Delta sigma toolbox." Online; accessed 24 September 2023, MathWorks. (2023), [Online]. Available: https://it.mathworks.com/matlabcentral/fileexchange/19-delta-sigma-toolbox (Cited on pages 16, 18).

[14] L. Hernandez, A. Wiesbauer, S. Paton, and A. Di Giandomencio, "Modelling and optimization of low pass continuous-time sigma delta modulators for clock jitter noise reduction," in *2004 IEEE International Symposium on Circuits and Systems*, IEEE, 2004, pp. I-1072–5. DOI: 10.1109/ISCAS.2004.1328384 (Cited on pages 26, 28, 34, 36).

[15] A. Ashry and H. Aboushady, "Modeling jitter in continuous-time sigma-delta modulators," in *2010 IEEE International Behavioral Modeling and Simulation Workshop*, IEEE, 2010, pp. 53–56. DOI: 10.1109/BMAS.2010.6156598 (Cited on pages 26, 28, 34, 36).

Chapter III:
Phase Noise Profile Model

THIS chapter delineates a framework [16] for simulating phase noise characteristics of a Voltage-Controlled Oscillator (VCO) in the clock source for the CTΔΣM. Positioned before the detailed design discussions, it provides the mathematical foundation to understand the influence of the clock source's phase noise spectrum on the CTΔΣM's in-band noise. This placement anticipates the design rationale and results discussed in Chapter V, offering the reader a logical progression of concepts despite the chronological inversion of the research process. The equations presented in Chapter IV will explain the design decisions for the clock source and support the thesis' conclusions.

3.1 First model characterisation

The first model studied to achieve the task employed an **Infinite Impulse Response** (IIR) filter is employed to approximate the *phase noise profile*, and is typically given by the following equation [17]:

$$G_{\text{IIR}}(z^{-1}) = \frac{\sum_{i=0}^{P} b_i z^{-i}}{1 + \sum_{j=0}^{Q} a_j z^{-j}}, \tag{3.73}$$

where P and Q are the order of the *numerator* and *denominator* polynomials, respectively, and b_i and a_j are the coefficients of the *FIR filter*.

For this model, only the *feedforward* path of the IIR filter is utilized. Therefore, the transfer function of this IIR filter is described by a summation of four singular transfer functions $H_i(z^{-1})$. These functions signify the *frequency breakpoints* of the phase noise profile (expressed in dBc/Hz), and the magnitude is determined using the α_i coefficients of equation 3.73. The following system of equations elaborates on this:

$$\begin{cases} H_i(z^{-1}) = \dfrac{A_i(1-\alpha_i)}{1-\alpha_i z^{-1}}, \\ \alpha_i = [1 + \dfrac{\omega_{\text{pole}_i}}{f_s}]^{-1} \qquad \text{for } i = 1, 2, \ldots, N \\ \dfrac{Y(z^{-1})}{U(z^{-1})} = \sum_{i=1}^{4} H_i(z^{-1}) \end{cases} \tag{3.74}$$

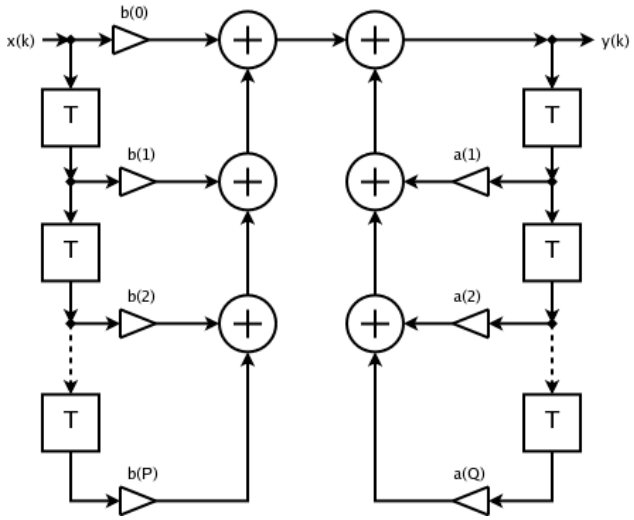


Figure 3.39: Generic IIR filter structure [17]

with

$$U(z^{-1}) = \mathscr{Z}\{u[n]\}, \quad u[n] \sim \mathcal{N}(0, \sigma^2) \tag{3.75}$$

Here, f_s denotes the *sampling frequency*, ω_{pole_i} is the *angular frequency* of the pole, and N represents the total number of poles. The input signal, denoted by $U(z^{-1})$, is a sequence $u[n]$ with a *Gaussian distribution* $\mathcal{N}(0, \sigma^2)$. This signal generates **random noise** around the profile created by the IIR filter coefficients to create the desired phase noise profile.

To effectively simulate and visualize the phase noise profile, a MATLAB script was initially developed. This script provides a suite of functionalities, including the setting of Bode plot options, the definition of the sampling frequency, and the configuration of the inverse z-transform variable. Its core operation, however, is constructing a transfer function designed to emulate a fundamental first-order low-pass filter characterized by a solitary pole at a specific frequency. This transfer function is initially described in the **s-domain**, as shown in Equation (3.76):

$$A(s) = k\frac{1}{1 + s\tau} \tag{3.76}$$

Here, k is the gain of the system, $p = -\frac{1}{\tau}$ is the pole, and $f_p = \frac{|p|}{2\pi}$ is the pole frequency.
To render this transfer function compatible with the digital world of the phase noise model, it is translated into the $\mathscr{Z}$**-domain**. This is achieved through the method of **forward differencing**, $s = \dfrac{z-1}{T_s}$, yielding the following

$\mathcal{Z}$-domain transfer function:

$$A(z^{-1}) = k\frac{T_s z^{-1}}{(1-z^{-1})\tau + T_s} \tag{3.77}$$

where T_s is the sampling period required in the discrete-time analysis, equal to the reciprocal of the **clock source output frequency**.

The script proceeds to compute the magnitudes of the s-domain and $\mathcal{Z}$-domain transfer functions across a logarithmically spaced frequency range. This explains how these transfer functions behave across the defined frequency spectrum. The results of these computations are depicted in two semi-log plots (Figure 3.40), which plot the magnitude of the transfer functions (expressed in decibels) against frequency. The first plot showcases the s-domain transfer function, while the second illustrates its $\mathcal{Z}$-domain counterpart. The Nyquist frequency, a pivotal reference point, is demarcated on the plots with a vertical dashed line.

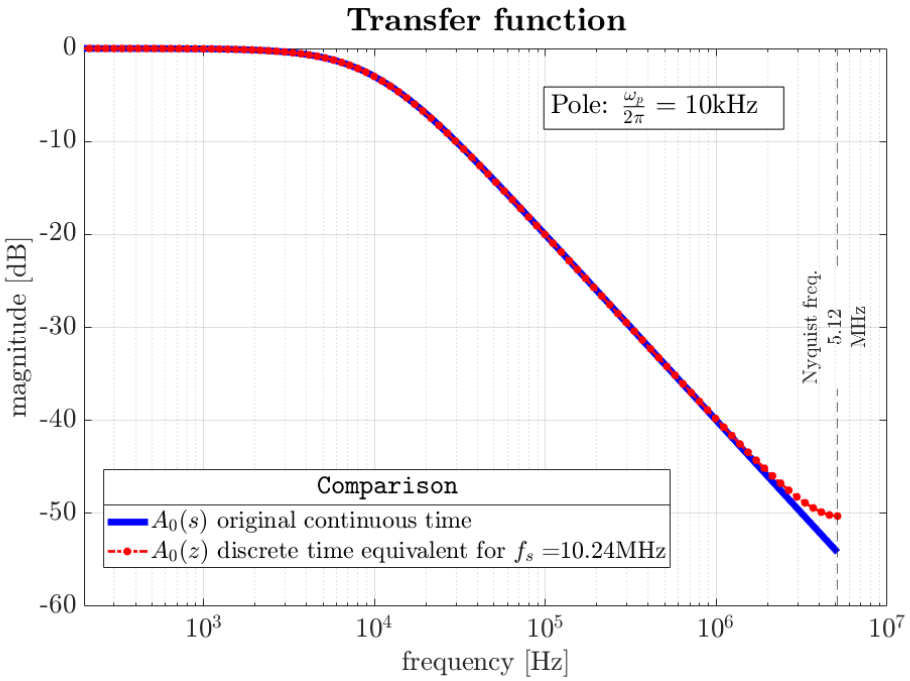


Figure 3.40: Conversion of a Laplace transform into a $\mathcal{Z}$-transform

The flowchart describing the Matlab code (see Figure 3.41) and the code itself are presented below, see Script 3.5.

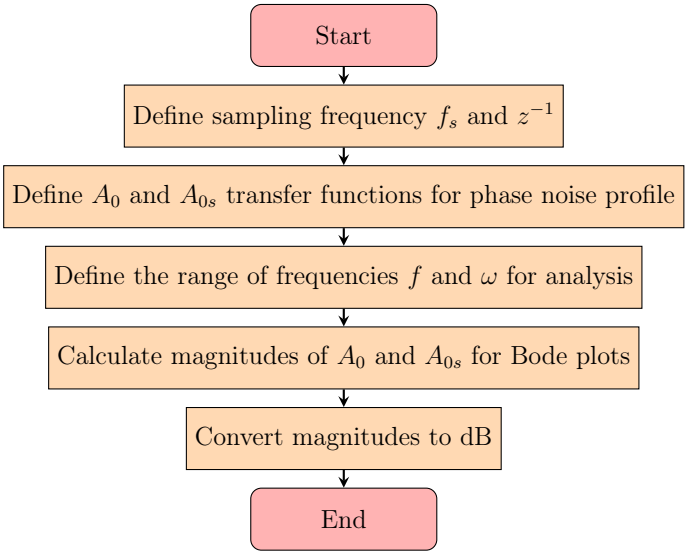


Figure 3.41: Flowchart of the Conversion into a $\mathcal{Z}$-transform.

Script 3.2: Conversion into a $\mathcal{Z}$-transform

```
% Clear the MATLAB environment before starting
clc
clear all
close all

%% Set options for Bode plots
opts = bodeoptions('cstprefs');
% Set interpreter for labels and title as 'latex'
opts.InputLabels.Interpreter = 'latex';
opts.OutputLabels.Interpreter = 'latex';
opts.XLabel.Interpreter = 'latex';
opts.YLabel.Interpreter = 'latex';
opts.Title.Interpreter = 'latex';
% Set font sizes
opts.Title.FontSize = 20;
opts.XLabel.FontSize = 16;
opts.YLabel.FontSize = 16;
opts.TickLabel.FontSize = 16;
% Set other options
opts.grid = 'on';
opts.FreqUnits = 'Hz';
opts.PhaseUnits = 'deg';
opts.PhaseWrapping = 'on';

%% Define sampling frequency and z^-1
% Set sampling frequency
fs = 10.24E6;
Ts = 1/fs;
% Define z^-1 with sampling period Ts
z_1 = tf([0 1], [1 0], Ts, 'Variable', 'z^-1');

%% Define Transfer Functions for Phase Noise Profile
% Set the dominant pole
wp = 2*pi*10E3;
```

```
% Define transfer function A0 with dominant pole wp
A0 = Ts*z_1/(Ts+ (1-z_1)/wp)

% Define the s-domain transfer function for comparison
s = tf([1 0], [0 1], 'Variable', 's');
A0_s = 1/(1+s/wp);

%% Define the range of frequencies for analysis
% Set range of angular velocities
f = logspace(2, log10(fs/2), 100);
w = 2*pi.*f;

%% Calculate magnitudes for Bode plots
% Calculate magnitude of A0 in s-domain
[A0_s_mag, A0_s_ph, w_s_out] = bode(A0_s, w);
f_s_out = w_s_out./(2*pi);
% Calculate magnitude of A0 in Z-domain
[A0_mag, A0_ph, w_out] = bode(A0, w);
f_out = w_out./(2*pi);
% Convert magnitudes to single-column vectors
A0_s_mag = squeeze(A0_s_mag);
A0_mag = squeeze(A0_mag);
% Convert magnitudes to dB
A0_s_mag_dB = 20*log10(A0_s_mag);
A0_mag_dB = 20*log10(A0_mag);
```

While this methodology facilitated a theoretical translation from the *s-domain* to the $\mathcal{Z}$-domain, its practical application in the **phase noise model** was not as successful as anticipated. Specifically, the transfer functions, intended to define the $H_i(z^{-1})$ functions for the phase noise model, did not yield the desired results. This indicates the **challenges** encountered when transitioning from *continuous-time* (s-domain) models to *discrete-time* ($\mathcal{Z}$-domain) models. Furthermore, this method was unsuccessful in *accurately replicating* the phase noise profile of the VCO. Despite the **IIR filter's** ability to approximate phase noise profiles, it *did not generate* a phase noise profile that *closely followed* the specified frequency points in dBc/Hz at particular frequency offsets. The inadequacy of this approach became apparent during the validation phase, leading to the omission of these results in favor of developing a more advanced model, as detailed in the subsequent chapter. This advanced model, which successfully aligns the phase noise with the specified frequency points, is illustrated in Figures 3.42, 3.60, and 3.61. Therefore, these experiences highlight the necessity for *further research* to develop more **precise methods** for modelling phase noise in Simulink.

3.2 Advanced model characterisation

In light of the challenges encountered in accurately modelling phase noise using an IIR filter, the focus now shifts towards investigating the impact of phase noise on the SNR of a Continuous-Time Delta-Sigma modulator.

Phase noise, characterized by its inherently *random phase fluctuations* in an otherwise perfect periodic signal, is essentially defined by its **spectrum** for this reason, the necessity of defining a *frequency profile* model to be applied to any signal.

3.3 How Phase Noise Affects CTΔΣM's SNR

To better understand how phase noise influences the spectral output of a (CTΔΣM), a *MATLAB*-based exploration was conducted. This involved **injecting phase noise** into a sinusoidal waveform and observing its consequent effect on the (CTΔΣM) output.
The MATLAB script is structured as follows:

- Phase Noise Generation: The script first generates a phase noise profile, defined by its PSD in dB/Hz.
- Frequency Shifting: The generated phase noise undergoes a frequency shift by a predefined amount. This is realized by multiplying the phase noise in the time domain with a complex exponential, equating to a frequency shift in the frequency domain.
- Phase Noise and Frequency Shift Plots: The script plots the original and frequency-shifted phase noise on the same graph, along with a phasor representing the frequency shift.
- CTΔΣM ADC Noise Modeling: The script evaluates how the frequency-shifted phase noise affects the output spectrum of a CTΔΣM ADC.
- In-Band Noise (IBN) Computation: The script computes the in-band noise due to the phase noise and the ADC's Noise Transfer Function (NTF).
- Convolution in Frequency Domain: The script performs a convolution between the differentiated NTF and the phase noise profile in the frequency domain.
- Final Plotting: The total in-band noise (IBN) is plotted, encompassing the IBN due to phase noise and the IBN due to the convolution operation.

The frequency profile of the **phase noise spectrum** is crucial in determining the Signal-to-Noise Ratio (SNR) of the CTΔΣM. The mathematical modelling of phase noise in electrical **oscillators**, as outlined in several works

[16], [18], [19], provided the foundational understanding necessary for this study. These models allowed for a detailed exploration of how phase noise variations can significantly influence the SNR, thereby impacting the overall performance of the CTΔΣM. The **connection** between phase noise and the performance of the CTΔΣM was further elucidated in another study [20]. Furthermore, the `script` used in this research offers a practical demonstration of several key signal processing concepts, including *frequency shifting*, *convolution in the frequency domain*, and the *impact of noise* on system performance.

3.3.1 Phase Noise profile

The MATLAB script begins by defining the breakpoints in the frequency range (f_{break}) and the corresponding phase noise levels ($\phi_{\text{noise_level}}$). These *break points* delineate the frequency points where the phase noise profile changes.

A different relationship between phase noise and frequency is considered for each delineated frequency range. The phase noise follows a $1/f^3$ law in the first range, a $1/f^2$ law in the second and third ranges, and remains constant in the fourth range. The phase noise levels are manually adjusted to match the slopes of the $1/f^3$ and $1/f^2$ laws, as shown in Figure 3.42. This pattern follows the phase characteristic of the **oscillator**, which is necessary to create an **integrated clock** to generate the *clock signal* required by the CTΔΣM ADC.
The script then defines the frequency vector for positive frequencies (f_{pos}) using a logarithmic scale. Following the frequency pattern described before, the phase noise profile, $\phi(f)$, is calculated directly as a *function of frequency*, independently for each frequency range.

$$\phi(f) = \begin{cases} -30log_{10}(f) & f \leq 10^4 \\ -20log_{10}(f) & 10^4 < f \leq 10^6 \\ -100\frac{\text{dBc}}{\text{Hz}} & f > 10^7 \end{cases} \tag{3.78}$$

This is accomplished by calculating the phase noise in each range as a linear function of the logarithm of the frequency, using known slopes and intercepts (q) that are calculated to match the predefined phase noise levels at the breakpoints at each *decade* from 1kHz up to 10MHz (see Script 3.3). The **breakpoints** and corresponding phase noise levels are indicated with red dots, and the plot is annotated with the mathematical expression of the phase noise profile. The flowchart describing the Matlab code (see Figure 3.43)

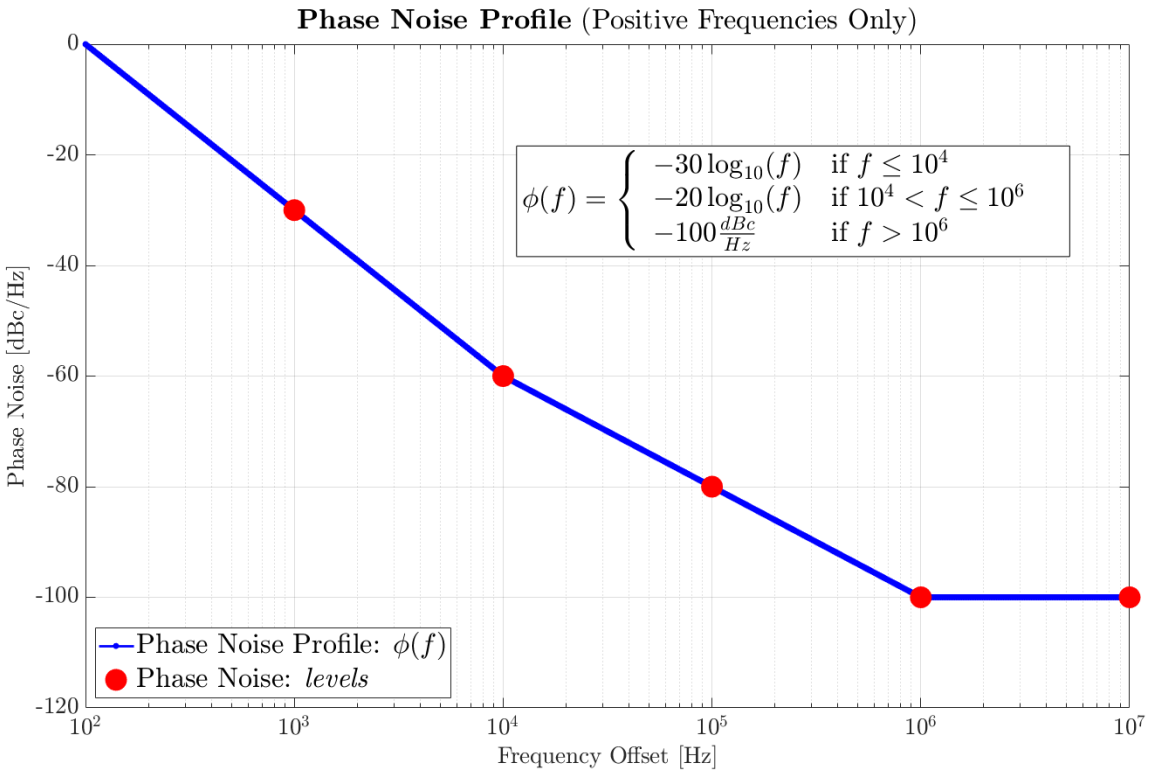
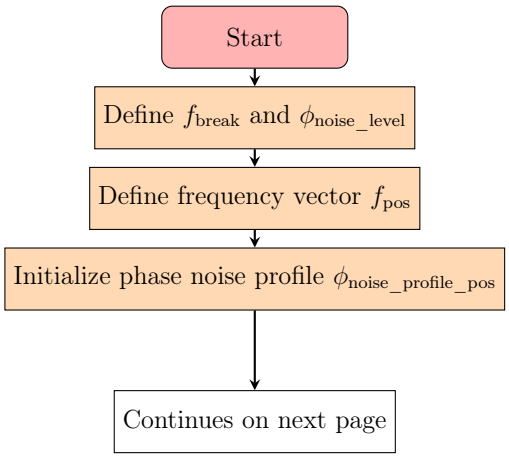


Figure 3.42: Phase noise profile (Single-Sided).

and the Matlab code itself, which is divided into generating a single-sided phase noise profile and extending it to a double-sided phase noise profile, are presented below. See Scripts 3.3 and 3.4.



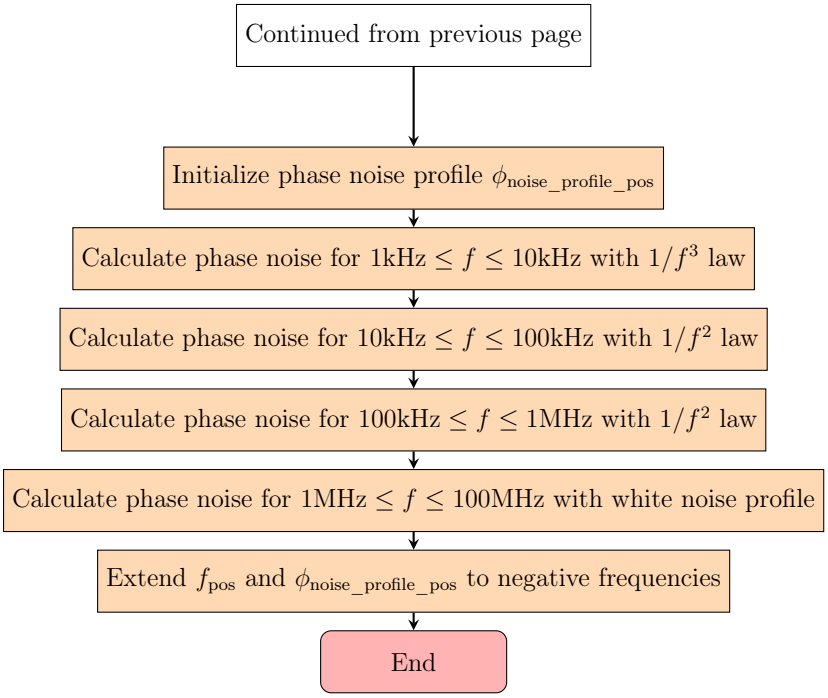


Figure 3.43: Flowchart of the Matlab code for generating the phase noise profile.

Script 3.3: Phase Noise profile (Single-Sided).

```
% Define the frequency points where the phase noise profile changes
f_break =           [1E3, 10E3, 100E3, 1E6, 10E6];
% Define the phase noise levels at the break points
phi_noise_level =   [-30, -60, -80, -100, -100];
% Define the frequency vector for positive frequencies
f_pos = logspace(2, log10(f_break(5)*1), N).';

% Define the phase noise profile in dBc/Hz for positive frequencies
phi_noise_profile_pos = zeros(N, 1);

% Generate the phase noise profile directly as a function of
    frequency
i = 1;
% For the FIRST frequency range, the phase noise follows a 1/f^3 law
    i = 2; %1kHz - 10kHz;
    index_2 = find(f_pos <= f_break(i));
    q_2 = abs(-30*log10(f_break(i)) - phi_noise_level(i) );
    %mx + q
    phi_noise_profile_pos(index_2) = - 30*log10(f_pos( index_2 )) +
        q_2;

% For the SECOND frequency range, the phase noise follows a 1/f^2
    law
    i = 3; %10kHz - 100kHz;
    index_3 = find(f_pos > f_break(i-1) & f_pos <= f_break(i) );
    q_3 = abs(-20*log10(f_break(i)) - phi_noise_level(i) );
    %mx + q
```

```
            phi_noise_profile_pos(index_3) = - 20*log10(f_pos( index_3 )) +
                q_3;

            i = 4; %100kHz - 1MHz;
            index_4 = find(f_pos > f_break(i-1) & f_pos <= f_break(i) );
            q_4 = abs(-20*log10(f_break(i)) - phi_noise_level(i) );
            %mx + q
            phi_noise_profile_pos(index_4) = - 20*log10(f_pos( index_4 )) +
                q_4;

        % For the THIRD frequency range, the phase noise follows a WHITE
            PROFILE
            i = 5; %1MHz - 100MHz
            index_5 = find(f_pos > f_break(i-1)); %DON'T NEED & f_pos <=
                f_break(i)
            %y=q
            phi_noise_profile_pos(index_5) = phi_noise_level(i);
```

The final step *extends* the frequency vector to negative frequencies by symmetry to obtain the **full frequency vector** (`f`), and similarly extends the phase noise profile to obtain the **full phase noise profile** ($\varphi_{\text{double_sided}}$). This double-sided phase noise profile is displayed in a linear plot (Figure 3.44), again indicating the breakpoints and *corresponding* phase noise levels.

Script 3.4: Phase Noise profile (Double-Sided).

```
        %% PHASE NOISE PROFILE in {Double side}
        % Extend the frequency vector to negative frequencies by symmetry
        f = [-flip(f_pos(2:end)); f_pos];

        % Extend the phase noise profile to negative frequencies by symmetry
        phi_noise_profile = [flip(phi_noise_profile_pos(2:end));
            phi_noise_profile_pos];
```

The MATLAB code provided in Script 3.4 performs this extension. The first line of the code *reverses* the vector of positive frequencies, excluding the DC component, and appends it to the start of the original frequency vector. The result is a full-frequency vector that spans both *positive* and *negative frequencies*. The second line of the code performs a similar operation on the phase noise profile. The phase noise profile of positive frequencies is reversed and appended to the start of the original profile, yielding a complete phase noise profile corresponding to the full frequency vector.

3.3.2 Generating the Phase Noise

In the new segment of the MATLAB script, phase noise in the frequency domain is produced using a **complex Gaussian random process**, denoted as $\mathcal{N}_{\mathcal{C}}$. This procedure is executed for both positive and all frequency ranges, respectively labelled as $L_{\phi}(f)$ and $L_{\varphi_{\text{DS}}}(f)$.

For each frequency, a complex Gaussian random variable with a zero *mean*

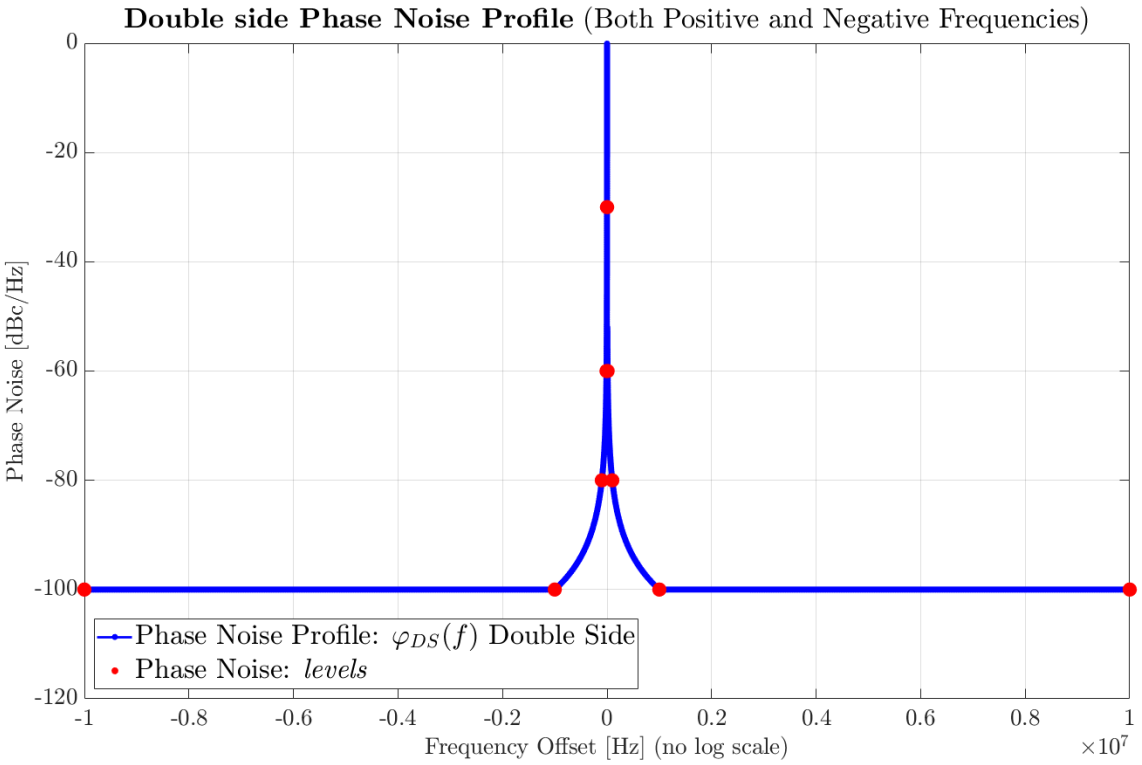


Figure 3.44: Phase Noise profile (Double-Sided).

is generated. The *standard deviation* of this random variable is determined by the square root of the power spectral density (PSD) of the phase noise profile at the specific frequency. Including a complex component in the random variable ensures the correct processing of the **phase**, so it adheres to the profile described in the previous step.

The following equations can mathematically represent this process of generating phase noise:

$$\begin{cases} L_{\phi}(f) \sim \mathcal{N}_{\mathcal{C}}\left(0, \sqrt{\dfrac{\phi(f)_{\text{dBc/Hz}}}{10}}\right), & f > 0 \\ L_{\varphi_{\text{DS}}}(f) \sim \mathcal{N}_{\mathcal{C}}\left(0, \sqrt{\dfrac{\varphi_{DS}(f)_{\text{dBc/Hz}}}{10}}\right), & -\infty < f < +\infty \end{cases} \tag{3.79}$$

Here, $L_{\phi}(f)$ represents the **generated phase noise** for positive frequencies, while $L_{\varphi_{\text{DS}}}(f)$ corresponds to the phase noise across the full frequency range.

Both are modelled as complex Gaussian random variables with zero mean and a standard deviation that reflects the square root of the PSD of the *phase noise profile*.

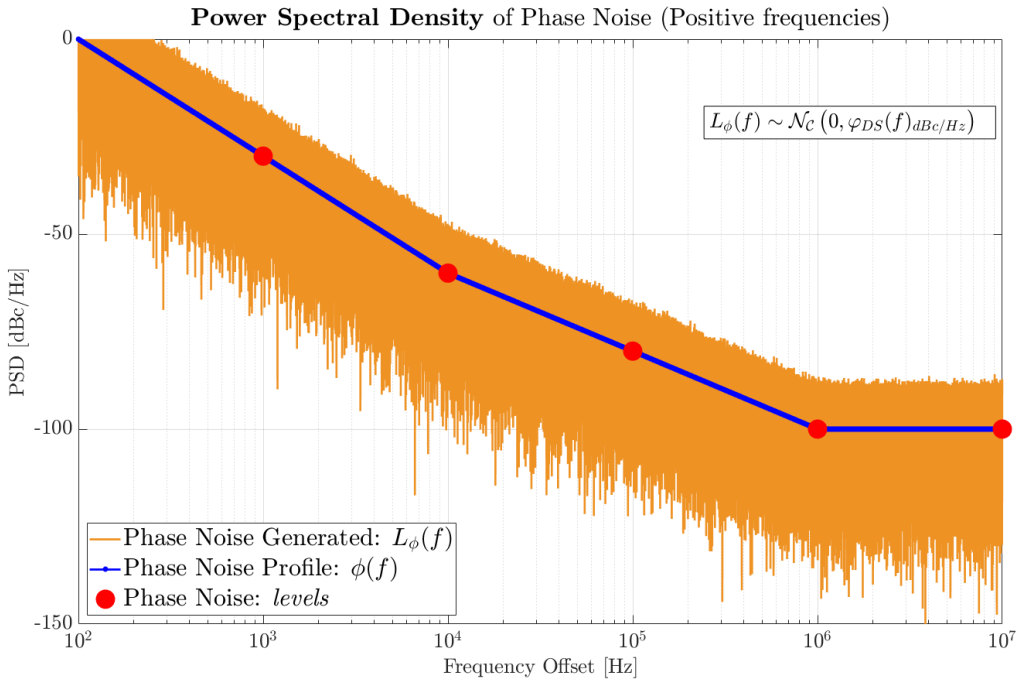


Figure 3.45: Phase Noise generated - positive frequencies.

The first plot (see Figure 3.45) is a semi-logarithmic plot showing the PSD of the generated phase noise for positive frequencies, along with the phase noise profile and the phase noise levels at the breakpoints. The second plot (see Figure 3.46) is a linear plot showing the PSD of the generated phase noise for both positive and negative frequencies, again along with the phase noise profile and the phase noise levels at the breakpoints.
The MATLAB code for this section is as follows in Script 3.5:

Script 3.5: Generating Phase Noise.

```
% Generate the phase noise in the frequency domain: PSD of the
    generated phase noise
phi_noise_freq = (randn(size(f)) + 1j*randn(size(f))) .* sqrt(10.^(
    phi_noise_profile/10));
% Calculate the power spectral density (PSD) of the phase noise
PSD_phi_noise = abs(phi_noise_freq).^2;
% Convert the PSD to dBc/Hz
PSD_phi_noise_dBc_Hz = 10 * log10(PSD_phi_noise);
```

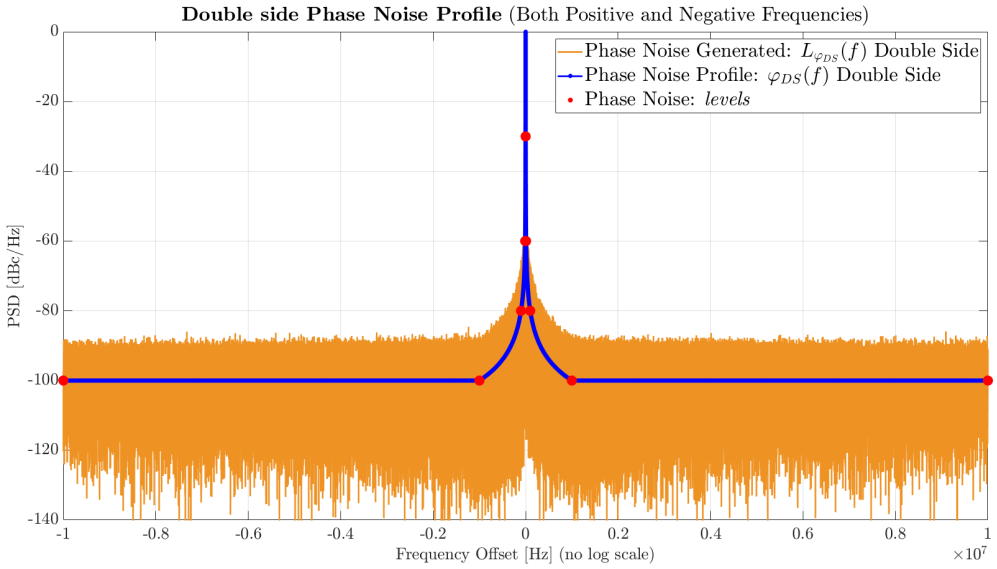


Figure 3.46: Phase Noise generated - full spectrum.

3.3.3 Frequency Shifting of Phase Noise

The phase noise spectrum now needs to be shifted to align with the signal's frequency at the input of the Analog-to-Digital Converter (ADC). This process is not as straightforward as simply scaling the amplitude array but involves a rigorous application of signal theory. The phase noise spectrum $L_{\varphi_{DS}}[f]$ is frequency shifted by $f_0 = 16\,\text{kHz}$, the input frequency of the ADC.

The equation mathematically represents this in equation (3.80):

$$L_{\varphi_{DS}}[f - f_0] = \mathscr{F}_k \left\{ \mathscr{F}_k^{-1}\{L_{\varphi_{DS}}[f]\} e^{-j(2\pi f_0)\frac{n}{f_s}} \right\} \tag{3.80}$$

where $\mathscr{F}_k$ and $\mathscr{F}_k^{-1}$ denote the forward and inverse Fourier transform, n is the time index, f_s is the sampling frequency, and the term $e^{-j(2\pi f_0)\frac{n}{f_s}}$ represents a complex exponential used to implement the frequency shift. This equation is derived from the Discrete Fourier Transform (DFT) properties. Since the original phase noise spectrum $L_{\varphi_{DS}}[f]$ is given in the *frequency domain*, the inverse Fourier transform $\mathscr{F}_k^{-1}$ is used to *shift* it to the time domain. The complex exponential $e^{-j(2\pi f_0)\frac{n}{f_s}}$ is then multiplied in the time domain, a process that **corresponds** to a *frequency shift* in the frequency domain. This operation is crucial because it adheres to the fundamentals of signal processing theory, ensuring the **correct** manipulation of the phase

noise **spectrum**. The resulting signal in the time domain represents the phase noise shifted by the desired frequency f_0. The forward Fourier transform $\mathscr{F}_k$ is then applied to bring the frequency-shifted phase noise back to the frequency domain for visualization and further processing. This rigorous process ensures that the phase noise spectrum is correctly shifted to align with the ADC input frequency, which is of utmost importance in this analysis; see Figure 3.47. The same frequency shift can be applied to the *phase*

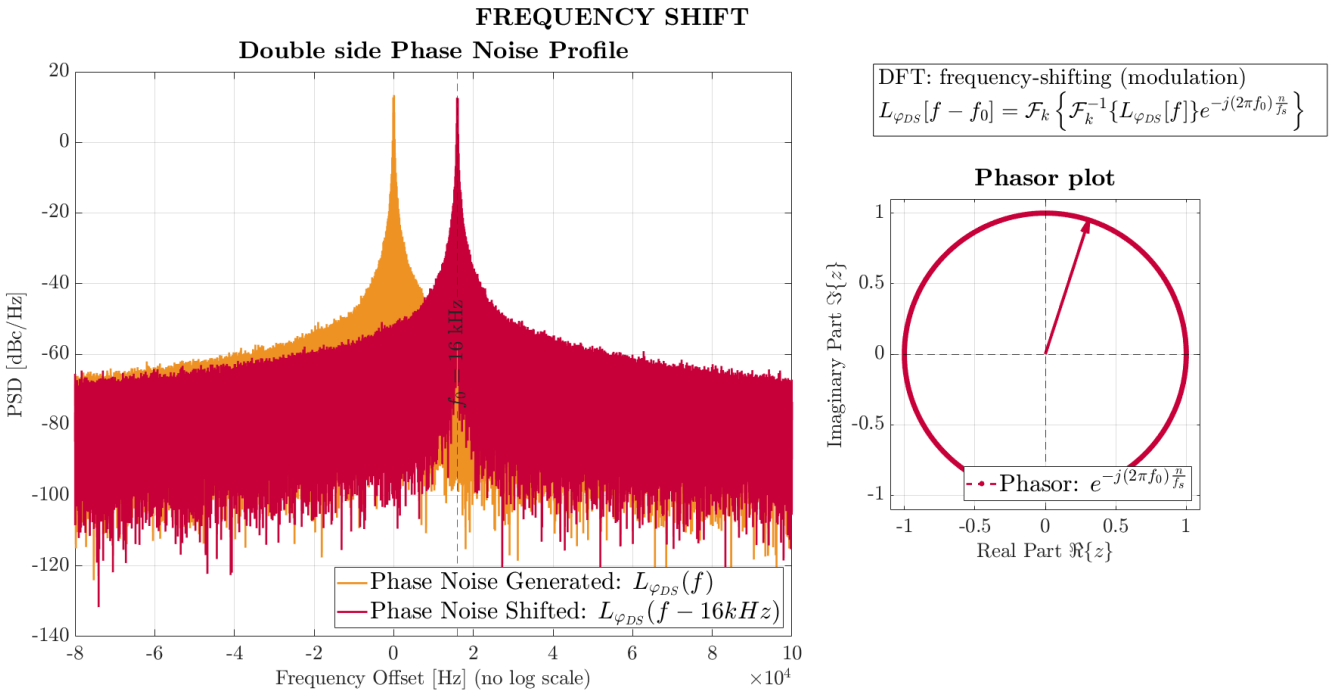


Figure 3.47: Phase Noise Spectrum Shifted to ADC Input Frequency.

noise levels which are originally defined in the equation 3.78, as shown in figure 3.48. The flowchart in Figure 3.49 illustrates the MATLAB code for the frequency shift operation, which involves converting from the frequency domain to the time domain, applying a frequency shift, and converting back to the frequency domain. See Script 3.6.

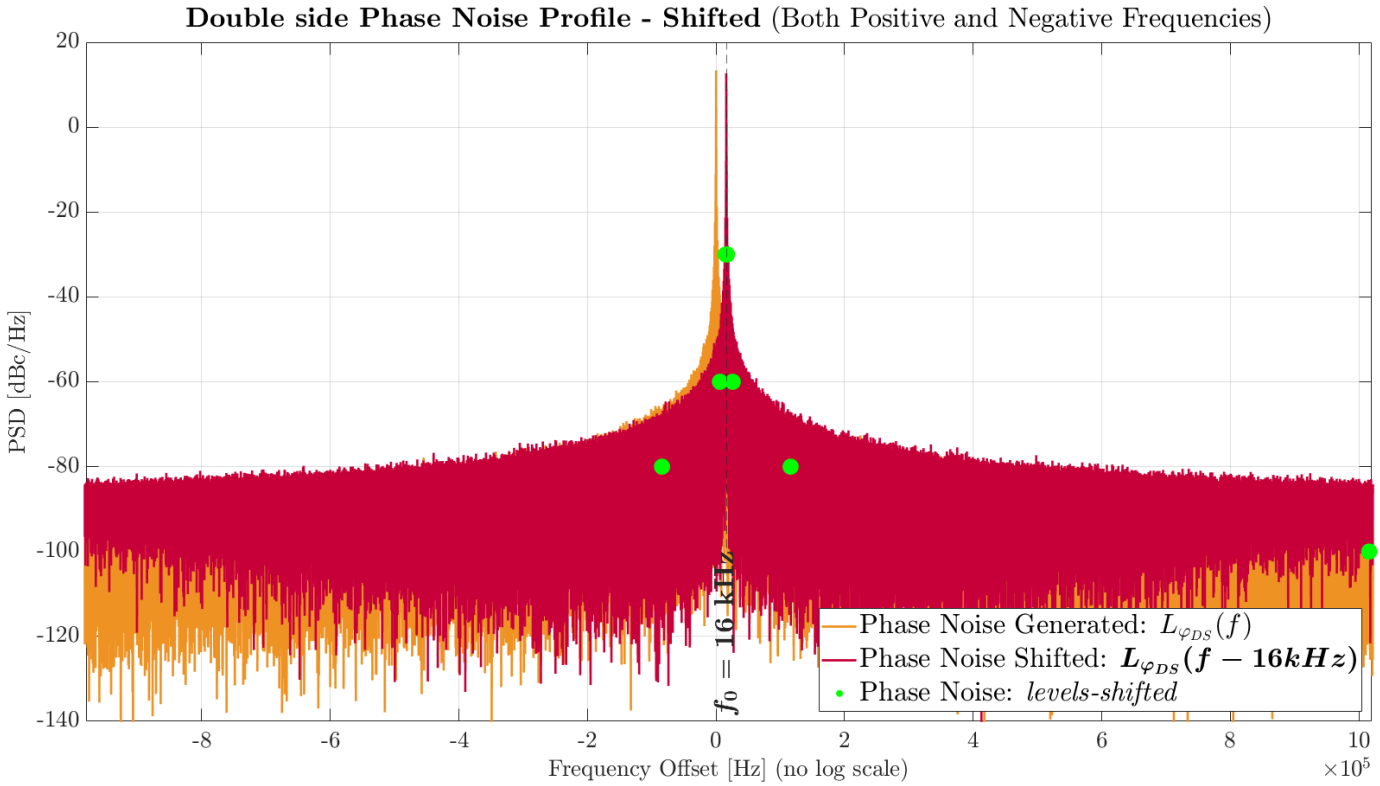


Figure 3.48: Phase Noise Spectrum Shifted.

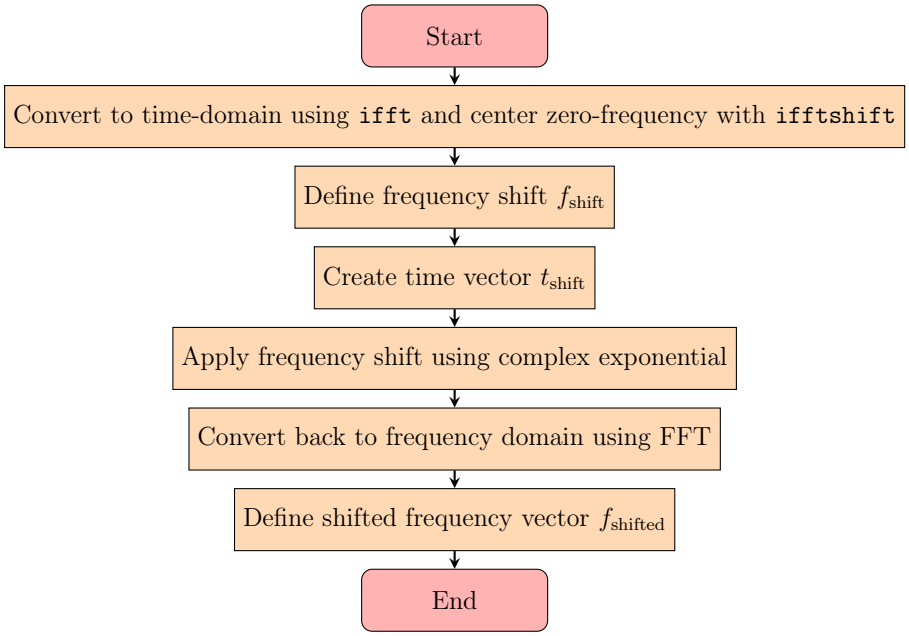


Figure 3.49: Flowchart of the Matlab code for performing the frequency shift operation.

Script 3.6: Frequency Shift Operation.

```
% Shift from frequency to time domain
freq_deviation_time = ifft(ifftshift(phi_noise_freq), 2*N-1);
% Define the frequency shift
f_shift = 16e3;
% Create a time vector to correspond to your data
t_shift = (0:length(f)-1).'/fs;
% Multiply with a complex exponential to implement the frequency
    shift
phi_noise_time_shifted = freq_deviation_time .* exp(-j * 2 * pi *
    f_shift .* t_shift);
% Shift back to the frequency domain
phi_noise_freq_shifted = fftshift(fft(phi_noise_time_shifted));
% Shifted frequency vector
f_shifted = f + f_shift;
```

Script 3.6 starts by shifting the phase noise spectrum, `phi_noise_freq`, from the frequency domain to the time domain. This is achieved using the inverse Fourier transform, `ifft`. The function `ifftshift` is used to rearrange the Fourier transform output, moving the zero-frequency component to the centre of the spectrum. Next, the frequency shift, denoted as `f_shift`, is defined to be 16 kHz. This value was chosen to match the input frequency of the ADC. To correspond to the data, a time vector, `t_shift`, is created with the same length as the frequency vector. The time vector increments in steps of the reciprocal of the sampling frequency `fs`. The frequency shift is then implemented in the time domain. This is done by multiplying the time-domain signal, `freq_deviation_time`, by a complex exponential, effectively shifting the frequency by `f_shift` in the frequency domain. The frequency-shifted signal in the time domain, `phi_noise_time_shifted`, is then transformed back to the frequency domain. The Fourier transform (`fft`) is used for this transformation, and the function `fftshift` is used to rearrange the output of the Fourier transform, moving the zero-frequency component to the centre of the spectrum. Lastly, a new frequency vector, `f_shifted`, is generated. This represents the original frequency vector `f` but shifted by `f_shift`.
In summary, the MATLAB code shown effectively translates the phase noise spectrum to align with the input frequency of the ADC. This is achieved by leveraging the properties of the Fourier transform to implement the frequency shift in the time domain, ensuring adherence to signal processing principles.

3.3.4 In-Band Noise Computation

The in-band noise of a CTΔΣM modulator can be computed by considering two primary sources [21]: the clock source **phase noise spectrum** $L_{\varphi_{DS}}$ and the ADC's **noise transfer function** (NTF).

The equation representing this is shown below (see equation 3.81).

$$\mathbf{IBN}\left(e^{j\omega}\right) = \frac{A^2}{2}\left(\frac{f_{\text{in}}}{f_s}\right)^2 \mathbf{L}_{\varphi_{DS}}\left(e^{j\omega - j2\pi(f_{in})}\right) + \\ + PSD\left\{\left(1 - e^{-j\omega}\right)\mathbf{NTF}(e^{j\omega})\right\} * \frac{L_{\varphi_{DS}}(e^{j\omega})}{4\pi^2} \tag{3.81}$$

Here, A is the amplitude of the ADC's input signal, f_{in} is its input frequency and f_s is the sampling frequency of the **clock source**. The variable ω is the normalized frequency variable, and $e^{j\omega}$ represents the complex exponential function, which is a key feature of digital signals. The equation is expressed in terms of $e^{j\omega}$ because the output of the ADC is discrete. This is a crucial point: although the input signal to the ADC is a continuous-time signal with amplitude A and frequency f_{in}, the ADC converts this continuous-time signal into a discrete-time signal. Hence, the digital output of the ADC is naturally represented in the complex exponential form $e^{j\omega}$, reflecting the discrete and periodic nature of the digital signal.

The first term in equation (3.81) represents the **contribution** of the *clock source* phase noise **shifted** by the input frequency of the ADC, 16kHz. This term is additionally weighted by the square of the amplitude of the input signal and the square of the ratio between the input frequency and the sampling frequency. This is expressed mathematically as:

$$\text{IBN}_1\left(e^{j\omega}\right) = \frac{A^2}{2}\left(\frac{16\text{kHz}}{f_s}\right)^2 L_{\varphi_{DS}}\left(e^{j\omega - j2\pi(16\text{kHz})}\right) \tag{3.82}$$

It is important to note that the phase noise of the clock source is crucial in the resulting noise characteristics of the ADC output. After being shifted to the ADC's input frequency, the phase noise becomes integral to the total **in-band noise**. Figure 3.50 shows how the phase noise generated by the clock source, when modulated by an input signal of amplitude 125mV and frequency of 16kHz, affects the in-band noise of the ADC output. The ADC operates at a sampling frequency of 10.24MHz:
As indicated in the textbox within the figure, the settings used for the Power Spectral Density (PSD) plot are important. The Equivalent Noise Bandwidth (ENBW) is set to 234.37 Hz to represent the power spectral density accurately. The number of points N used for the FFT is 2^{21}, providing high-resolution frequency data. Furthermore, the sampling frequency (f_s) used for the FFT is 327.7MHz. These settings are crucial for obtaining an accurate depiction of the in-band noise. The ADC operates at a clock frequency of 10.24MHz due to the clock source. Further *explanation* of these settings can

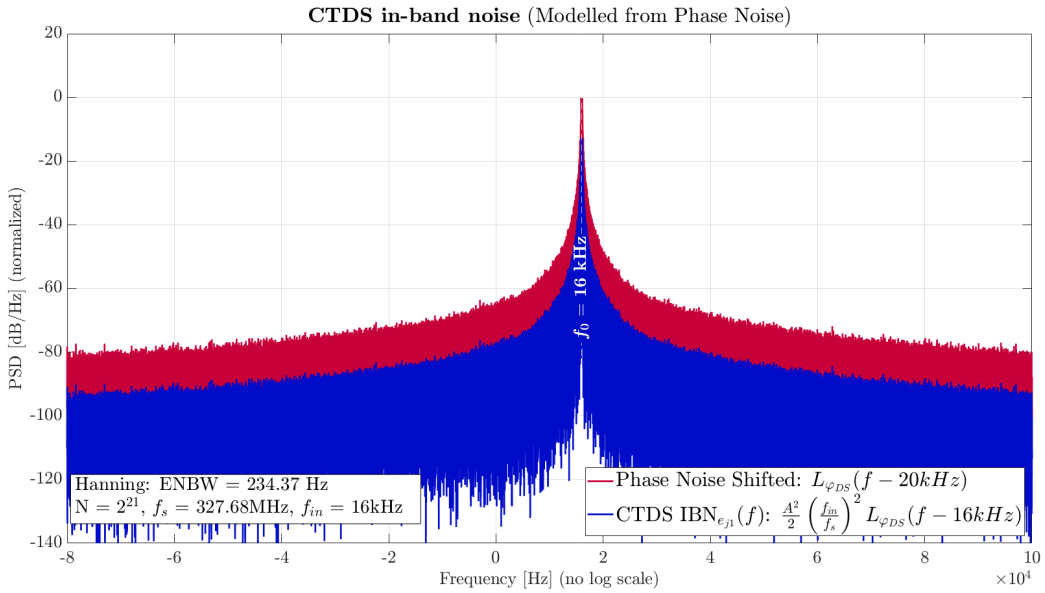


Figure 3.50: Phase Noise shifted by the input frequency of the ADC.

be found in Appendix A of the thesis.

The second term of equation (3.81) represents the convolution of the phase noise spectrum with the power spectral density (PSD) of the product of the first difference, which models the behaviour of the modulator's feedback loop, and the noise transfer function (NTF).

$$\mathrm{IBN}_2\left(e^{j\omega}\right) = \mathrm{PSD}\left\{\left(1-e^{-j\omega}\right)\mathrm{NTF}(e^{j\omega})\right\} * \frac{L_{\varphi_{DS}}(e^{j\omega})}{4\pi^2} \tag{3.83}$$

This convolution operation in the frequency domain corresponds to a multiplication operation in the time domain. Thus, the second term models the **interaction** of the phase noise with the modulator's dynamics. As shown in the textbox within Figure 3.51, the convolution operation in the frequency domain is implemented leveraging the Discrete Fourier Transform (DFT) *properties*. The sequence of operations involves first transforming the PSD and the phase noise into the time domain via the inverse DFT. This allows the execution of a simple *multiplication* operation in the time domain, **corresponding** to the *convolution* operation in the frequency domain. The result of this multiplication is then transformed back into the frequency domain using the DFT, as expressed in the following equation:

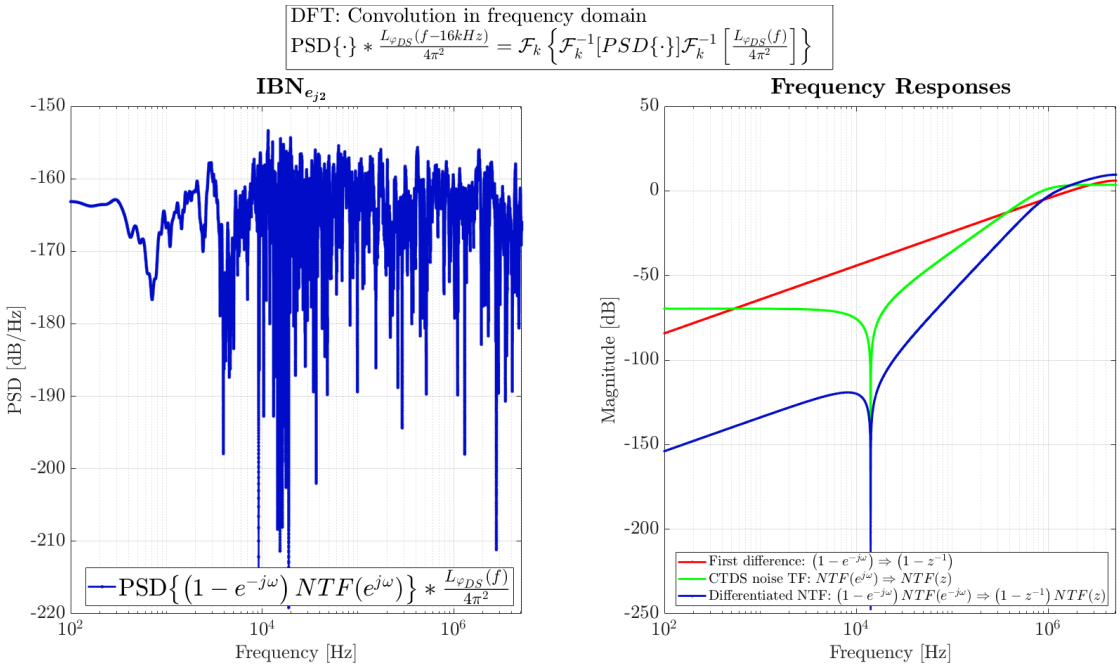


Figure 3.51: Phase Noise convoluted with the NTF.

$$\text{PSD}\left\{\cdot\right\} * \frac{L_{\varphi_{DS}}(f)}{4\pi^2} = \mathscr{F}\left\{\mathscr{F}^{-1}\left[\text{PSD}\left\{\cdot\right\}\right]\mathscr{F}^{-1}\left[\frac{L_{\varphi_{DS}}(f)}{4\pi^2}\right]\right\} \tag{3.84}$$

In this equation, $\mathscr{F}$ and $\mathscr{F}^{-1}$ denote the DFT and the inverse DFT, respectively. This approach effectively captures the impact of the phase noise interaction on the modulator's dynamics.

Conclusion on the IBN:

The total in-band noise (IBN) of the Continuous-Time Delta-Sigma (CT$\Delta\Sigma$M) modulator is computed by combining the contributions from two primary sources: the *phase noise* of the clock source shifted by the ADC input frequency and the convolution of the phase noise spectrum with the power spectral density (PSD) of the product of the first difference and the noise transfer function (NTF).
Figure 3.52 illustrates the total IBN computed for a low-pass ADC with a bandwidth of 20 kHz. We limit the spectrum to the in-band range as we are specifically evaluating the in-band noise. The analysis reveals that the *phase noise* contribution, when shifted to the ADC input frequency due to

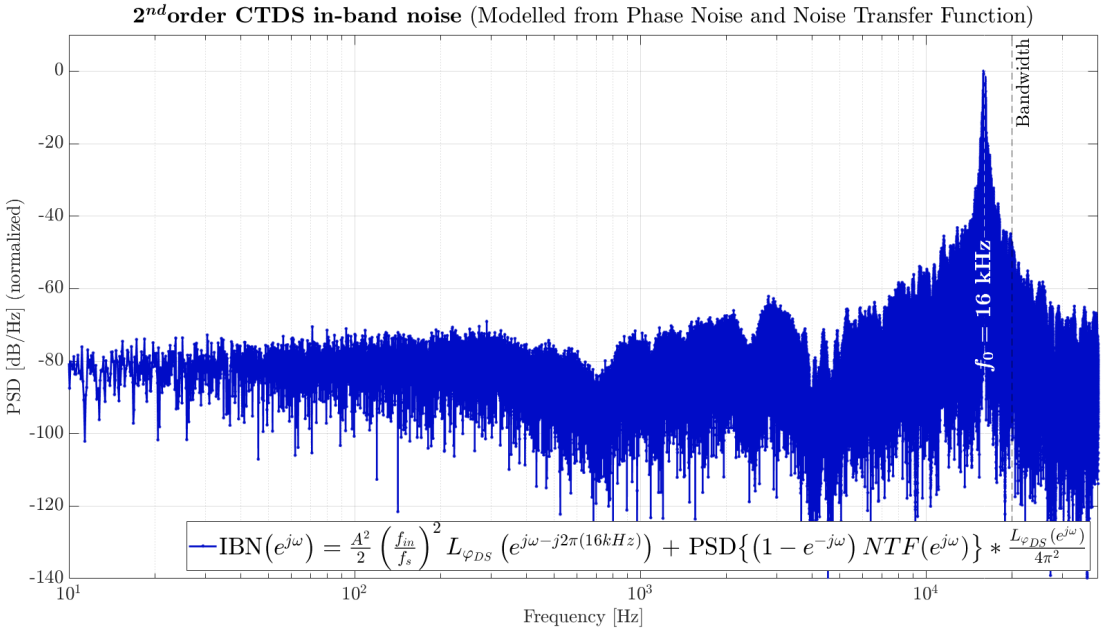


Figure 3.52: Total IBN until the CTΔΣM bandwidth of 20kHz.

the *closed-loop nature* of the ADC, can lead to a **reduction** in the Signal-to-Noise Ratio (SNR). This reduction in SNR essentially translates into a *loss in resolution*, impacting the overall performance of the CTΔΣM modulator. Consequently, understanding and mitigating the impact of phase noise on the IBN becomes a critical **aspect** of CTΔΣM modulator **design**.

The total IBN can be approximated by:

$$\text{IBN}\left(e^{j\omega}\right) = \frac{A^2}{2}\left(\frac{f_{in}}{f_s}\right)^2 L_{\varphi_{DS}}\left(e^{j\omega - j2\pi(f_{in})}\right) + \mathcal{O}\left(L_{\varphi_{DS}}(e^{j\omega})\right) \tag{3.85}$$

This equation suggests that the first term, representing the *phase noise* shifted by the ADC input frequency, **dominates** the total IBN [22]. The equation can be further simplified, resulting in:

$$\text{IBN}\left(e^{j\omega}\right) \approx \frac{A^2}{2}\left(\frac{f_{in}}{f_s}\right)^2 L_{\varphi_{DS}}\left(e^{j\omega - j2\pi(f_{in})}\right) \tag{3.86}$$

Here, the $\mathcal{O}\left(L_{\varphi_{DS}}(e^{j\omega})\right)$ term is dropped, which acknowledges the existence of additional contributing factors. This approximation seems justified, given the observed dominance of the first term in the analysis. However, it is

important to note that this is an approximation, and the *full complexity* of the system can only be understood with the complete equation. It is also crucial to consider the impact of the omitted term when the system *parameters change* in a way that might make it significant.

Simulation Results

The simulation results, which will be discussed in Chapter V and depicted in Figure 3.53, provide compelling evidence supporting the approximation given by Equation (3.86). The reduction in SNR observed in the simulation closely aligns with the theoretical expectations. The clock source phase noise spectrum, superimposed on the input frequency signal, decreases the power ratio between the input signal and the ADC noise.

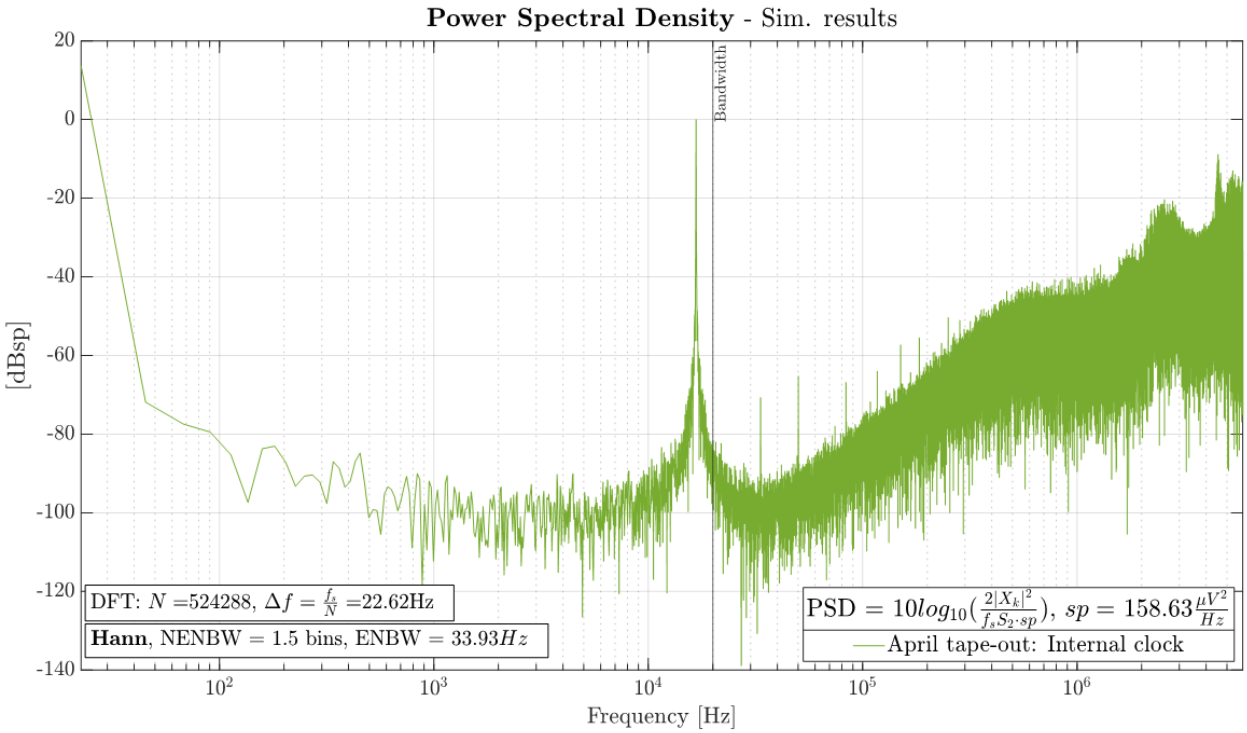


Figure 3.53: ADC output spectrum with internal clock.

It is worth noting that the resolution of the FFT plays a crucial role in accurately characterizing this phenomenon. The results shown here employ a high FFT resolution of 34Hz, made possible by using a Hanning window for spectrum evaluation. This high resolution, encompassing 2^{19} FFT bins,

allows for a more accurate estimation of the impact of $L_{\varphi_{DS}}\left(e^{j\omega - j2\pi(f_{in})}\right)$. Utilising a **lower** FFT resolution can potentially lead to misleading results by overestimating the SNR. This overestimation occurs because a lower resolution *obscures* the true noise characteristics of the system. Therefore, using a sufficiently high FFT resolution is vital for reliable results and an honest representation of system performance.

3.4 Phase Noise Analysis in the Discrete-Time Domain

This section is dedicated to **validating** the developed model for *phase noise*. A *sinusoidal discrete-time signal* will serve as the meansn for this validation. Once converted into the time domain, the phase noise will be added to the sinusoidal signal. This will facilitate an in-depth examination of the effects of phase noise on the signal. The waveforms and spectra of the original and the phase noise-affected signals in the time domain will be compared to affirm the accuracy of the phase noise generation process. The process is bifurcated into two parts. The **first** part details the phase noise **conversion** from the *frequency* domain to the *time* domain. The **second** part delves into the modulation of a sinusoidal signal with the generated phase noise and the ensuing analysis. This structured approach comprehensively verifies the **phase noise model**.

3.4.1 Phase Noise into the Discrete-Time Domain

This section delves into the process of converting the frequency deviation of the *generated phase noise*, denoted as $L_{\varphi_{\mathrm{DS}}}(f)$ or **phase noise generated**, into a real quantity in the *time domain*, represented as $\Phi_{DS}[n]$. This conversion is accomplished using the Inverse Fast Fourier Transform (IFFT). The transformation is significant as it ensures that the frequency deviation in the time domain corresponds to a real deviation in frequency at every point in time.

The mathematical representation of this conversion process is as follows:

$$\Phi_{DS}[n] = \sum_{k=-N}^{N} \Re\left\{\mathscr{F}_k^{-1}\left\{L_{\varphi_{\mathrm{DS}}}[k]\right\}\right\} \tag{3.87}$$

Equation 3.87 illustrates the transformation of frequency deviation from the frequency domain to the time domain. The elements of this equation are

explained below:

- $\Phi_{DS}[n]$ stands for the frequency deviation in the time domain, a real quantity.
- $\mathscr{F}_k^{-1}$ represents the Inverse Fast Fourier Transform (IFFT), applied to the frequency deviation of the phase noise $L_{\varphi_{\mathrm{DS}}}[k]$.
- $\Re\{\cdot\}$ is the real part of the result, ensuring that the frequency deviation in the time domain is a real quantity.
- The summation $\sum$ indicates the integration of phase deviation over time, resulting in a phase noise sequence in the time domain.
- This integration process is crucial since the phase of a signal at any given time is the integral of the frequency deviation of the signal up to that point.

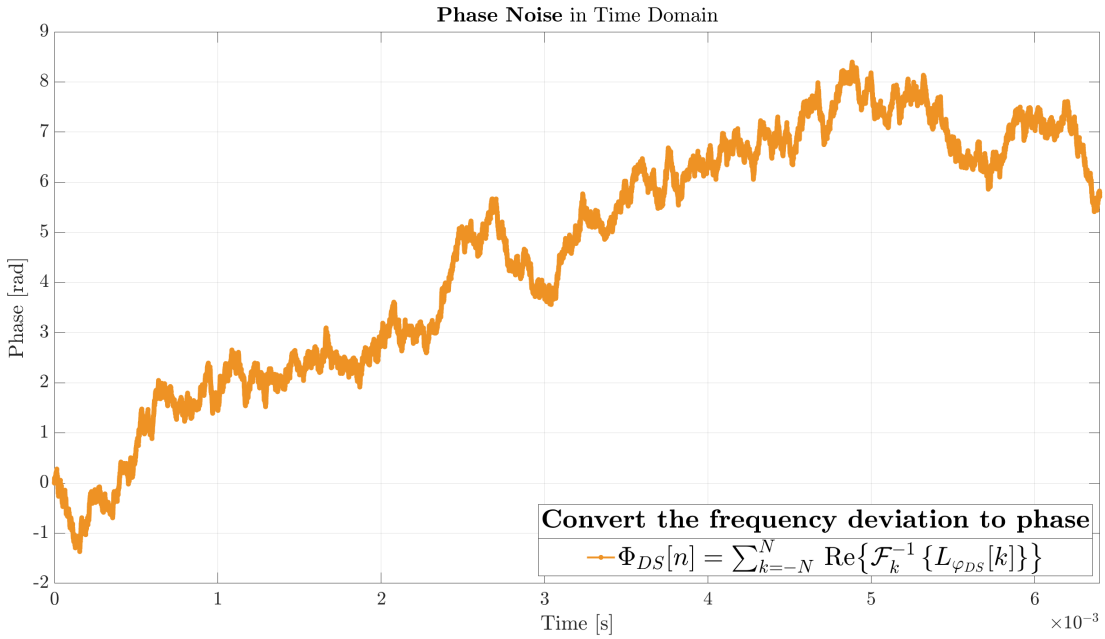


Figure 3.54: Conversion in Time Domain.

One fundamental assumption that underlies Equation 3.87 is the symmetry of the phase noise $L_{\varphi_{\mathrm{DS}}}(f)$ around zero frequency, as depicted in Figure 3.46. If this assumption is not met, the formula might need to be adjusted. The

symmetry assumption ensures that the inverse FFT operation results in a real-valued signal in the time domain.

Furthermore, it is important to note that the frequency deviation, $\Phi_{DS}[n]$, is **randomly generated** each time the code runs, due to the Complex Gaussian $\mathcal{N}_{\mathcal{C}}$ that generates the phase noise. However, the noise **profile** remains **consistent** despite this randomness, as depicted in Figure 3.44.

The flowchart in Figure 3.55 illustrates the MATLAB code for converting frequency deviation to the time domain, ensuring the result is real, and converting it to phase noise by integrating over time. See Script 3.7.

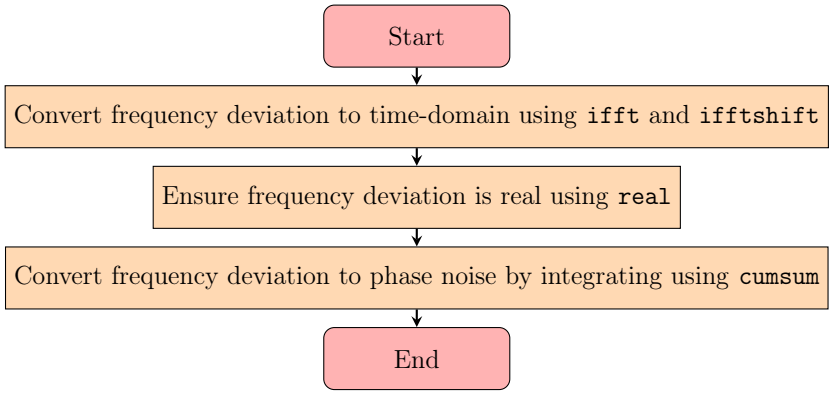


Figure 3.55: Flowchart of the Matlab code for converting frequency deviation to phase noise in the time domain.

Script 3.7: Converting Frequency Deviation to Time Domain.

```
%% CONVERT FREQUENCY DEVIATION INTO PHASE NOISE in discrete-time domain

% Convert the frequency deviation to the time domain using the inverse FFT
freq_deviation_time_N = ifft(ifftshift(phi_noise_freq), N);

% Ensure the frequency deviation is real (the imaginary part should be very small due to numerical precision issues)
freq_deviation_time_real = real(freq_deviation_time_N);

% Convert the frequency deviation to phase by integrating over time
phi_noise_time = 2 * pi * cumsum(freq_deviation_time_real);
```

In the mathematical representation of the conversion process, the term $\Phi_{\mathrm{DS}}[n]$ denotes the *phase noise* in the *time domain*, which is essentially the integral of the frequency deviation. However, the equation does not explicitly specify the method of integration. In the MATLAB script 3.7 of the same process,

the cumulative sum function `cumsum` converts the frequency deviation to phase by integrating over time. The factor of 2π is introduced when dealing with **digital signals**, as the frequency here is expressed in terms of radian frequency (radians/sample), not in Hz. In *digital signal processing*, frequency is often normalized by the sampling rate, yielding the *digital frequency*. The 2π term is instrumental in converting this digital frequency into phase by considering the 2π periodicity. Hence, the mathematical **equation** and the MATLAB **code** *accurately represent the conversion process*, albeit **differently**. Equation 3.87 offers a more theoretical, continuous-time perspective, while the MATLAB code provides a practical implementation that considers the **discrete nature** of digital signals.

3.4.2 Verifying the Frequency Deviation

Once the phase noise sequence is derived in the **time domain**, it is utilized to modulate the phase of the original signal. In this instance, a generic sinusoidal signal $x[n]$ is employed, characterized by amplitude A, angular frequency ω_{sig}, and phase offset ϕ_{offset}. The original signal is decomposed into amplitude and phase components to incorporate the generated frequency deviation. The phase noise is then added to the original phase, and the signal is reconstructed with the original amplitude and the modulated phase:

$$\begin{cases} \text{Signal:} & x[n] = A\sin\left(\omega_{sig}n + \phi_{\text{offset}}\right) \\ \text{Signal with noise:} & x_{\Phi_{DS}}[n] = A\sin\left(\omega_{sig}n + \phi_{\text{offset}} + \Phi_{DS}[n]\right) \end{cases} \tag{3.88}$$

Before proceeding further, the time-domain representation of the phase noise sequence $\Phi_{\text{DS}}[n]$ is plotted, with phase values constrained between $-\pi$ and π. Figure 3.56 allows a visual representation of how the frequency variation changes **randomly**. As with Figure 3.54 and Figure 3.56 changes at each turn of the code.

The phase noise sequence is then superimposed onto the original signal to generate a noise-affected signal $x_{\Phi_{DS}}[n]$. The time-domain plots of both the original and noise-affected signals are presented below; see Figure 3.57.

The noisy signal comprises the original signal with its phase **modulated** by the *phase noise* generated in the frequency domain and then converted into a discrete time domain. Although this illustration demonstrates how phase noise can alter a signal in the time domain, it is important to underscore that phase noise *effects* are **challenging to discern visually** in the time domain, see Figure 3.57. Therefore, a **spectral analysis** of the two signals is necessitated to validate the accuracy of the noise generation process; see Figures 3.58a and 3.58b below.

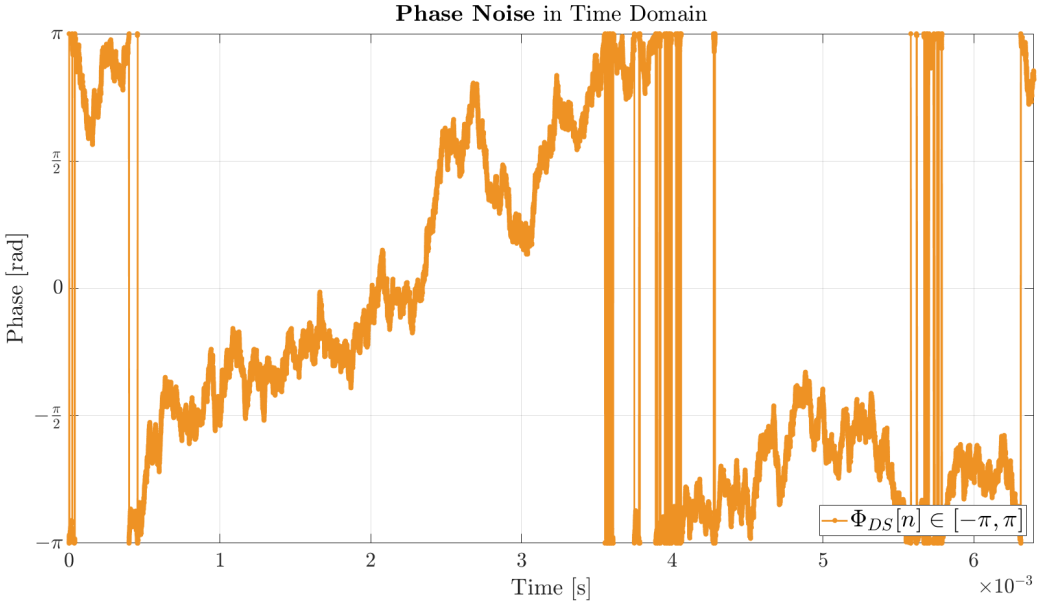


Figure 3.56: Conversion in Time Domain within the range $[-\pi, \pi]$.

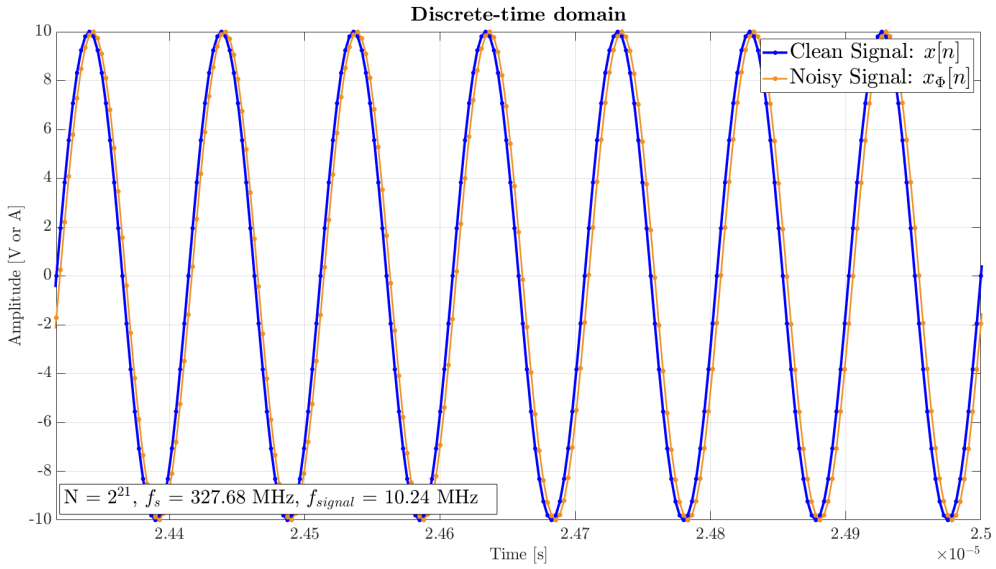
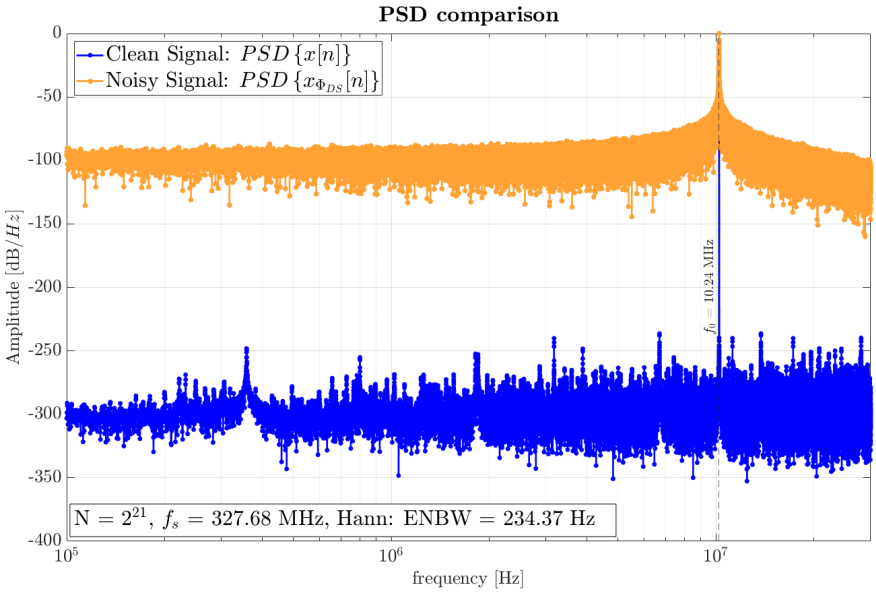


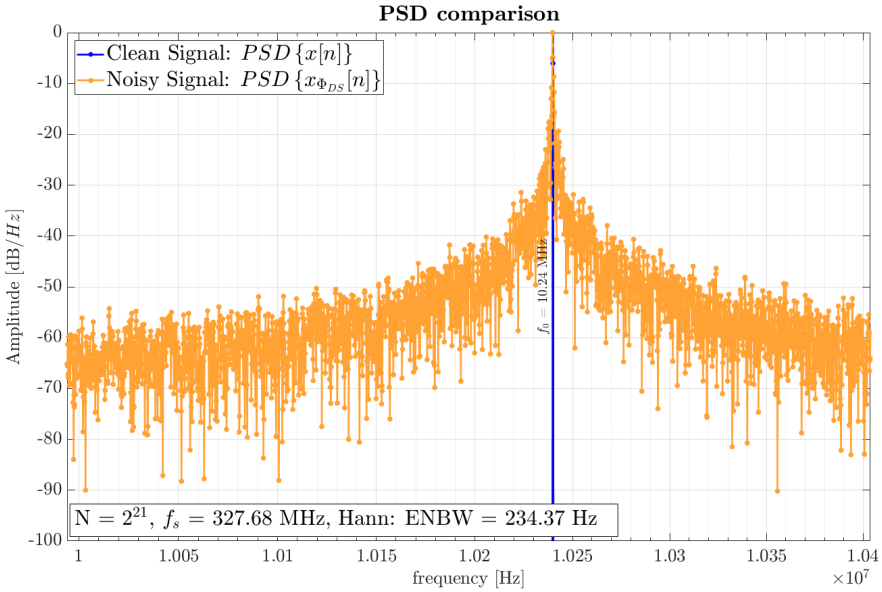
Figure 3.57: Original and Noisy Signal in Time Domain.

Spectral Analysis and Verification

To validate the accuracy of the *phase noise generation process* and its impact on the signal, spectral analysis is performed on both the original signal $x[n]$



(a) Full spectrum



(b) Magnification around the signal carrier frequency

Figure 3.58: PSD of signal $x[n]$ and $x_{\Phi_{DS}}[n]$.

and the signal with phase noise $x_{\Phi_{DS}}[n]$. As seen in Figures 3.58a and 3.58b, the Power Spectral Density (PSD) of the signals is plotted. Figure 3.58a

presents the full spectrum of both signals. Significant aspects to be noted from the figure are:

- The spectrum of the non-noised signal $x[n]$ appears to have a noise floor between -300 dB/Hz and -250 dB/Hz. However, this is not actual noise deriving from the movement but instead a manifestation of the inherent noise of the FFT algorithm.

- The spectrum of the signal with added phase noise, $x_{\Phi_{DS}}[n]$, shows a roll-off at high frequencies. This phenomenon occurs because the generated phase noise profile (refer to Figure 3.45) has a **limited** frequency array. The conversion into the time domain cannot account for all possible frequencies. This frequency span was intentionally reduced to simplify the process and expedite the code execution. While it results in a limited spectrum, it *does not* compromise the correctness of the procedure shown.

Figure 3.58b provides a magnified view around the signal's carrier frequency, further illustrating these points. The spectral analysis demonstrates that the spectrum of $x_{\Phi_{DS}}[n]$ closely aligns with the *phase noise profile* that was initially defined. This correlation confirms the successful application of the generated phase noise. These results verify the *correctness of the phase noise generation* and its *application to the signal* but also highlight the effects of phase noise in the frequency domain. It is essential to note that while phase noise effects are challenging to discern visually in the time domain (as shown in Figure 3.57), their impact becomes **evident** when examining the signal in the frequency domain. This insight underlines the significance of spectral analysis in understanding and diagnosing the effects of phase noise on signals. The flowchart in Figure 3.59 illustrates the MATLAB code for modulating a signal with phase noise, which involves decomposing the original signal, adding phase noise, and generating the noisy signal. See Script 3.8.

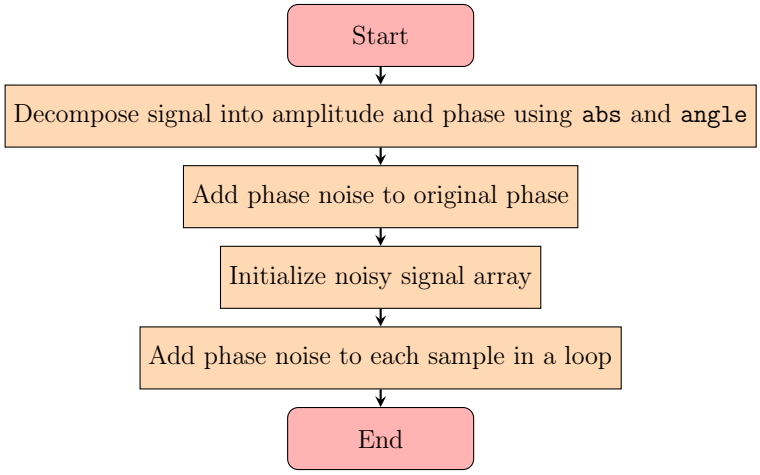


Figure 3.59: Flowchart of the Matlab code for modulating a signal with phase noise.

Script 3.8: Modulating Signal with Phase Noise.

```
% Decompose the original signal into amplitude and phase
amp_signal = abs(signal);
phi_signal = angle(signal);

% Add the phase noise to the original phase
phi_noisy = phi_signal + phi_noise_time;

% Initialize the noisy signal
noisy_signal = zeros(N, 1);

% Add phase noise to each sample
for i = 1:length(t)
    % Add phase noise to the phase of the current sample
    noisy_phase = 2 * pi * f0 * t(i) + phi_offset + wrapToPi(
        phi_noise_time(i));
    % Generate the noisy sample
    noisy_signal(i) = A * sin(noisy_phase);
end
```

One of the key elements in the MATLAB code 3.8 is the function `wrapToPi`, which is used to wrap the phase to the range $[-\pi, \pi]$. The role of this function is pivotal in maintaining the cyclical nature of the phase of a sinusoid, which has a period of 2π radians. The addition of phase noise can potentially cause phase *variations* that **exceed** π, resulting in signal **discontinuities** due to the sinusoidal function's inherent periodicity. Such discontinuities are not physically meaningful and often lead to spurious *artefacts* in the signal, distorting the accurate representation of the phase-modulated signal. The `wrapToPi` function **prevents** this issue by wrapping the phase back into the $[-\pi, \pi]$ interval whenever it exceeds these limits. This ensures a smooth and continuous phase transition, a *crucial* aspect of a phase-modulated signal.

The appropriate use of this function exemplifies an essential aspect of signal processing – the necessity to consider and **preserve** the *physical* and *mathematical* characteristics of the signal during the processing operations.

Measuring the Phase Noise

To provide a more comprehensive validation of the *phase noise model*, the phase noise profile of the signal with added phase noise, denoted $x_{\Phi_{DS}}[n]$, was quantitatively assessed. The function `phaseNoiseMeasure` from the MATLAB Communications Toolbox was employed for this task. This function is specifically designed to measure the phase noise profile in the time domain, aligning with the overall objective of this study. It takes several parameters, including the time vector, the noisy signal (considering only the real part), the Effective Noise Bandwidth (ENBW) rounded to two decimal places, the breakpoint frequencies, and the phase noise levels used in the phase noise profile. The measured phase noise profile is stored in the variable `PNMeasure` as the output of this function.

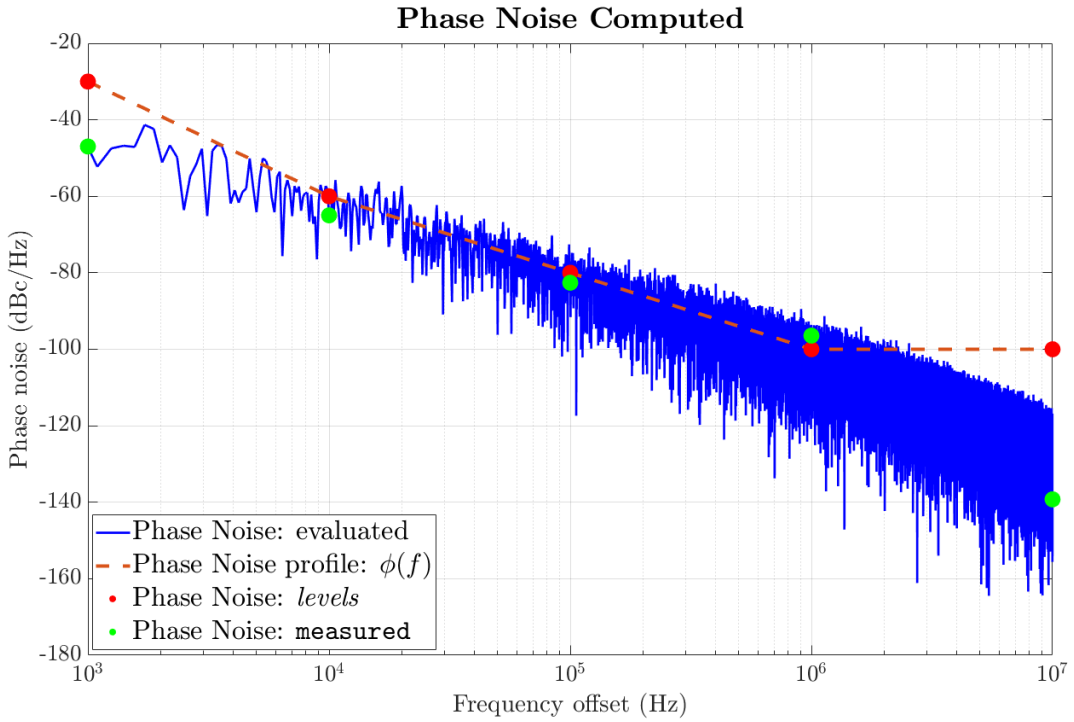


Figure 3.60: Phase Noise measured

As shown in Figure 3.60, the phase noise profile was evaluated and measured.

The red circles represent the *levels* of the phase noise profile that was initially defined, while the green circles indicate the measured phase noise profile. The phase noise profile, as initially defined, is represented by the curve labelled '*Phase Noise profile:* $\phi(f)$', and the evaluated phase noise is depicted by the curve labelled '*Phase Noise: evaluated*'. There are, however, two areas where the measured phase noise does not align with the initial phase noise levels:

- The first discrepancy is observed at the first dBc/Hz point. This is because the `phaseNoiseMeasure` calculates the FFT of a discrete signal. Since $x_{\Phi_{DS}}[n]$ does not have infinite points, the FFT algorithm has a few bins at low frequency from the carrier.

- The second discrepancy is found at the last dBc/Hz point, which **does not follow** the characteristic of the phase noise profile $\phi(f) = -100\frac{\text{dBc}}{\text{Hz}}$ for $f > 10^7$. The FFT bin defines the FFT resolution using the last breaking point frequency, 10MHz. This might have caused the wrong estimation of the phase noise.

Despite these discrepancies, the measured phase noise closely aligns with the initial phase noise levels, thus successfully demonstrating the *accuracy* of the phase noise model.

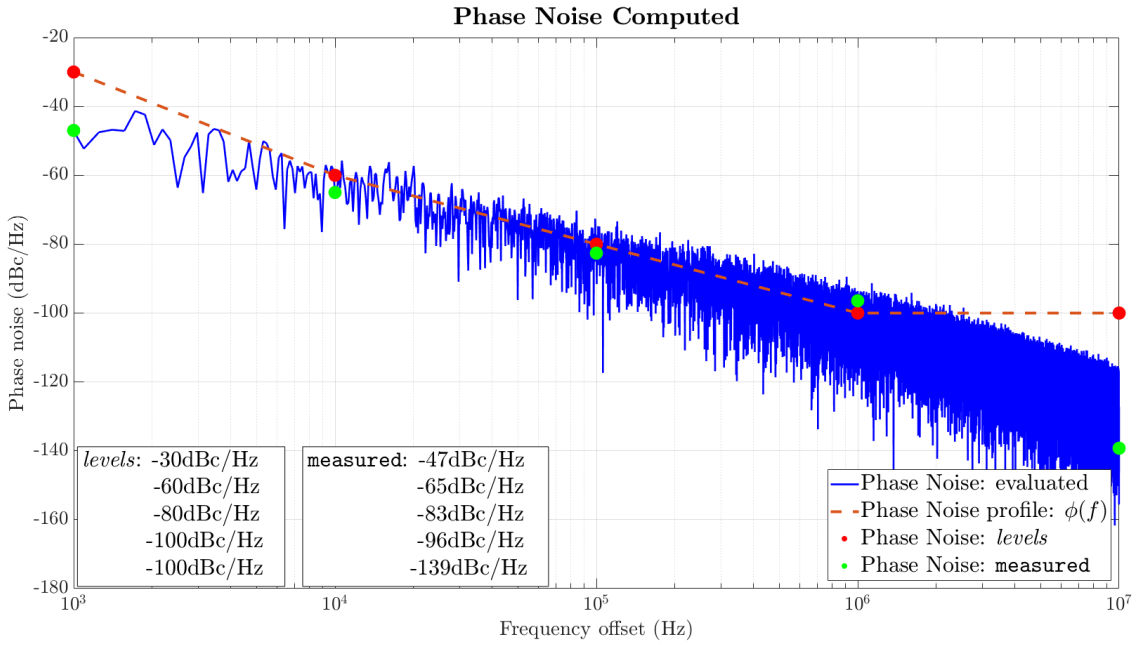


Figure 3.61: Phase Noise measured with levels.

Figure 3.61 shows the text boxes in the figure that provide the phase noise levels that were initially defined (in red) and the phase noise levels as measured (in green), reinforcing the visual comparison of the initial and measured phase noise levels. This figure highlights the effectiveness of the phase noise measurement method in accurately quantifying the phase noise profile in the time domain.

The MATLAB code for this process is presented below:

Script 3.9: Measuring Phase Noise Profile.

```
% Compute phase noise measurement
PNMeasure = phaseNoiseMeasure(t, real(noisy_signal), round(ENBW,
    2), f_break, 'on', 'Phase noise', phi_noise_level, 'Type',
    'Time');
```

The function `phaseNoiseMeasure` is part of the MATLAB Communications Toolbox, specifically designed to measure the phase noise profile in the time domain. Further details about this function can be found in the official MATLAB documentation [23]. Several parameters are passed to this function:

- `t`: the time vector.
- `real(noisy_signal)`: the real part of the noisy signal.
- `round(ENBW, 2)`: the Effective Noise Bandwidth (ENBW) rounded to two decimal places.
- `f_break`: the breakpoint frequencies used in the phase noise profile.
- `on`: an option to turn on some function feature.
- `Phase noise`: a label indicating that the measurement is for phase noise.
- `phi_noise_level`: the phase noise levels used in the phase noise profile.
- `Type`: an option indicating the type of measurement being performed.
- `Time`: specifying that the measurement is being performed in the time domain.

3.5 Summary of Chapter III

This chapter delineates a framework for simulating the phase noise characteristics of a **Voltage-Controlled Oscillator (VCO)** in the clock source

for the CT$\Delta\Sigma$M. Positioned before the detailed design discussions, it provides the mathematical foundation to understand the influence of the clock source's phase noise spectrum on the CT$\Delta\Sigma$M's in-band noise.

Key points discussed include:

- The use of an **Infinite Impulse Response (IIR) filter** to approximate the *phase noise profile*, utilising only the feedforward path to describe the frequency breakpoints of the phase noise profile.
- Development of a MATLAB script to simulate and visualise the phase noise profile, defining the transfer function in the *s-domain* and converting it to the $\mathcal{Z}$-domain using forward differencing.
- Analysis of the limitations of the IIR filter approach in accurately replicating the phase noise profile and the transition to a more advanced model.
- Exploration of the impact of phase noise on the SNR of the CT$\Delta\Sigma$M, with a MATLAB-based investigation injecting phase noise into a sinusoidal waveform and analysing its effect.
- Introduction of a phase noise profile model in MATLAB, defining breakpoints in the frequency range and the corresponding phase noise levels to generate the desired phase noise profile.
- Computation of in-band noise (IBN) by considering the phase noise spectrum and the ADC's noise transfer function (NTF), demonstrating how phase noise shifted by the ADC input frequency impacts the in-band noise.

The chapter concludes by verifying the developed phase noise model using a discrete-time signal, confirming the accuracy of the phase noise generation process through time-domain and spectral analyses.

The subsequent chapter will introduce the **FLL as the Clock Source for CT$\Delta\Sigma$M**, discussing the choice of a Frequency Locked Loop (FLL) as the clock source and its implications on the CT$\Delta\Sigma$M's performance.

References

[16] F. X. Kärtner, "Analysis of white and f-α noise in oscillators," *International Journal of Circuit Theory and Applications*, vol. 29, no. 2, pp. 131–148, 2001 (Cited on pages 63, 69).

[17] Wikipedia, *Infinite impulse response — Wikipedia, the free encyclopedia*, [Online; accessed 26-November-2023], 2023. [Online]. Available: https://en.wikipedia.org/wiki/Infinite_impulse_response (Cited on pages 63, 64).

[18] A. Demir *et al.*, "Phase noise in oscillators: A unifying theory and numerical methods for characterization," *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*, vol. 47, no. 5, pp. 655–674, 2000 (Cited on page 69).

[19] A. Hajimiri and T. H. Lee, "A general theory of phase noise in electrical oscillators," *IEEE Journal of solid-state circuits*, vol. 33, no. 2, pp. 179–194, 1998 (Cited on page 69).

[20] C. De Berti, P. Malcovati, L. Crespi, and A. Baschirotto, "Colored clock jitter model in audio continuous-time $\Sigma\Delta$ modulators," pp. 1–4, 2015, Accessed: 2023-09-24. DOI: 10.1109/NEWCAS.2015.7182021. [Online]. Available: https://ieeexplore.ieee.org/document/7182021 (Cited on pages 69, 98, 124, 129, 150).

[21] M. Aqamolaei and M. M. Ahmadi, "Modelling an oscillator phase noise whose spectral density contains both 1/f 2 and 1/f 3 regions," *Electronics Letters*, vol. 58, 2021. DOI: 10.1049/ell2.12393 (Cited on page 78).

[23] MathWorks. "Phase noise measure." Online; accessed 24 July 2023, MathWorks. (2023), [Online]. Available: https://www.mathworks.com/help/comm/ref/phasenoisemeasure.html (visited on 07/24/2023) (Cited on page 94).

Chapter IV:
FLL as the Clock Source for CTΔΣM

In ultra-low-noise applications such as magnetic sensing for minimally invasive procedures, the precise tracking of surgical instruments makes the clock source for the CTΔΣ modulator is critically important. The specification of CTΔΣM model, which has been defined in Chapter II, delineates the specifications for the clock source design, as presented in Table 4.5[6]:

Parameters	**Topology A**	**Topology B**
OGB	1.495	1.495
OSR	256	256
f_s	10.24MHz	10.24MHz
Signal Bandwidth	20kHz	20kHz
SNR	$>$ 70dB	$>$ 70dB
$\sigma_{\text{Periodic Jitter}}$	$\leq$ 100 ps rms	$\leq$ 100 ps rms

Table 4.5: CTΔΣM Model Specifications for Topologies A and B

Therefore, the key constraints central to the clock design are delineated in Table 4.6:

Parameters	**Value**
$\boldsymbol{\sigma}_{\text{Periodic Jitter}}$	$\leq$ **100** ps rms
$\boldsymbol{f_{out}(t)}$ (Output Frequency)	$\approx$ **10.24**MHz

Table 4.6: Key Design Constraints for Clock Source

The CTΔΣM design necessitates the clock source's output to approximate the ideal value of 10.24MHz. The ADC's Simulink model portrays the jitter constraint, referring solely to the white spectrum, as shown in Figure 2.7 and explained in equation 2.28. This contrasts the discoveries for the in-band spectrum, which is profoundly influenced by phase noise, as evident in

[6]The *out-of-band gain* is expressed in magnitude and not dB [9].

equation 3.81. This understanding emerged *post-clock design* phase and *chip testing*. Such constraints, already explored in literature, particularly within audio frequency bandwidths [20] and more comprehensive CTΔΣM models [22], [9], gained renewed significance. While it was often assumed that the clock source's typical phase noise spectrum would have minimal impact on the CTΔΣM ADC, the primary design goal for the clock source was to optimize **area** and **power consumption**. Consequently, a *resistor capacitor-based clock source* was chosen for the Frequency-Locked Loop (FLL) structure. However, post-chip measurement analyses disproved this assumption, underscoring the significant **influence** of the clock source's phase noise spectrum on ADC performance [20], [22].

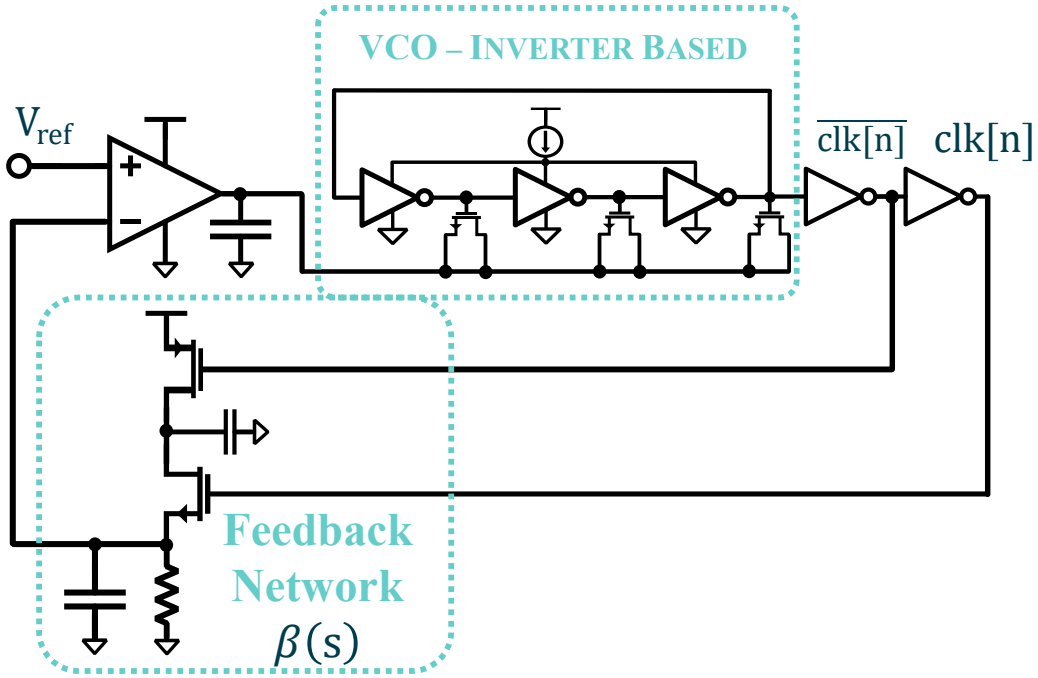


Figure 4.62: FLL Schematic

Before those findings, the Anser EMT Team referenced the model presented in [24]. Although the design mirrors the published design, it introduced no novel elements. Given the Anser chip's **in-vivo** placement, temperature compensation was deemed unnecessary, as the human body's temperature rarely exceeds 40°C. In light of these design specifications, the schematic, depicted in Figure 4.62, incorporates key components:

- **VCO (Voltage-Controlled Oscillator)**: A ring oscillator comprising an odd number of three inverters for oscillation, driven by a control

voltage reference $V_c(t)$.

- **Feedback Mechanism**: Converts the VCO's squared waveform output into the voltage reference, denoted as $V_{\text{feedback}}(t)$.

- **Voltage reference**: An operational amplifier with the positive input connected to the reference voltage $V_{\text{ref}}(t)$. By comparing this with $V_{\text{feedback}}(t)$ at the negative input, the VCO's control voltage is determined.

This closed-loop system ensures the VCO's control's voltage $V_c(t)$ remains stable, stabilising the output frequency and countering potential deviations. It is worth noting that the entire FLL circuit is designed and simulated using *Cadence* as the simulation tool. The design leveraged the **TSMC** library accessed through MCCI, utilizing a **65nm** architecture. This work was facilitated by the **Europractice license**, which provided essential resources and tools for the development process, ensuring compliance with industry standards and facilitating access to advanced technological capabilities.

4.1 Frequency Reference choise

Digital systems demand precision, and frequency references are integral in achieving this. Their selection, however, is influenced by several parameters, including period accuracy, stability, power consumption, and footprint. Historically, **crystal oscillators** have been the preferred choice due to their consistent and remarkable clock precision, irrespective of their implementation overheads. However, even the smallest commercially available crystal oscillator exceeds the size constraints for our application [25]. Conversely, while **LC tanks** are optimised for GHz operations, migrating them to MHz applications is challenging, primarily due to the requisite large LC components, leading to an expanded chip area. For precision and reliability, **Phase Locked Loops (PLLs)**, see Figure 4.63, are often deployed [26]. Essential components within a PLL encompass:

- **Phase Comparator**: This component is responsible for phase alignment between the input signal clock (ϕ_{ref}) and the feedback signal (ϕ_{fb}), ensuring the stability of the output frequency.

- **Low-pass Filter**: It serves to convert the phase comparator's output into the control voltage (V_C) for the VCO.

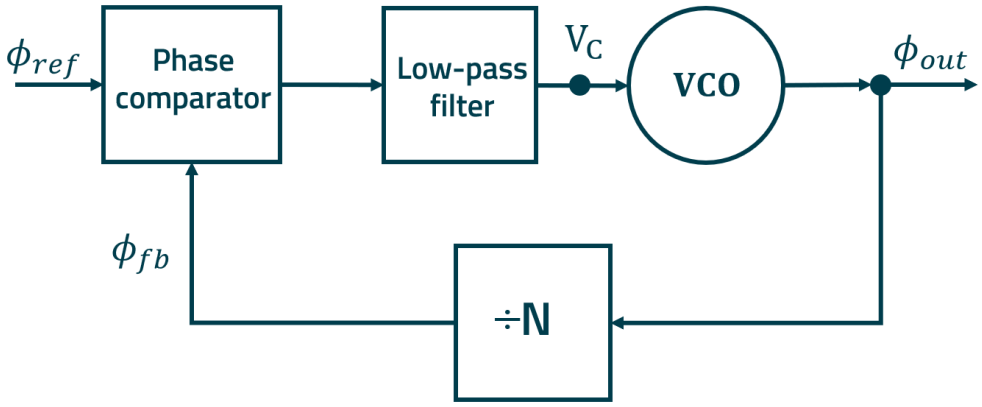


Figure 4.63: Fundamental configuration of a PLL.

- **Voltage-Controlled Oscillator (VCO)**: Designed to generate an output clock proportional to the input signal frequency, its frequency of oscillation, modulated by the control voltage (V_C), is routed to both the system output and feedback loop.

- **Frequency Divider**: Essential for scaling down the output frequency, it assists in phase alignment by the phase comparator.

The need for a clock source that does not rely on another clock for its reference prompts the introduction of the *FLL*. While the structure of the FLL is reminiscent of the PLL, distinct variations exist between them. Both architectures endeavour to stabilise the output frequency, albeit with differing components [27], as illustrated in Figure 4.64.

To gain a comprehensive understanding of the system, it is crucial to consider the signal transfer functions at various loop stages. Based on the block diagram presented in Figure 4.64, the ensuing equations elucidate these transfer functions:

$$\begin{cases} T(s) \triangleq H(s) K_{\text{VCO}} \beta(s) \\ \beta(s) = \dfrac{V_{\text{feedback}}(s)}{f_{out}} \end{cases} \tag{4.89}$$

Where $T(s)$ represents the *open-loop gain*, $\beta(s)$ is the transfer function of the feedback mechanism, and K_{VCO} signifies the VCO's static gain. It is imperative to highlight that these equations are premised on ideal conditions.

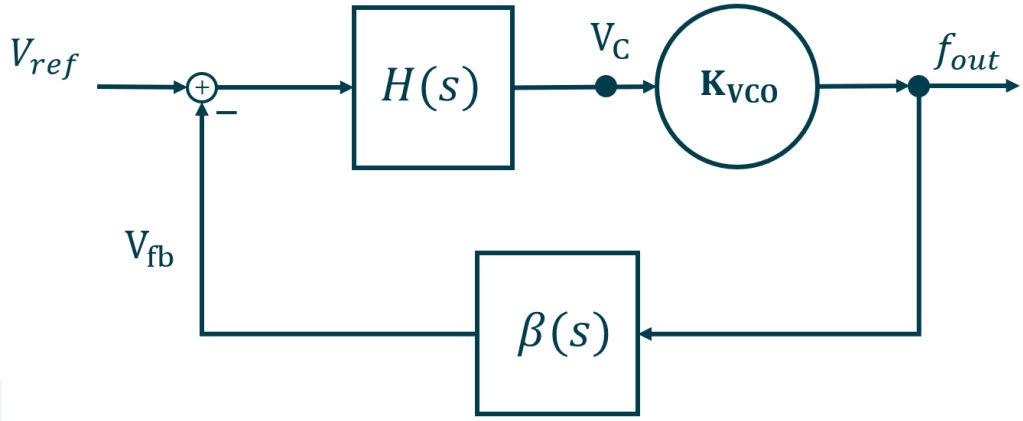


Figure 4.64: Architectural overview of an FLL [26].

Practical deployments may necessitate refinements, accounting for aspects like noise, variations in power supply, and other external determinants.

4.2 VCO Design and Analysis

The design of the Voltage-Controlled Oscillator is the first step for the clock source, especially when precision and stability are paramount. The essential role of the VCO is to yield an output frequency that can be modulated based on an input voltage. Table 4.7 contains the notation used in the description of the ring oscillator:
All average values in the table are computed in Cadence Virtuoso using the equation:

$$\bar{x} = \frac{1}{\texttt{simT} - \texttt{sim0}} \int_{\texttt{sim0}}^{\texttt{simT}} x(t)\, dt \tag{4.90}$$

where $x(t)$ represents a generic signal in the transient simulation, and `simT` and `sim0` are the final and initial time points, respectively, of the transient simulation in Cadence Virtuoso.

Following the methodologies outlined in [26], the VCO design was approached systematically. To reduce the output frequency, several modifications were made:

- The initial design comprised three stages, oscillating at a minimum, with dimensions $\left(\frac{W}{L}\right)_{P-N} = \frac{200\text{ nm}}{60\text{ nm}}$. This setup resulted in an output frequency $f_{out} = 17.7$ GHz, the maximum frequency attainable.

Symbol	Description
W	Width of the MOS channel
L	Length of the MOS channel
δ	Duty cycle of the ring oscillator
$\sigma_{\text{Periodic Jitter}}$	Standard deviation of the periodic jitter noise
f_{out}	Average output frequency
$\hat{P}$	Average power consumption
N	Number of stages
V_{DD}	Supply voltage of $1.2V$

Table 4.7: Notation utilised in the VCO design discussion.

- To achieve a near 50% duty cycle, PMOS compensation required larger widths: $\left(\frac{W}{L}\right)_P = \frac{400 \text{ nm}}{60 \text{ nm}}$ and $\left(\frac{W}{L}\right)_N = \frac{200 \text{ nm}}{60 \text{ nm}}$, resulting in a duty cycle $\delta = 49.97\%$.

Reduced Output Frequency

- The number of stages was expanded from 3 to 5, rendering $f_{out} = 10$ GHz.

- The MOS's length was quadrupled to 240 nm, further reducing the output frequency to $f_{out} = 2$ GHz with a periodic jitter $\sigma_p = 200$ fs.

Adjustment of Load Capacitor

- Incorporation and subsequent adjustment of a load capacitor C_L led to $f_{out} = 10.4$ MHz and $\sigma_p = 14$ ps. This was based on the relation:

$$\hat{P} = N f_{out} C_{i,\text{tot}} V_{DD}^2 \begin{cases} \hat{P}_{\text{without } C_L} \approx \hat{P}_{\text{with } C_L} \\ \dfrac{2 \text{ GHz}}{11 \text{ MHz}} \cdot \dfrac{\hat{P}}{N f_{out} V_{DD}^2} = 446 \text{ fF} \end{cases} \tag{4.91}$$

- Refinement of C_L to 390 fF yielded $f_{out} = 11.1$ MHz and $\sigma_p = 13$ ps.

- Further manipulation was required to minimise the jitter's standard deviation by increasing the MOS multiplier four times. Consequently, the load capacitance $\mathbf{C_L = 1.56pF}$ and the multiplier provided $f_{out} =$

10.7 MHz, σ_p = 8.6 ps, a duty cycle δ = 51%, and average power consumption of 127.8 μW.

- By employing a NAND gate as the primary inverter, with specifications W_P = 200 nm and W_N = 400 nm, the output frequency was maintained at $\mathbf{f_{out} = 10.7\ MHz}$ with a periodic jitter of $\boldsymbol{\sigma_p = 9\ ps}$, a duty cycle $\boldsymbol{\delta = 51}$% while the power consumption remained unchanged at $\mathbf{127.8\mu W}$.

Transistors with *thick gate oxide* were incorporated into the design, as recommended in the literature [24]. Typically, such transistors are chosen for their ability to operate at higher supply voltages without compromising reliability, resulting in increased drive current and improved performance. In this context, including thick gate oxide helped mitigate the short-channel effects in modern CMOS processes, ensuring optimal VCO operation [24].

The *initial* design incorporated five stages, targeting the reduction of periodic jitter noise. Increasing the number of stages inherently mitigated the flicker noise from transistors, a dominant source of phase noise in the spectrum [26], [27]. The design, as visualized in Cadence, is presented in Figure 4.65. The first inverter was replaced with a NAND gate to facilitate the integration of a control signal, `'V_EnableClk'`. This signal activates the oscillation when set to V_{DD} (logical 1), disrupting the loop when set to zero. In this state, the NAND gate functions as a buffer, terminating the oscillation in the ring. A subsequent inverter, positioned after the ring, refines the oscillator's output. This inverter, identical in size to its counterparts, ensures uniform timing. An additional inverter, sourced from the TSMC 65nm library, provides feedback for both the signal $clk_{out}(t)$ and its inverse, $\overline{clk_{out}}(t)$. Figure 4.66 showcases the distinction between the output of the ring oscillator (depicted in red) and the refined output (illustrated in green), labelled as "Inv. output".

Figure 4.67 depicts the implementation of a *varactor* from the TSMC 65nm library, specifically the `nmoscap`, which functions as a capacitive load and facilitates the voltage control of the ring oscillator. While employing an NMOS as a capacitive load is feasible, it mandates coupling the control voltage, V_C, to both the source and drain, thereby utilizing the transistor's entire capacitance.

The total gate capacitance, when the source and drain are connected to the same potential, can be decomposed into three primary components: the intrinsic capacitance of the gate over the channel, C_{channel}, and the overlap capacitances due to the gate overlapping the source, C_{GS}, and the drain,

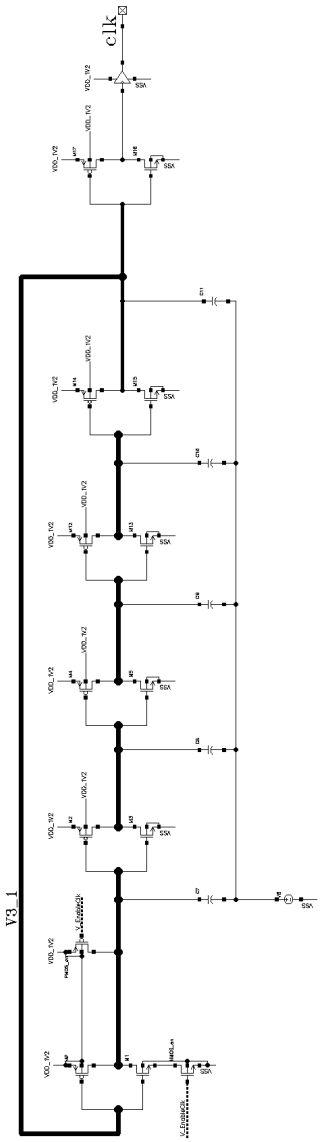


Figure 4.65: Schematic Representation of a Five-Stage VCO Designed in Cadence.

C_{GD}. Representing this relationship mathematically by Equation (4.92).

$$C_{\text{total}} = C_{\text{channel}} + 2C_{\text{overlap}} \tag{4.92}$$

where $C_{\text{overlap}} = C_{\text{GS}} = C_{\text{GD}}$. However, using the NMOS for capacitance in this manner proves less efficient than the designed varactor. As shown in Figure 4.68a, the average output frequency does not increase linearly with the control voltage. In contrast, as portrayed in Figure 4.68b, the varactor man-

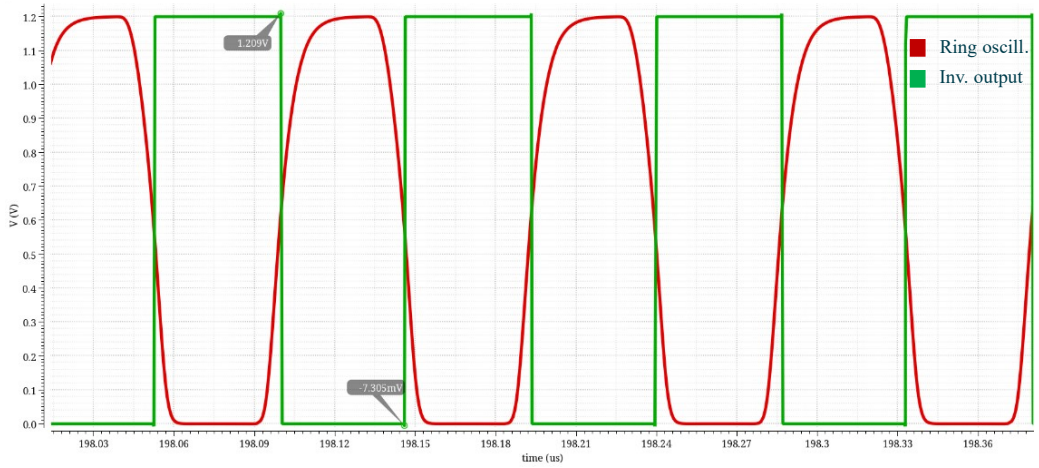


Figure 4.66: VCO transient simulation.

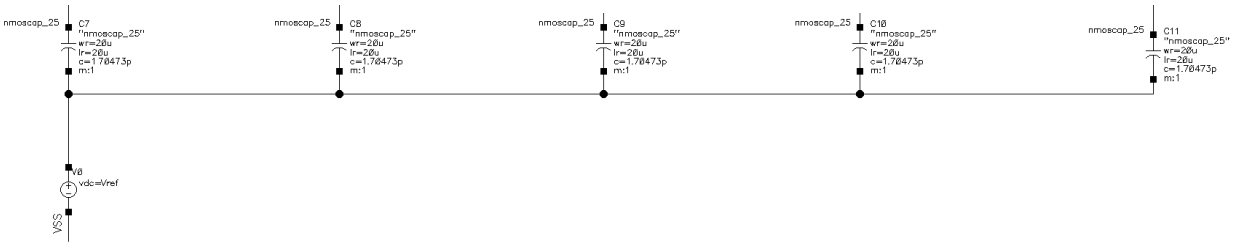


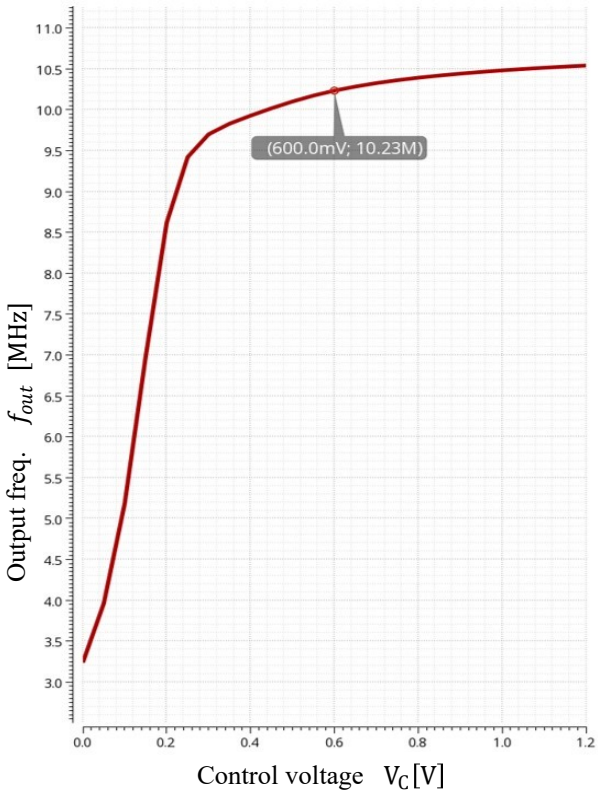
Figure 4.67: Varactor used in the design.

ifests a more *linear response* to voltage changes. It is worth noting that the central frequency in this representation is not set at 600mV but is intended to underscore the linearity comparison between the NMOS and varactor.

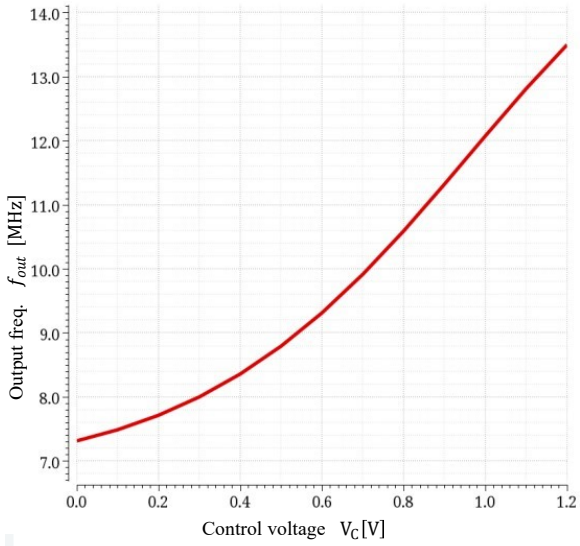
Building upon the detailed analysis and design considerations presented above, a crucial parameter emerges for the VCO's performance: the K_{VCO}. In the block model described in Figure 4.64, the $\mathbf{K_{VCO}}$ is quantified as **5.23 MHz/V**. This parameter, often termed the VCO's sensitivity, measures how the output frequency varies with changes in the control voltage. Mathematically, K_{VCO} can be expressed as:

$$K_{\text{VCO}} = \frac{\Delta f_{\text{out}}}{\Delta V_{\text{C}}} \quad \text{or} \quad K_{\text{VCO}} \triangleq \frac{df_{out}(V_C)}{dV_C} \tag{4.93}$$

where Δf_{out} represents the change in output frequency and ΔV_{C} denotes the change in control voltage. Understanding this sensitivity is paramount, as it offers insights into the VCO's responsiveness to voltage variations. This concept is further elaborated upon in the works of [27]–[29].



(a) NMOS as load.



(b) Varactor as load.

Figure 4.68: Output frequency responses of VCOs with different loads.

4.3 Feedback Mechanism

Feedback is a fundamental component in the FLL, ensuring the output remains synchronised with the desired output frequency[7].

The feedback mechanism, as depicted in Figure 4.69, plays a crucial role in maintaining the stability and performance of the system. The architecture is essentially a voltage divider, where the equivalent resistance connected to V_{DD} is formed by a switched capacitor. The two switches, PMOS and NMOS, are connected to the VCO's output. This configuration ensures that the feedback voltage V_{fb} is a fraction of the supply voltage V_{DD}, determined by the ratio of R_l to the sum of R_l and the equivalent resistance R_{eq} of the switch-capacitor circuit. The equivalent resistance R_{eq} is inversely proportional to the product of the switching capacitance C_s and the output frequency f_{out}. Therefore, the feedback voltage can be expressed as:

$$\begin{cases} V_{\text{fb}} = V_{DD} \cdot \dfrac{R_l}{R_l + R_{eq}} \\ R_{eq} = \dfrac{1}{C_s f_{out}} \end{cases} \tag{4.94}$$

Solving for V_{fb} :

$$V_{\text{fb}} = V_{DD} \cdot \frac{R_l C_s f_{out}}{R_l C_s f_{out} + 1}$$

Equation (4.94) highlights the dependency of the feedback voltage on the resistive load, switching capacitance, and the output frequency. The switched capacitor acts as a dynamic resistance, which varies with the output frequency, thus influencing the feedback voltage.

The primary model, as referenced in [24], provides an equation to compute the feedback voltage $V_{\text{fb}}(t)$. This equation captures the relationship between the feedback voltage and the output frequency, factoring in the resistive and capacitive loads and the number of clock cycles, N_{cycle}. Equation (4.95) is given by:

$$V_{\text{fb}}(t) = V_{DD} \frac{R_l C_l}{10 N_{\text{cycle}} \pi} f_{out} \quad \text{at} \quad \frac{f_{out}}{N_{\text{cycle}}} \text{Hz} \tag{4.95}$$

[7]Special thanks are extended to PhD candidate *Mikhail Gaidukov* for his invaluable assistance with the **equations** described in this section.

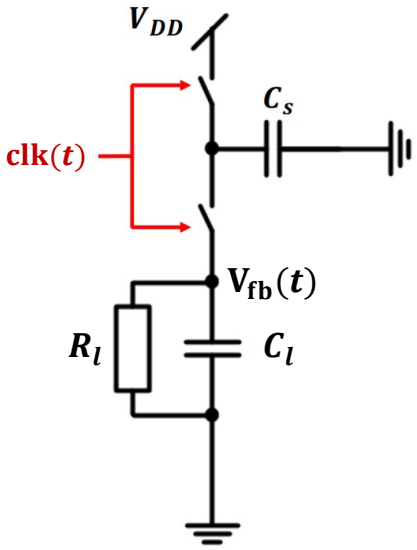


Figure 4.69: Feedback architecture.

It provides a simplified model for the feedback voltage. However, a more comprehensive model is introduced for a more accurate representation:

$$V_{\text{fb}}\left[\frac{n}{f_{out}}\right] = \frac{q_0}{C_l}\sum_{n=1}^{+\infty}\left(\frac{C_l}{C_l+C_s}\right)^n \exp\left\{-\frac{n}{f_{out}R_l(C_l+C_s)}\right\} \tag{4.96}$$

Solving the summation for $V_{\text{fb}}(t)$:

$$V_{\text{fb}}\left[\frac{n}{f_{out}}\right] = \frac{q_0}{C_l}\left[\frac{1}{1-\left(\frac{C_l}{C_l+C_s}\right)\exp\left\{-\frac{1}{f_{out}R_l(C_l+C_s)}\right\}} - 1\right]$$

And simplifying it:

$$V_{\text{fb}}\left[\frac{n}{f_{out}}\right] = -\frac{q_0}{C_l-(C_l+C_s)\exp\left\{\frac{1}{R_l f_{out}(C_l+C_s)}\right\}} \tag{4.97}$$

Equation (4.96) represents the feedback voltage as a function of time. The term $\frac{q_0}{C_l}$ signifies the initial charge divided by the capacitive load. The summation captures the cumulative effect of the feedback mechanism over multiple cycles. Each term in the summation represents the feedback voltage's

decay over time due to the resistive and capacitive loads. The factor $\left(\frac{C_l}{C_l+C_s}\right)^n$ indicates the fraction of the feedback voltage retained after each cycle, while the exponential term $e^{-\frac{n}{f_{out}R_l(C_l+C_s)}}$ models the **switching** of the feedback voltage over time. Since the voltage is switching, the peak-to-peak amplitude of the feedback voltage is given by:

$$V_{\text{fb, peak-to-peak}} = V_{\text{fb}} \left[\frac{n}{f_{out}}\right] \left(1 - e^{-\frac{1}{f_{out}R_l(C_l+C_s)}}\right) \tag{4.98}$$

Equation (4.98) quantifies the maximum variation in the feedback voltage over a given period. This peak-to-peak amplitude is especially significant in systems where the voltage undergoes periodic or oscillatory changes. The equation is derived from the time-dependent feedback voltage $V_{\text{fb}}(t)$ and incorporates an exponential decay factor. The term $e^{-\frac{1}{f_{out}R_l(C_l+C_s)}}$ represents the rate of decay of the feedback voltage, influenced by the output frequency f_{out}, the load resistance R_l, and the combined capacitive loads C_l and C_s. The factor $1-e^{-\frac{1}{f_{out}R_l(C_l+C_s)}}$ thus captures the proportion of the feedback voltage that remains stable around the **average value**. Therefore, $V_{\text{fb, peak-to-peak}}$ provides a measure of how much the feedback voltage fluctuates around its average value over time, offering insights into the stability and performance of the feedback mechanism in the system.

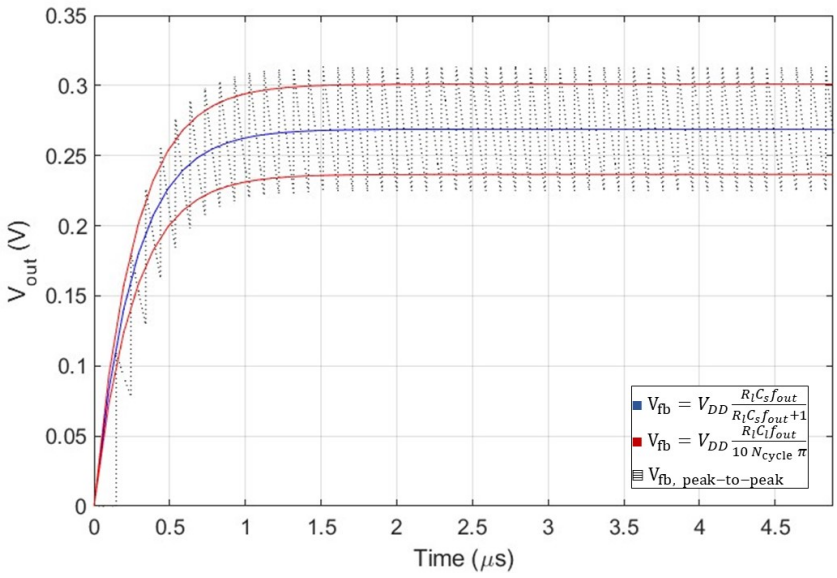


Figure 4.70: Feedback time response

In Figure 4.70, the y-axis is labeled as $V_{out}(t)$, referring to the output feedback voltage of Equation 4.97. The blue line represents the average value of the feedback voltage as given by Equation (4.94). The dashed line shows the actual feedback voltage $V_{\text{fb}}(t)$ when considering the switching capacitor acting as a resistance. The two red lines show the peak-to-peak estimation calculated by equation 4.98. The feedback time response visually represents how the feedback voltage varies over time.

4.3.1 Components and Configuration

The system's stability and jitter reduction capability depend on the feedback values [28], [29]. The chosen values, listed in Table 4.8, are based on the recommendations from the reference paper [24], given the comparable working frequency with the Anser EMT project.

Component	**Configuration/Value**
MOS $\left(\frac{W}{L}\right)_{\text{P-N}}$	$\frac{500nm}{300nm}$
MOS Fingers	2
C_s	$147fF \cdot 2 = 294fF$
C_l	$147fF \cdot 20 = 2.94pF$
R_l	$100k\Omega$

Table 4.8: Component values and configurations.

The component configuration directly influences the system's performance. A *base unit* of $147fF$ `crtmom` capacitor was chosen based on layout considerations[8]. The schematic in Figure 4.71 displays these values and their connections. The specifications for the switches, PMOS and NMOS are: $W = 500nm$, $L = 300nm$, fingers $= 2$, and multiplier $= 2$.

4.3.2 Determining the Ideal V_{fb}

The feedback voltage, denoted as V_{fb}, is integral to the operation of the FLL system. The frequency output, represented as f_{out}, is intrinsically linked to

[8] *Aleksandr Sidun*, the layout design engineer of the team, provided valuable input in this aspect.

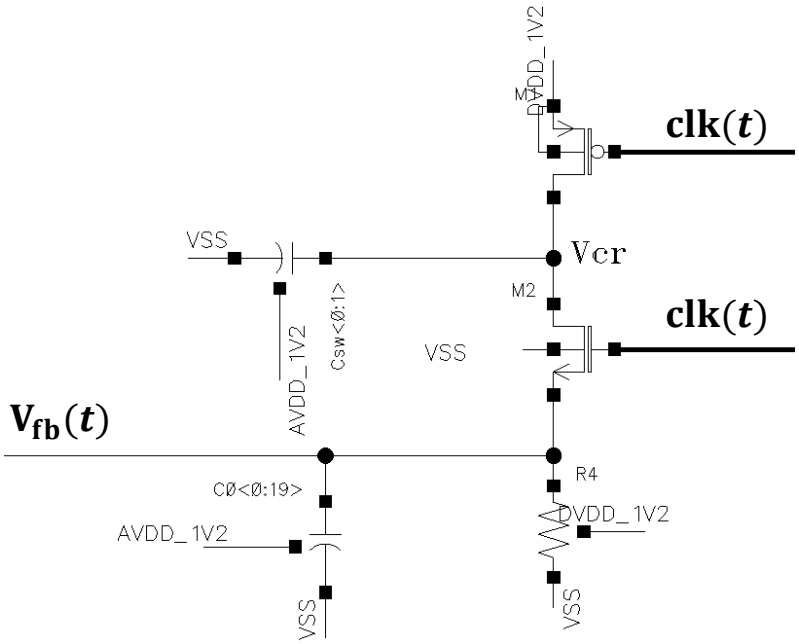


Figure 4.71: Feedback schematic in Candence, as described in Figure 4.69.

the control voltage, V_C, which is applied to the VCO [26]. The relationship between V_C and f_{out} was ascertained through a transient simulation using the Cadence Virtuoso platform. The simulation, visualised in Figure 4.72, involved varying V_C from its minimum value of 0V to its maximum value of V_{DD} (1.2V). The conditions for this simulation were set to a temperature of **27°C**, corresponding to room temperature, and utilized the **tt corner** specification. This simulation did not incorporate factors such as *mismatch* and *transient noise*.

V_C	$f_{out} = g(V_C)$	Var.% from 10.24MHz	$\bar{V}_{\text{fb}} = h(V_C)$
0V	7.71MHz	-24.6%	225.2mV
1.2V	11.18MHz	+9.2%	299.3mV

Table 4.9: Correlation between V_C, f_{out}, and $\bar{V}_{\text{fb}}$.

Table 4.9 shows the results of the simulations illustrating the relationship between the VCO and feedback. The expression $f_{out} = g(V_C)$ denotes that the output frequency, f_{out}, is a function of V_C, governed by the inherent

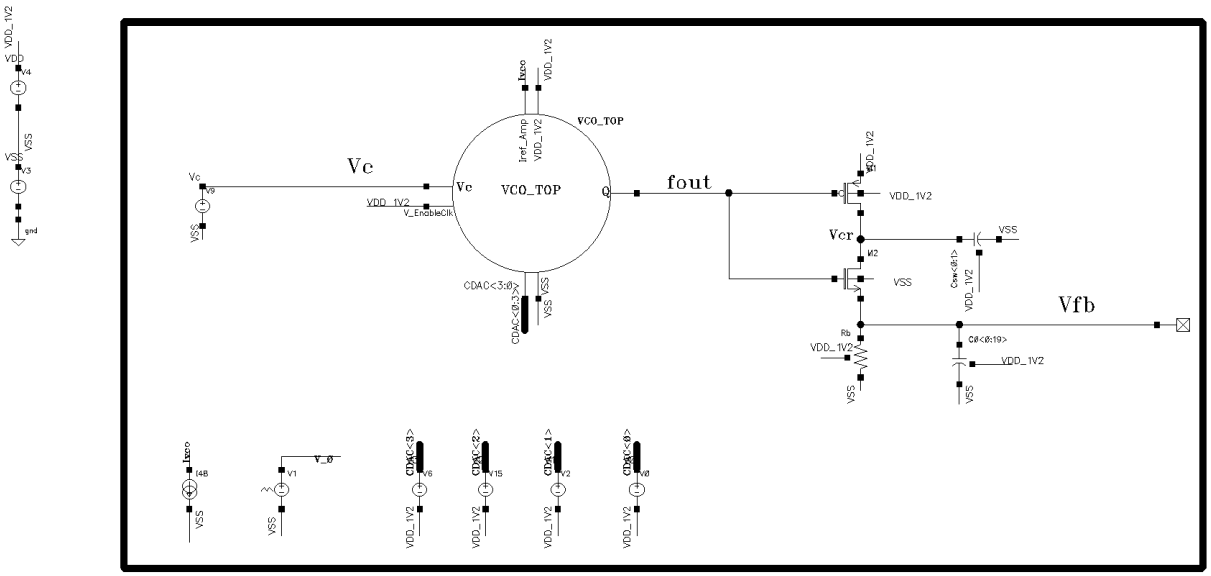


Figure 4.72: Specific Cadence's schematic utilized for the simulation of the ideal feedback voltage.

characteristics of the VCO. This relationship was elaborated in equation 4.93. Similarly, $\bar{V}_{\text{fb}} = h(V_C)$ conveys that the average feedback voltage is a function of V_C, as detailed in the equation 4.94.

The optimal value for V_{fb} is determined to be **277.7mV** when V_C is set to 600mV, as depicted in Figure 4.73. This value is inherently associated with the voltage reference, V_{ref}. The V_{ref} is applied to the non-inverting input of the *operational amplifier*. The primary function of this operational amplifier is to *nullify* the deviation between the ideal voltage value and the actual value when the VCO deviates from the central frequency. Consequently, this ensures the stabilisation of the system around the desired frequency.

4.4 Jitter Analysis of the Feedback

This subsection provides a detailed analysis of *jitter* within the feedback mechanism as a function of the VCO's output. The signal spectra analysis is essential for evaluating the performance of the system model. Using Cadence, the spectrum of the signal $V_{\text{fb}}(t)$ is examined, with details on transient simulation and parameters to prevent spectrum leakage found in Appendix B.

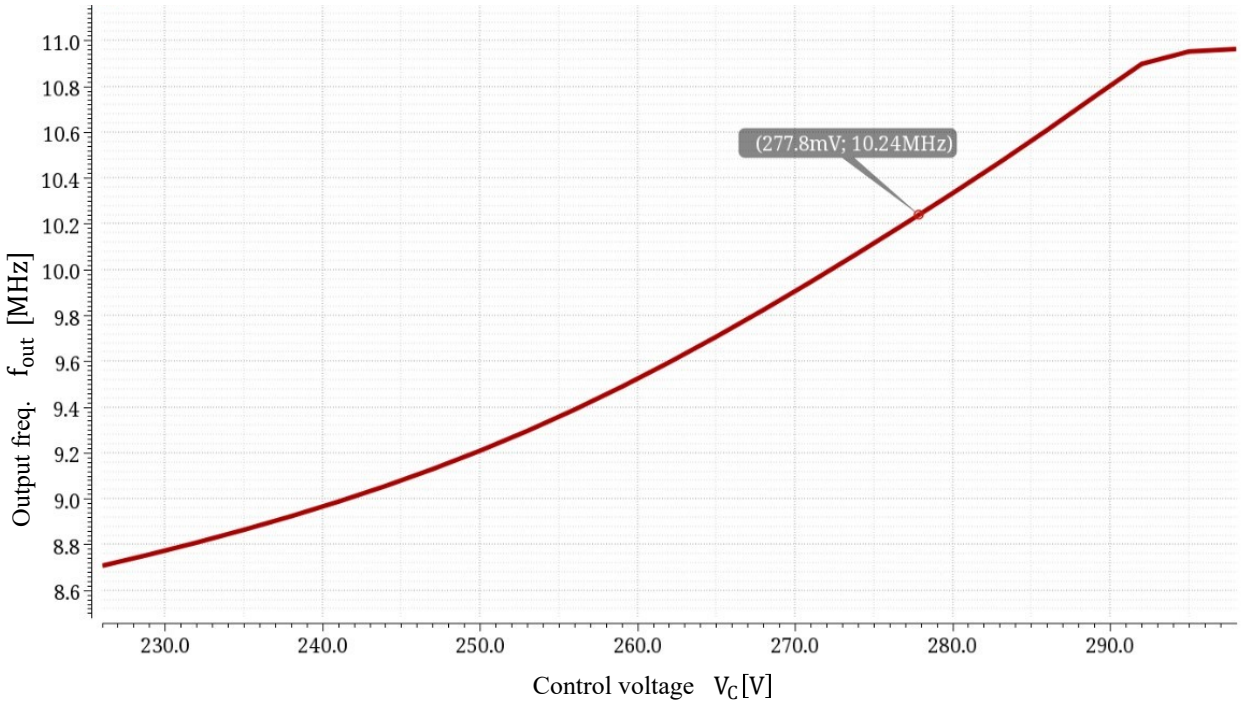


Figure 4.73: VCO and feedback output frequency in relation to V_C.

4.4.1 Ring oscillator output

To commence the spectrum analysis, consider the *ideal* periodic square wave $clk(t)$ characterized by a period T, where $f_{out} = \frac{1}{T}$ is the fundamental frequency, and amplitude V_{DD}. This wave is represented as a **Fourier series**:

$$clk(t) = \sum_{n=1,3,5,\ldots}^{+\infty} \frac{2V_{DD}}{n\pi} \sin(n\pi f_{out} t)$$

$$= \frac{V_{DD}}{\pi} \left[\sin(2\pi f_{out} t) + \frac{1}{3}\sin(6\pi f_{out} t) + \frac{1}{5}\sin(10\pi f_{out} t) + O\left(\frac{1}{n}\right) \right] \tag{4.99}$$

The Fourier series representation of $clk(t)$ elucidates its frequency components, essential for understanding its behaviour in the frequency domain and potential noise sources. Equation (4.99) is truncated at the third term for brevity and to provide a clearer understanding of the trend. This truncation justifies the approximation of higher-order terms as $O\left(\frac{1}{n}\right)$ since the amplitude diminishes for each further term of the series. The Fourier Transform

of $clk(t)$, therefore, is a series of delta impulses at harmonic frequencies:

$$\mathscr{F}\{clk(t)\}(f) = \sum_{k=1,3,5,\dots}^{+\infty} \frac{V_{DD}}{k}\delta(f - kf_{out})$$

$$= V_{DD}\left[\frac{1}{2}\delta(f - f_{out}) + \frac{1}{6}\delta(f - 3f_{out}) + \frac{1}{10}\delta(f - 5f_{out}) + O\left(\frac{1}{n}\right)\right] \tag{4.100}$$

The term $O\left(\frac{1}{n}\right)$ in equation (4.100) indicates the same approximation as the amplitude of harmonics decreases with frequency increase. Harmonics are present at odd frequencies such as f_{out}, $3f_{out}$, $5f_{out}$, and so on. An ideal system necessitates infinite bandwidth for the clock signal's precise transmission. However, real systems, constrained by bandwidth, can only transmit a finite set of harmonics.

With the Fourier representation of the ideal clock signal established, the impact of **phase noise** is considered. Incorporating a double-sided phase noise profile $L_{\varphi_{DS}}(f)$, as referenced in Section 3.4 and depicted in Figure 3.58b, the Fourier Transform of the noisy clock signal $clk_{noise}(t)$ is derived as:

$$\mathscr{F}\{clk_{noise}(t)\}(f) = \mathbf{L_{\varphi_{DS}}(f)} * \sum_{\mathbf{k=1,3,5},\dots}^{+\infty} \frac{\mathbf{V_{DD}}}{\mathbf{k}}\delta(\mathbf{f - kf_{out}})$$

$$= \sum_{k=1,3,5,\dots}^{+\infty} \frac{V_{DD}}{k}\mathbf{L_{\varphi_{DS}}(f - kf_{out})}$$

$$= V_{DD}\left[\frac{1}{2}L_{\varphi_{DS}}(f - f_{out}) + \frac{1}{6}L_{\varphi_{DS}}(f - 3f_{out}) + \frac{1}{10}L_{\varphi_{DS}}(f - 5f_{out}) + O\left(\frac{1}{n}\right)\right] \tag{4.101}$$

Equation 4.101 represents the **convolution** of the phase noise profile $L_{\varphi_{DS}}(f)$ with the Fourier Transform of the ideal clock signal, resulting in a series of phase noise characterize the latter profiles at multiple frequencies of f_{out}. Indeed, the phase noise profile, being intrinsically linked to the VCO's characteristics [28], can be approximated in a first-order manner as:

$$\mathscr{F}\{clk_{noise}(t)\}(f) \approx \frac{V_{DD}}{2}L_{\varphi_{DS}}(f - f_{out}) + O\left(\frac{1}{n}\right) \tag{4.102}$$

However, it is important to note that this approximation, equation 4.102 is only valid if the higher harmonics truly have a negligible effect. If this is not

true, the approximation may not be accurate [29].

The Cadence calculator offers a formula to determine a signal's **phase noise profile**. For accurate results, it is essential to sample the signal $clk(t)$ using the same number of DFT points, N, and employ a faster sampling period T_s than the clock. This ensures the square waveform values are captured at their central voltage for both high and low states, which is crucial for preventing spectrum leakage. The calculator might not flag errors but could yield a spectrum with noise levels higher than actual if these precautions aren't taken. Hence, the functions below utilise `Clk_nTs` instead of directly sourcing `'v("/clk" ?result "tran")'` from transient simulations[9]. The provided Cadence calculator formulas are derived from the theoretical foundations discussed, offering a practical approach to evaluate the phase noise profile:

- PN = PN(Clk_nTs "rising" (VAR("VDD") / 2) ?Tnom nil ?windowName "Hanning" ?smooth 1 ?windowSize 524288 ?detrending "None" ?cohGain "default" ?methodType "absJitter")
- PN_correct= PN(Clk_nTs "rising" (VAR("VDD") / 2) ?Tnom nil ?windowName "Hanning" ?smooth 1 ?windowSize 524288 ?detrending "None" ?cohGain "default" ?methodType "absJitter") - dB10(VAR("deltaF"))

The two formulas might seem similar, but the accurate phase noise profile, particularly the **single side**, is extracted from the latter. This formula adjusts for the DFT's bin resolution, `deltaF`, defined by $\frac{fs}{N}$ (where $N = 2^{19}$), which corresponds to 19.24Hz. The dBc/Hz scale is standardised to a **1Hz resolution** [30]. Therefore, the derived value is adjusted by the DFT's resolution, translating to a subtraction in the logarithmic scale[10].

Figure 4.74 illustrates the distinction: the *orange* plot displays the phase noise without the 1Hz resolution adjustment, while the *dark red* plot presents the appropriately scaled value. Both plots display multiple harmonics at multiples of f_{out}, a characteristic of square waveforms in the frequency domain, as referenced in equation 4.100. However, the peak is inverted due to the carrier frequency value's normalised phase noise, hence the term dB**c**/Hz of the vertical axis [30]. Each peak exhibits a triangular spectrum inherent to the VCO's design [26]. At this juncture, no other specifications rather than Table 4.6 were available for the VCO, as Section 3.4 was analysed post-clock

[9]The underlying methodology, which involves the DFT for spectrum evaluation, is detailed in Appendix B.

[10]Thanks is extended to co-supervisor *Dr. Daniel O'Hare* for highlighting the necessary correction for accurate phase noise profile computation using the formula `'PN - dB10(VAR("deltaF"))'`.

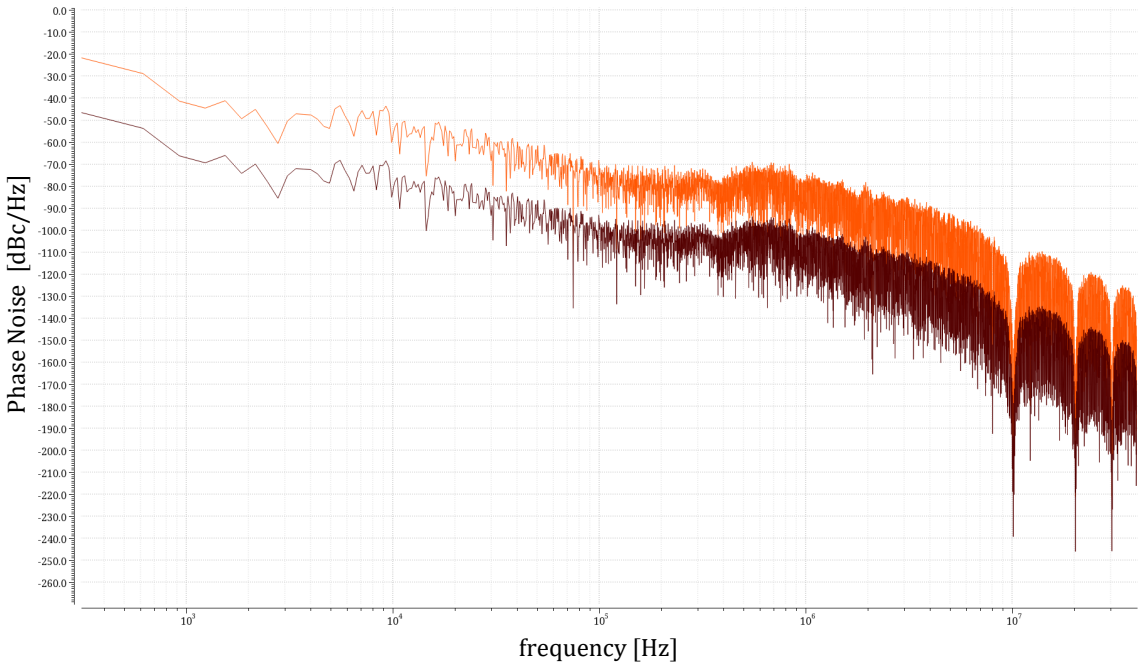


Figure 4.74: VCO's Phase Noise calculated

source implementation, leaving *no constraints* for the *phase noise spectrum*. Figure 4.74 offers an initial insight into equation 4.101, where phase noise is shifted at multiples of the carrier frequency and modulated in amplitude.

The subsequent figures (Fig. 4.75, 4.76a and 4.76b) provide visual representations that complement the theoretical discussions, illustrating the practical manifestations of the concepts discussed. Analysing the **clock's output spectrum** provides a clearer visual representation of the jitter spectrum. To ensure optimal resolution, the DFT computation utilised 2^{19} samples, maximising transient simulation time and balancing it with DFT bin resolution. The schematic set $V_C = 600\text{mV}$ for the VCO, yielding an output frequency $f_{out} = 10.7\text{MHz}$, as depicted in Figure 4.75, where only the first harmonic is shown.
In conjunction with transient noise simulations, the spectrum reveals that the VCO clock signal's spectrum profile exhibits $1/f^3$ and $1/f^2$ tendencies around the ring oscillators' output frequency [26], [29].

4.4.2 Feedback frequency response

With the PSD of the ring oscillator established, along with its phase noise spectrum, it is pertinent to analyze the impact of jitter on the output of the

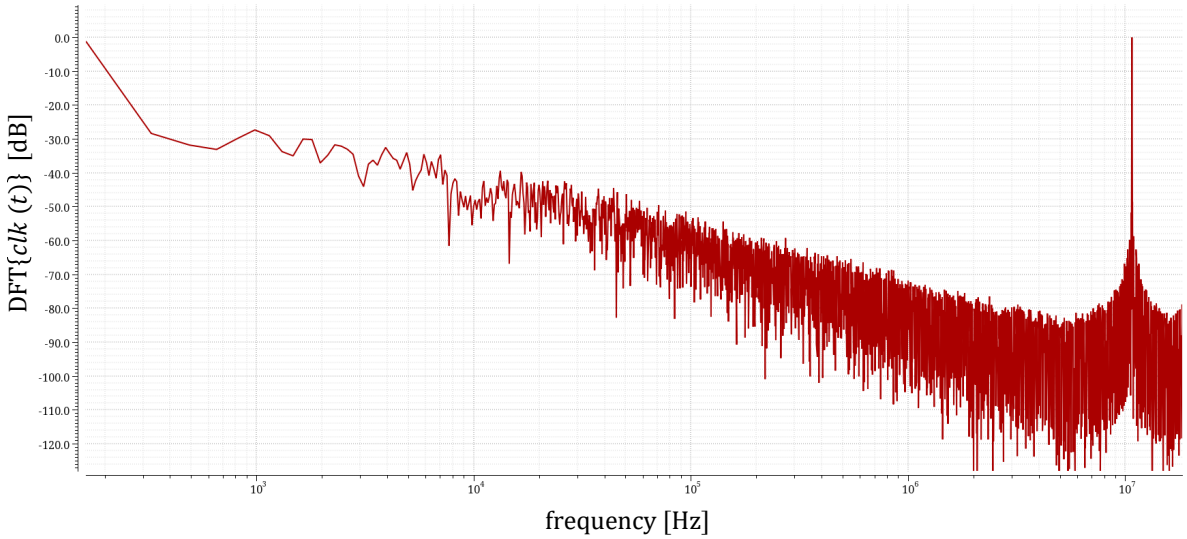


Figure 4.75: VCO's output spectrum

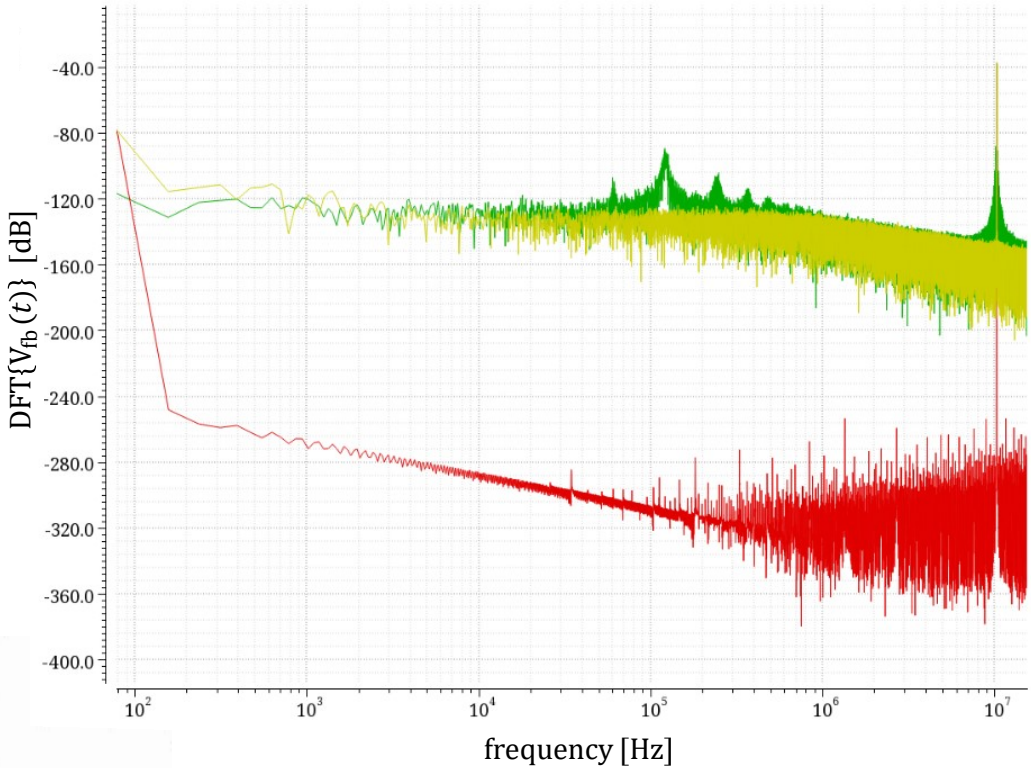
feedback network, as depicted in Figure 4.72.

Figures 4.76a and 4.76b originate from a schematic that integrates both an ideal clock source and a VCO, each followed by the feedback mechanism in an open loop. The aim is to assess the **influence** of the jitter profile on the feedback output:

- The *red* plot showcases the spectrum produced by the feedback network using an **ideal clock** generator **without thermal noise**. The inherent jitter-free ideal clock from Cadence's `'analog lib'` introduces noise around the output frequency due to the DFT's computational noise. It is significant to note that these peaks remain below -240 dB.

- The *yellow* plot in Figure 4.76a incorporates both the **ideal clock** and **thermal noise**, capping the noise frequency at $4 \cdot f_{out} = 40MHz$. Although this threshold can be raised for better precision, it would prolong the simulation time. The fourfold increase in f_{out} is empirically substantiated, emphasizing that the noise missing in the red simulation is effectively neutralized, exceeding the initial DFT noise.

- The *green* plot represents the outcome, utilizing the **VCO** to drive the feedback and apply **noise** analysis for the simulation. This graph illustrates how the VCO's spectrum from Figure 4.75 is directly translated to the feedback's output at the same output frequency [27]. The plot



(a) All feedback spectra simulations.

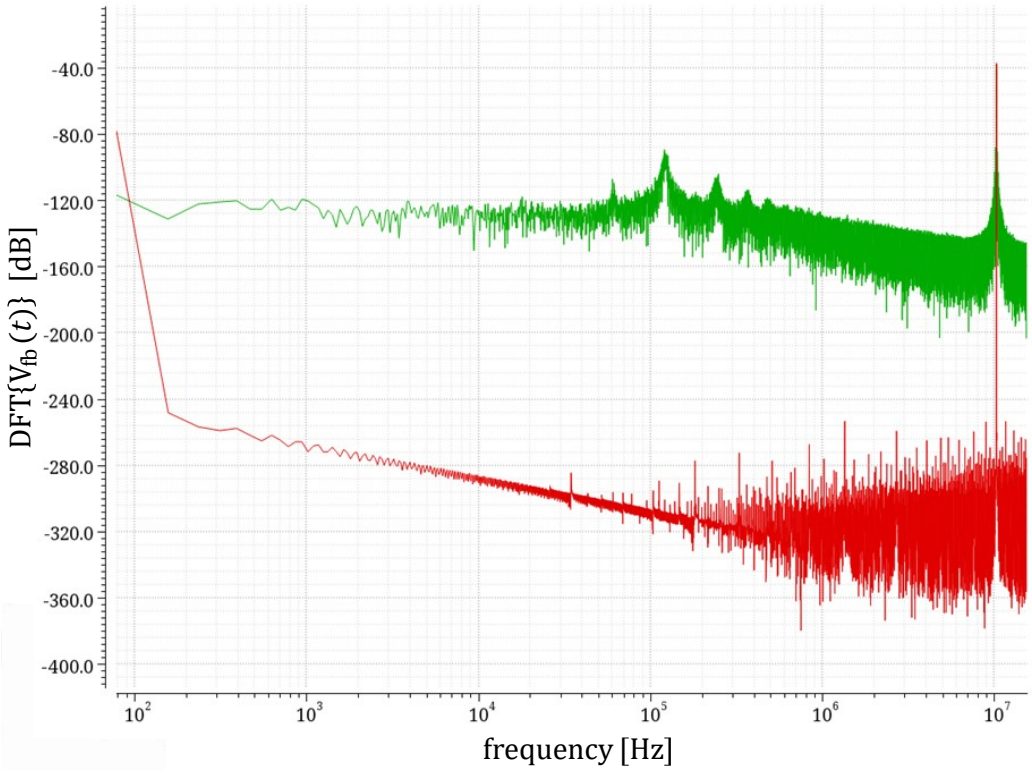
Figure 4.76: FLL's Feedback DFT analysis.

displays harmonics in the 100kHz range, indicating that some are also shifted backwards from the fundamental frequency.

The feedback network's transfer function plays a pivotal role in shaping the spectrum profile observed the green and yellow plots shown in Figure 4.76. As delineated in [24], the transfer function for the feedback network is given by a **dominant pole** characteristic:

$$\beta(s) = \frac{K_{F2V}}{1 + \frac{s}{\omega_{F2V}}} \tag{4.103}$$

Where K_{F2V} represents the static gain of the "frequency to voltage" feedback mechanism, and ω_{F2V} denotes the pole's pulsation. The parameters are



(b) Comparative analysis with and without noise activation.

Figure 4.76: FLL's Feedback DFT analysis.

defined as:

$$\begin{cases} K_{F2V} = \dfrac{V_{DD}(R_l C_s)}{(1 + R_l C_s f_{out})^2} \\ \\ \omega_{F2V} = \dfrac{(1 + R_l C_s f_{out})}{R_l (C_l + C_s/2)} \end{cases} \tag{4.104}$$

The transfer function coefficients are given by Equation (4.104). The gain K_{F2V} represents the amplitude response of the network to input variations, while the pole ω_{F2V} indicates the frequency at which the response starts to attenuate, evident at 1MHz, as shown in the *yellow* plot of Figure 4.76a with an ideal clock. The components R_l, C_s, and C_l are integral to the design of the feedback network and are detailed in Table 4.8. The spectrum in the figures illustrates the feedback network's response, shaped by both the transfer function (Equation 4.103) and the input signal characteristics (Equation 4.102).

4.4.3 Feedback Noise

The noise sources within the feedback network are analyzed in this section. The objective is to formulate an equation for subsequent system analysis. Analyzing these noise characteristics is crucial for evaluating their impact on system performance. The output voltage's total noise contribution consists of multiple components [26]. This is attributed to the non-ideal nature of MOSFETs and the thermal noise from resistors. The PSD at the feedback output is segmented as defined in Equation 4.105.

$$S_{v_{\text{out}}}(f) = S_{v_{\text{out,eq}}}(f) + S_{v_{\text{out,2}}}(f) + \left(\frac{R_2}{R_{\text{eq}} + R_l}\right)^2 S_{v_{\text{MOS}}}(f) + S_{v_{\text{out,flicker}}}(f) \approx$$

$$\approx S_{v_{\text{out,eq}}}(f) + S_{v_{\text{out,2}}}(f) \tag{4.105}$$

Components of the noise are:

- $S_{v_{\text{out,eq}}}(f)$ Noise contribution of the equivalent resistance R_{eq} of the switch capacitor, see Equation 4.94.
- $S_{v_{\text{out,2}}}(f)$ Noise contribution of R_l.
- $S_{v_{\text{MOS}}}(f)$ Noise generated by the MOSFET's "on" on "off" resistance.
- $S_{v_{\text{out,flicker}}}(f)$ Flicker noise (1/f) of the MOSFETs.

In [24], the effect of the R_{off} of transistors is analyzed better to describe the charging and discharging of the switched capacitor. However, to focus the noise analysis of the feedback at the VCO frequencies, switches can be simplified as *ideal* [27], [28], with the switched capacitor's noise contribution being that of an equivalent resistor R_{eq}. This is viewed as an independent noise source through the resistive divider. The noise fraction of R_{eq} at the output is influenced by the resistive division between R_{eq} and R_l:

$$S_{v_{\text{out,eq}}}(f) = \left(\frac{R_2}{R_{\text{eq}} + R_2}\right)^2 \cdot 4kTR_{\text{eq}} \tag{4.106}$$

Instead, the noise contribution of R_l in the Laplace domain is:

$$\begin{cases} S_{v_{\text{out,2}}}(s) = |G(s)|^2 \cdot 4kTR_l \\ G(s) = \dfrac{R_l}{R_{\text{eq}}(1 + sR_lC_l) + R_l} \end{cases} \tag{4.107}$$

The noise generated by R_l is directly at the output of the divider, and its contribution to the output noise is modulated by the transfer function $G(s)$. The noise expression can be further simplified:

$$S_{v_{\text{out}}}(s) = 4kT \left[\left(\frac{R_2}{R_{\text{eq}} + R_2} \right)^2 R_{\text{eq}} + \left| \frac{R_l}{R_{\text{eq}}(1 + sR_lC_l) + R_l} \right|^2 R_l \right] \approx$$

$$\approx \frac{4kT}{R_2 C_l^2} \cdot \frac{1}{s^2} \tag{4.108}$$

This equation provides a practical noise estimate in the FLL feedback network. The formula is deemed suitable, given:

- The term $\frac{1}{s^2}$ captures the high-frequency behavior.
- The term $\frac{4kT}{R_2 C_l^2}$ denotes the appropriate scale for noise amplitude.

This approximation holds primarily at **high frequencies** and does not depict the circuit's low-frequency behavior [26].

4.5 Final FLL Structure

The final structure of the FLL results from systematic design iterations and comprehensive evaluations. The following sections delineate the characteristics and functionalities of the established design, placing it in context with theoretical models and benchmarks. While the design is a significant step towards completion, it requires integrating *additional components* to achieve full functionality. Collaborative efforts have played a pivotal role in reaching this stage[11].

4.5.1 Voltage reference

The voltage reference for the positive terminal of the amplifier, illustrated in Figure 4.62, is established using a resistive divider. The reference voltage aims to align with the ideal average voltage feedback detailed in Section 4.3.2.

[11] Contributions from colleagues at MCCI and the Anser project team have been instrumental. Noteworthy contributions include the FLL structure **modelling** by PhD candidate *Mikhail Gaidukov*, layout insights related to the **resistor divider** and the **trimming capacitive circuit** in the VCO by *Aleksandr Sidun*, and the **amplifier** block provided by PhD Candidate *Manish Srivastava*.

To meet layout requirements and minimize component mismatch [26], a single capacitor of type `'rppolywo'` with a resistance of $7k\Omega$ is utilized. The resistor configuration comprises 13 series resistors for the equivalent R_1 and 4 for R_2, as depicted in Figure 4.77.

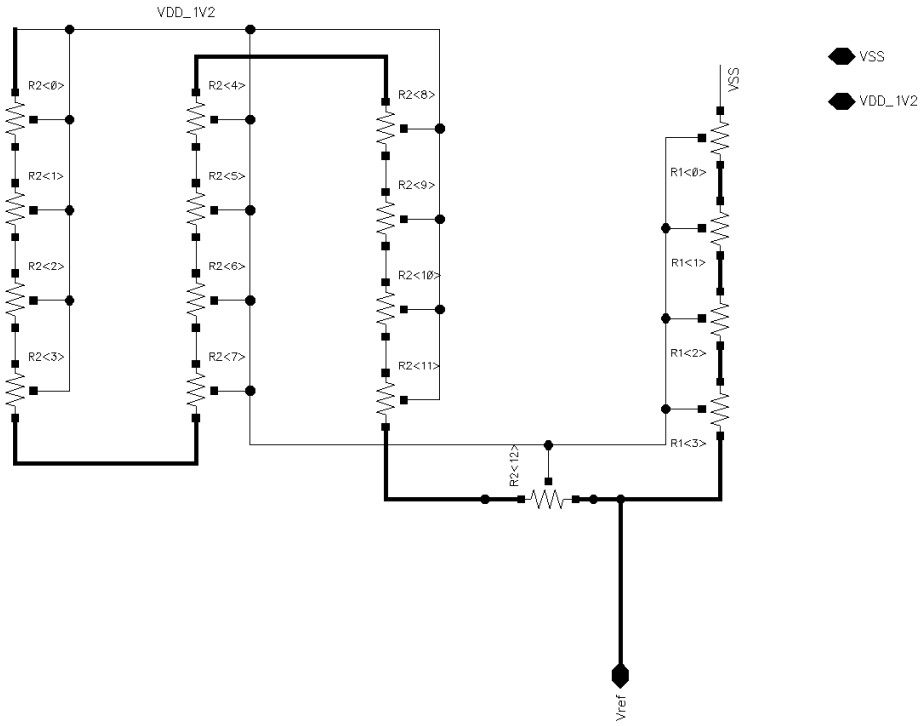


Figure 4.77: Resistor Divider Schematic on Cadence.

Given the design equivalence of all resistors, the output voltage is determined by Equation 4.109, **approximating** the desired ideal voltage:

$$V_{\text{ref}} = V_{DD}\frac{4}{4+13} = 282mV, \qquad \text{where} \quad V_{\text{ref, Ideal}} = 278mV \tag{4.109}$$

The resistor divider is powered directly by the chip's LDO to ensure a stable supply throughout the chip. Notably, the Anser chip features two LDOs: one dedicated to *analog* components, to which this divider is connected, and another for *digital* components.

Noise contribution

As delineated in Section 4.4.3, the noise contribution for the feedback is analogous but simplified for this context. Equation 4.110 gives the PSD :

$$S_{v_{\text{ref}}}(f) = \left(\frac{R_2}{R_1 + R_2}\right)^2 4kT(R_1 + R_2) \tag{4.110}$$

As further detailed in Section 4.5.4, the noise should not impact the FLL performance. The cumulative resistance, $R_1 + R_2$, should be constrained to $120k\Omega$ or less.

4.5.2 Amplifier

The amplifier, a crucial component of the FLL structure, is based on a folded cascode OTA architecture, and its schematic is shown in Figure 4.78. The

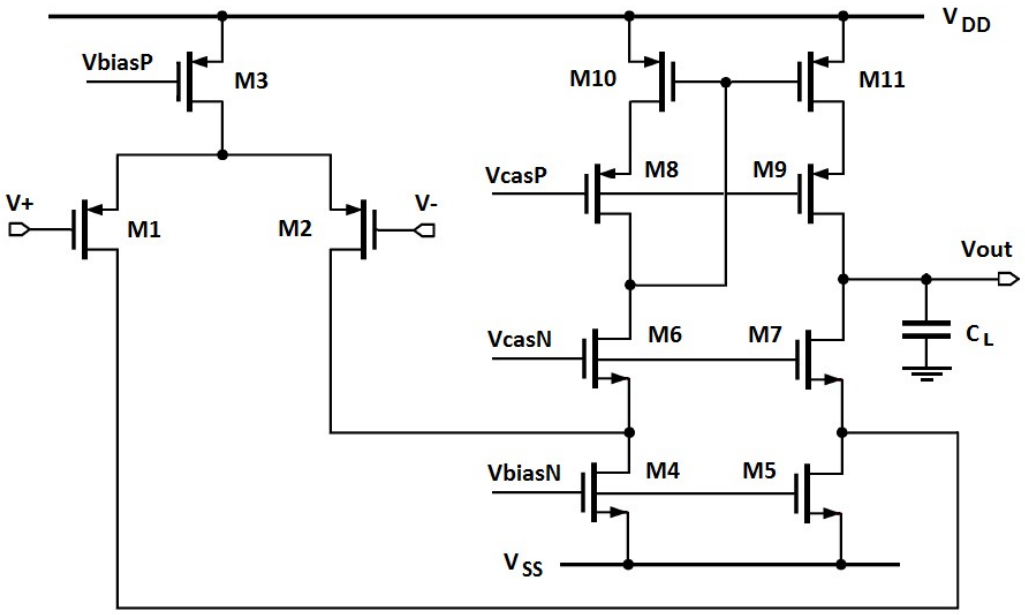


Figure 4.78: Amplifier schematic used in the FLL

PMOS differential pair, implemented in PMOS, has the following specifications in Table 4.10.
The transconductance (g_m) of the OTA, representing the variation in the output current to the input voltage, is $2.1\mu S$. This value is derived from the open-loop analysis of the amplifier, as illustrated in Figure 4.79. The load

Parameter	Value
$\left(\frac{W}{L}\right)_P$	$\frac{4\mu m}{2\mu m}$
Fingers	2
Multiplier	60

Table 4.10: PMOS differential pair

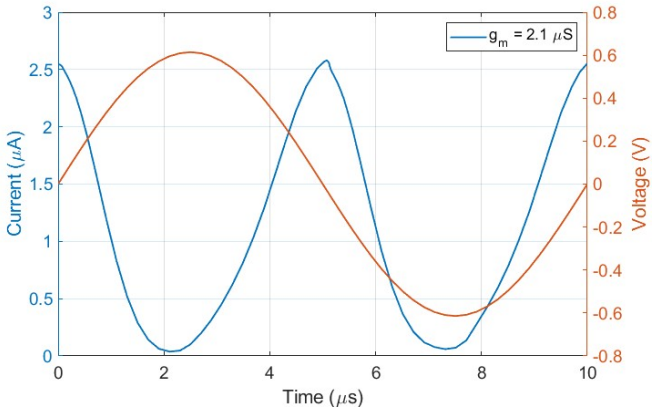


Figure 4.79: Time analysis of OTA Input Voltage and Output Current for Transconductance g_m Evaluation.

capacitance (C_L) is set at $1pF$, aligning with the integration requirements of the FLL structure, which leads to the OTA's transfer function shown in Equation 4.111.

$$H(s) = \frac{g_m}{sC_L} \tag{4.111}$$

Flicker noise was a challenge in the design, as it has impacted the FLL's performance [26], [27]. This issue was identified through **top-level** chip simulations, leading to design iterations of the amplifier to address it.

4.5.3 Final VCO

Top-level simulations revealed *discrepancies* in the FLL phase noise spectrum compared to the expected outcomes for the CTΔΣM model detailed in Section 2.5. When simulating the clock source in conjunction with the ADC, while enabling thermal noise in the transient simulation for spectrum evaluation, the achieved SNR was below the 70 dB target. This discrepancy highlighted the limitations of Equation (2.28) and the associated white noise jitter model given by Equation 2.9. The model's constraints, as defined by Equation 2.10, became evident. The analysis in Section 3.4 further emphasized the significance of the clock's phase noise profile and its **impact** on the CTΔΣM [20], [22], [30]. The observed SNR was attributed to Equation 3.86. Referencing [26], [27], [29], the VCO was modified to have three stages, following the design process outlined in Section 4.2. The updated version used the thick gate oxide MOSFET for the ring oscillator and introduced a current

mirror to supply the PMOS. This adjustment aimed to enhance the VDD impact and reduce jitter in the FLL. Additionally, a trimming circuit was added for the capacitive load. The final VCO architecture and its components are detailed in Table 4.11 referring to Figure 4.80 and 4.84.

Part	$\left(\frac{\mathbf{W}}{\mathbf{L}}\right)$	**Fingers**	**Multiplier**
Current mirror PMOS	$\left(\frac{500nm}{8\mu m}\right)$	2	2, $46_{\text{output current}}$
Ring Oscillator PMOS	$\left(\frac{1\mu m}{300nm}\right)_{\text{NAND}}, \left(\frac{2\mu m}{300nm}\right)$	2	3
Buffers PMOS	$\left(\frac{500nm}{200nm}\right)$	2	4
Ring Oscillator NMOS	$\left(\frac{1\mu m}{2\mu m}\right)_{\text{NAND}}, \left(\frac{2\mu m}{4\mu m}\right)$	2	3
Buffers NMOS	$\left(\frac{500nm}{200nm}\right)$	2	3

Table 4.11: Three stages ring oscillator

In Figure 4.80, the current mirror circuit is not depicted due to its elementary nature. The sources of the PMOS transistors appear to be connected to an unmarked node. However, all those in the ring oscillator are connected to the output of the PMOS in the current mirror, specifically the one with a 46 multiplier as indicated in Table 4.11. Only a single branch of the current mirror is required to supply current to the VCO, as during the oscillation, only one stage of the ring oscillator is active at any given time, with the others being inactive [24]. On the other hand, the PMOS transistors of the two buffers are connected to the *digital* LDO.

The VCO described in Section 4.2 had an output frequency of 10.7 MHz

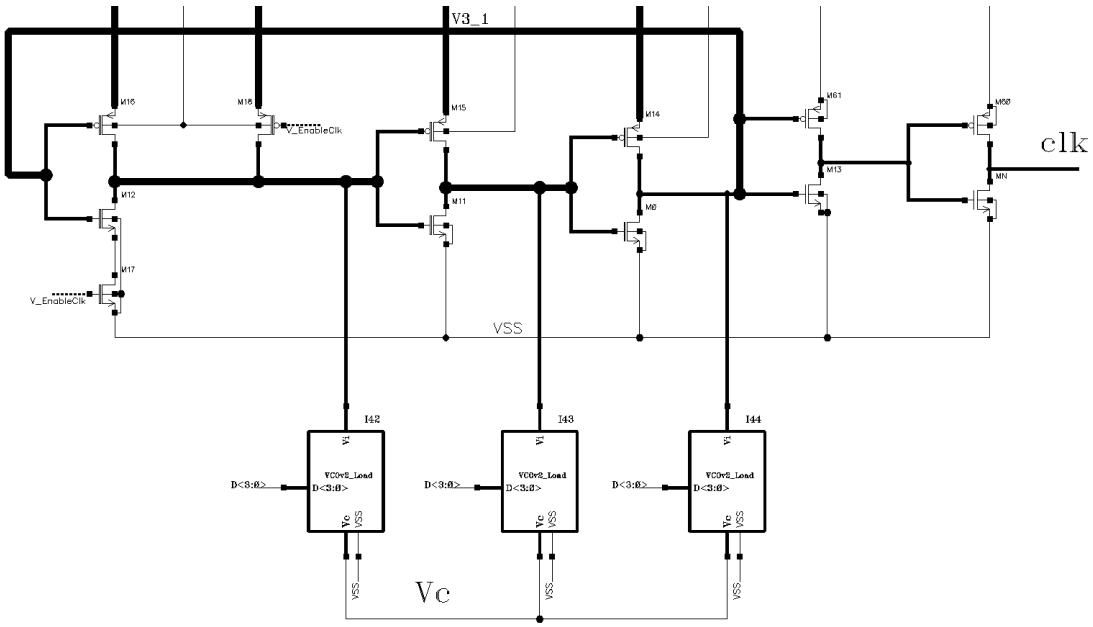


Figure 4.80: Three stages VCO design in Cadence.

with a periodic jitter standard deviation of 9ps and utilized a capacitive load of **1.56pF**. In contrast, the updated VCO achieves an output frequency closer to the ideal value, as outlined in Table 4.6, with $\mathbf{f_{out} = 10.12\ MHz}$. However, the jitter remains comparable at $\boldsymbol{\sigma_p = 10\ ps}$. Given that the output frequency remains within the same operational range and the *initial* capacitive load remains unchanged, the primary benefit is observed in the spectrum produced at the FLL's output, as illustrated in Figure 4.81. The average power consumption remains consistent with the previous VCO, registering at approximately $\mathbf{130\mu W}$. Additionally, the gain $\mathbf{K_{VCO}}$ is quantified as $\mathbf{5.25\ MHz/V}$.

Figure 4.81 presents two overlapping PSDs centred at the output frequencies to analyse the spectrum distribution. In the initial depiction, one spectrum is emphasized, while the subsequent image accentuates the other, facilitating a comprehensive comparison between the two spectral densities. Both FLLs utilize the OTA delineated in Section 4.5.2, denoted as `new Amp`. The primary PSD, labelled `FLL[new Amp, VCOv3]` in blue, corresponds to the FLL output spectrum with the five-stage ring oscillator from Section 4.2. Con-

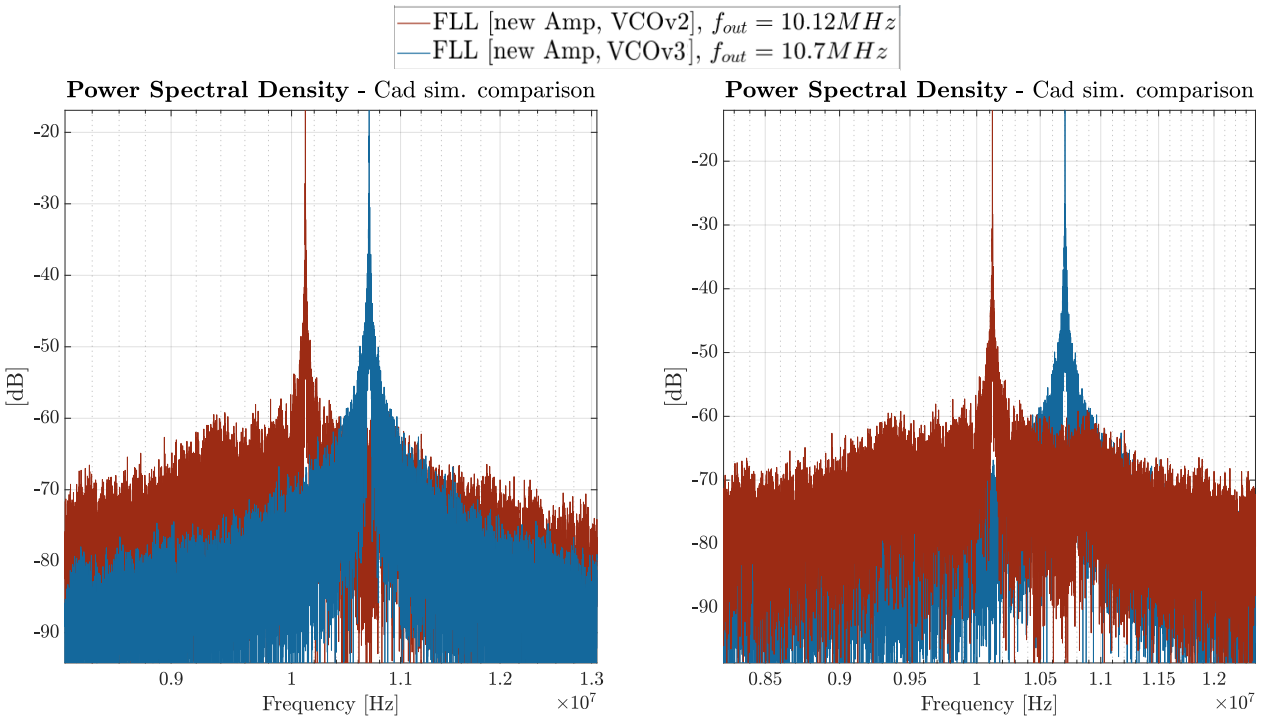


Figure 4.81: Comparison of FLL spectra

versely, `FLL[new Amp, VCOv2]` relates to the **revised** three-stage VCO[12]. As discernible from the figure, the spectrum of the first plot appears considerably narrower than the latter. Like white noise, the red spectrum exhibits a flat characteristic, spanning a broader frequency range than the blue. However, it displays a diminished $1/f^3$ and $1/f^2$ characteristic around the central frequency compared to the blue, which, despite its narrower frequency spread, lacks a consistent noise feature in its spectrum. This suggests that the blue PSD possesses a reduced phase noise profile compared to the red [30], as it extends less in frequency but exhibits a broader local phase noise around the output frequency. This distinction clarifies the **difference** between the standard deviation of periodic jitter, σ_p, and its phase noise profile $L(f)$. Considering the definition of jitter noise's standard deviation [26] , it can be evaluated as:

$$\sigma_p = \sqrt{\frac{2}{2\pi f_{out}} \int_0^{\frac{f_{out}}{2}} L_{\text{single side}}(f) \left|1 - e^{-\frac{j\omega}{f_{out}}}\right| df} \tag{4.112}$$

[12]The letter `'v'` adjacent to the VCO name signifies the design version, not the oscillator stages.

Equation 4.112 suggests that the standard deviation employed to characterize the CTΔΣM model is the root mean squared value of the phase noise spectrum's first difference transfer function magnitude, in this context, single-sided. Consequently, the plot in Figure 4.81 represented in blue exhibits a reduced σ_p compared to the plot depicted in red.

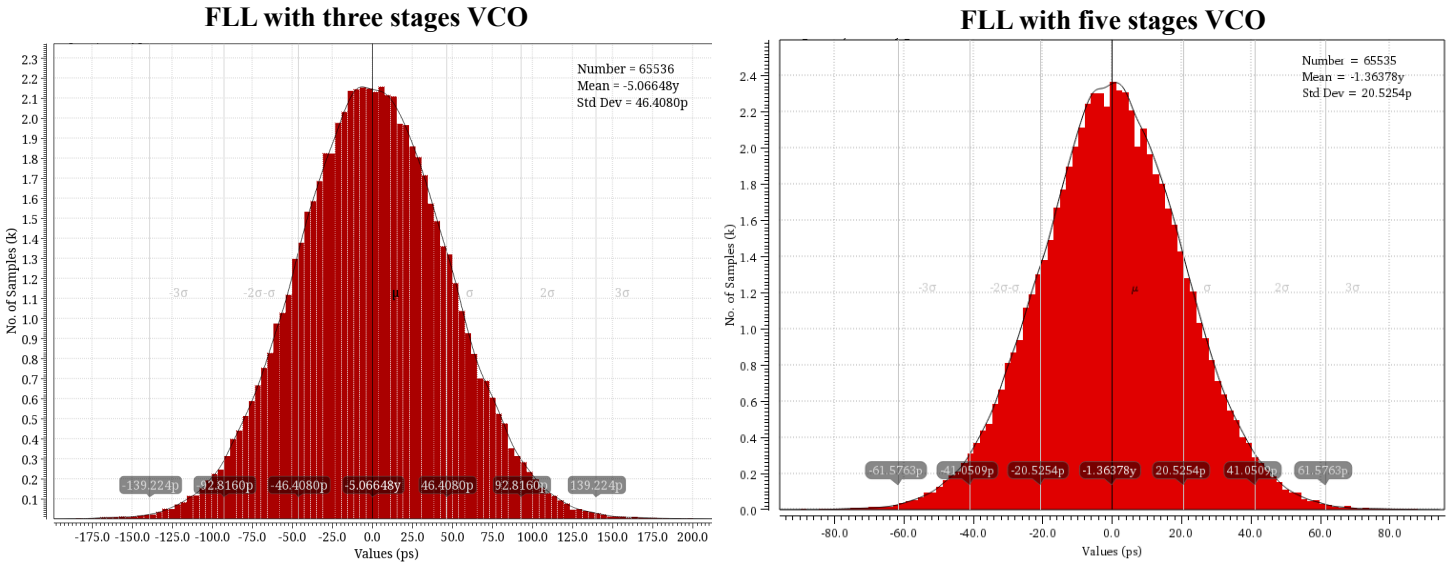


Figure 4.82: Comparison of FLL's σ_p

Utilizing the Cadence calculator facilitates the computation of the periodic jitter's standard deviation by employing the inherent function and generating the histogram, which subsequently auto-evaluates the statistics in terms of mean and variance.

- ◇ CLK = v("/out_fll" ?result "tran")
- ◇ period_jitter(CLK "rising" ?mode "auto" ?binSize 0 ?xName "time" ?outputType "plot")
- ◇ histogram2D(period_jitter(CLK "rising" ?mode "auto" ?binSize 0 ?xName "time" ?outputType "plot") 200 "standard" "stddev" t nil)

As depicted in Figure 4.81, the FLL equipped with the updated VCO exhibits a $\boldsymbol{\sigma_p = 46ps}$ value that is more than double that of the initial design at $20ps$. However, the spectrum's flat region proves advantageous for the IBN, as indicated by Equation 3.85. The CTΔΣM can mitigate the jitter's impact as in the model Simulink of Section 2.5, **only** if the clock spectrum is completely

white [20], [22], [30]. This understanding drove the decision to refine the FLL to align more closely with the initial clock model.

Trimming Capacitive Load

The initial approach to design the trimming circuit involved testing the three-stage VCO in a PVT simulation, varying the corner cases (`ss`, `tt`, `sf`, `fs`, `ff`), voltage supply (1.14V, 1.2V, 1.26V), and temperature (27°C and 40°C). This analysis facilitated the assessment of how the ring oscillator's output frequency deviated from the ideal.

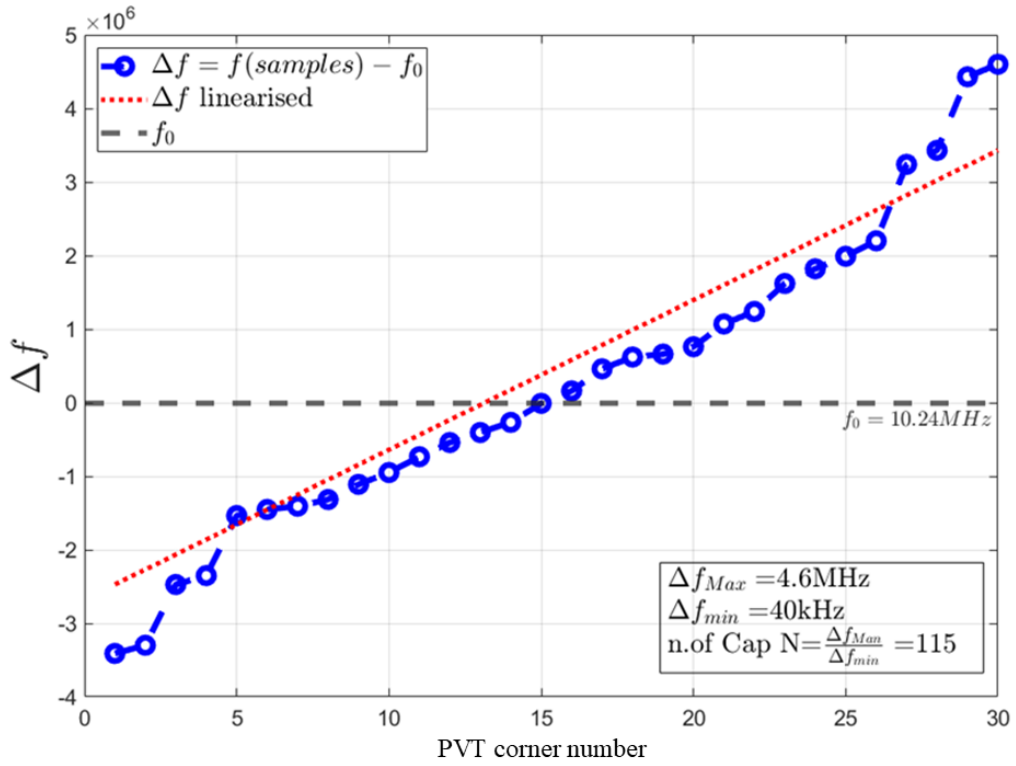


Figure 4.83: PVT Analysis of VCO Frequency Variation

As depicted in Figure 4.83, the frequency variation is nearly linear. The centre represents the `tt` simulations, the top-right primarily showcases the ff simulations with a maximum frequency deviation of 4.6MHz from the ideal, and the bottom-left predominantly displays the ss simulations. The remaining plots represent a linear combination of all PVT analysis factors, totalling 30 measured output frequencies. The initial estimate for the number of capacitors required to adjust for all possible variations is determined by the

ratio of maximum to minimum frequency variation, yielding a rounded value of 115 capacitors. While the smallest unit of the `crtmom` capacitor, $3fF$, can be used to adjust the minimum frequency step and verified through simulation, this approach is area-inefficient.

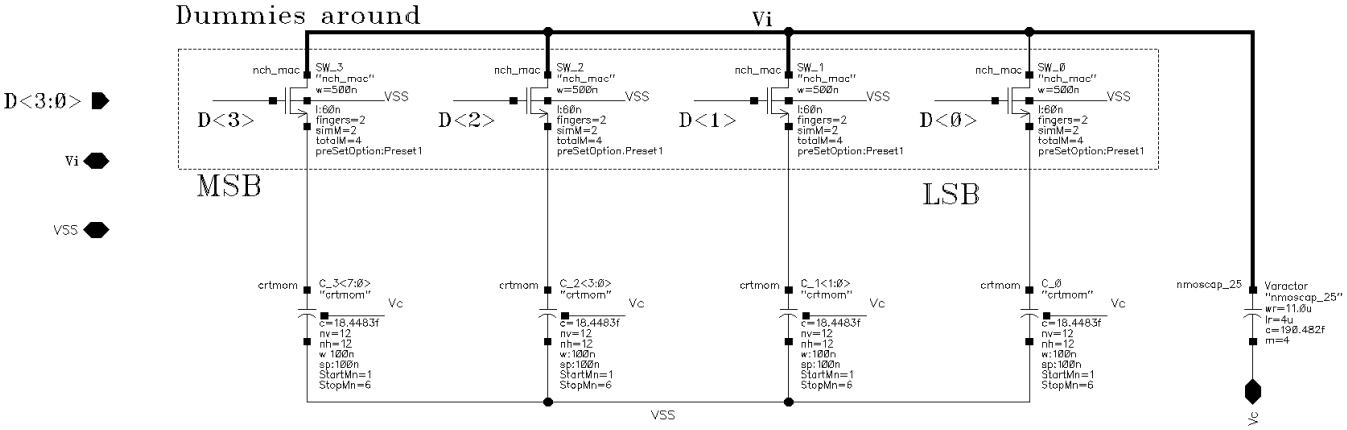


Figure 4.84: Capacitive load with trimming circuit

Based on the methodology outlined in [26], Figure 4.84 depicts the capacitive load trimming circuit with a unit capacitance of $190fF$, with subsequent capacitors being integral multiples of this base unit. The trimming mechanism employs a binary code, `D<3:0>`. This code controls the gates of the NMOS switches, which have dimensions of $w = 500nm$ and $l = 60nm$ and are configured with two fingers and a multiplier of 2. In this binary representation, the code `'D<3>D<2>D<1>D<0>'` indicates that the leftmost bit, `D<3>`, corresponds to $2^3 \times 19fF$, followed by $2^2 \times 19fF$, and so on. This trimming circuit facilitates adjustments to the central frequency, aligning it with desired values. A reference table detailing the potential codes to be applied at 1.2V and $40°C$ is provided in Table 4.12.

It is worth noting that the codes do not span the entire binary range, i.e., from '0000' for `ss` to '1111' for `ff`. This deliberate choice provides a margin to accommodate potential mismatches during fabrication. While the 65 nm manufacturing process is renowned for its reliability [31] and often yields results close to the typical (`tt`) corner, it is prudent to account for unforeseen variations. The maturity of the 65 nm process does lend confidence in its consistency, but as with all manufacturing processes, there's always a possibility, albeit small, of deviations.

Code D<3>D<2>D<1>D<0>	Corner
0010	ss
1000	sf
0110	tt
1000	fs
1010	ff

Table 4.12: Code for trimming circuit

4.5.4 Final Model for the Clock Source

With all the parameters of the clock source defined, it is feasible to analyse the system from a block diagram perspective, as depicted in Figure 4.64. Taking into account all the previously evaluated model components, we reference the *open loop gain* (Equation 4.89), the OTA and feedback transfer functions (Equations 4.111 and 4.103), and the static VCO gain K_{VCO}. Additionally, we consider the noise spectra of the feedback and resistor divider (Equations 4.105 and 4.110). Consequently, the system's noise contributions to the **output frequency** are defined in the Laplace domain by the system of equations 4.113 for each building block.

$$\begin{cases} N_{v_{\text{ref}}}(s) = \left| \dfrac{H(s)K_{\text{VCO}}}{1+T(s)} \right|^2 S_{v_{\text{ref}}}(s) \\ N_{\text{OTA}}(s) = \left| \dfrac{K_{\text{VCO}}}{1+T(s)} \right|^2 S_{\text{OTA}}(s) \\ N_{\text{VCO}}(s) = \left| \dfrac{1}{1+T(s)} \right|^2 S_{\text{VCO}}(s) \\ N_{\text{feedback}}(s) = \left| -\dfrac{H(s)K_{\text{VCO}}}{1+T(s)} \right|^2 S_{v_{\text{feedback}}}(s) \end{cases} \tag{4.113}$$

By summing together all these *independent* noise sources components scaled on the system transfer function where they are injected, we can determine the **total noise contribution**. This is visualized in Figure 4.85.

$$N_{\text{total}}(s) = N_{v_{\text{ref}}}(s) + N_{\text{OTA}}(s) + N_{\text{VCO}}(s) + N_{\text{feedback}}(s) \tag{4.114}$$

Utilising Equation 4.112, we can assess the expected standard deviation of the FLL's periodic jitter by integrating the noise of Equation 4.114.

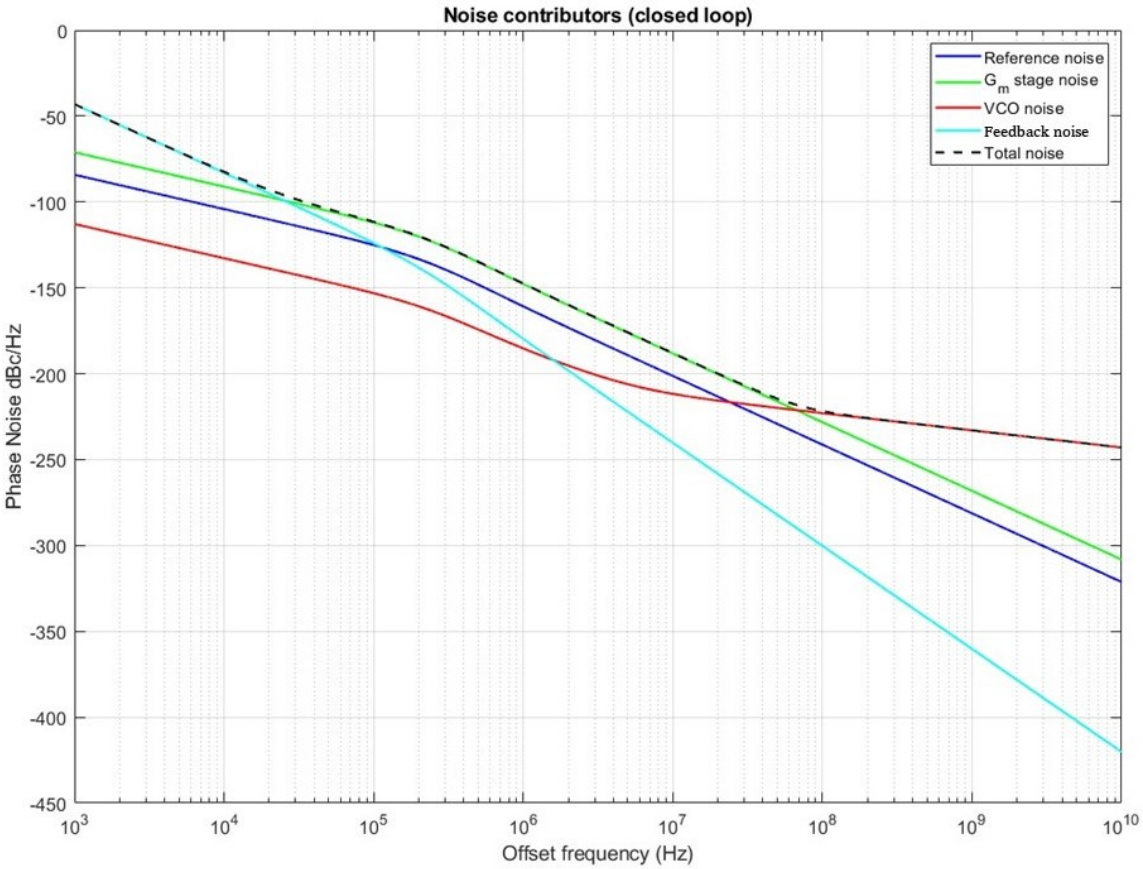


Figure 4.85: Clock source noise transfer functions

As illustrated in Figure 4.86, the resultant $\boldsymbol{\sigma_p}$ is $38ps$, closely aligning with the 46ps determined through Cadence simulations and depicted in the noise transient simulation histogram of Figure 4.82. It is essential to note that this value served as a starting reference during the design phase to ascertain all system noise components. In the interest of maintaining a coherent sequence in the component development narrative, this model is introduced at the conclusion.

For further clarity, the standard deviation of the **absolute jitter**, denoted as σ_a, is also assessed. The absolute jitter represents the cumulative deviation of a clock signal over an observation period relative to an ideal jitter-free signal. Unlike the periodic jitter, which varies with each period, the observation duration influences the absolute jitter's magnitude. Its value can be derived

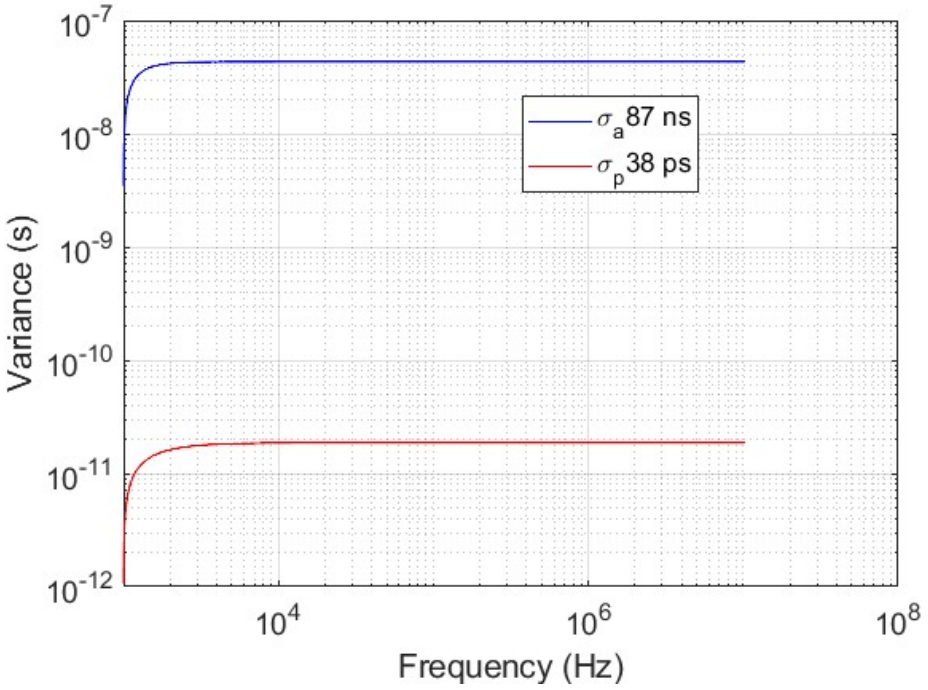


Figure 4.86: Expected standard deviation of absolute jitter σ_a and periodic jitter σ_p.

from the total noise spectrum, as described by Equations 4.115 [27], [28].

$$\begin{cases} \sigma_a = \sqrt{\dfrac{2}{2\pi f_{out}} \displaystyle\int_0^{\frac{f_{out}}{2}} N_{\text{total, single side}}(f) df} \\ \sigma_p = \sqrt{\dfrac{2}{2\pi f_{out}} \displaystyle\int_0^{\frac{f_{out}}{2}} N_{\text{total, single side}}(f) \left|1 - e^{-\frac{j\omega}{f_{out}}}\right| df} \end{cases} \tag{4.115}$$

It is noteworthy that the estimated value of σ_a at 87ns serves as a reference point. The stark contrast between this and the σ_p value of 46ps—almost a thousand-fold difference—is expected. This significant disparity arises because the integration spans the entire spectrum without being weighted by the factor $\left|1 - e^{-\frac{j\omega}{f_{out}}}\right|$. This unweighted integration naturally results in a much larger absolute jitter value, emphasizing the importance of considering spectral characteristics when evaluating jitter metrics.

4.5.5 Full Model Results

The final configuration of the schematic is depicted in Figure 4.87. The parameters obtained from the Cadence simulations are summarized in Table 4.13 and align with the initial *specifications* of the CTΔΣM model, as referenced in Table 4.6.

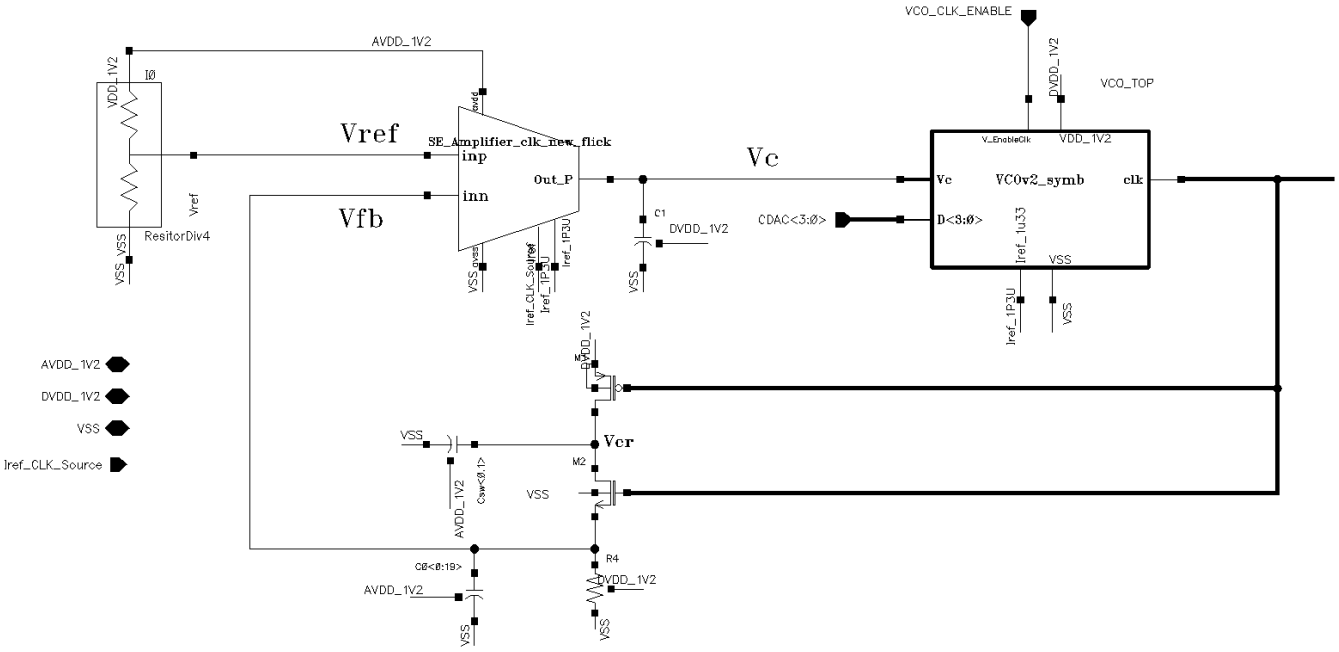


Figure 4.87: Complete Clock schematic, the FLL, in Cadence.

f_{out}	$\lvert\Delta f_{out}\rvert$	$\widehat{P}$	σ_p	δ	Area
10.25MHz ± 8.9kHz	17.81kHz	165μW	46ps	51.4%	0.01931mm^2

Table 4.13: Summary of Key Parameters.

Thanks to the trimming load, the output frequency achieved its best result, reaching an average output frequency of $\mathbf{f}_{\text{out}} = \mathbf{10.25MHz}$, while the absolute frequency deviation, $|\Delta f_{\text{out}}|$, is 17.81kHz.

Continuing with the description of simulation results, the periodic jitter standard deviation, σ_p, is 46ps as shown in Figure 4.82. At the same time, the duty cycle, δ, remains at 51.4%, indicating a nearly balanced waveform. The

Block	Component	Configuration/Value
Res. divider	R_1	$91k\Omega$
	R_2	$28k\Omega$
	R_{tot}	$119k\Omega$
OTA	g_m	$2.1\mu S$
	C_L	$1pF$
VCO	K_{VCO}	$5.25MHz/V$
Feedback	C_s	$147fF \cdot 2 = 294fF$
	C_l	$147fF \cdot 20 = 2.94pF$
	R_l	$100k\Omega$

Table 4.14: Component values and configurations.

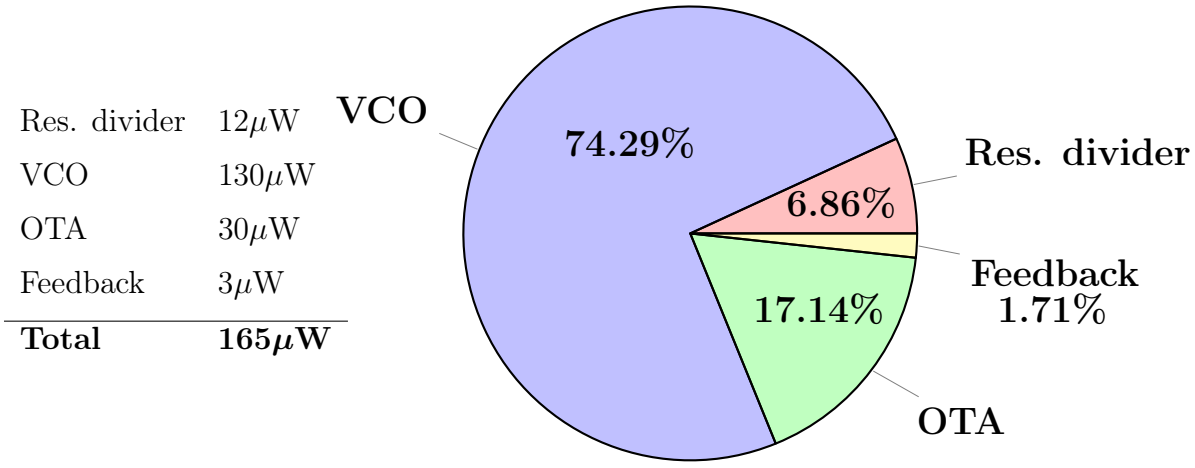


Figure 4.88: Power distribution in the FLL.

design's footprint is 0.01931mm^2, suitable for the Anser EMT medical system's **low area** integration requirements. The final configuration, segmented by the FLL building blocks, is detailed in Table 4.14, encapsulating the design parameters discussed in the preceding Sections of the current Chapter. Figure 4.88 elucidates the average *power* distribution for each block, highlighting the total power consumption. This showcases that the FLL, when

employed as a clock source for the Anser EMT chip, aligns with the initial vision of **minimal area** and **low power** consumption. It meets the stipulated requirements of a periodic jitter standard deviation less than or equal to 100ps and an output frequency better than 10.24MHz.

4.6 Summary of Chapter IV

This chapter explores the design and implementation of a **Frequency-Locked Loop (FLL)** as the clock source for the CTΔΣM. In applications requiring ultra-low-noise performance, such as magnetic sensing for minimally invasive procedures, the clock source's precision and stability are crucial.

Key points discussed include:

- The **specifications** for the CTΔΣM clock source, focusing on parameters like **periodic jitter** and output frequency, as outlined in Table 4.5.
- Analysis of the **clock source design constraints** and the impact of phase noise on the ADC performance, highlighting the importance of minimizing jitter and ensuring a stable output frequency.
- Design of the **Voltage-Controlled Oscillator (VCO)**, including its characteristics, simulation results, and the impact of component choices on the overall performance. The use of *Cadence* for simulation and the **TSMC 65nm** architecture is detailed.
- Examination of the **feedback mechanism**, its components, and configuration, including the derivation of the transfer functions and noise analysis, as shown in Figures 4.69 and 4.70.
- Detailed discussion on the **frequency reference choice** and the comparison between different clock source technologies, and their applicability to the CTΔΣM.
- Implementation of the **final FLL structure**, integrating additional components to achieve full functionality and optimize performance, as depicted in Figures 4.62, 4.78, and 4.67.
- Analysis of **jitter** and **phase noise** in the feedback network and its impact on the overall system performance, with spectrum analysis provided in Figures 4.75 and 4.74.

The chapter concludes by verifying the FLL design through comprehensive simulations, demonstrating its capability to meet the required specifications for the CT$\Delta\Sigma$M. The insights gained from these simulations guide the final design adjustments.

The subsequent chapter will present the **Final Results and Conclusion**, summarizing the overall findings and discussing the implications for future research and applications in the field of ultra-low-noise ADC design.

References

[8] R. Schreier, S. Pavan, and G. C. Temes, *Understanding Delta-Sigma Data Converters*. John Wiley & Sons, Inc., 2017 (Cited on pages 10–15, 31, 32, 36, 37, 41–43, 52).

[20] C. De Berti, P. Malcovati, L. Crespi, and A. Baschirotto, "Colored clock jitter model in audio continuous-time $\Sigma\Delta$ modulators," pp. 1–4, 2015, Accessed: 2023-09-24. DOI: 10.1109/NEWCAS.2015.7182021. [Online]. Available: https://ieeexplore.ieee.org/document/7182021 (Cited on pages 69, 98, 124, 129, 150).

[22] M. Aqamolaei and M. M. Ahmadi, "Modelling an oscillator phase noise whose spectral density contains both $1/f^2$ and $1/f^3$ regions," *Electronics Letters*, vol. 58, no. 4, pp. 139–141, 2022. DOI: https://doi.org/10.1049/ell2.12393. eprint: https://ietresearch.onlinelibrary.wiley.com/doi/pdf/10.1049/ell2.12393. [Online]. Available: https://ietresearch.onlinelibrary.wiley.com/doi/abs/10.1049/ell2.12393 (Cited on pages 82, 98, 124, 129, 150, 151).

[24] A. Khashaba, J. Zhu, N. Pal, M. G. Ahmed, and P. K. Hanumolu, "A 32-mhz, 34-μw temperature-compensated rc oscillator using pulse density modulated resistors," *IEEE Journal of Solid-State Circuits*, vol. 57, no. 5, pp. 1470–1479, 2022. DOI: 10.1109/JSSC.2021.3121014 (Cited on pages 98, 103, 107, 110, 118, 120, 125).

[25] Murata Manufacturing Co., Ltd. "Murata introduces world's smallest low-profile crystal unit size 1.2x1.0x0.30mm." Accessed: 2023-12-17. (2017), [Online]. Available: https://passive-components.eu/murata-introduces-worlds-smallest-low-profile-crystal-unit-size-1-2x1-0x0-30mm/ (visited on 12/17/2023) (Cited on page 99).

[26] B. Razavi, *Design of CMOS Phase-Locked Loops: From Circuit Level to Architecture Level*. Cambridge University Press, 2020. DOI: 10.1017/9781108626200 (Cited on pages 99, 101, 103, 111, 115, 116, 120–122, 124, 127, 130).

[27] B. Razavi, "A general theory of phase noise in electrical oscillators," in *Phase-Locking in High-Performance Systems: From Devices to Architectures*. 2003, pp. 189–204. DOI: 10.1109/9780470545492.ch20 (Cited on pages 100, 103, 105, 117, 120, 124, 133).

[28] A. Hajimiri and T. Lee, "A general theory of phase noise in electrical oscillators," *IEEE Journal of Solid-State Circuits*, vol. 33, no. 2, pp. 179–194, 1998. DOI: 10.1109/4.658619 (Cited on pages 105, 110, 114, 120, 133).

[29] A. Hajimiri, S. Limotyrakis, and T. Lee, "Jitter and phase noise in ring oscillators," *IEEE Journal of Solid-State Circuits*, vol. 34, no. 6, pp. 790–804, 1999. DOI: 10.1109/4.766813 (Cited on pages 105, 110, 115, 116, 124).

[30] A. Devices. "Converting oscillator phase noise to time jitter." Accessed: 24 September 2023. (2009), [Online]. Available: https://www.analog.com/media/en/training-seminars/tutorials/MT-008.pdf (Cited on pages 115, 124, 127, 129, 151).

[31] TSMC. "65nm technology." (2022), [Online]. Available: https://www.tsmc.com/english/dedicatedFoundry/technology/logic/l_65nm (Cited on page 130).

Chapter V:
Final Results and Conclusions

THIS chapter consolidates the findings and analyses related to the design and evaluation of the Anser EMT chip, with a primary focus on the Frequency-Locked Loop (FLL) and the Continuous-Time Delta-Sigma Modulator (CT$\Delta\Sigma$M). The results obtained from the chip's testing are presented, emphasizing the impact of the clock source's phase noise spectrum on the CT$\Delta\Sigma$M's In-Band Noise (IBN). As highlighted in previous sections, the significance of the phase noise spectrum is reiterated in the context of the chip's overall performance. The chapter concludes by discussing the implications of these findings for future designs and research in this domain, referencing the challenges and considerations encountered during the design process.
Before proceeding with the final part of the chapter, a recap of the key points discussed in the previous chapters is provided.

In Chapter II, the creation of the CT$\Delta\Sigma$M model was explored in depth. Starting from scratch, a 2^{nd}-order modulator was developed and validated to meet the given project specifications, achieving a Signal-to-Noise Ratio (SNR) of 67dB. The mathematical modelling using Matlab/Simulink included the derivation of the Signal Transfer Function (STF), Noise Transfer Function (NTF), and Loop Filter $L(s)$. An FIR filter was implemented in the feedback loop to **mitigate** clock jitter effects, resulting in **improved** stability and noise performance.

Chapter III focused on understanding why the FLL's phase noise spectrum prevented achieving the desired SNR. This chapter developed a specific model to study the interaction between the FLL and CT$\Delta\Sigma$M, using an Infinite Impulse Response (IIR) filter to approximate the phase noise profile. A MATLAB script was created to simulate and visualize the phase noise profile, and a detailed analysis was conducted on the limitations of the IIR filter approach, leading to the transition to a **more advanced model**. The impact of phase noise on the SNR of the CT$\Delta\Sigma$M was explored, with MATLAB-based investigations highlighting the effects on in-band noise (IBN).

Chapter IV examined the design and implementation of the FLL as the clock source for the CT$\Delta\Sigma$M. The chapter analyzed the clock source design constraints and the impact of phase noise on ADC performance, emphasizing the importance of minimizing jitter and ensuring a stable output frequency. It detailed the design of the **Voltage-Controlled Oscillator (VCO)**, including its characteristics, simulation results, and the impact of component

choices on overall performance. The final FLL structure was implemented and evaluated, demonstrating its capability to meet the required specifications for the CTΔΣM.

Collectively, these chapters provide a comprehensive overview of the design process, challenges, and solutions implemented in the development of the Anser EMT chip. This final chapter synthesizes these findings, presents the results of the chip's testing, and discusses the implications for future research and design in the field of ultra-low-noise ADCs for medical applications.

5.1 CTΔΣM with different VCOs in the FLL

This chapter elucidates the rationale behind the transition from a 5-stage to a 3-stage VCO, as detailed in Section 4.5.3. While the aforementioned section provided insights into the output spectrum of the FLL, it did not delve into the combined SNR value with the CTΔΣM. The analyses were conducted using the test bench depicted in Figure 5.89.

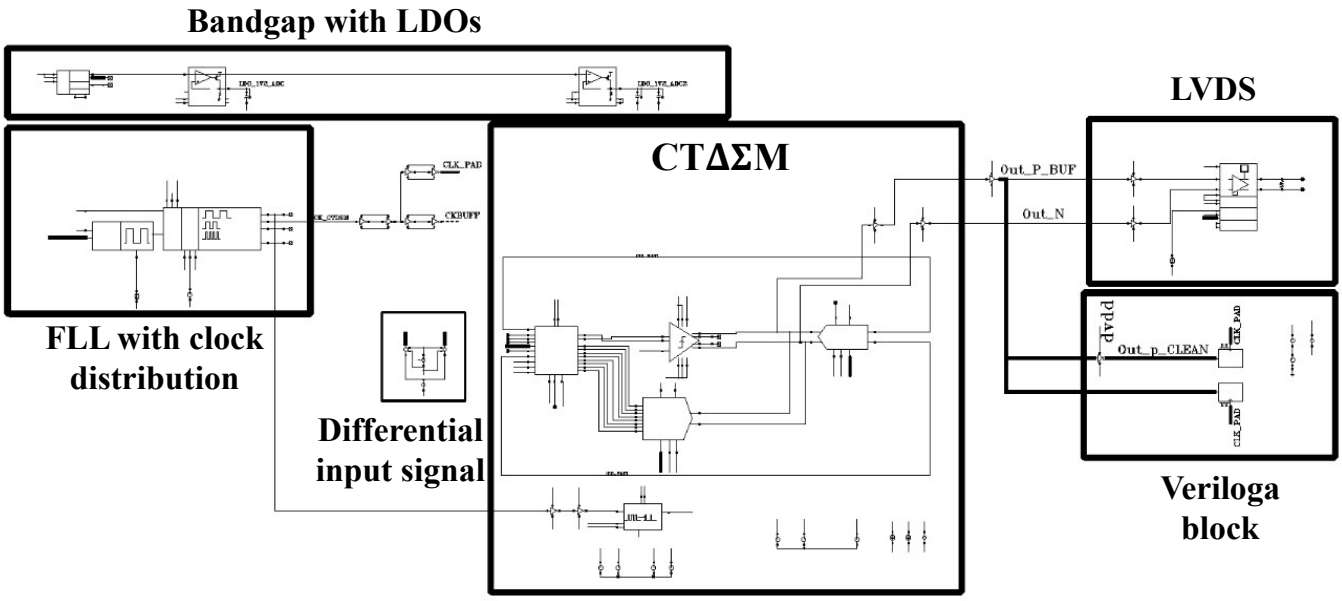


Figure 5.89: Schematic of the Top-Level testing setup in Cadence.

This setup employed a combined **Bandgap** voltage reference with **LDOs** for voltage generation, an integrated **FLL** with its **clock source distribution** block, an **LVDS** [32] for real chip data transmission, and a **VerilogA-coded** block to extract the ADC output. The simulation was executed in transient

mode with noise enabled for both FLL and CTΔΣM to expedite the process. Additionally, both blocks incorporated PEX extraction, provided by the layout engineer as a schematic substitute. This inclusion of PEX extraction ensured that the transient simulations accounted for layout parasitics, enhancing the accuracy and relevance of the simulation results. The following flowchart, see Figure 5.90, illustrates the key operations and parameter settings of the Verilog-A code 5.10 used in the analysis:

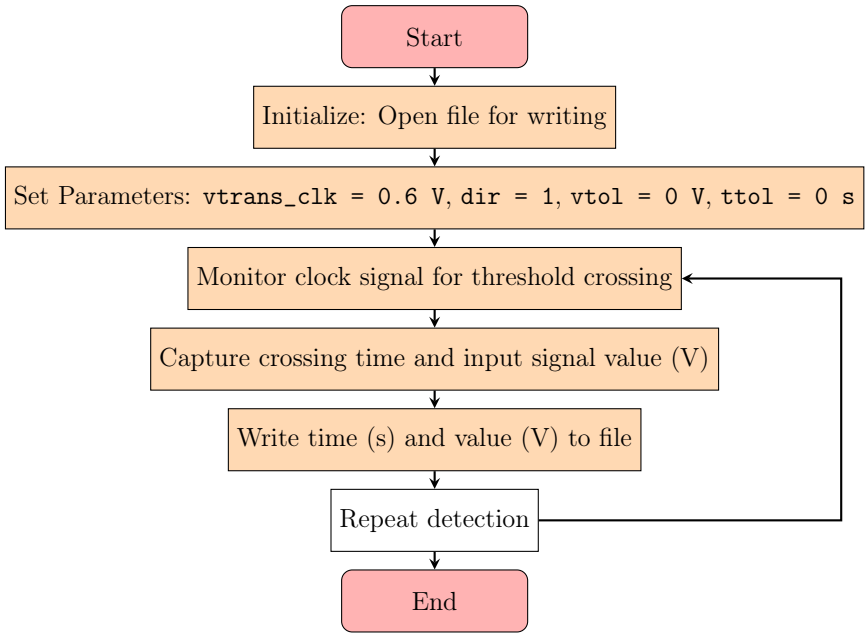


Figure 5.90: Verilog-A flowchart to send data to the workspace.

Script 5.10: Verilog-A code for TWS_VerilogA

```
1  // VerilogA for anser_alessandro, TWS_VerilogA, veriloga
2
3  `include "constants.vams"
4  `include "disciplines.vams"
5
6  module TWS_VerilogA9(in, clk);
7
8  input in;
9  input clk;
10
11 electrical in;
12 electrical clk;
13
14 parameter real    vtrans_clk = 0.6;          // treshold value
15 parameter integer dir = 1;                   /* Rising  edge:  dir =1
16                                              Falling edge:  dir = -1
17                                              both    edges: dir = 0
18                                           */
```

```
19  parameter real     vtol = 0;                  // signal tolerance on the ↩
        clk
20  parameter real     ttol = 0;                  // time tolerance on the clk
21
22  real out;
23  real Cross_point;
24  integer file;
25
26  analog begin
27  @(initial_step) begin
28      file = $fopen("/home/Docs1/alessandro.ferro/linuxhome/ALLokVref_[↩
            N2_15][kBin53][NoiseALL][normal]PEX01clk.txt");
29      end
30  @(cross(V(clk)-vtrans_clk, dir, vtol, ttol)) begin
31      Cross_point = $abstime;                              //$strobe(" Cross ↩
            Time = %e \n", abstime);
32      out = V(in);
33      $fwrite(file, "%e \n", out);                         //$fwrite(" fp, %e↩
             \n", Cross_point);
34      end
35  end
36
37  endmodule
```

Figure 5.91 displays the PSD at the CTΔΣM output, as extracted by the Veriloga block detailed in Script 5.10. Annotations on the image provide the DFT characteristics and simulation specifics, including the 2^{16} data points and the resolution of each DFT bin using a Hanning window. To accentuate the visual differentiation and facilitate a comparative analysis of the spectral densities, the ADC input signal frequency underwent minor adjustments between the two simulations. Additionally, the spectrum was normalized to the peak power of the input signal, where the central bin of the spectral window equates to the value ***sp*** indicated in the figure. This normalization allows for a standardized spectral analysis, revealing that the total in-band noise is positioned 100 dB below the input signal power.

Solely based on Figure 5.91, the two PSDs appear analogous, yet there is a 7-dB difference in SNR. This discrepancy can be attributed to the phase noise profile $L_\varphi(f)$, as illustrated in Figure 4.81. Figure 5.92 demonstrates that despite the 5-stage VCO having a lower periodic jitter standard deviation, it degrades the IBN more significantly (represented by the *blue* plot). In contrast, the 3-stage VCO with its near-white spectrum properties around the fundamental frequency (see Figure 4.81) enhances the ADC's performance, as previously discussed in Equation (3.85).

In conclusion, the system achieves an **SNR** of **67dB**. While this does not meet the desired 70dB or the 80dB observed in the Simulink model, it remains suitable for the Anser EMT application. The insights garnered from this analysis underscore the importance of meticulous VCO selection and its

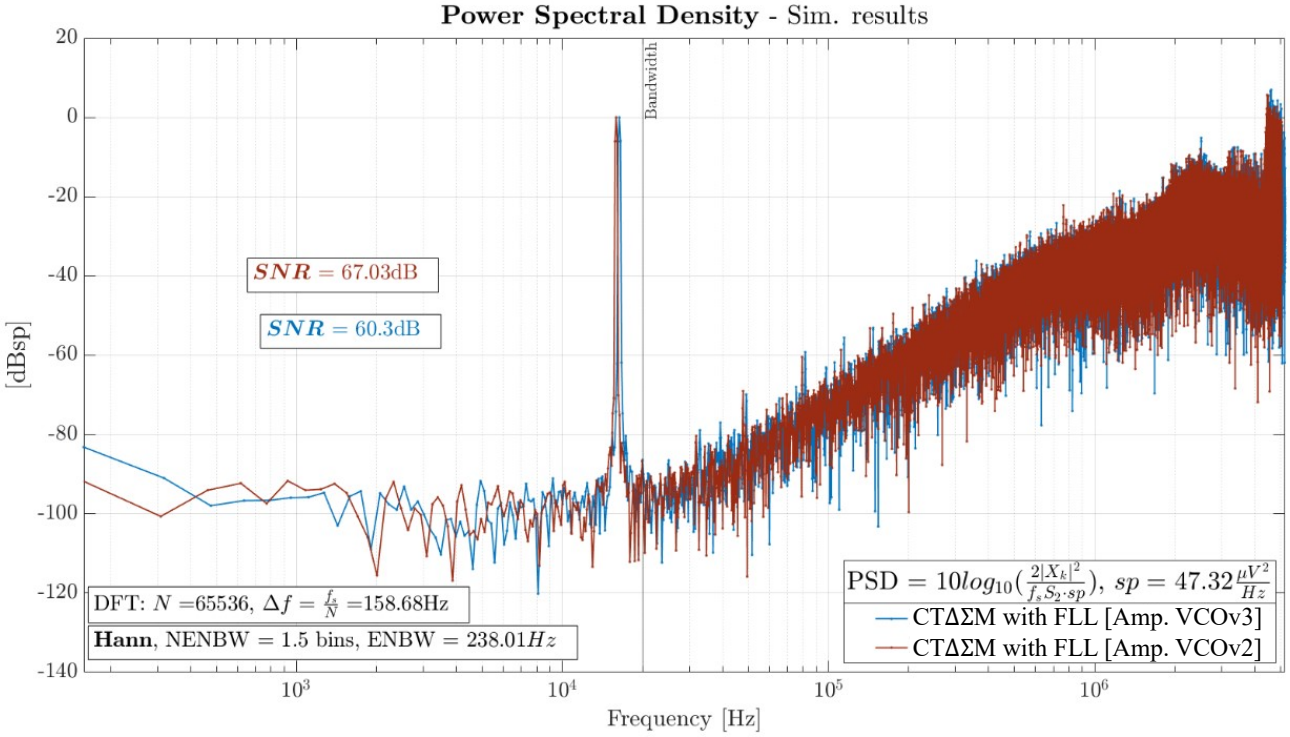


Figure 5.91: Simulated CTΔΣM's PSD comparison.

profound impact on the overall system performance. Future endeavours in this domain should prioritize refining the VCO design and its integration with the CTΔΣM to optimize the SNR and meet the stringent requirements of electromagnetic tracking applications.

5.2 Cadence: Phase noise clock model

A clock model was developed using Cadence's `analoglib` library. The model utilized an ideal block with an added noise profile in dBc/Hz from a real clock source for faster simulations. While the model accurately represented the noise profile in *standalone* measurements, it exhibited ideal behaviour with a **white noise** phase noise spectrum when integrated with other blocks, specifically with CTΔΣM schematic. The adjustable parameter was **limited** to the jitter's standard deviation. Further details are discussed in Appendix B.

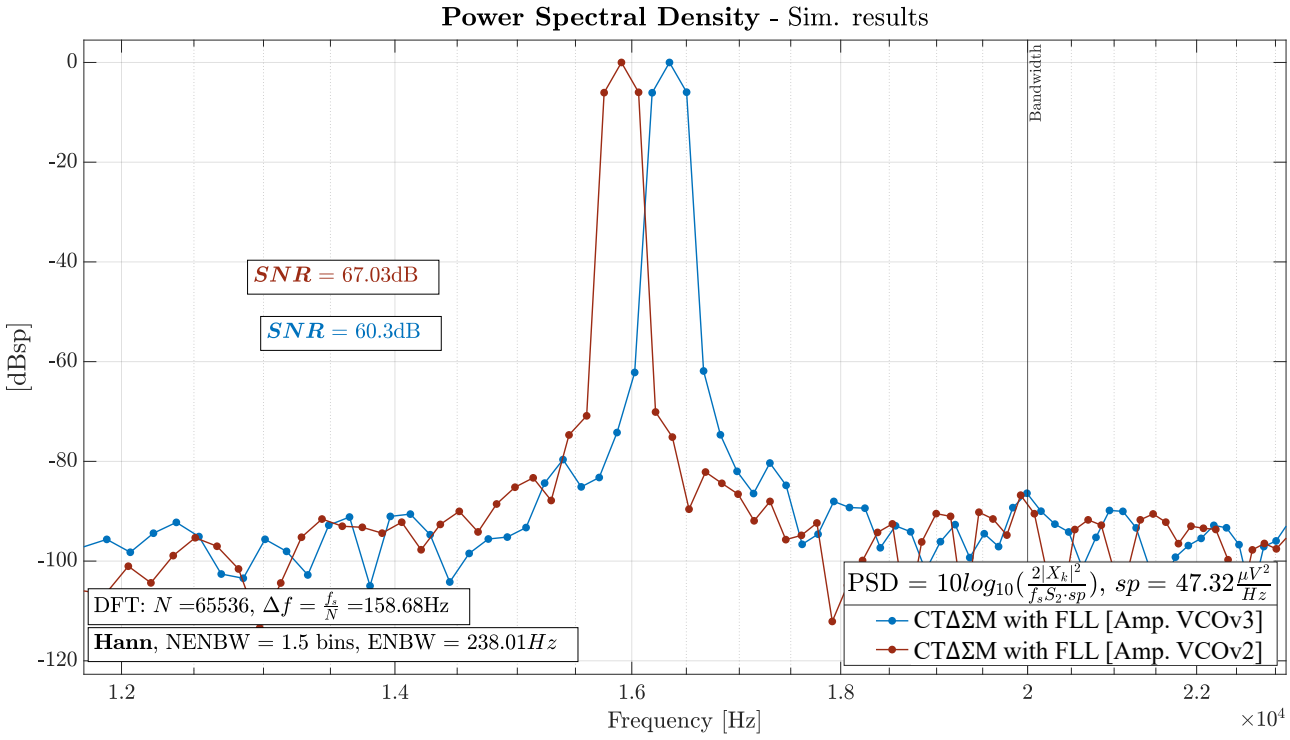


Figure 5.92: CTΔΣM's SNR comparison of Figure 5.91

5.3 Anser EMT Results

The chip's performance was evaluated using its integrated ADC, as depicted in Figure 5.93. This figure illustrates the testing setup, comprising several key components:

- **Power Supply**: Provides consistent and regulated power to the chip.
- **LDO with Reference Voltage Buffer**: The integrated LDO with a reference voltage buffer ensures a stable voltage input.
- **On-Chip Blocks**: Represent the chip's core functional modules.
- **8-tones Signal Generator**: Acts as the chip's input [33], typical for frequency-domain magnetic tracking applications [4].
- **Clock Generator**: A SMA100B from Rohde-Schwarz with a `RMS` absolute jitter standard deviation of **18.7fs** [34].
- **Matlab program**: Essential for data processing tasks.

It is noteworthy that the input bandwidth of the signal generator spans from $20 - 34$kHz. This range not only aligns with typical requirements for magnetic tracking but also surpasses the ADC's initial design bandwidth of 20kHz, highlighting the chip's enhanced capability.

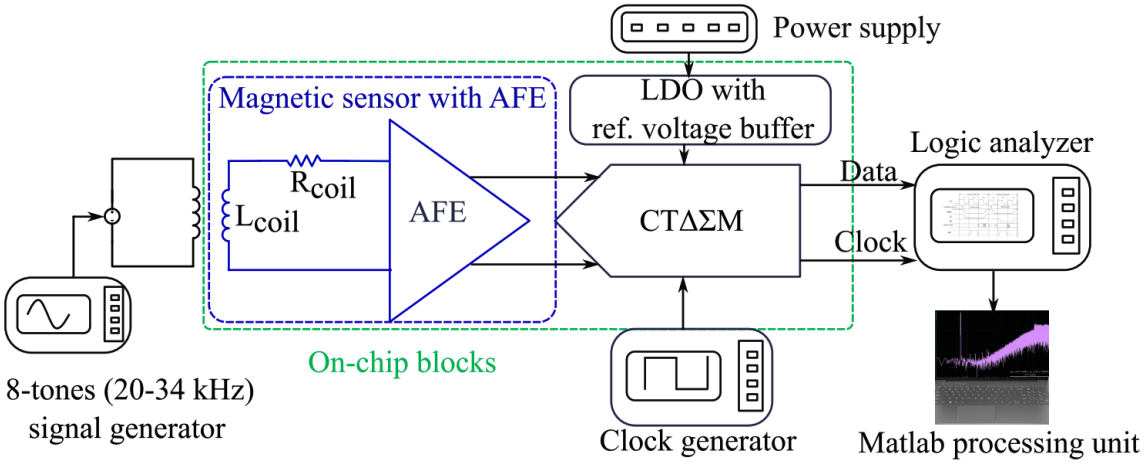


Figure 5.93: Test setup for the CTΔΣM and on-chip magnetic sensor.

The PSD presented in Figure 5.94 was obtained using a 130K-point DFT, corresponding to 2^{17} points. For enhanced clarity and noise reduction, the spectrum was averaged eight times. This spectrum represents the CTΔΣM's output for the input source from the on-chip sensor, captured by the **Digital Discovery** logic analyzer [35]. The CTΔΣM's output data and clock signals were sampled at a rate of $100MS/s$. Notably, the spectrum exhibits characteristics akin to the NTF detailed in Figure 3.53 of Chapter III. Upon closer inspection, a pronounced high-frequency tone is evident, resulting from applying chopping to an Analog Front-End (AFE) Capacitively Coupled Instrument Amplifier (CCIA). This tone was subsequently *filtered* externally.

Figure 5.95 provides an enlarged perspective of Figure 5.94, highlighting the extracted tones. These tones are instrumental in determining the magnetic sensor's position and orientation (pose), a vital component in the Anser EMT electromagnetic tracking system [4]. The exceptionally low jitter of the **Clock Generator**, an SMA100B from Rohde-Schwarz [34], ensures a white frequency phase noise profile. This characteristic allows the CTΔΣM to operate at its peak performance, even when the eight tones are transmitted closely in frequency. This optimal performance is fully realized as described by Equation (2.28) in Chapter II. Such a spectrum cannot be guaranteed if the clock source had a phase noise profile $L_{\varphi}(f)$ as described in Equa-

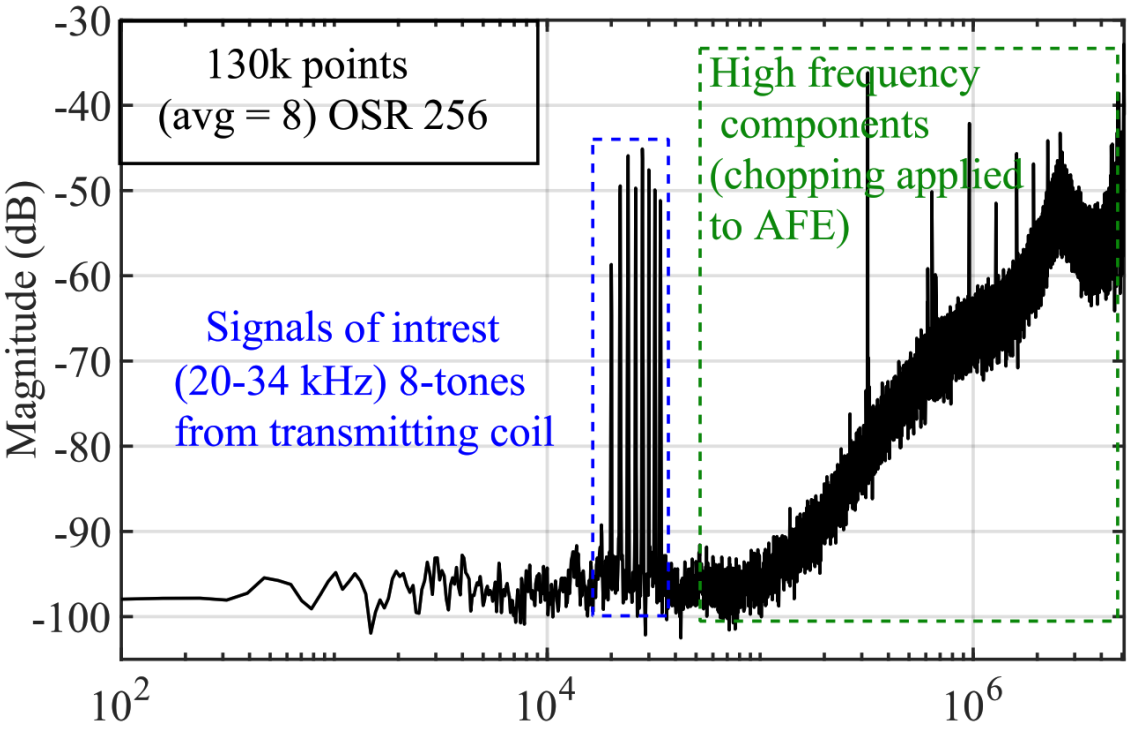


Figure 5.94: Magnitude plot showing the performance of the CTΔΣM with magnetic sensor with the time-varying 8-tone (20-34 kHz) magnetic field.

tion (3.85).

5.3.1 CTΔΣM and Integrated Clock Results

An assessment of the phase noise spectrum's influence on the ADC's IBN was conducted using a configuration similar to that delineated in the preceding Section. This setup was enhanced by the inclusion of a DSOX6004A Oscilloscope [36] to directly capture the output waveforms from both the integrated chip's *internal* and *external* clock sources [34].

Figure 5.96 delineates the experimental arrangement at the Marconi Lab within Tyndall National Institute, where the PSD measurements were conducted. The setup facilitated the simultaneous acquisition of spectra from both clock sources, represented by the *pink* plots on the oscilloscope's monitor. The Anser EMT testing board allows to extract the internal clock signal directly from the fully integrated chip, which was previously detailed in Figure 1.2 of Chapter I.

The PSDs delineated in Figure 5.96 are further represented in Figure 5.97

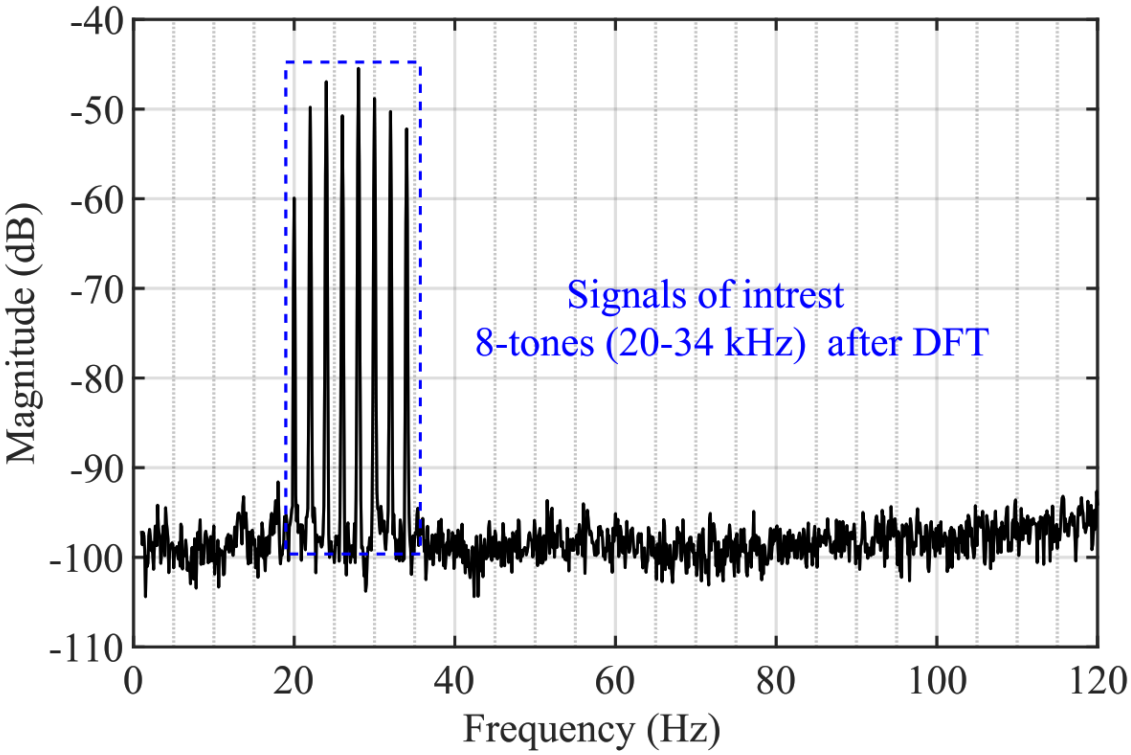


Figure 5.95: Detailed view of the PSD from Figure 5.94

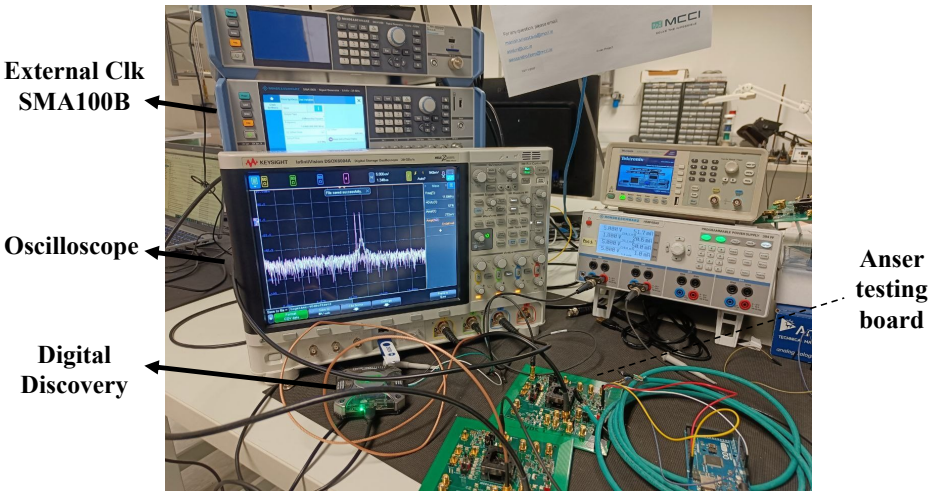


Figure 5.96: Detailed view of the testing setup at Tyndall National Institute.

and plot using Matlab for an enhanced frequency domain visualisation, facilitating the distinction of spectra with white phase noise, indicated by the

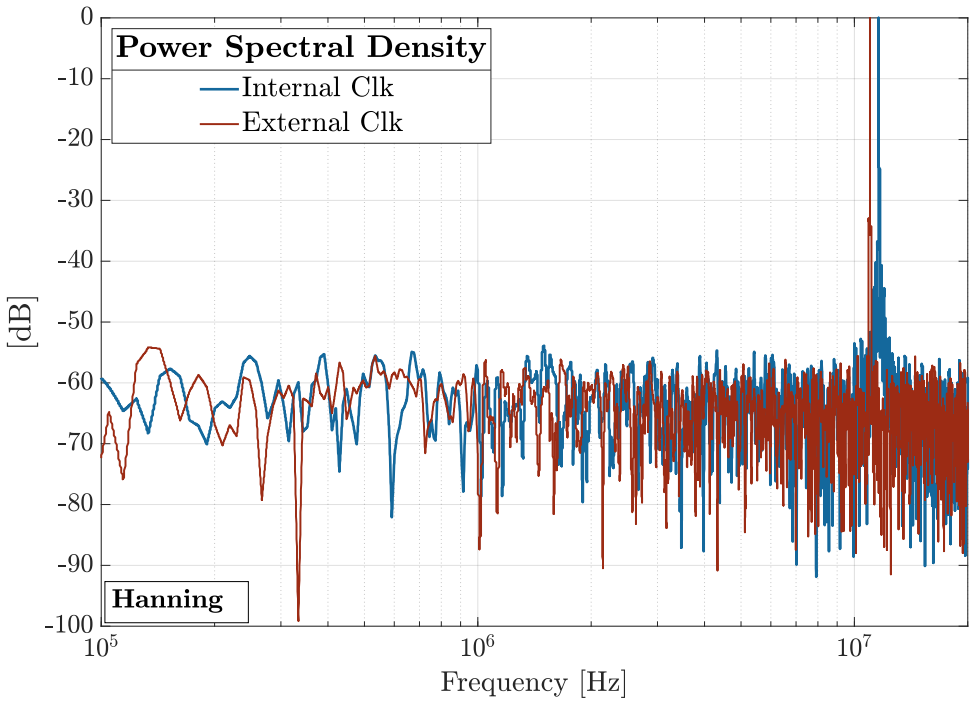


Figure 5.97: Oscilloscope clock's spectra comparison.

red plot, against non-white noise, as seen in the *blue* plot. Note that the *red* plot nearly exemplifies an ideal PSD, characterised by a Dirac delta-like peak at the output frequency (for elucidation, refer to Equation (4.100) in Chapter IV), assuring the best performance for the CT$\Delta\Sigma$M.

By isolating the PSDs as depicted in Figure 5.98, the disparity between the expected and observed frequencies becomes more pronounced. Both PSDs graphs conclude at the primary harmonic; therefore, it is feasible to augment the PSD resolution proximate to its output frequency by trimming the oscilloscope setting. Figure 5.100 shows this visualisation improvement by analysing the internal clock PSD and its related phase noise spectrum around the output frequency. However, the observed output frequency of 11.85 MHz, as opposed to the ideal 10.25 MHz predicted by Cadence simulations, known that the analysis was conducted without enabling the trimming capacitive circuit to test how different the standard output frequency in the circuit compared to the ideal one tested on Cadence. The difference can be attributed to the inherent variances in the physical implementation of the circuitry. Such discrepancies are not uncommon in practice and can arise from factors such as parasitic capacitance in the manufacturing process. Despite this frequency shift, phase noise analysis on the ADC's IBN remains valid, as the primary

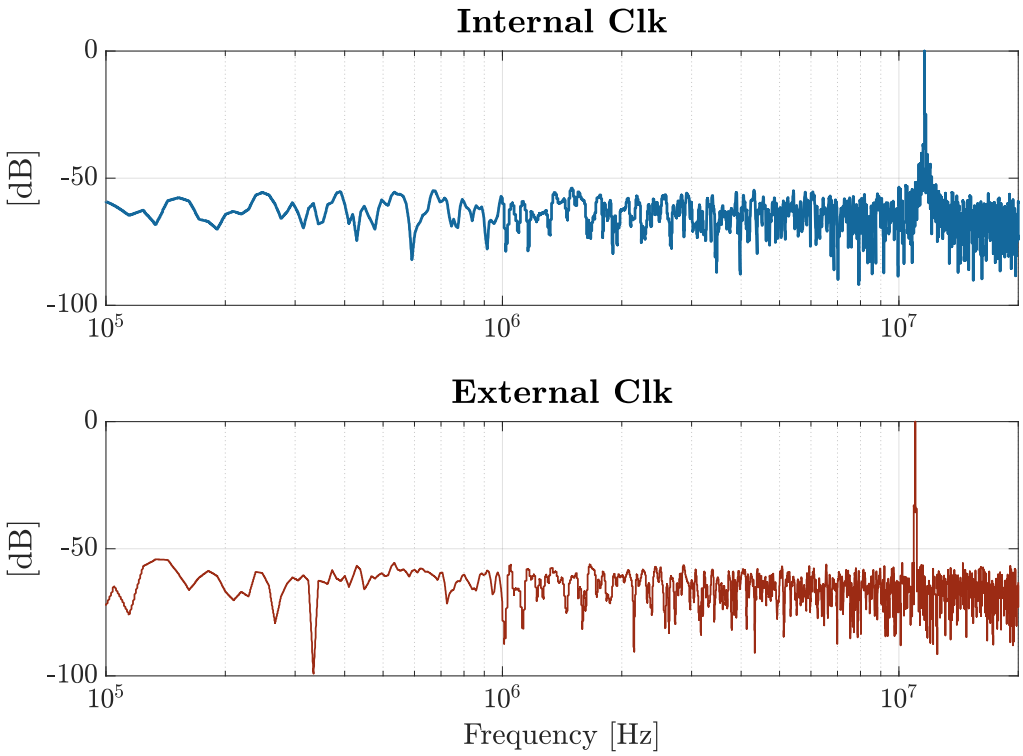


Figure 5.98: Clock PSD from Figure 5.97.

interest is relative noise characteristics rather than absolute frequency values.

Unfortunately, the direct measurement of the periodic jitter standard deviation in-chip was not conducted, as the oscilloscope available through the MCCI package did not include the specific license required *specifically* for periodic jitter measurement functionality. This additional license, necessary for enabling detailed jitter analysis, was not a standard component of the provided equipment. Despite this, the absence of these specific measurements does *not compromise* the overall findings regarding the phase noise's impact on the ADC's IBN. Future work could aim to secure the appropriate tools to fill this gap in the data set.

Referencing the CTΔΣM's output PSD from Figure 3.53 in Chapter III, Figure 5.100 demonstrates the phase noise profile of the clock source **shifted** to the ADC's *input signal* frequency at 16.71 kHz. The Digital Discovery tool [35] was operated at maximum resolution, with the DFT for the CTΔΣM's output computed with $N = 2^{19}$ points, yielding a frequency band

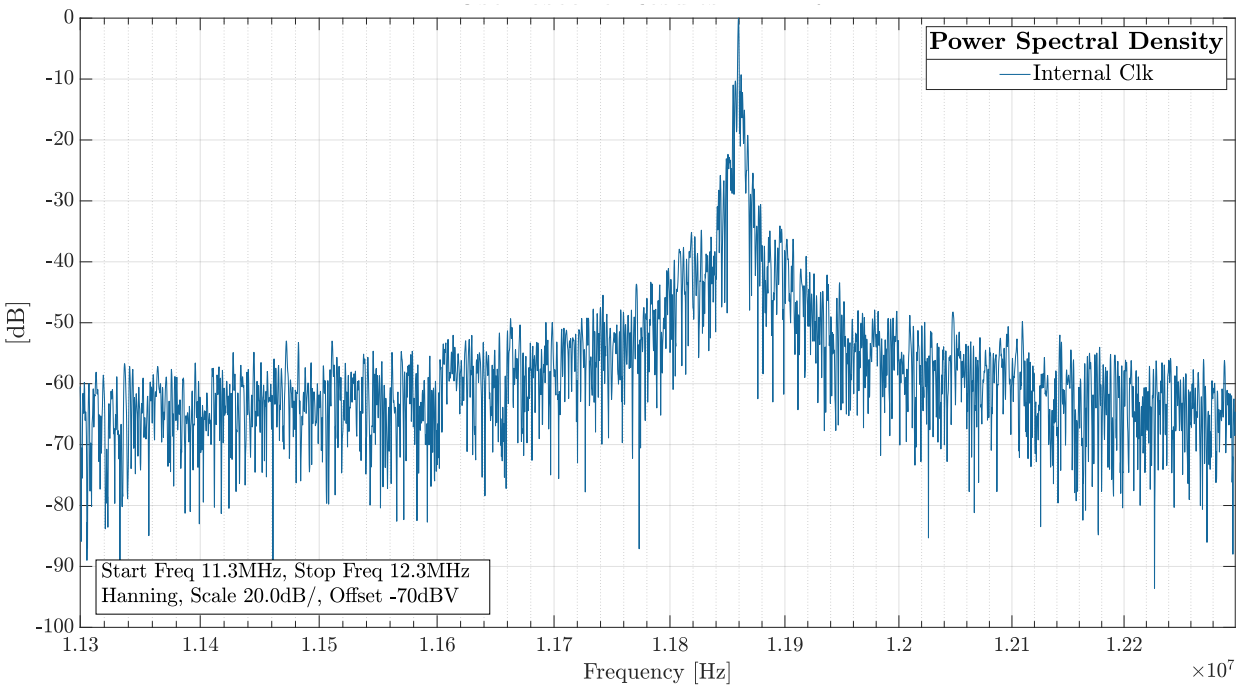


Figure 5.99: Internal's clock PSD from Figure 5.97

resolution number defined in Equation 5.116.

$$n^{\circ}BW_{\text{bins}} = \frac{BW}{\Delta f} = \frac{20kHz}{f_s}N \tag{5.116}$$

The number of in-band bins scales with the DFT resolution, N, providing 1024 bins for spectrum analysis and 1021 for phase noise scrutiny, resulting in a resolution of 33.9 Hz per bin. An ideal resolution of 1 Hz per bin would facilitate a comprehensive visualisation of the ADC's phase noise spectrum. Therefore, the IBN equation [20], [22] can be redefined as:

$$\text{IBN}\,(f) \approx \frac{A^2}{2}\left(\frac{f_{in}}{f_s}\right)^2 L_{\varphi_{DS}}\,(f - f_{in}) \tag{5.117}$$

This analysis elucidates the complex interplay between phase noise and ADC in-band noise. Despite some practical limitations, the results contribute to the foundational knowledge required for future advancements in integrated clock designs and ADC performance. These findings are instrumental in developing precise and reliable electromagnetic tracking systems in medical technology.

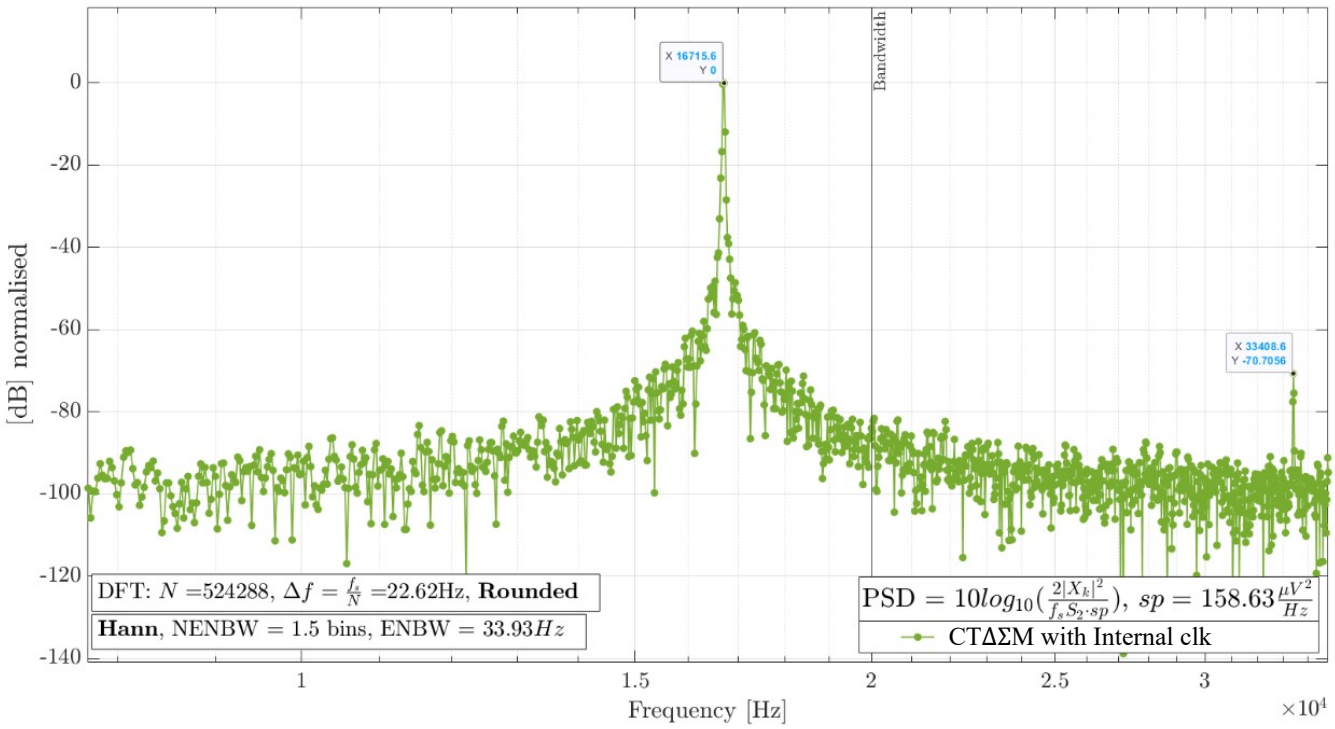


Figure 5.100: CTΔΣM's IBN PSD plot.

5.4 Conclusions and Future Work

The design journey for the Anser EMT chip's **Frequency-Locked Loop (FLL)**, modelled after the **CTΔΣM** Matlab/Simulink **model**, appeared straightforward at the outset. The FLL met the stipulated specifications, as referenced in Table 4.6. However, deeper research into the literature revealed a well-documented challenge: the clock source's phase noise spectrum severely constrains the CTΔΣM's In-Band Noise (IBN). This realization, although seemingly evident from Equation (3.85) [22], was initially overlooked by our team. A detailed exploration of this phenomenon can be found in Section 3.4.

Given this newfound understanding, it became evident that while the FLL is apt for specific requirements, it might not be the optimal choice for the CTΔΣM. A Phase-Locked Loop (PLL) might serve better [30], [37], but introduces the challenge of requiring an external clock source. This poses a dilemma, especially when the design goals are geared towards **low-area** and **low-noise** solutions for the EMT chip for medical interventions.

The results of this research have demonstrated significant advancements in the design of ultra-low-noise ADCs for medical applications. The primary contributions include:

- Development and validation of a 2^{nd}-order CT$\Delta\Sigma$M model from scratch, achieving a **Signal-to-Noise Ratio (SNR)** of 67dB. This was in line with the project's specifications, although not reaching the ideal 70dB or 80dB observed in Simulink models, it is suitable for the Anser EMT application.

- Implementation of an FIR filter in the feedback loop to mitigate clock jitter effects, resulting in improved stability and noise performance.

- Detailed phase noise analysis, elucidating why the FLL's phase noise spectrum hindered achieving the desired SNR. This analysis provided a framework for future improvements in VCO design and its integration with the CT$\Delta\Sigma$M.

These findings underscore the importance of meticulous VCO selection and its profound impact on the overall system performance. Future endeavours in this domain should prioritize refining the VCO design and its integration with the CT$\Delta\Sigma$M to optimize the SNR and meet the stringent requirements of electromagnetic tracking applications.

In closing, this thesis has not only contributed to the academic discourse on IC design but also highlighted practical considerations for the development of medical devices for in-vivo sensing. The challenges encountered serve as a testament to the dynamic nature of engineering research, where theoretical understanding must continually be reconciled with practical design constraints.

Future research should focus on:

- Exploring alternative clock source technologies, such as PLLs, to further reduce phase noise and improve the SNR of the CT$\Delta\Sigma$M.

- Investigating the integration of adaptive filtering techniques to dynamically compensate for phase noise variations in real-time.

- Conducting extensive in-vivo testing to validate the performance of the Anser EMT chip in clinical settings, ensuring its reliability and accuracy in practical applications.

Ultimately, the advancements and insights gained from this research lay a strong foundation for future innovations in the field of ultra-low-noise ADCs. By addressing the identified challenges and exploring the proposed research directions, the potential for significant improvements in the precision and reliability of medical devices is substantial, paving the way for enhanced patient care in minimally invasive procedures.

References

[4] H. Jaeger, A. Franz, K. donoghue, *et al.*, "Anser emt the first open-source electromagnetic tracking platform for image-guided interventions," *International Journal of Computer Assisted Radiology and Surgery*, vol. 12, Mar. 2017. DOI: 10.1007/s11548-017-1568-7 (Cited on pages 3, 144, 145).

[20] C. De Berti, P. Malcovati, L. Crespi, and A. Baschirotto, "Colored clock jitter model in audio continuous-time $\Sigma\Delta$ modulators," pp. 1–4, 2015, Accessed: 2023-09-24. DOI: 10.1109/NEWCAS.2015.7182021. [Online]. Available: https://ieeexplore.ieee.org/document/7182021 (Cited on pages 69, 98, 124, 129, 150).

[22] M. Aqamolaei and M. M. Ahmadi, "Modelling an oscillator phase noise whose spectral density contains both $1/f^2$ and $1/f^3$ regions," *Electronics Letters*, vol. 58, no. 4, pp. 139–141, 2022. DOI: https://doi.org/10.1049/ell2.12393. eprint: https://ietresearch.onlinelibrary.wiley.com/doi/pdf/10.1049/ell2.12393. [Online]. Available: https://ietresearch.onlinelibrary.wiley.com/doi/abs/10.1049/ell2.12393 (Cited on pages 82, 98, 124, 129, 150, 151).

[30] A. Devices. "Converting oscillator phase noise to time jitter." Accessed: 24 September 2023. (2009), [Online]. Available: https://www.analog.com/media/en/training-seminars/tutorials/MT-008.pdf (Cited on pages 115, 124, 127, 129, 151).

[32] Wikipedia. "Low-voltage differential signaling." Italian. Accessed: 2023-11-26, Wikipedia. (2023), [Online]. Available: https://it.wikipedia.org/wiki/Low-voltage_differential_signaling (visited on 11/26/2023) (Cited on page 140).

[33] K. Technologies, *33500b and 33600a series trueform waveform*, https://www.keysight.com/it/en/assets/7018-05928/data-sheets/5992-2572.pdf, Datasheet, Accessed: 19 October 2023, 2023 (Cited on page 144).

[34] Rohde-Schwarz, *Sma100b signal generator*, https://www.rohde-schwarz.com/it/prodotti/misura-e-collaudo/generatori-di-segnali-analogici/rs-sma100b-rf-and-microwave-signal-generator_63493-427776.html, Accessed: 19 October 2023, 2023. [Online]. Available: https://scdn.rohde-schwarz.com/ur/pws/dl_downloads/pdm/cl_brochures_and_datasheets/specifications/5215_1018_22/SMA100B_specs_en_5215-1018-22_v0704.pdf (Cited on pages 144–146).

[35] Digilent, *Digital discovery: Start*, https://digilent.com/reference/test-and-measurement/digital-discovery/start, Accessed: 19 October 2023, 2023 (Cited on pages 145, 149).

[36] Keysight Technologies, *Dsox6004a oscilloscope - 1GHz to 6GHz, 4 analog channels*, https://www.keysight.com/it/en/product/DSOX6004A/oscilloscope-1-ghz-6-ghz-4-analog-channels.html, Accessed: 2023-09-24, 2023 (Cited on page 146).

[37] H. Zheng, "Mixed signal compensation of sampling errors in adcs due to noisy dpll clock sources," Available from: https://cora.ucc.ie/items/667c56e2-462d-4a9b-a2ec-0574e2f2f20e, M.S. thesis, University College Cork, 2021 (Cited on page 151).

Appendix

A. Discrete Fourier transform

The Fourier transform implies a deep understanding of the spectrum evaluation algorithm; it involves a trade-off between the simulation's time duration and spectral precision tolerated in the design. Firstly, it is necessary to underline the differences between the various use of a Fourier transform. Based on nature, *continuous* or *discrete*, to be analysed, it is possible to use three types of transforms:

1. **Time - domain** Fourier transform $\mathbf{X}(f)$.

   The first one, the Fourier Transform (**FT**), requires a real-continuous function $x(t) : \mathbb{R} \mapsto \mathbb{C}$, and it produces a complex continuous function in the *frequency* domain, $\mathscr{F}\{x(t)\} = \int_{-\infty}^{+\infty} x(t)e^{j2\pi ft}\,dt = \mathbf{X}(f) \in \mathbb{C}$. All of the transform's properties are well-defined [38].

2. **Discrete - Time** Fourier transform $\mathbf{X}_{\frac{1}{\mathbf{T_s}}}(f)$.

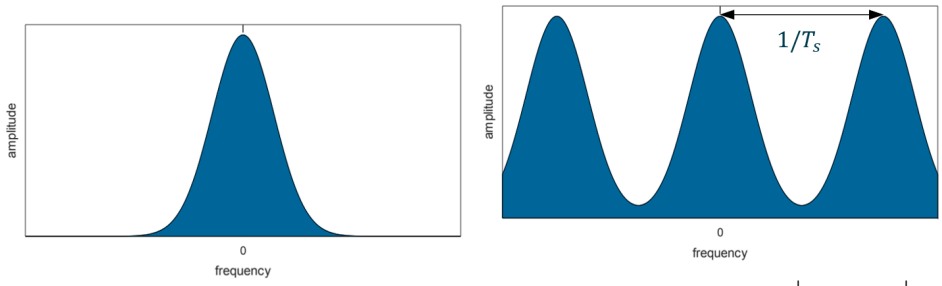
   The second type of transform, the Discrete-Time Fourier Transform (**DTFT**), is used to analyse a **discrete-time** signal $x[n]$ with a period of T_s seconds. It produces a complex continuous function in the frequency domain, $\mathscr{F}\{x[n]\} = \sum_{n=-\infty}^{\infty} x[n]e^{-j2\pi fnT_s} = \mathbf{X}_{\frac{1}{\mathbf{T_s}}}(f) \in \mathbb{C}$. However, this transform is *impractical* for digital signal processing because it requires **unlimited data**.

3. **Discrete** Fourier transform $\mathbf{X}[k]$.

   The third type of transform, the Discrete Fourier Transform (**DFT**), is used to analyse a **finite-length** sequence of **discrete-time samples**, $x[0], x[1], ..., x[N-1]$. It produces a **finite series** of complex numbers in the frequency domain, $\mathscr{F}_k\{x[n]\} = \sum_{n=0}^{N-1} x[n]e^{-j2\pi kn/N} = \mathbf{X}[k] \in \mathbb{C}$, where $k = 0, 1, ..., N-1$.

A.1 FT, DTFT and DFT

Fourier Transform and Discrete-Time Fourier Transforms are *continuous functions*. The FT maps a continuous time-domain signal, $x(t)$ to a continuous frequency-domain representation, $\mathbf{X}(f)$, while the DTFT maps a discrete time-domain signal, $x[n]$ with period T_s, to a continuous frequency-domain representation $\mathbf{X}_{\frac{1}{\mathbf{T_s}}}(f)$, as shown in figure 1a and 1b:



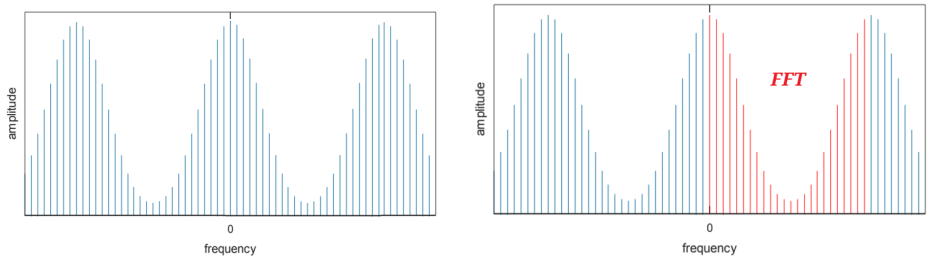
(a) FT magnitude $|\mathbf{X}(f)|$ (b) DTFT magnitude $\left|\mathbf{X}_{\frac{1}{\mathbf{T_s}}}(f)\right|$

Figure 1: Comparison of FT and DTFT Magnitudes

In addition, the DTFT is defined for both *finite-length* and *infinite-length* discrete-time signals, and it is related to the FT by Poisson summation formula[39]:

$$\mathbf{X}_{\frac{1}{\mathbf{T_s}}}(f) = \sum_{k=-\infty}^{+\infty} \mathbf{X}\left(f + \frac{k}{T_s}\right) \qquad \text{(Eq. 1.1)}$$

The Discrete Fourier Transform (DFT), instead, is a **sampled version** of the DTFT and is computed using a **finite number** of input signal **samples**.



(a) DFT magnitude $|\mathbf{X}[k]|$ (b) FFT magnitude $\left|\mathbf{X}[0, \frac{N}{2} - 1]\right|$

Figure 2: Comparison of DFT and FFT Magnitudes

The DFT can be viewed as a discrete-time Fourier series with a **period** of N samples, as shown in Fig. 2a.

$$\mathbf{X}_{\frac{1}{T_s}}(f) = \frac{1}{N} \sum_{k=-\infty}^{+\infty} \mathbf{X}[k]\delta\left(f - \frac{k}{NT_s}\right) \qquad \text{(Eq. 1.2)}$$

The equation [40] Eq. 1.2 states that the frequency-domain representation of a discrete-time signal, obtained using the DTFT, can be expressed as a sum

of impulse functions centred at frequencies that are integer multiples of the fundamental frequency $\frac{1}{NT_s}$, where T_s is the sampling period and N is the length of the DFT. The amplitude of each impulse function is proportional to the value of the DFT coefficient $\mathbf{X}[k]$ at the corresponding frequency. This relationship is important because it allows us to calculate the *frequency-domain representation* of a **discrete-time signal** using the DFT, which is much **faster** to compute than the DTFT.

A.2 Fast Fourier transform FFT

The DFT can be efficiently **computed** using the Fast Fourier Transform (**FFT**) algorithm, which has a computational complexity of $\mathcal{O}(N \log_2 N)$ [41]. The choice of which transform to use depends on the specific application and the trade-off between *time duration* and *spectral precision*.

A.2.1 From DFT to FFT

Since the time series of the digitised input signal is always real[42], the output array of the DFT follows certain **symmetries**, which can be exploited to reduce computational cost. Specifically, the following properties hold:

- if N is even, then $\mathbf{X}[0],\ \mathbf{X}\left[\frac{N}{2}\right] \in \mathbb{R}$

- $\mathbf{X}[N-m] = \mathbf{X}^*[m], \quad m \in \left[0, \frac{N}{2} - 1\right] \in \mathbb{N}$ (Eq. 2.1)

Where N is the maximum value for a finite-length sequence of discrete-time samples $x[n]$, and $\mathbf{X}^*[\cdot]$ denotes the complex conjugate of the DFT. The upper half of the transformation is **redundant**, and there is no need to compute since it can not be evaluated **efficiently** by the FFTW package used in Matlab/Simulink [43]. The terms DTF and FFT will be used **interchangeably**, even though there is a substantial difference between the two.

A.3 Window function

The FFT algorithm requires that the input signal be finite in length, and this can lead to **spectral leakage** or loss of frequency resolution if the signal does *not* have **integer periods**. One way to mitigate these issues is to use window functions applied to the signal **before** performing the FFT. A

window function is a mathematical function multiplied with the input signal to reduce the effects of spectral leakage and improve frequency resolution. Various types of window functions are available, such as rectangular, *Hanning*, Hamming, and Blackman, each with advantages and disadvantages [44].

When dealing with a discrete finite-length signal $x[nT_s]$ and a window function $w[nT_s]$, it is essential to remember that windowing a signal involves **multiplication** in the *time domain* and **periodic convolution** in the *frequency domain*. The Fourier transform of the windowed signal can be expressed as:

$$\mathscr{F}_k\{x[nT_s]w[nT_s]\} = \mathbf{X}[k] \circledast_p \mathbf{W}[k] \qquad \text{(Eq. 3.1)}$$

Here, $\circledast_p$ denotes the periodic convolution operator, which takes into account the periodic extension of the Discrete Fourier transforms $\mathbf{X}[k]$ and $\mathbf{W}[k]$. The periodic convolution is computed as follows:

$$\mathbf{X}[k] \circledast_p \mathbf{W}[k] = \frac{1}{N} \sum_{n=0}^{N-1} \mathbf{X}[n]\mathbf{W}[(k-n) \bmod N] \qquad \text{(Eq. 3.2)}$$

Where N is the length of the input signal, and mod denotes the modulo operator.

A.3.1 NENBW and ENBW

In the context of the Discrete Fourier Transform (DFT), the normalized equivalent noise bandwidth (NENBW) of a window function is a measure of the **effective width** of the window's frequency response. It is commonly used to quantify the **impact** of the window on the signal's power *spectrum*. The NENBW can be defined as the *equivalent rectangular bandwidth* (ERB) of a rectangular window with the same power spectrum as the given window function. When used to sample a white noise signal, the ERB is the width of a rectangular window that would have the same energy as the given window. In other words, the NENBW measures how much **noise** the windowing process **introduces**. Mathematically, the NENBW can be calculated using the following equation:

$$\int_0^{\frac{BW}{2}} |\mathbf{W}(f)|^2 df = \text{NENBW} \cdot |\mathbf{W}(f_0)|^2 \qquad \text{(Eq. 3.3)}$$

where $\mathbf{W}(f)$ is the Fourier transform of the window function, BW is the signal bandwidth, f_0 is the center frequency of the window, and $|\mathbf{W}(f_0)|^2$ is the maximum value of $|\mathbf{W}(f)|^2$ in the frequency domain. The integral on the

left-hand side of the equation represents the power spectrum of the window function up to half the signal bandwidth, while the right-hand side represents the power obtained with an equivalent rectangular window, as shown in Fig. 3. The NENBW is then calculated as the ratio of these two powers.

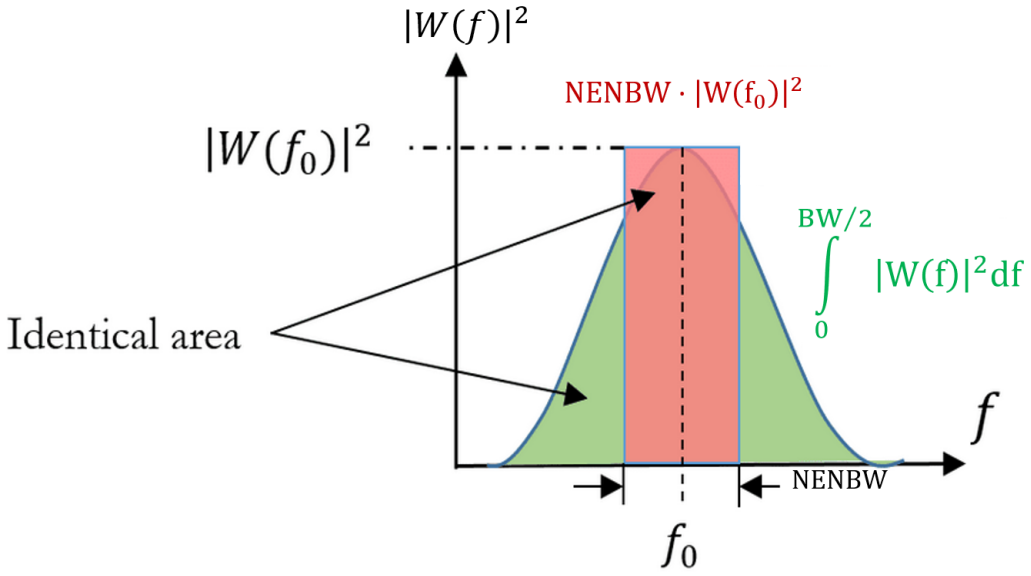


Figure 3: NENBW example

A normalized window function has a NENBW of 1, meaning it *does not* introduce any *additional* noise. In practice, however, most window functions have a NENBW greater than 1, which raises some extra noise. The NENBW is a vital parameter when selecting a window function for a particular application, especially in signal processing and communications systems. A **low** NEBW indicates that a window function has good **frequency selectivity** and minimises the amount of noise introduced by the windowing process, which can improve the overall performance of different window functions.

In the following equation, $w(t)$ is a window function of length N, $\mathbf{W}(f)$ represents the continuous Fourier transform of $w(t)$, and $\mathbf{W}[k]$ represents the discrete Fourier transform (DFT) of $w(t)$ at index k. NENBW is the normalized equivalent noise bandwidth, BW is the signal's bandwidth, and f_0 and k_0 represent the DFT's centre frequency and index:

$$NENBW = \frac{\int_0^{\frac{BW}{2}} |\mathbf{W}(f)|^2 df}{|\mathbf{W}(f_0)|^2} \xrightarrow{\mathscr{F}_k} \frac{\sum_{k=0}^{N-1} |\mathbf{W}[k]|^2}{|\mathbf{W}[k_0]|^2} \qquad \text{(Eq. 3.4)}$$

It can be converted from the *frequency domain* to the *time domain* using

Parseval's theorem[45]. The NENBW can be expressed in terms of the window function $w[n]$ of length N in the time domain, as shown in the following equation:

$$NENBW = N\frac{\sum_{k=0}^{N-1}|w[n]|^2}{|\sum_{k=0}^{N-1}w[n]|^2} = N\frac{S_2}{S_1^2}, \quad n \in [0, N] \in \mathbb{N} \qquad \text{(Eq. 3.5)}$$

Where $w[n]$ represents the window function in the time domain at time index n, and S_1 and S_2 are the first and second moments of the window function, respectively. The first moment S_1 equals the sum of the window function values, while the second moment S_2 equals the sum of the squared magnitudes of the window function values. Using these moments, the NENBW can be **calculated** directly from the window function in the **time domain**, and the resulting **scale** is **FFT bins** as $n \in [0, N] \in \mathbb{N}$. Thus, the NENBW is a fraction of the FFT size and represents the number of FFT bins that contain the same amount of noise power as a rectangular window of the same length, see Fig. 3.

The formula for NENBW, shown in Eq. Eq. 3.5, provides an alternative method to compute the same value as the built-in MATLAB function `enbw`:

```
%Noise bandwidth
S1 = sum(window); S2 = sum(window.^2);
%NORMALIZED EQUIVALENT Noise bandwidth
%[Bins]
NENBW = N*S2/(S1^2);
%Matlab function
NENBW = enbw(window)
```

Figure 4: NENBW formula for a window function in MATLAB

To illustrate, let's take the example of a *Hanning window*, defined as an array of length equal to $N = 2^{16}$ as shown in Fig. 5.

```
% Example usage:
enbw(hann(2^16, 'periodic'))
% Output:
ans = 1.5000
```

Figure 5: Calculating the NENBW for a Hanning window of length $N = 2^{16}$.

This code snippet indicates that the normalized equivalent noise bandwidth of the **Hanning window** is equal to **1.5** of the FFT **bins**.

ENBW

The Effective Noise Bandwidth of a window function is the *bandwidth* of a rectangular window that produces the *same amount* of noise power as the given window. It measures the effective bandwidth of the window in terms of noise power. It provides the same information as the NENBW, but it is given in **units** of **frequency** $[Hz]$:

$$ENBW = NENBW \cdot \Delta f = f_s \frac{S_2}{S_1^2}, \quad \Delta f = \frac{f_s}{N} \qquad \text{(Eq. 3.6)}$$

In MATLAB, the ENBW can be calculated using the formula in Eq. 3.6 or with the built-in function `enbw(window, fs)`, where the sampling frequency f_s must be specified:

```
%Noise bandwidth
S1 = sum(window); S2 = sum(window.^2);
%EFFECTIVE Noise bandwidth %[Hz]
ENBW = fs*S2/(S1^2);
%Matlab function
NENBW = enbw(window, fs)
```

Figure 6: ENBW formula for a window function in MATLAB

```
%E.g.
enbw(hann(2^16, 'periodic'), fs)
%Results
ans = 234.3750
```

Figure 7: ENBW for a Hannin window of length $N = 2^{16}$

This result completes the example of Fig.5 and indicates that the effective noise bandwidth of the **Hanning** window is **234.37 Hz**, alternatively the *minimun resultion* of the FFT applied to the windows signal.

Example: Suppose we have a generic window function $w[n]$ of length $N = 256$ and a sampling frequency of $f_s = 1000$ Hz. We can compute the ENBW of this window as follows:

1. Compute the NENBW using the formula:

$$NENBW = N \frac{S_2}{S_1^2}$$

where S_1 and S_2 are the first and second moments of the window function, respectively. For the given window, we can compute S_1 and S_2 as:

$$S_1 = \sum_{n=0}^{N-1} w[n] = 11.8438$$
$$S_2 = \sum_{n=0}^{N-1} |w[n]|^2 = 3.9246$$

Therefore, the NENBW is:

$$NENBW = N\frac{S_2}{S_1^2} = 256 \times \frac{3.9246}{(11.8438)^2} = 0.2243$$

2. Compute the FFT frequency resolution Δf as:

$$\Delta f = \frac{f_s}{N} = \frac{1000}{256} = 3.9063 \text{ Hz}$$

3. Compute the ENBW as:

$$ENBW = NENBW \cdot \Delta f = 0.2243 \times 3.9063 = 0.875 \text{ Hz}$$

The results obtained in the computation of the effective noise bandwidth (ENBW) for a window function of length $N = 256$ and a sampling frequency of $f_s = 1000Hz$ are:

1. a NENBW of 0.2243 bins;
2. an FFT frequency resolution of 3.9063 Hz;
3. an ENBW of 0.875 Hz.

These results can be used to analyze and design signal-processing systems that employ windowing techniques. The ENBW provides a useful **metric** for quantifying the *spectral leakage* **caused** by **windowing**.

Example for a Hann window: Suppose we have a Hanning window[1] function $w[n]$ of length $N = 256$ and a sampling frequency of $f_s = 1000$ Hz. We can compute the ENBW of this window as follows:

[1]The Hann window is named after Julius von Hann, an Austrian meteorologist, while the Hanning window is named after Wilhelm Hanning, a German physicist. However, the two names are sometimes used interchangeably, and both refer to the same window function, a raised cosine window type.

1. Compute the NENBW using the formula:

$$NENBW = N\frac{S_2}{S_1^2}$$

where S_1 and S_2 are the first and second moments of the window function, respectively. For the Hann window, we can compute S_1 and S_2 as:

$$S_1 = \sum_{n=0}^{N-1} w[n] = 1$$
$$S_2 = \sum_{n=0}^{N-1} |w[n]|^2 = \frac{1}{3}$$

Therefore, the NENBW is:

$$NENBW = N\frac{S_2}{S_1^2} = 256 \times \frac{\frac{1}{3}}{(1)^2} = 85.3333$$

2. Compute the FFT frequency resolution Δf as:

$$\Delta f = \frac{f_s}{N} = \frac{1000}{256} = 3.9063 \text{ Hz}$$

3. Compute the ENBW as:

$$ENBW = NENBW \cdot \Delta f = 85.3333 \times 3.9063 = 333.3333 \text{ Hz}$$

Note that the Hann window's first moment is 1, and the second moment is 1/3.

A.4 Periodogram

The Periodogram is a method used to estimate the power spectrum of a signal. It is obtained by computing the **magnitude squared** of the Discrete Fourier Transform (DFT) of the discrete time-domain signal **without** using a window function. The Periodogram provides a *rough estimate* of the power spectrum but suffers from **high variance** and **poor spectral resolution**, especially when the signal contains multiple frequency components.
The Periodogram is calculated as follows:

$$P(f_k) = \frac{1}{N}\left|\sum_{n=0}^{N-1} x[n]e^{-i2\pi f_k n}\right|^2 = |\mathbf{X}[k]|^2$$

where $x[n]$ is the discrete time-domain signal, N is the length of the signal and k represents the *index* of the frequency bin. Specifically, $f_k = k/N$ represents the frequency corresponding to the kth bin. The Periodogram estimates the power spectral density (PSD) by averaging the periodogram values over adjacent frequency bins[46].

A.5 FFT using a Window function

Before calculating the FFT of a signal, it is important to understand the differences between various spectral analysis techniques. These include the Periodogram, Window function (to reduce FFT leakage), and the concept of effective noise bandwidth (NENBW) and equivalent noise bandwidth (ENBW).

Considering a **time windowed signa**l $y[n] = x[n]w[n]$ of length N, where $x[n]$ is the initial signal and $w[n]$ is the window function, there are several important concepts[47] to be aware of when analyzing the power content of a signal, including:

- Power Spectrum (PS)

  measures the *power content* of a signal as a *function of frequency*. It is computed using the Periodogram of the *windowed time-domain signal*, which contains only the non-redundant part of the DFT, as explained in Eq. 2.1. This means that the PS is only defined for frequencies $k \in [0, \frac{N}{2}]$, where $\frac{N}{2}$ is *half* of the signal and window length. Therefore, the PS value for each frequency component must be **doubled** to obtain information about the **full spectrum**. The PS is commonly expressed in Volts or Amperes squared units, $[V^2]$ or $[A^2]$.

$$PS[k] = 2\frac{|\mathbf{Y}[k]|^2}{S_1^2} \qquad \text{(Eq. 5.1)}$$

  Where S_1^2 is referred to Eq. 3.5.

- Power Spectral Density (PSD)

  measures the signal's *power content* per *unit frequency bandwidth*. It is computed by **dividing** the **power spectrum** by the effective noise bandwidth of the window function. The PSD is commonly expressed in units of Volts or Ampere squared per Hertz per unit bandwidth, $[\frac{V^2}{Hz}]$ or $[\frac{A^2}{Hz}]$.

$$PSD[k] = 2\frac{PS[k]}{ENBW} = 2\frac{|\mathbf{Y}[k]|^2}{f_s S_2} \qquad \text{(Eq. 5.2)}$$

Where f_s is the sampling frequency used in the Periodogram or squared DFT magnitude, and S_2 is referred to Eq. 3.5.

- Amplitude Spectrum (AS)

  measures the *amplitude* of signal $y[n]$ as a *function of frequency*. It is equivalent to the **square root** of the **power spectrum** and is commonly expressed in units of Volts $[V]$ or Ampere $[A]$.

$$AS[k] = \sqrt{PS[k]} \tag{Eq. 5.3}$$

- Amplitude Spectral Density (ASD)

  measures the *amplitude* of the signal $y[n]$ per *unit frequency bandwidth*. It is computed by dividing the amplitude spectrum by the adequate noise bandwidth of the window function. Alternatively, it is equivalent to the **square root** of the **power spectral density**. The ASD is commonly expressed in units of Volts per Hertz per unit bandwidth $[\frac{V}{\sqrt{Hz}}]$ or $[\frac{A}{\sqrt{Hz}}]$.

$$ASD[k] = \sqrt{PSD[k]} \tag{Eq. 5.4}$$

A.6 Conclusion and Summary of FFT using a Window Function

In conclusion, using a window function in FFT analysis is essential to reduce **FFT leakage** and obtain **accurate** power spectral information. Understanding the concepts of Power Spectrum (PS), Power Spectral Density (PSD), Amplitude Spectrum (AS), and Amplitude Spectral Density (ASD) is crucial to analyze the power content of a signal as a function of frequency and unit frequency bandwidth. The PS and PSD are calculated *using* the Periodogram of the windowed **discrete time-domain signal**, while the AS and ASD are obtained by taking the square root of the PS and PSD, respectively. Additionally, normalized equivalent noise bandwidth (NENBW) and effective noise bandwidth (ENBW) must be considered when calculating the PSD and ASD, as they determine the frequency resolution and the frequency bandwidth of the analysis. Therefore, by using a **proper** window function and understanding these concepts, one can **accurately** analyze the power content of a signal in the frequency domain.

References

[38] Wikipedia, *Fourier transform*, Apr. 2023. [Online]. Available: https://en.wikipedia.org/wiki/Fourier_transform (Cited on page 155).

[39] Wikipedia, *Poisson summation formula*, Apr. 2023. [Online]. Available: https://en.wikipedia.org/wiki/Poisson_summation_formula (Cited on page 156).

[40] Wikipedia, *Discrete fourier transform dft*, Apr. 2023. [Online]. Available: https://en.wikipedia.org/wiki/Discrete_Fourier_transform# (Cited on page 156).

[41] Wikipedia, *Fast fourier transform*, Mar. 2023. [Online]. Available: https://en.wikipedia.org/wiki/Fast_Fourier_transform (Cited on page 157).

[42] G. Heinzel, A. Rüdiger, and R. Schilling, *Spectrum and spectral density estimation by the discrete fourier transform (dft)*, Jan. 1970. [Online]. Available: https://pure.mpg.de/pubman/faces/ViewItemOverviewPage.jsp?itemId=item_152164 (Cited on page 157).

[43] FFTW.org, *Fftw subroutine package*. [Online]. Available: http://www.fftw.org/ (Cited on page 157).

[44] Wikipedia, *Window function*, Mar. 2023. [Online]. Available: https://en.wikipedia.org/wiki/Window_function (Cited on page 158).

[45] Wikipedia, *Parseval's theorem*, Mar. 2023. [Online]. Available: https://en.wikipedia.org/wiki/Parseval%27s_theorem (Cited on page 160).

[46] Wikipedia, *Periodogram*, Mar. 2023. [Online]. Available: https://en.wikipedia.org/wiki/Periodogram (Cited on page 164).

[47] F. Harris, "On the use of windows for harmonic analysis with the discrete fourier transform," *Proceedings of the IEEE*, vol. 66, no. 1, pp. 51–83, 1978. DOI: 10.1109/PROC.1978.10837 (Cited on page 164).

B. Configuring Cadence Virtuoso

The successful analysis of an analogue-to-digital converter (ADC) is heavily contingent upon the precise configuration of the simulation settings within Cadence Virtuoso. This chapter delineates a comprehensive guide to correctly setting up the simulation, sampling and evaluating the ADC output spectrum, and modelling clock sources with varying jitter characteristics.

B.1 Output Spectrum in ADC

Precise configuration of the transient simulation is crucial in accurately assessing the output spectrum signals. Improper settings can introduce artificial spectrum leakage, leading to resolution degradation and diminished SNR. The simulation time, denoted as `simT`, should be calculated using the following equation:

$$\texttt{simT} = \frac{1}{\texttt{DeltaF}} + t_0 \qquad \text{(Eq. 1.1)}$$

Where `DeltaF` is equivalent to $\frac{f_s}{N}$ and denotes the DFT bin or frequency resolution in Hertz. Incorporating `N` and `f_s` into the variable area is recommended to streamline the process. To ensure an accurate transient simulation, carefully select these parameters:

- Stop time: Set to `VAR("simT")`.
  This controls the duration of the simulation.

- Accuracy: Set to "Conservative".
  This setting improves the precision of the simulation at the cost of computational speed.

- Enable "Transient Noise".
  This option allows the simulation of noise effects in the transient analysis.

- Noise Fmax: Set to `5*VAR("fs")`.
  This determines the maximum frequency of noise considered in the simulation.

Choose the maximum noise frequency as a multiple of f_s, based on the number of harmonics you wish to have in the simulation. All these settings are necessary to obtain the correct number of points, N, required for accurate spectrum analysis of the **ADC output**.

B.1.1 Clipping the Signal

Before proceeding with the analysis, it is crucial to ascertain that the transient signal under consideration, namely `vtime('tran "/v_out[n]"')`, includes an integer multiple of the number of periods. This condition is essential to maintain the periodicity of the DFT, which employs a *circular convolution*. To ensure this condition, the signal can be "clipped" (truncated by removing the extra time) with the following command:

```
◇ v_clip = clip(vtime('tran "/v_out[n]"'), VAR("t0"), VAR("simT"))
```

Alternatively, the same operation can be performed using this variant:

```
◇ v_clip = clip(v("/v_out[n]" ?result "tran"), VAR("t0"), VAR("simT"))
```

The utilization of these commands ensures that the transient signal is appropriately adjusted for the subsequent DFT operation, thereby minimizing convolution errors and enhancing the overall accuracy of the spectral analysis.

B.1.2 Sampling the Clipped Signal

An essential precondition for a successful DFT operation is to obtain N points by *sampling* the clipped signal in the time domain, using the sampling frequency of the ADC. An effective way to establish f_s is by determining the **average** value of your clock source and subsequently defining it as a variable within your workspace. All variables, including `deltaF` and `simT`, should be configured to align with this empirically derived f_s, owing to its pivotal role in the transient simulation settings. Employing a high degree of precision post-decimal point is recommended, as this practice enhances the quality of the resultant sampled signal. Note, however, that this step may be superfluous in the context of a clock source *model* that does *not* incorporate jitter.

After the adjustments, the following formula can be deployed for the task at hand:

```
◇ v_nTs = sample(v_clip VAR("t0"), VAR("simT") "linear" (1/VAR("fs")))
```

In the case of **real** clock schematics laden with jitter, the above function could potentially lead to information loss and introduce sampling noise into the sampled signal, thereby compromising the integrity of the spectrum evaluation. To navigate this issue, the preferred approach is to employ a **VerilogA** code that generates a text file comprising the values of the ADC output, sampled by the actual signal:

```
1   include "constants.vams"
2   include "disciplines.vams"
3   module Block_Name(in, clk);
4   input in;
5   input clk;
6   electrical in;
7   electrical clk;
8   //threshold value
9   parameter real vtrans_clk = 0.6;
10  /* Rising edge: dir = 1
11  Falling edge:  dir = -1
12  both edges: dir = 0
13  */
14  parameter integer dir = 1;
15  // signal tolerance on the clk
16  parameter real vtol = 0;
17  // time tolerance on the clk
18  parameter real ttol = 0;
19  real out;
20  real Cross_point;
21  integer file;
22  analog begin
23  @(initial_step) begin
24      file = $fopen("/home/userName/linuxhome/Folder/fileName.txt");
25      end
26  @(cross(V(clk) - vtrans_clk,dir,vtol,ttol)) begin
27      Cross_point = $abstime;
28      out = V(in);
29      $fwrite(file,"%e \n",out);
30      end
31  end
32  endmodule
```

This code creates a VerilogA block with two **inputs**: the ADC output, `in`, and the clock source output, `clk`. The `vtrans_clk` parameter signifies the half-amplitude of the clock signal. Although it is currently set to $600mV$, this value should be adjusted to correspond with the amplitude of the clock source in use. The advantage of this approach is that the generated file is sampled with the same jitter noise as the clock source, enabling us to assess its effect on the ADC output spectrum. One highly recommended approach is to compute the PSD using Matlab, as the generated *text file* can be easily integrated into the Matlab environment.

B.1.3 Spectrum Evaluation

In the context of an **ideal** clock source free from jitter, the spectrum can be accurately computed using a periodogram, as exemplified in the formula:

◇ `PSD_v = dB20( dft( v_nTs, VAR("t0"), VAR("simT"), VAR("N"), "Hanning", 1, 1, 1.0) )`

This function calculates the PSD as $10log_{10}\left(|\mathbf{X}[k]|^2\right)$, where $\mathbf{X}[k]$ is the Discrete Fourier Transform (DFT) of the ADC output employing the *Hanning*

window. A notable benefit of utilizing `dB20(dft(·))` over `psd(·)` stems from the former's capacity to use `VAR("N")` to specify the DFT's number of points, thus leveraging workspace variables. On the other hand, the latter function returns an error when using `VAR("N")`, necessitating manual specification of the number. For clarification, the alternative function is given as:

◇ `PSD2_v = psd(v_nTs, VAR("t0"), VAR("simT"), 2**(18), ?windowName "Hanning", ?smooth 1, ?windowSize 2**(18), ?detrending "None", ?cohGain 1)`

B.2 White Noise Clock model

A clock source **model** that generates jitter as **white noise** can be created by using a random variable with a Gaussian distribution, denoted as $\mathbf{x} \sim \mathcal{N}(0, \sigma_{\mathbf{x}})$. Please note that the jitter noise is **not** purely white, thus, this method represents a *basic approach* to incorporating clock noise into the system. Nevertheless, the analogLib allows the easy implementation of this model by employing the `vsource` block and selecting `bit` as the `Source Type`. Figure 1 displays the configuration options for this block, where:

- Zero and One values: correspond to the minimum and maximum amplitudes of the clock signal, respectively.
- Period of the waveform: should be equal to twice the frequency f_s.
- Rise and Fall time: should be chosen in accordance with the standard deviation of the noise, $\sigma_{\mathbf{x}}$.
- Trigger: should be set to `internal`.
- **RJ(rms)**: should be set to $\frac{\sigma_{\mathbf{x}}}{\sqrt{2}}$. As it is in root-mean-square value, this setting ensures that the desired standard deviation in seconds is achieved. In this example, $\sigma_{\mathbf{x}} = 50ps$.

All other settings, such as `Number of Periodic Jitters` and all `Pattern Parameters`, should be configured as shown in Figure 1 to guarantee accurate clock model behavior.

B.3 Non-White Clock Noise

Please Note Implementing phase noise model requires Spectre to be at version 20.1.0.382.isr12 64bit or higher, dated 5 Nov 2021. If your Spectre version is lower than this, you will encounter issues when applying the phase

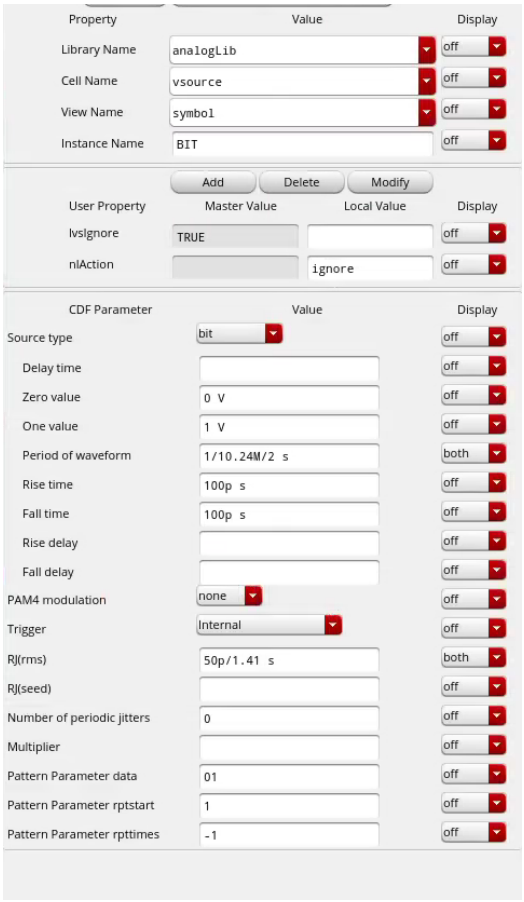


Figure 1: White Noise Jitter model

noise model. Please ensure you have the correct version before proceeding with the following steps.

To expedite transient simulations of your ADC and its clock source, the clock schematic can be modelled as a block emulating the performance characteristics of the original circuit. Evaluating the **phase noise** of the clock schematic is a crucial initial step and can be performed using the function below:

- ◇ `PN_[2_19] = PN(Clk_nTs "rising" (VAR("VDD")/2) ?Tnom nil ?windowName "Hanning" ?smooth 1 ?windowSize 524288 ?detrending "None" ?cohGain "default" ?methodType "absJitter") - dB10(VAR("DeltaF"))`

The function calculates the phase noise in dBc/Hz before subtracting the DFT resolution `DeltaF` (converted to the dB scale), which adjusts the phase noise result to a $1Hz$ bandwidth resolution, complying with the definition of phase noise. Prior to employing this function, ensure that the clock source

signal is properly *clipped* and *sampled* for an accurate DFT computation. Be aware that the signal's name, `Clk_nTs`, is based on the definition provided in the preceding sections. It is important to note that `PN(·)` shares a common limitation with `psd(·)` - the necessity of numerically stating the number of points, denoted as N = 2**(19).

After computation, the phase noise plot can be stored as a `.out.spectre` file. When opened with a text editor on Linux, this file presents two column arrays: frequency and dBc/Hz. It is instrumental in establishing the desired spectrum of the clock model. Start by setting the `Source Type` to `pulse`, and

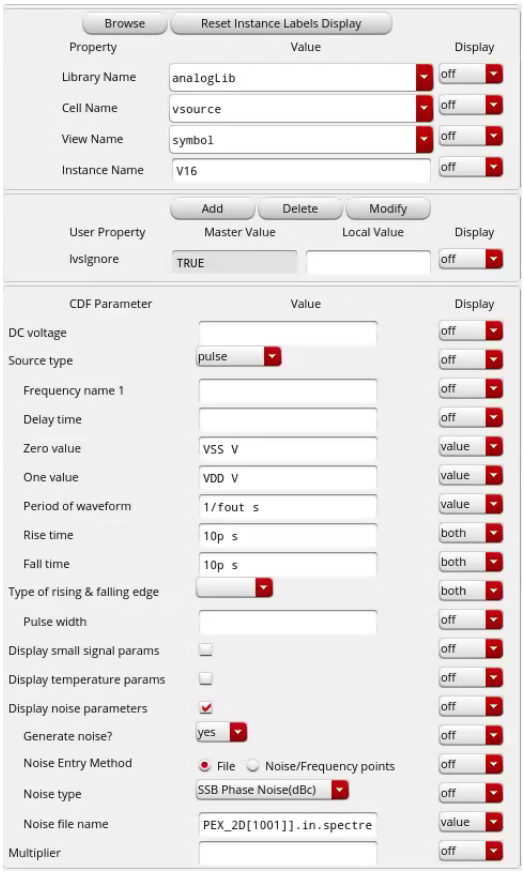


Figure 2: Phase Noise Jitter model

keep the initial settings consistent with those detailed in Section II. Follow the instructions presented in Figure 2 to finish the setup:

- Generate noise: select `yes`.
- Noise entry model: select `file`.

- Noise type: choose Single Side Bandwidth phase noise.
- Noise file name: enter the **full path** of the file (saved from the transient simulation and evaluated using `PN_[2_19]`).

While the method described above [48] is executable in Virtuoso®, the resultant clock source waveform, which aligns with the defined settings, **may not** affect the ADC as expected. It is probable that the clock model's phase noise spectrum **will not** influence the ADC output spectrum, thereby challenging the current model's *validity*.

B.3.1 Possible solutions

One solution is to set the maximum noise frequency inversely proportional to the rise and fall time to expose all frequency components of the noise, which requires changing:

- Noise Fmax: set to $\frac{1}{\texttt{RiseTime}}$.

Alternatively, an empirical approach can be taken to scale the maximum noise frequency to the necessary level (`k*VAR("fs")`), where $k \in \mathbb{R}$, guided by the simulation results.
Otherwise, implementing the model proposed in the Cadence support forum [49] and [50] or developing an own VerilogA block that generates noise following the described functions are also valid solutions.

B.4 Conclusion

This guide detailed an approach to analyse how *jittery clock sources* might affect ADCs. The proposed strategy involves setting up *transient simulations*, accurately sampling signals, and carrying out thorough *spectrum evaluations*.

The method includes developing a clock model using a *random variable* with a Gaussian distribution to emulate *white noise*. This involves configuring a `vsource` block in analogLib with specific parameters, including Zero and One values, Period of the waveform, Rise and Fall time, and RJ(rms).

To emulate *non-white clock noise*, *phase noise characteristics* of the clock source were utilised, requiring an evaluation of the original clock schematic's phase noise and its emulation using a phase noise model leveraging the analogLib library.

Although this guide provides a comprehensive strategy, *further refinements and solutions* could enhance the modelling and analysis of jitter noise. These methods **could be tailored** depending on the *specific characteristics* of the clock source and the required precision of the analogue-to-digital conversion process.

References

[48] Cadence Support. "How to specify phase noise as an instance parameter in spectre sources (e.g. vsource, isource, port)." (), [Online]. Available: https://support.cadence.com/apex/ArticleAttachmentPortal?id=a1Od0000000tiraEAA&pageName=ArticleContent (visited on 05/18/2023) (Cited on page 173).

[49] Cadence Support. "Modeling oscillators with arbitrary phase noise profiles." (), [Online]. Available: https://support.cadence.com/apex/ArticleAttachmentPortal?id=a1Od0000000nTjxEAE&pageName=ArticleContent (visited on 05/18/2023) (Cited on page 173).

[50] Cadence Community. "Vco phase noise modeling using a transient noise simulation." (), [Online]. Available: https://community.cadence.com/cadence_technology_forums/f/custom-ic-design/55478/vco-phase-noise-modeling-using-a-transient-noise-simulation (visited on 05/18/2023) (Cited on page 173).

[51] Cadence Community. "Comparing transient noise, pnoise, and pnoise with lorentian approximation of a ring oscillator." (), [Online]. Available: https://community.cadence.com/cadence_technology_forums/f/rf-design/51484/comparing-transient-noise-pnoise-and-pnoise-with-lorentian-approximation-of-a-ring-oscillator (visited on 05/18/2023).

Index of References

This index provides references used in the research for the Master's thesis project. The sources encompass a range of insights in the field. They cover both theoretical foundations and practical applications, supporting the findings and conclusions of this thesis. For ease of navigation, the references are arranged in **alphabetical order**:

[R.1] S. Ahmed and V. Kakkar, "Modeling and simulation of an eight-bit auto-configurable successive approximation register analog-to-digital converter for cardiac and neural implants," *SIMULATION*, vol. 94, no. 1, pp. 11–29, 2018. DOI: 10.1177/0037549717716537.

[R.2] M. Aqamolaei and M. M. Ahmadi, "Modelling an oscillator phase noise whose spectral density contains both 1/f 2 and 1/f 3 regions," *Electronics Letters*, vol. 58, 2021. DOI: 10.1049/ell2.12393.

[R.3] M. Aqamolaei and M. M. Ahmadi, "Modelling an oscillator phase noise whose spectral density contains both $1/f^2$ and $1/f^3$ regions," *Electronics Letters*, vol. 58, no. 4, pp. 139–141, 2022. DOI: https://doi.org/10.1049/ell2.12393. eprint: https://ietresearch.onlinelibrary.wiley.com/doi/pdf/10.1049/ell2.12393. [Online]. Available: https://ietresearch.onlinelibrary.wiley.com/doi/abs/10.1049/ell2.12393.

[R.4] A. Ashry and H. Aboushady, "Modeling jitter in continuous-time sigma-delta modulators," in *2010 IEEE International Behavioral Modeling and Simulation Workshop*, IEEE, 2010, pp. 53–56. DOI: 10.1109/BMAS.2010.6156598.

[R.5] Cadence Community. "Comparing transient noise, pnoise, and pnoise with lorentian approximation of a ring oscillator." (), [Online]. Available: https://community.cadence.com/cadence_technology_forums/f/rf-design/51484/comparing-transient-noise-pnoise-and-pnoise-with-lorentian-approximation-of-a-ring-oscillator (visited on 05/18/2023).

[R.6] Cadence Community. "Vco phase noise modeling using a transient noise simulation." (), [Online]. Available: https://community.cadence.com/cadence_technology_forums/f/custom-ic-design/55478/vco-phase-noise-modeling-using-a-transient-noise-simulation (visited on 05/18/2023).

[R.7] Cadence Support. "How to specify phase noise as an instance parameter in spectre sources (e.g. vsource, isource, port)." (), [Online]. Available: https://support.cadence.com/apex/ArticleAttachmentPortal?id=a1Od0000000tiraEAA&pageName=ArticleContent (visited on 05/18/2023).

[R.8] Cadence Support. "Modeling oscillators with arbitrary phase noise profiles." (), [Online]. Available: https://support.cadence.com/apex/ArticleAttachmentPortal?id=a1Od0000000nTjxEAE&pageName=ArticleContent (visited on 05/18/2023).

[R.9] S. Christie, "Electromagnetic navigational bronchoscopy and robotic-assisted thoracic surgery," *AORN J*, vol. 99, no. 6, pp. 750–763, 2014. DOI: 10.1016/j.aorn.2013.09.016.

[R.10] C. De Berti, P. Malcovati, L. Crespi, and A. Baschirotto, "Colored clock jitter model in audio continuous-time $\Sigma\Delta$ modulators," pp. 1–4, 2015, Accessed: 2023-09-24. DOI: 10.1109/NEWCAS.2015.7182021. [Online]. Available: https://ieeexplore.ieee.org/document/7182021.

[R.11] A. Demir *et al.*, "Phase noise in oscillators: A unifying theory and numerical methods for characterization," *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*, vol. 47, no. 5, pp. 655–674, 2000.

[R.12] A. Devices. "Converting oscillator phase noise to time jitter." Accessed: 24 September 2023. (2009), [Online]. Available: https://www.analog.com/media/en/training-seminars/tutorials/MT-008.pdf.

[R.13] Digilent, *Digital discovery: Start*, https://digilent.com/reference/test-and-measurement/digital-discovery/start, Accessed: 19 October 2023, 2023.

[R.14] FFTW.org, *Fftw subroutine package*. [Online]. Available: http://www.fftw.org/.

[R.15] A. Hajimiri and T. Lee, "A general theory of phase noise in electrical oscillators," *IEEE Journal of Solid-State Circuits*, vol. 33, no. 2, pp. 179–194, 1998. DOI: 10.1109/4.658619.

[R.16] A. Hajimiri, S. Limotyrakis, and T. Lee, "Jitter and phase noise in ring oscillators," *IEEE Journal of Solid-State Circuits*, vol. 34, no. 6, pp. 790–804, 1999. DOI: 10.1109/4.766813.

[R.17] A. Hajimiri and T. H. Lee, "A general theory of phase noise in electrical oscillators," *IEEE Journal of solid-state circuits*, vol. 33, no. 2, pp. 179–194, 1998.

[R.18] F. Harris, "On the use of windows for harmonic analysis with the discrete fourier transform," *Proceedings of the IEEE*, vol. 66, no. 1, pp. 51–83, 1978. DOI: 10.1109/PROC.1978.10837.

[R.19] F. Harris, "On the use of windows for harmonic analysis with the discrete fourier transform," *Proceedings of the IEEE*, vol. 66, no. 1, pp. 51–83, 1978. DOI: 10.1109/PROC.1978.10837.

[R.20] G. Heinzel, A. Rüdiger, and R. Schilling, *Spectrum and spectral density estimation by the discrete fourier transform (dft)*, Jan. 1970. [Online]. Available: https://pure.mpg.de/pubman/faces/ViewItemOverviewPage.jsp?itemId=item_152164.

[R.21] L. Hernandez, A. Wiesbauer, S. Paton, and A. Di Giandomencio, "Modelling and optimization of low pass continuous-time sigma delta modulators for clock jitter noise reduction," in *2004 IEEE International Symposium on Circuits and Systems*, IEEE, 2004, pp. I-1072–5. DOI: 10.1109/ISCAS.2004.1328384.

[R.22] H. Jaeger, A. Franz, K. donoghue, *et al.*, "Anser emt the first open-source electromagnetic tracking platform for image-guided interventions," *International Journal of Computer Assisted Radiology and Surgery*, vol. 12, Mar. 2017. DOI: 10.1007/s11548-017-1568-7.

[R.23] F. X. Kärtner, "Analysis of white and f-α noise in oscillators," *International Journal of Circuit Theory and Applications*, vol. 29, no. 2, pp. 131–148, 2001.

[R.24] Keysight Technologies, *Dsox6004a oscilloscope - 1GHz to 6GHz, 4 analog channels*, https://www.keysight.com/it/en/product/DSOX6004A/oscilloscope-1-ghz-6-ghz-4-analog-channels.html, Accessed: 2023-09-24, 2023.

[R.25] A. Khashaba, J. Zhu, N. Pal, M. G. Ahmed, and P. K. Hanumolu, "A 32-mhz, 34-μw temperature-compensated rc oscillator using pulse density modulated resistors," *IEEE Journal of Solid-State Circuits*, vol. 57, no. 5, pp. 1470–1479, 2022. DOI: 10.1109/JSSC.2021.3121014.

[R.26] P. Malcovati, S. Brigati, F. Francesconi, F. Maloberti, P. Cusinato, and A. Baschirotto, "Behavioral modeling of switched-capacitor sigma-

delta modulators," *IEEE Trans. Circuits Syst. I*, vol. 50, no. 3, pp. 352–364, 2003. DOI: 10.1109/TCSI.2003.808892.

[R.27] MathWorks. "Delta sigma toolbox." Online; accessed 24 September 2023, MathWorks. (2023), [Online]. Available: https://it.mathworks.com/matlabcentral/fileexchange/19-delta-sigma-toolbox.

[R.28] MathWorks. "Phase noise measure." Online; accessed 24 July 2023, MathWorks. (2023), [Online]. Available: https://www.mathworks.com/help/comm/ref/phasenoisemeasure.html (visited on 07/24/2023).

[R.29] Murata Manufacturing Co., Ltd. "Murata introduces world's smallest low-profile crystal unit size 1.2x1.0x0.30mm." Accessed: 2023-12-17. (2017), [Online]. Available: https://passive-components.eu/murata-introduces-worlds-smallest-low-profile-crystal-unit-size-1-2x1-0x0-30mm/ (visited on 12/17/2023).

[R.30] T. Nandi, K. Boominathan, and S. Pavan, "Continuous-time $\Delta\Sigma$ modulators with improved linearity and reduced clock jitter sensitivity using the switched-capacitor return-to-zero dac," *IEEE J. Solid-State Circuits*, vol. 48, no. 8, 2013. DOI: 10.1109/JSSC.2013.2259012.

[R.31] M. Ortmanns and F. Gerfers, *Continuous-Time Sigma-Delta A/D Conversion: Fundamentals, Performance Limits and Robust Implementations*. Springer Berlin Heidelberg, 2006.

[R.32] B. Razavi, "A general theory of phase noise in electrical oscillators," in *Phase-Locking in High-Performance Systems: From Devices to Architectures*. 2003, pp. 189–204. DOI: 10.1109/9780470545492.ch20.

[R.33] B. Razavi, *Design of CMOS Phase-Locked Loops: From Circuit Level to Architecture Level*. Cambridge University Press, 2020. DOI: 10.1017/9781108626200.

[R.34] K. Reddy and S. Pavan, "Fundamental limitations of continuous-time delta-sigma modulators due to clock jitter," *IEEE Trans. Circuits Syst. I*, vol. 54, no. 10, pp. 2184–2194, 2007. DOI: 10.1109/TCSI.2007.905648.

[R.35] Rohde-Schwarz, *Sma100b signal generator*, https://www.rohde-schwarz.com/it/prodotti/misura-e-collaudo/generatori-di-segnali-analogici/rs-sma100b-rf-and-microwave-signal-generator_63493-427776.html, Accessed: 19 October 2023, 2023.

[Online]. Available: https://scdn.rohde-schwarz.com/ur/pws/dl_downloads/pdm/cl_brochures_and_datasheets/specifications/5215_1018_22/SMA100B_specs_en_5215-1018-22_v0704.pdf.

[R.36] R. Saad, S. Hoyos, and S. Palermo, "Analysis and modeling of clock-jitter effects in delta-sigma modulators," in *MATLAB - A Fundamental Tool for Scientific Computing and Engineering Applications - Volume 1*, V. Katsikis, Ed., InTech, 2012.

[R.37] R. Schreier, S. Pavan, and G. C. Temes, *Understanding Delta-Sigma Data Converters*. John Wiley & Sons, Inc., 2017.

[R.38] M. Srivastava, A. Ferro, A. Sidun, *et al.*, "A small-area 2nd-order adder-less continuous-time $\Delta\Sigma$ modulator with pulse shaping fir dac for magnetic sensing," *IEEE Open Journal of Circuits and Systems*, vol. 5, pp. 42–54, 2024. DOI: 10.1109/OJCAS.2024.3378653.

[R.39] M. Srivastava, K. O'Donoghue, A. Sidun, *et al.*, "3d position tracking using on-chip magnetic sensing in image-guided navigation bronchoscopy," *IEEE Transactions on Biomedical Circuits and Systems*, pp. 1–17, 2024. DOI: 10.1109/TBCAS.2024.3384016.

[R.40] K. Technologies, *33500b and 33600a series trueform waveform*, https://www.keysight.com/it/en/assets/7018-05928/data-sheets/5992-2572.pdf, Datasheet, Accessed: 19 October 2023, 2023.

[R.41] TSMC. "65nm technology." (2022), [Online]. Available: https://www.tsmc.com/english/dedicatedFoundry/technology/logic/l_65nm.

[R.42] Wikipedia, *Discrete fourier transform dft*, Apr. 2023. [Online]. Available: https://en.wikipedia.org/wiki/Discrete_Fourier_transform#.

[R.43] Wikipedia, *Fast fourier transform*, Mar. 2023. [Online]. Available: https://en.wikipedia.org/wiki/Fast_Fourier_transform.

[R.44] Wikipedia, *Fourier transform*, Apr. 2023. [Online]. Available: https://en.wikipedia.org/wiki/Fourier_transform.

[R.45] Wikipedia, *Infinite impulse response — Wikipedia, the free encyclopedia*, [Online; accessed 26-November-2023], 2023. [Online]. Available: https://en.wikipedia.org/wiki/Infinite_impulse_response.

[R.46] Wikipedia. "Low-voltage differential signaling." Italian. Accessed: 2023-11-26, Wikipedia. (2023), [Online]. Available: https://it.wikipedia.org/wiki/Low-voltage_differential_signaling (visited on 11/26/2023).

[R.47] Wikipedia, *Parseval's theorem*, Mar. 2023. [Online]. Available: https://en.wikipedia.org/wiki/Parseval%27s_theorem.

[R.48] Wikipedia, *Periodogram*, Mar. 2023. [Online]. Available: https://en.wikipedia.org/wiki/Periodogram.

[R.49] Wikipedia, *Poisson summation formula*, Apr. 2023. [Online]. Available: https://en.wikipedia.org/wiki/Poisson_summation_formula.

[R.50] Wikipedia, *Window function*, Mar. 2023. [Online]. Available: https://en.wikipedia.org/wiki/Window_function.

[R.51] H. Zheng, "Mixed signal compensation of sampling errors in adcs due to noisy dpll clock sources," Available from: https://cora.ucc.ie/items/667c56e2-462d-4a9b-a2ec-0574e2f2f20e, M.S. thesis, University College Cork, 2021.