NOVA SCHOOL OF SCIENCE & TECHNOLOGY

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

JOÃO FRANCISCO DE ALMEIDA REBELO SOARES

BSc in Electrical and Computer Engineering

ANALYSIS AND DESIGN OF RING AMPLIFIERS FOR PIPELINE ADCS

MASTER IN ELECTRICAL AND COMPUTER ENGINEERING

NOVA University Lisbon
December, 2024

NOVA SCHOOL OF
SCIENCE & TECHNOLOGY

DEPARTMENT OF ELECTRICAL
AND COMPUTER ENGINEERING

ANALYSIS AND DESIGN OF RING AMPLIFIERS FOR PIPELINE ADCS

JOÃO FRANCISCO DE ALMEIDA REBELO SOARES

BSc in Electrical and Computer Engineering

Adviser: Nuno Filipe Silva Veríssimo Paulino
Associate Professor, NOVA University Lisbon

Co-adviser: Marco Filipe Ribeiro Rodrigues
Principal Analog/Mixed Signal Design Engineer, Renesas Electronics Lisbon

Examination Committee

Chair: Anikó Katalin Horváth da Costa
Assistant Professor, NOVA University Lisbon

Rapporteurs: Nuno Filipe Silva Veríssimo Paulino
Associate Professor, NOVA University Lisbon

João Pedro Oliveira
Associate Professor, NOVA University Lisbon

MASTER IN ELECTRICAL AND COMPUTER ENGINEERING

NOVA University Lisbon
December, 2024

Analysis and Design of Ring Amplifiers for Pipeline ADCs

This document was created with the (pdf/Xe/Lua)LaTeX processor and the NOVAthesis template (v6.10.10) [55].

To my friend Manuel Luz, who is no longer with us.

Acknowledgements

I want to express my deepest gratitude to my advisor, Professor Nuno Paulino, for his invaluable guidance and support throughout the development of this work. His availability and expertise were crucial in resolving key challenges encountered along the way.

I am also sincerely grateful to my co-advisor, Engineer Marco Rodrigues, for his continuous assistance, insightful suggestions, and patience in addressing important issues that occurred during this process.

A heartfelt appreciation goes to my family for their unwavering support throughout this journey, both during this project and throughout my academic career.

Finally, I would like to express my profound recognition to my friends André Carvalho, Afonso Tavares, Cezar Petrea, Diogo Nunes, and Francisco Silva, who accompanied me throughout the course. I am also deeply grateful to André Pinto, Carolina Junqueiro, João Matos, José Pica and Rodrigo Leal, whose support beyond the academic sphere was essential to the success of this project.

" Let the future tell the truth, and evaluate each one according to his work and accomplishments. The present is theirs; the future, for which I have really worked, is mine. " (Nikola Tesla)

Abstract

The main focus of this thesis is the study of ring amplifier configurations for use as a residue amplifier in pipeline analog-to-digital converters. Two implementations have been studied: the conventional and the critically damped configurations. The conventional amplifier has been analyzed, leading to the development of a robust parametric simulation testbench and a deeper understanding of the fundamentals of ring amplification, from which a theoretical computer-based model has been created. For the critically damped ring amplifier, the computational model was adapted and a design approach based on the g_M/I_D method was developed to assess its suitability for pipeline analog-to-digital converters. This method uses fast computational iterations reducing the need for extensive simulations, yet remaining consistent with practical simulation results. Although employed in a flip-around configuration with a 3-bit resolution, the design strategy is versatile enough to be adapted to other environments simply by changing testing framework parameters.

A critically damped ring amplifier in 65 nm CMOS technology was successfully developed, featuring two designs: one optimized for open-loop gain, achieving 53.6 dB, and another focused on optimizing GBW, reaching 5.7 GHz with a correspondent settling time of 345.7 ps. The gain-oriented design delivered an ENOB of 10.7 bits, a THD of -67.1 dB, and consumed 1.0 mW. In contrast, the GBW-oriented design achieved an ENOB of 13.0 bits, a THD of -80.4 dB, and consumed 2.0 mW.

Additionally, an extensive literature review presents the evolution of ring amplification as a residue amplifier, providing a detailed explanation of the fundamental configurations and highlighting their evolution as they converge towards a specific design focus.

Keywords: ADC, analog-to-digital converter, pipeline, RA, residue amplifier, ring amplifier, critically damped, gm/ID method

Resumo

O foco principal desta tese é o estudo de configurações de amplificadores em anel para uso como amplificador de resíduo em conversores analógico para digital concorrencionais. Foram estudadas duas implementações: a configuração convencional e a criticamente amortecida. O amplificador convencional foi analisado, levando ao desenvolvimento de um banco de testes de simulação paramétrica robusto e a uma compreensão profunda dos fundamentos da amplificação em anel, a partir dos quais foi criado um modelo teórico baseado em computador. Para o amplificador em anel criticamente amortecido, o modelo computacional foi adaptado e foi desenvolvida uma abordagem de projeto baseada no método g_m/I_D para avaliar a sua adequação a conversores analógico para digital concorrencionais. Este método utiliza iterações computacionais rápidas, reduzindo a necessidade de simulações extensas, mas mantendo-se consistente com os resultados de simulações práticas. Embora utilizado numa configuração invertida com uma resolução de 3 bits, a estratégia de projeto é suficientemente versátil para ser adaptada a outros ambientes, bastando alterar os parâmetros da estrutura de ensaio.

Um amplificador em anel criticamente amortecido em tecnologia CMOS de 65 nm foi desenvolvido com sucesso, apresentando dois dimensionamentos: um otimizado para o ganho em malha aberta, atingindo 53.6 dB, e outro focado na otimização do GBW, alcançando 5.7 GHz com um tempo de estabelecimento correspondente de 345.7 ps. O projeto orientado para ganho forneceu um ENOB de 10.7 bits, uma THD de -67.1 dB e consumiu 1.0 mW. Em contraste, o projeto orientado para GBW alcançou um ENOB de 13.0 bits, uma THD de -80.4 dB e consumiu 2.0 mW.

Adicionalmente, uma extensa revisão da literatura apresenta a evolução da amplificação em anel como amplificador de resíduo, fornecendo uma explicação detalhada das configurações fundamentais e destacando sua evolução à medida que convergem para um foco de desempenho específico.

Palavras-chave: ADC, conversor analógico para digital, concorrencial, RA, amplificador de resíduo, amplificador em anel, criticamente amortecido, método gm/ID

Contents

List of Figures

List of Tables

Acronyms

ADC analog-to-digital converter *(pp. 4, 5, 8, 9, 11, 15, 17–19, 41–43)*

CDL capacitive degeneration linearization *(pp. 31–33)*
CMD common-mode detection *(p. 30)*
CMFB common-mode feedback *(p. 65)*
CMOS complementary-metal-oxide-semiconductor *(pp. 27, 38, 39, 48)*
CMRR common-mode rejection ratio *(p. 26)*

DA dynamic amplifier *(pp. 30, 33)*
DAC digital-to-analog converter *(pp. 13–19)*
DNL differential nonlinearity *(p. 7)*

ENOB effective number of bits *(pp. 8, 41, 43, 67–70)*

FFT fast Fourier transform *(pp. 8, 48–50, 65, 67, 70)*
FIR finite impulse response *(p. 14)*
FOM figure of merit *(pp. 9, 27)*

GBW gain-bandwidth product *(pp. 23, 27, 63, 64, 68, 71)*

INL integral nonlinearity *(p. 7)*

LSB least significant bit *(pp. 5–7, 9, 15, 16)*

MCS merged capacitor switching *(pp. 16, 18)*
MDAC multiplying digital-to-analog converter *(pp. 17, 18, 48, 49, 65)*
MSB most significant bit *(pp. 5, 15, 16)*

OSR oversampling ratio *(p. 12)*

OTA operational transconductance amplifier *(pp. 28, 30, 33)*
OVS output voltage swing *(p. 26)*

PSD power spectral density *(p. 27)*
PSRR power supply rejection ratio *(p. 26)*
PVT process voltage temperature *(pp. 37, 40, 76)*

RA residue amplifier *(pp. 17, 19)*

SAR successive-approximation register *(p. 15)*
SC switched capacitor *(p. 18)*
SFDR spurious free dynamic range *(pp. 8, 68, 70)*
SH sample-and-hold *(pp. 11–13, 15–17)*
SNDR signal-to-noise and distortion ratio *(pp. 8, 68, 70)*
SNR signal-to-noise ratio *(pp. 8, 12, 68, 70)*
SR slew rate *(p. 25)*

THD total harmonic distortion *(pp. 8, 20, 67–70)*
TL track-and-latch *(p. 10)*

Symbols

A	gain *(pp. 18, 22, 24)*
A_F	feedback gain *(pp. 18, 23, 24)*
B_{OUT}	binary output code *(pp. 4, 9, 12)*
B_S	binary stream *(p. 12)*
B_T	thermometer binary code *(p. 9)*
BW	bandwidth *(pp. 4, 22)*
BW_F	feedback bandwidth *(p. 24)*
β	feedback factor *(pp. 23, 24)*
C_L	load capacitance *(pp. 27, 30–32)*
D_{cycle}	duty cycle *(p. 18)*
ϵ_A	gain error *(p. 6)*
ϵ_{OS}	offset error *(p. 6)*
ϵ_Q	quantization error *(p. 5)*
f_N	Nyquist frequency *(p. 4)*
f_{OS}	oversampling frequency *(p. 12)*
f_S	sampling frequency *(pp. 4, 12)*
FOM_S	Schreier figure of merit *(p. 9)*
FOM_W	Walden figure of merit *(p. 9)*
ϕ_A	amplification phase *(p. 18)*
ϕ_{CLK}	clock phase *(p. 13)*
ϕ_S	sampling phase *(p. 18)*
G_m	transconductance *(pp. 28, 31)*

L channel length *(p. 27)*
LIN linearity *(p. 20)*

M order *(pp. 13, 14)*

N resolution *(pp. 4, 9, 15, 18)*

P_d dissipated power *(pp. 9, 27)*
P_D harmonic distortion power *(p. 8)*
P_H highest spur power *(p. 8)*
P_N noise power *(p. 8)*
P_{QN} quantization noise power *(p. 6)*
P_S signal power *(p. 8)*

R_{out} output resistance *(pp. 28, 35)*

T temperature *(p. 27)*
T_A amplification time *(p. 31)*
T_{CLK} clock period *(p. 18)*
t_d delay time *(p. 18)*
t_f fall time *(p. 18)*
t_r rise time *(p. 18)*
t_s settling time *(p. 25)*
τ time constant *(p. 25)*

V_{CLK} clock voltage *(p. 30)*
$V_{CM_{IN}}$ input common mode voltage *(p. 27)*
$V_{CM_{OUT}}$ output common mode voltage *(pp. 27, 30)*
V_{DAC} digital to analog voltage *(pp. 13, 15–17)*
V_{DD} positive supply voltage *(pp. 27, 30)*
V_{DZ} dead zone voltage *(p. 34)*
V_{IN} input voltage *(pp. 4, 9, 11, 15, 17, 33)*
V_{LSB} least significant bit voltage *(p. 5)*
V_N nyquist voltage *(pp. 12, 13)*
V_{OS} offset voltage *(p. 34)*
V_{OUT} output voltage *(p. 33)*
V_{QN} quantification noise voltage *(pp. 5, 14)*
V_{REF} reference voltage *(pp. 4, 9)*
V_{RES} residue voltage *(p. 17)*
V_{RESA} amplified residue voltage *(p. 17)*

V_{SH} sampled and hold voltage *(pp. 13, 15, 17)*

V_{SS} negative supply voltage *(p. 27)*

W channel width *(p. 27)*

1

Introduction

1.1 Motivation

The growing demand for System-on-Chip devices has emerged as a result of continuous advancements in semiconductor technology. In parallel, the design of analog-to-digital converters has evolved to keep pace with these technological advancements. One significant advantage of this is the scaling property. As technology advances, associated with a shorter channel length size, converters achieve higher bandwidth and require lower supply voltages. These improvements in speed and efficiency are highly sought after in the semiconductor industry.

Among the various analog-to-digital converters, pipeline converters have become a leading choice for applications demanding both high speed and higher resolution. As the name implies, pipeline converters use a series of stages, each handling a portion of the conversion process. The final digital output is achieved by combining the results from all these stages.

A critical component in developing pipeline converters is the residue amplifier. As the name implies, the residue amplifier serves to amplify the residue, defined as the difference between a stage's input signal and the portion already resolved by that stage, in order to align it with the input of the subsequent stage.

A promising solution for residue amplification is the ring amplifier, which leverages CMOS scalability to enhance performance as it efficiently adapts to advancing semiconductor technologies, aligning with the design goals of high-speed, high-resolution, and low-power pipeline converters.

1.2 Objectives

The main objective of this dissertation is to investigate different methods of ring amplification for incorporation into a pipeline analog-to-digital converter.

To ensure successful integration into a 3-bit pipeline architecture, the ring amplifier must meet specific requirements associated with the converter. These specifications include a feedback gain of 4 V/V, a THD level of at least -60 dB, an ENOB of no less than 10 bits, a load capacitance of 400 fF, and a supply voltage of 1.2 V. The amplifier is implemented using 65 nm CMOS technology. Additionally, to maintain stability, it must have a phase margin of at least 60° and a minimum amplifier gain of 40 dB.

Keeping this focus in mind, the study will examine two distinct ring amplifier configurations. The first approach will analyze a basic ring amplifier setup to understand its fundamental operation. The second approach will involve designing a more sophisticated ring amplifier configuration optimized for two specific performance metrics: one for open-loop gain and the other for gain-bandwidth product, while also considering efficiency and trying to meet the desired specifications.

1.3 Structure

This document is divided into four chapters, outlined below. Every chapter contains a brief introduction and conclusion to maintain a smooth flow of the text.

In Chapter 1 the study's framework is outlined, providing clear insights into the motivations and objectives. In addition, a detailed description of the overall structure is provided.

Chapter 2 establishes the theoretical background crucial to comprehend the context of the study. It covers essential concepts related to analog-to-digital converters and amplifiers. Additionally, it explores various techniques for residue amplification in pipeline converters, with a particular emphasis on ring amplifiers. The chapter provides an extensive survey of the evolution of ring amplification technology over the years and concludes with a detailed explanation and comparison of the most significant configurations in the field.

Chapter 3 begins by introducing the conventional ring amplifier, offering an exhaustive theoretical analysis of the circuit, supported by simulations to examine its operation. The chapter also presents the standard testbench used to evaluate the critically damped amplifier, laying the foundation for the main analysis. It then transitions to a practical approach, discussing the critically damped ring amplifier, providing a comprehensive circuit analysis, and introducing a detailed design methodology that incorporates graphical techniques based on the g_m/I_D method. This methodology is applied to two distinct designs: one optimized for open-loop gain and the other for gain-bandwidth product, with careful attention to power dissipation in both. The chapter concludes with simulation results that validate the designs and a comparison between theoretical predictions and practical outcomes.

Finally, Chapter 4 concludes the dissertation by summarizing its key contributions to the field, discussing its limitations, and suggesting potential future work that could build upon the research presented.

The annexes provide all the tables and code used in the design process, the testbenches employed for electrical simulations, and a table highlighting the evolution of ring amplifiers in pipeline converters to support the literature review.

2

Theory Fundamentals

2.1 Introduction

This chapter presents an overview of the essential principles behind analog-to-digital converters, delving into different architectures and their characteristics, with a particular focus on pipeline implementation. It then offers a detailed explanation of residue amplification techniques, particularly ring amplification. The fusion of pipeline architecture with residue amplification techniques takes center stage in this exploration.

2.2 Analog-to-Digital Converter

2.2.1 Basic Concepts

An analog-to-digital converter (ADC) is a device that converts an analog input signal, V_{IN} into a digital output code, B_{OUT}. The input signal is previously sampled at a frequency f_S and quantified to 2^N possible output values, corresponding to $2^N - 1$ transition voltages, where N is the converter resolution. A conversion range, V_{REF} sets an input voltage limit [6]. Figure 2.1 shows the symbol representation of an ADC.

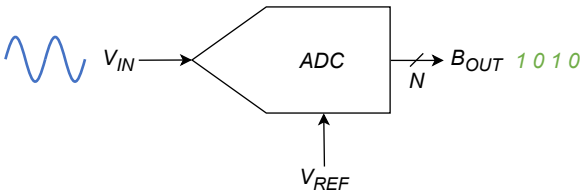


Figure 2.1: ADC symbol.

Sampling must comply with the Shannon-Nyquist theorem, as expressed in Equation (2.1), where BW represents the bandwidth of the analog signal, and $f_S/2$ corresponds to the Nyquist frequency f_N, ensuring accurate reconstruction of the analog signal.

$$f_s > 2BW \tag{2.1}$$

In a digital word like the one represented by Equation (2.2), the most significant bit (MSB) is the leftmost bit (b_{N-1}) and has the most weight, on the other hand, the least significant bit (LSB) is the rightmost bit (b_0) and has the lowest weight [6].

$$B_{OUT} = b_{N-1}2^{N-1} + ... + b_2 2^2 + b_1 2^1 + b_0 2^0 \tag{2.2}$$

The LSB voltage V_{LSB} corresponds to the minimum change in the input voltage that results in a change of the LSB. This is expressed by Equation (2.3).

$$V_{LSB} = \frac{V_{REF}}{2^N} = \frac{2V_R}{2^N} \tag{2.3}$$

The transition voltages are derived from Equation (2.4).

$$V_{T_K} = k \cdot \frac{V_{LSB}}{2} - V_R, k \in \left[1, 2^N - 1\right] \tag{2.4}$$

A simple way to relate the input with the output is characterized by Equation (2.5).

$$B_{OUT} = \left\lfloor \frac{V_{IN}}{V_{LSB}} \right\rfloor \tag{2.5}$$

In theory, the transfer function of an ADC is expected to be a straight line. However, in practice, it adopts a staircase configuration with a width of a single step equivalent to 1 LSB. Limitations imposed by the materials used in the converter implementation introduce deviations from the ideal transfer function, leading to an irregular staircase real transfer function [56]. Figure 2.2 illustrates the ideal transfer function of a 3-bit ADC.

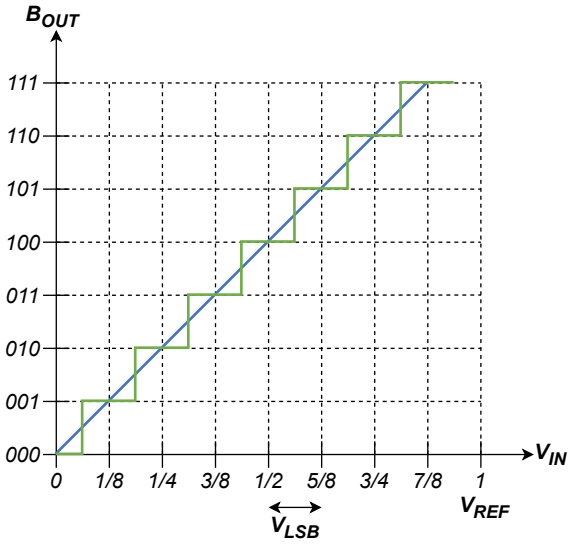


Figure 2.2: Ideal transfer function of a 3-bit ADC.

The discrepancy between continuous analog and discrete digital scales causes quantization error ϵ_Q, where different input signals produce identical digital outputs [56]. This error is modeled in Figure 2.3, where it is represented as the difference between the analog signal and the constrained digital value, known as quantization noise voltage V_{QN}.

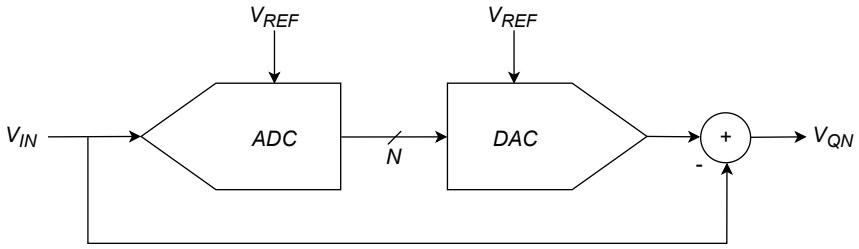


Figure 2.3: Quantization noise model.

The quantification noise voltage is limited within ±1/2 LSB, implying that input voltages within this range of the center of a step are resolved to the same digital code.

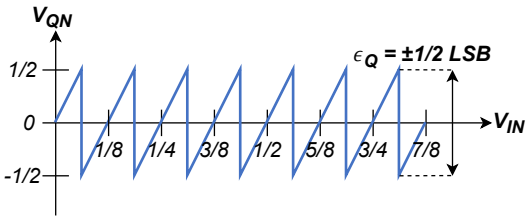


Figure 2.4: Quantization error in a 3-bit ADC.

This error can be quantified in terms of quantization noise power (P_{QN}) as described by Equation (2.6). It's worth mentioning that higher resolution leads to a decrease in this value, which improves the overall accuracy of digital representation.

$$P_{QN} = \frac{V_{LSB}^2}{12} \tag{2.6}$$

Static Performance

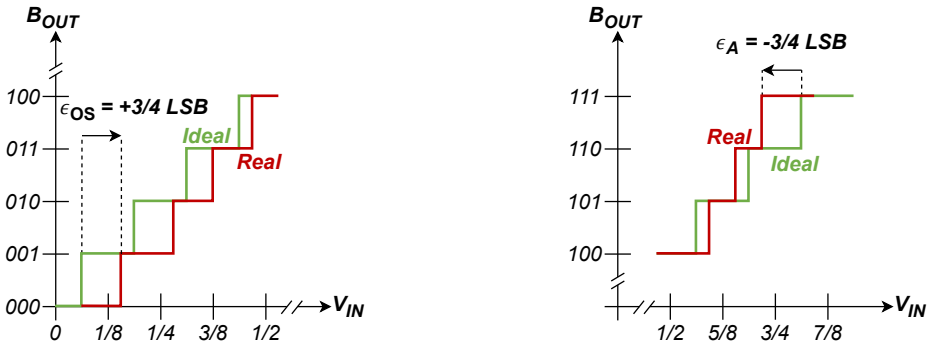
The offset error ϵ_{OS} is the deviation between the origin of the real transfer function and the ideal transfer function [56]. A positive value implies that the real function is shifted to the right of the ideal one, while a negative value suggests it is shifted to the left of the ideal function. This error affects all codes uniformly and can be corrected through a calibration process. This measure can be given in LSB by Equation (2.7).

$$\epsilon_{OS} = \frac{V_{T_{1_{real}}} - V_{T_{1_{ideal}}}}{V_{LSB_{ideal}}} \tag{2.7}$$

The gain error ϵ_A is the difference between the slope of the real transfer function and the ideal transfer function [56]. The characteristics observed in the previous error are also evident here. This error can be expressed in units of LSB using Equation (2.8).

$$\epsilon_A = \frac{V_{T_{2^n-1_{real}}} - V_{T_{2^n-1_{ideal}}}}{V_{LSB_{ideal}}} \tag{2.8}$$

An example illustrating these linear errors is shown in Figure 2.5, demonstrating an offset error of +3/4 LSB (2.5a) and a gain error of −3/4 LSB (2.5b).



(a) Offset error. (b) Gain error.

Figure 2.5: Linear errors in a 3-bit ADC.

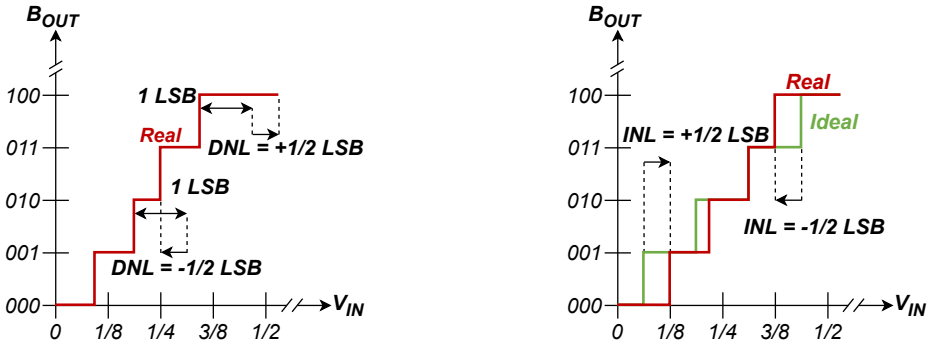
The differential nonlinearity (DNL) [35] is the difference between a real step size and the ideal value of 1 LSB, without linear errors. A positive value indicates that the step is larger than 1 LSB, whereas a negative value suggests it is smaller. A DNL less than 1 LSB guarantees no missing codes. This measure is calculated by Equation (2.9), for $2^N - 1$ points [36].

$$DNL[k] = \frac{V_{T_{real}}[k+1] - V_{T_{real}}[k]}{V_{LSB_{ideal}}} - 1, k \in [0, 2^N - 2] \tag{2.9}$$

The integral nonlinearity (INL) [35] is the dissimilarity between the real and ideal transition voltage, excluding linear errors. Also, it can be interpreted as a sum of DNL. A positive value implies that the real transition voltage falls to the right of the ideal transition voltage, while a negative value signifies it is to the left. A INL less than 1/2 LSB guarantees no missing codes. This measure is calculated by Equation (2.10), for 2^N points [36].

$$INL[k] = \frac{V_{T_{real}}[k] - V_0}{V_{LSB_{ideal}}} - k, k \in [0, 2^N - 1] \tag{2.10}$$

A demonstrative instance of these nonlinear errors is showcased in Figure 2.6, depicting a DNL of $-1/2$ and $+1/2$ LSB (2.6a) and a INL of $-1/2$ and $+1/2$ LSB (2.6b).



(a) DNL. (b) INL.

Figure 2.6: Nonlinear errors in a 3-bit ADC.

Dynamic Performance

The signal-to-noise ratio (SNR) [35], defined by Equation (2.11), quantify signal quality by expressing the ratio of the signal power P_S to noise power P_N) and is given in dB [67].

$$SNR = 10 \log_{10} \left(\frac{P_S}{P_N} \right) \tag{2.11}$$

The ideal SNR, considering only quantization noise, is defined by Equation (2.12).

$$SNR_{ideal} = 6.02N + 1.76 \tag{2.12}$$

The total harmonic distortion (THD) [35], defined by Equation (2.13) and expressed in dB, measures the degree to which harmonic components contribute to the distortion of the signal, by calculating the ratio of the harmonic distortion power P_D to the signal power P_S. Typically, five harmonics are considered to calculate it [67].

$$THD = 10 \log_{10} \left(\frac{P_D}{P_S} \right) \tag{2.13}$$

The signal-to-noise and distortion ratio (SNDR) [35], given by Equation (2.14) and expressed in dB, is a measure of the performance of an ADC. This value is obtained as the ratio of the signal power P_S to the combined power of noise P_N and distortion P_D components [67].

$$SNDR = 10 \log_{10} \left(\frac{P_S}{P_N + P_D} \right) \tag{2.14}$$

The effective number of bits (ENOB) [35], specified by Equation (2.15), is a measure of the accuracy of an ADC. It represents the actual resolution the converter can achieve, accounting for the impact of noise and distortion [67].

$$ENOB = \frac{SNDR - 1.76}{6.02} \tag{2.15}$$

The spurious free dynamic range (SFDR) [35], given by Equation (2.16), is a measure of the range an ADC can precisely perform conversions. This value is obtained as the ratio of the signal power P_S to the power of the highest spur P_H [67].

$$SFDR = 10 \log_{10} \left(\frac{P_S}{P_H} \right) \tag{2.16}$$

The measurements mentioned above are based on the application of a fast Fourier transform (FFT) with a bin count equal to half the number of samples used and the input frequencies must be carefully chosen according to Equation (2.17) where the number of cycles and the number of FFT samples must be mutually prime numbers to suppress spectrum leakage, which can otherwise distort the spectral analysis. Also, is recommended to use the Blackman window [35].

$$f_{IN} = \frac{N_{cycles}}{N_{FFT}} \cdot f_s \tag{2.17}$$

Figure 2.7 illustrates the spectrum of a typical sine signal, offering a visual representation to enhance understanding of the previously discussed measurements. In this spectrum, the DC component is identified as DC, the fundamental signal carrier is labeled as S, the highest spur is marked as H, the distortion carriers are denoted as D, and the noise floor is represented as N.

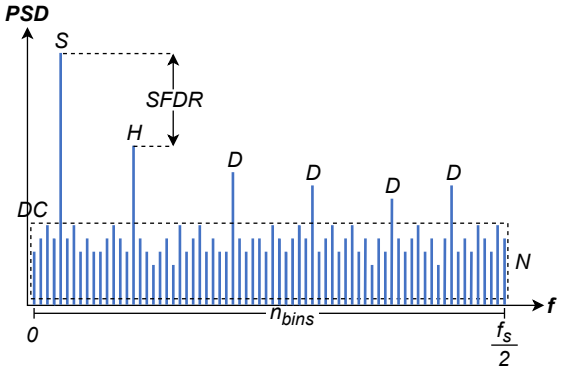


Figure 2.7: Spectrum of a sine signal.

The figure of merit (FOM) evaluates the performance of an ADC. It can be defined in various ways, such as the Walden FOM (FOM_W), given by Equation (2.18) and expressed in J/cs, and the Schreier FOM (FOM_S), given by Equation (2.19) and expressed in dB [12]. Also, the dissipation power is denoted as P_d.

$$FOM_W = \frac{P_d}{2^{ENOB} \cdot 2BW} \tag{2.18}$$

$$FOM_S = SNDR + 10\log_{10}\left(\frac{BW}{P_d}\right) \tag{2.19}$$

2.2.2 Flash ADC

A N resolution flash ADC, as depicted in Figure 2.8, consists of $2^N - 1$ comparators and a resistive divider with 2^N resistors, which produce $2^N - 1$ reference voltages meticulously spaced 1 LSB apart. Each comparator outputs a logical high when the input (V_{IN}) is greater than the applied reference voltage (V_{REF}). The array of outputs forms a thermometer code (B_T), transitioning from 'ones' to 'zeros' where the input becomes less than the respective reference voltage. An encoder then processes this thermometer code to produce the binary digital output (B_{OUT}) [57] [72].

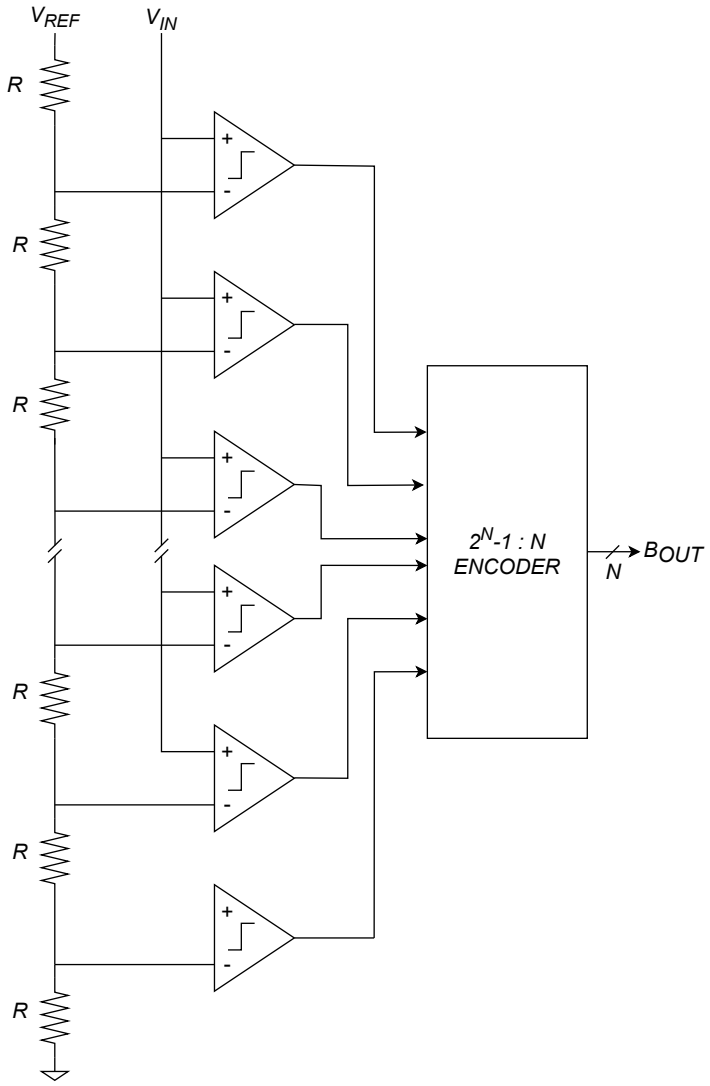


Figure 2.8: Flash ADC block diagram.

The comparator consists of a differential amplifier, which minimizes kickback, followed by a track-and-latch (TL) stage employing positive feedback to rapidly generate full-swing digital signals, as illustrated in Figure 2.9 [6].

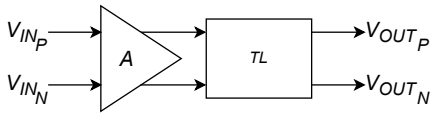


Figure 2.9: Comparator block diagram.

A fat-tree encoder consisting of OR gates, like the one represented in Figure 2.10, can be used to process the thermometer code. A $2^N - 1$ code is assigned to the leaf nodes, while the roots of the tree receive a N-bit binary code. This digital implementation is recognized for having high speed and low power consumption [47].

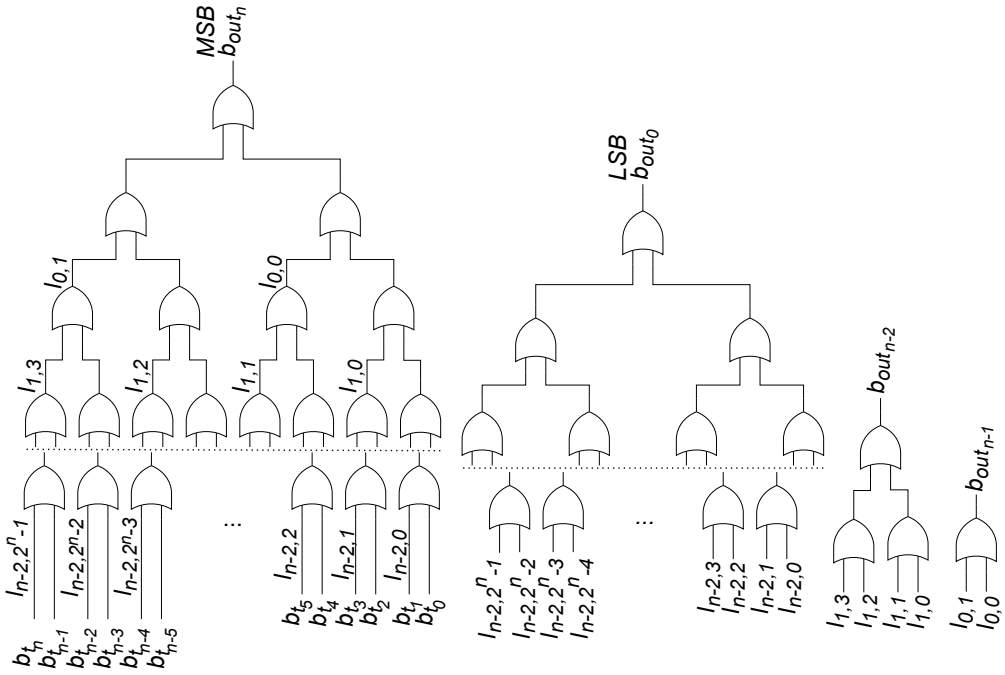


Figure 2.10: Fat-tree encoder block diagram.

Figure 2.11 showcases an example of a 3-bit code derived from a thermometer code, providing a visual representation of the digital encoding scheme employed.

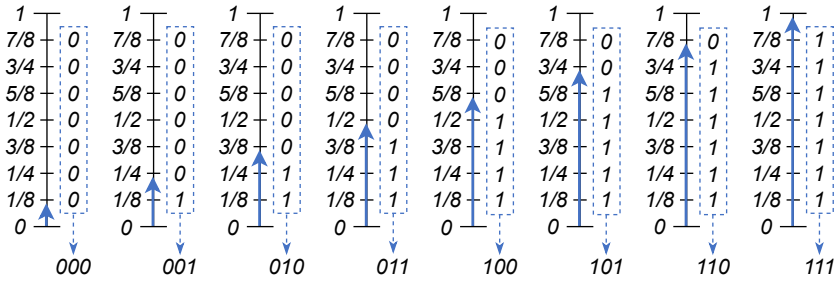


Figure 2.11: 3-bit code derived from a thermometer code.

The flash architecture has the advantage of being very fast, performing conversions in a single cycle leading to very low latency. This makes it particularly suitable for applications demanding very high frequencies. However, it has the disadvantage of consuming a lot of power and taking up a considerable amount of space. As the resolution rises, an exponential component increase is required, leading to an associated rise in power dissipation. This aspect confines the architecture to relatively low resolutions [57] [3].

2.2.3 Sigma-Delta ADC

A sigma-delta ADC, as shown in Figure 2.12, is implemented with an anti-aliasing analog filter, a sample-and-hold (SH), a sigma-delta modulator, and a decimation digital filter. Initially, the anti-aliasing filter restrings the input signal (V_{IN}) to frequencies less than the

Nyquist frequency. Subsequently, the continuous-time signal (V_N) undergoes sampling by the SH and the sigma-delta modulator processes the acquired signal into a bit stream (B_S), characterized by a low-resolution and high-frequency (f_{OS}). Finally, the decimation digital filter converts the preceding signal into a digital signal (B_{OUT}) with high-resolution and low-frequency (f_S) [6].

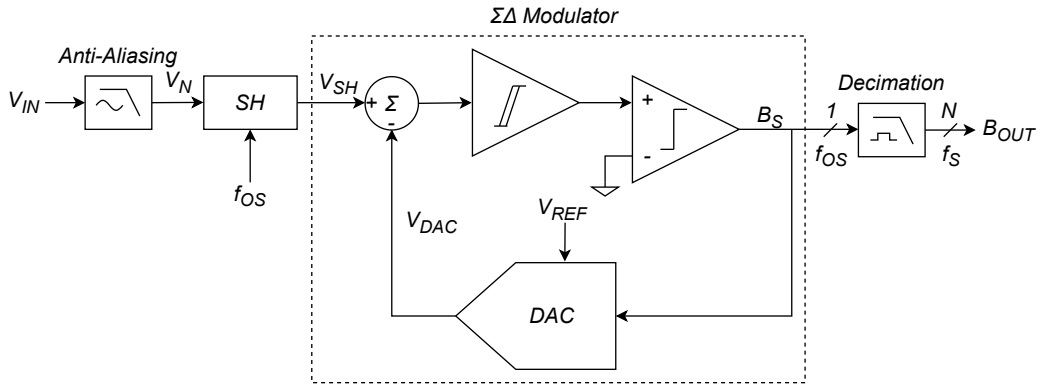


Figure 2.12: Sigma-delta ADC block diagram.

This architecture applies the technique of oversampling, which involves sampling at a much higher frequency than the Nyquist frequency, as indicated by the oversampling ratio (OSR) in Equation (2.20). By doing so, the quantization noise is distributed across a wider frequency range. This allows effective noise filtering, contributing to an improvement in the ideal SNR, as described in Equation (2.21) [57] [14].

$$OSR = \frac{f_S}{2BW} \tag{2.20}$$

$$SNR_{ideal} = 6.02N + 1.76 + \log_{10}(OSR) \tag{2.21}$$

Additionally, noise shaping is implemented, where integration is used in a feedback loop. This method serves as a high-pass filter for quantization noise. If a filter is applied to the loop, it removes more noise than simple oversampling. Consequently, an improvement in the ideal SNR is observable, for first-order integration and even more drastic for second-order integration, as outlined in Equation (2.22) [57] [14].

$$SNR_{ideal} = \begin{cases} 6.02N + 1.76 + 30\log_{10}(OSR) - 5.17, M = 1 \\ 6.02N + 1.76 + 50\log_{10}(OSR) - 12.59, M = 2 \end{cases} \tag{2.22}$$

Figure 2.13 illustrates a noise spectrum of the techniques associated with sigma-delta conversion. It provides insights into quantization noise within the Nyquist band, the dispersion of quantization noise to higher frequencies through oversampling, and the impact of quantization noise shaping. The digital filter allows for the observation that oversampling results in significant noise reduction, and noise shaping amplifies this reduction even further.

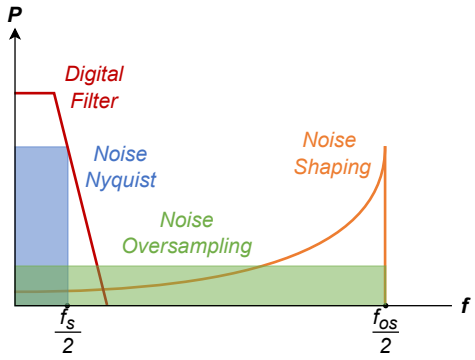


Figure 2.13: Noise spectrum of the techniques used in sigma-delta conversion.

The anti-aliasing analog filter can be achieved through a simple low-pass RC filter, as illustrated in Figure 2.14 [6].

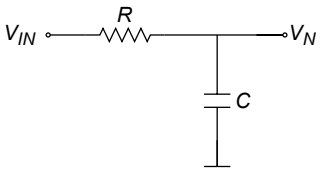


Figure 2.14: Low-pass RC filter schematic.

The transfer function of the anti-aliasing analog filter is described by Equation (2.23), where $\tau = RC$ represents the time constant. This time constant is related to the cutoff frequency f_c by the formula $f_c = \frac{1}{2\pi\tau}$.

$$AF(s) = \frac{V_{IN}(s)}{V_N(s)} = \frac{1}{1 + RCs} \tag{2.23}$$

The SH can be executed as demonstrated in Figure 2.15. In this configuration, when ϕ_{CLK} is high the capacitor follows V_N, and when it is low, the capacitor ideally maintains a constant value equal to the value when ϕ_{CLK} went low [6].

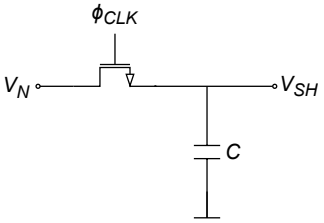


Figure 2.15: Sample and hold schematic.

The M order sigma-delta modulator comprises an integrator, a comparator, and a single-bit digital-to-analog converter (DAC). The subtraction of the DAC output voltage (V_{DAC}) from the sampled signal (V_{SH}) is fed into the integrator. The integrator accumulates this difference with its previous output. Afterward, the resulting signal is passed through the comparator, which converts it into a single-bit value—set to a logical high for a positive

signal and a logical low for a negative one. This value becomes the input for the DAC, determining whether to set to a positive or negative reference voltage. The density of 'ones' in the modulator output is directly proportional to the characteristics of the input signal. A greater number of 'ones' is generated for an increase in the input signal, and vice versa for a decrease [3] [14]. The linear model of this component is depicted in Figure 2.16, where is implied the comparator's quantification noise voltage (V_{QN}).

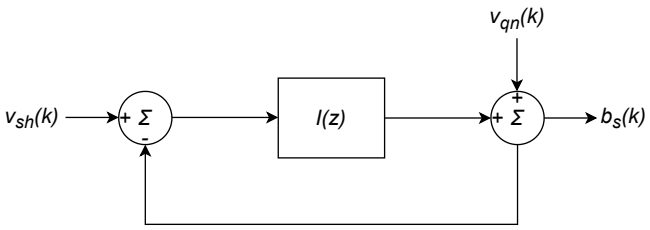


Figure 2.16: Sigma-delta modulator linear model.

The signal and noise transfer functions for both first-order and second-order systems are given by Equation (2.24) and Equation (2.25), respectively.

$$S(z) = \frac{B_S(z)}{V_{SH}(z)} = z^{-1} \tag{2.24}$$

$$N(z) = \frac{B_S(z)}{V_{QN}(z)} = \begin{cases} 1 - z^{-1} \\ (1 - z^{-1})^{-2} \end{cases} \tag{2.25}$$

The bit stream combines signal and noise contributions, as shown in Equation (2.26).

$$B_S(z) = S(z)V_{SH}(z) + N(z)V_{QN}(z) \tag{2.26}$$

The digital decimation filter can be implemented as a finite impulse response (FIR) low-pass sinc filter with an order of $M + 1$, as shown in Figure 2.17. This filter design consists of $M + 1$ integrators followed by $M + 1$ differentiators [6].

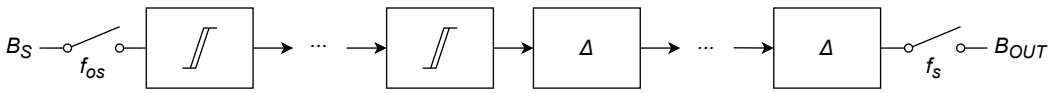


Figure 2.17: Low-pass sinc filter block diagram.

Equation (2.27) provides the transfer function of the decimation filter.

$$DF(z) = \frac{B_S(z)}{B_{OUT}(z)} = \frac{1}{OSR^{M+1}} \left(\frac{1 - z^{-OSR}}{1 - z^{-1}} \right)^{M+1} \tag{2.27}$$

The integration of oversampling and noise-shaping techniques makes this architecture suitable for low-frequencies and high-resolution applications. Despite its low power consumption, a drawback arises in its latency attributed to the decimation filter, exceeding significantly that of other types [57] [3].

2.2.4 SAR ADC

Figure 2.18 represents the block diagram of a successive-approximation register (SAR) ADC, where it compromises a SH, a comparator, a SAR, and a DAC. In this setup, the input signal is sampled and held before being compared to successive reference voltages imposed by both the SAR and the DAC, facilitated by a high-speed comparator [57].

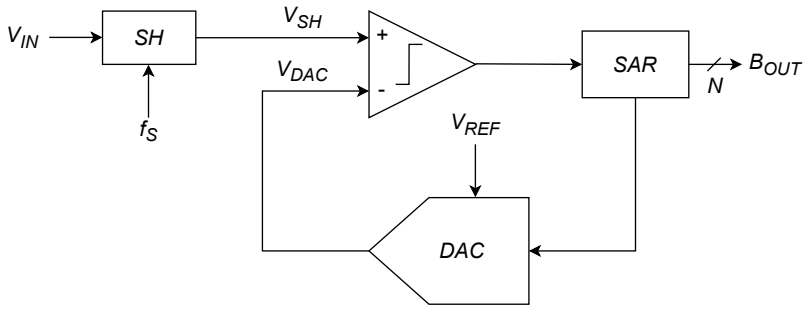


Figure 2.18: SAR ADC block diagram.

The binary search algorithm, illustrated in Figure 2.19, is used to generate reference voltages and determine the closest digital representation of an analog input signal. Initially, the input signal (V_{IN}) is sampled and held (V_{SH}), and the N-register is set to midscale, thereby causing the output of the V_{DAC} to be $V_{REF}/2$. Following this, a comparison is conducted to determine whether V_{IN} is greater or less than V_{DAC}. If V_{IN} is greater, the comparator outputs a logic high, keeping the MSB in the register at 1, and V_{DAC} is increased by $V_{REF}/4$. Otherwise, if V_{IN} is less, the comparator outputs a logic low, the MSB on the register is set to 0, and V_{DAC} is decreased by the same amount. This iterative process continues, with V_{DAC} adjustment voltages progressively decreasing until N iterations are completed, resulting in the acquisition of the LSB [58] [74].

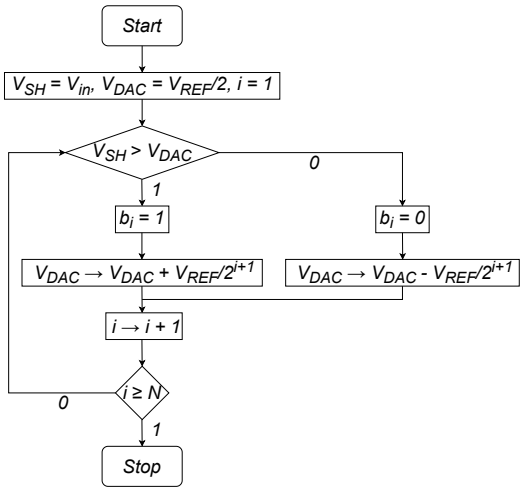


Figure 2.19: Binary search algorithm flowchart.

Figure 2.20 illustrates a 3-bit conversion example with a 1 V input. Initially, the register is set to 100_2, and V_{DAC} is 1/2 V. In the first comparison, where $V_{IN} > V_{DAC}$, the MSB is 1,

keeping the register at 100_2. V_{DAC} is then updated to 3/4 V for the next comparison. Since $V_{IN} < V_{DAC}$, the next bit remains 0, updating the register to 101_2 and setting V_{DAC} to 5/8 V. Finally, as $V_{IN} < V_{DAC}$, the LSB is set to 0. The final output is 100_2, and a new input signal is prepared for sampling [58].

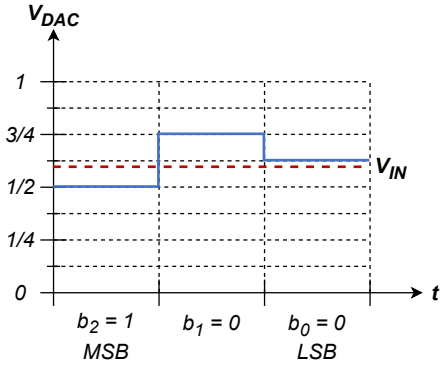


Figure 2.20: Binary search algorithm for 3-bit.

One straightforward approach to incorporate both the SH and the DAC functionalities into a single switch-capacitor circuit is to employ the charge-redistribution architecture with merged capacitor switching (MCS) technique, depicted in Figure 2.21. This setup employs an array of 2^N capacitors binary weighted. The conversion proceeds as follows:

1. *Sample mode*: Switch S_0 connects to V_{IN}, switch S_1 and the array switches $b_{N-1}...b_0$ connect to a common terminal, and switch S_2 closes. This configuration charges all capacitor bottom plates with V_{IN}.
2. *Hold mode*: Switch S_0 is connected to V_{REF}, switch S_1 and the array switches link to ground, and switch S_2 opens. This sets V_{DAC} to $-V_{IN}$.
3. *Redistribution mode*: Switch b_{N-1} connects to V_{REF} while the other switches are grounded, setting V_{DAC} to $-V_{IN} + V_{REF}/2$. If $V_{IN} > V_{REF}/2$, V_{DAC} is negative, and the comparator outputs a logic high, keeping the MSB at V_{REF}. If $V_{IN} \leq V_{REF}/2$, the MSB is grounded, and the comparator outputs a logic low. This process is iterated N times, progressively switching smaller capacitors to determine the LSB. Throughout, switch S_1 remains connected to ground [74] [64].

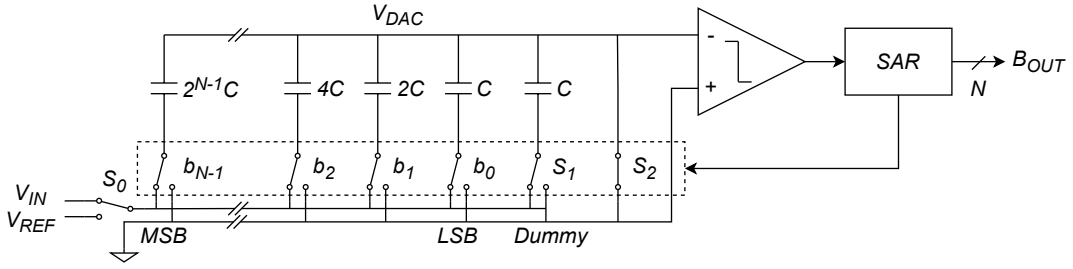


Figure 2.21: Charge-redistribution SAR ADC schematic.

This architecture offers versatility in both frequency and resolution, demonstrating efficient power use. As the resolution increases, there is a corresponding increase in the number of comparisons, creating a conflict with the sampling rate. Since the digital output data is accessible at the end of the sampling period, there is no latency. Furthermore, increased resolutions may necessitate calibration [57] [3].

2.2.5 Pipeline ADC

The pipeline implementation, as the name suggests, consists of a sequence of stages depicted in Figure 2.22. Each stage, except the final one, which is essentially an ADC, includes a SH, an ADC, and a multiplying digital-to-analog converter (MDAC). The latter comprises a DAC, a different amplifier, and a residue amplifier (RA). Within each stage, except the final one, the input signal (V_{IN}) is subject to both sampling and holding (V_{SH}) followed by quantification to the stage resolution through the SH and the ADC, respectively. The resulting digital representation is then converted back to analog (V_{DAC}) by the DAC. Subsequently, the analog output is subtracted from the input signal leading to a residue voltage (V_{RES}) that is then amplified (V_{RESA}) by a factor of 2^{n-1}, being n the stage resolution, before being sampled by the next stage. Ultimately, the final stage, m resolves the remaining bits and establishes a definitive resolution [57] [73].

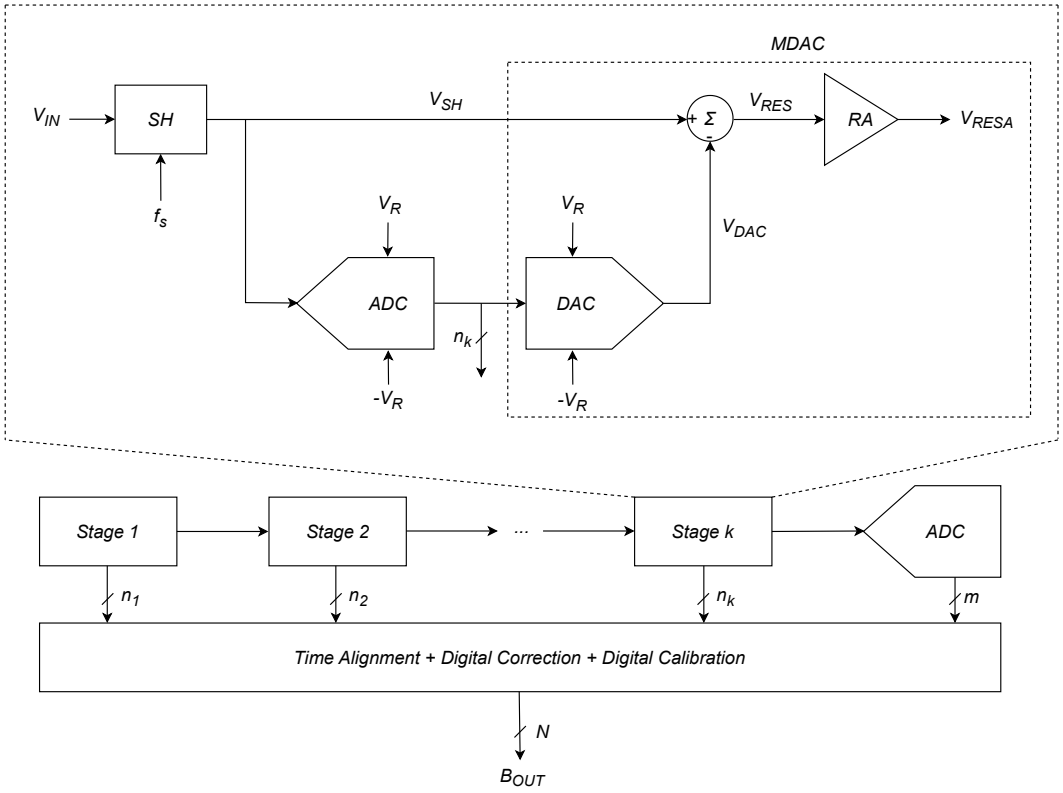


Figure 2.22: Pipeline ADC block diagram.

To achieve digital correction, the resolution N of the system can be determined using Equation (2.28).

$$N = n_1 + n_2 - 1 + ... + n_k - 1 + m - 1 \tag{2.28}$$

The MDAC can be implemented using a flip-around configuration, as illustrated in Figure 2.23. The conversion begins with the input signal sampled by the two capacitors, C_F and C_S, during the sampling phase ϕ_S. Next, this signal is transferred to C_F during the amplification phase ϕ_A. The voltage produced by the DAC depends on the value of the ADC, which normally is a flash ADC to achieve high throughput. This sets the voltage within the range of $\pm V_R$. Although the DAC is simplified to its voltage, it can be implemented using the MCS technique described earlier [68]. Additionally, this implementation can be configured for differential operation.

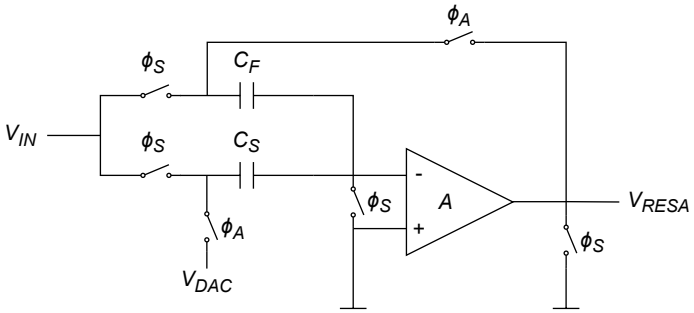


Figure 2.23: Flip-around MDAC schematic.

Equation (2.29) describes the residual voltage in this configuration.

$$V_{RESA} = \frac{A}{1 + A\frac{C_F}{C_S+C_F}} V_{IN} - \frac{A}{\frac{C_S+C_F}{C_S} + A\frac{C_F}{C_S}} V_{DAC} \tag{2.29}$$

The feedback gain A_F can be derived by considering only the input and residual voltages, as shown in Equation (2.30). For a higher amplifier gain A, typically around 40 dB as a reference, an approximation can be made while neglecting the parasitic capacitance C_P.

$$A_F = \frac{A}{1 + A\frac{C_F}{C_S+C_F}} \approx \frac{1}{\beta} = \frac{C_S + C_F}{C_S} \equiv 2^{n-1} \tag{2.30}$$

As a switched capacitor (SC) circuit, this configuration employs switches controlled by nonoverlapping phases, thereby ensuring proper sampling and transfer. These phases are generated by a single clock (T_{CLK}) and arranged so that they are never both high simultaneously, with a delay (t_d) between their edges, as evidenced in Figure 2.24 [6]. The rise time (t_r) is the transition from low to high, and the fall time (t_f) is the opposite.
The pulse width is given by Equation (2.31), where a duty cycle, D_{cycle}, of less than 50% is required to ensure nonoverlap.

$$t_p = D_{cycle} \cdot T_{clk} - (t_r + t_f) \tag{2.31}$$

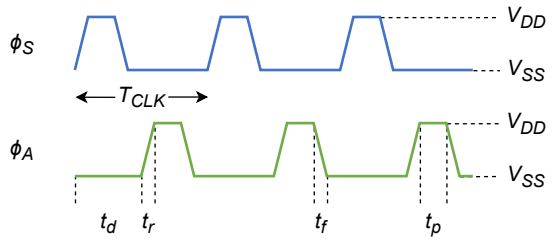


Figure 2.24: Nonoverlapping phases.

It should be emphasized that once a stage completes the conversion of a sample and transfers the residue to the next stage, another sample can be acquired, contributing to a high throughput. However, since the bits in each stage are determined at different time intervals, time alignment is necessary to synchronize the bits from a single sample. Additionally, it is crucial to note that a single sample must navigate through all stages before an output is generated, resulting in a latency that cannot be overlooked [57] [73]. In the example in Figure 2.25, the latency is approximately 7 cycles.

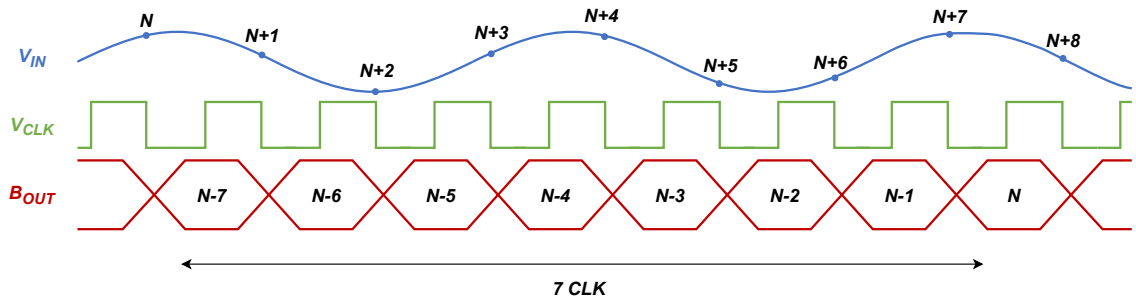


Figure 2.25: Data latency in pipeline ADC.

After a word is successfully created from a sample, digital error correction is applied to mitigate ADC comparator mismatch [70]. A method called 1-bit overlap is utilized, halving the gain and limiting the subsequent stage's input dynamic to $\pm V_R/2$. This decreases resolution by 1 bit in each stage except the first which has a full input. This allows for significant comparator offset inaccuracies in the residue voltage generation to be contained within the subsequent stage's range, allowing for correction [73]. Moreover, calibration is often necessary to enhance linearity, impacted by DAC capacitor mismatch and finite RA gain [18].

The residue amplifier, central to this study, is the element responsible for amplifying the residue, as expressed by Equation (2.32). This amplification is carried out to align with the input range of the subsequent stage. Proper amplification is crucial for maintaining signal integrity and achieving optimal performance across the entire ADC.

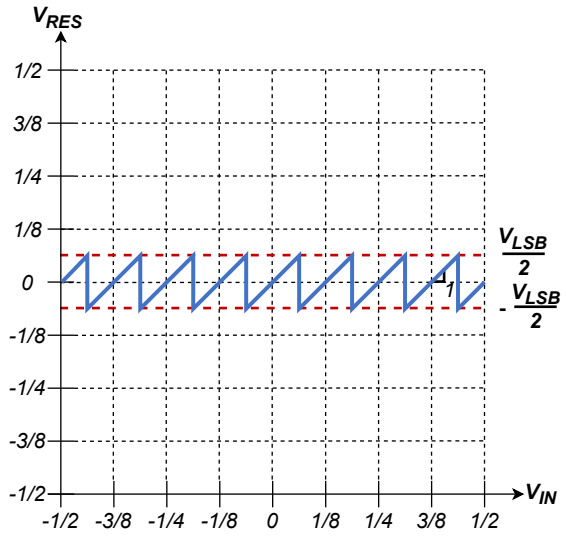
$$V_{RESA}(B) = (V_{IN} - V_{DAC}(B)) \cdot 2^{n-1} \tag{2.32}$$

Equations (2.33) and (2.34) can be used to estimate the amplifier's linearity (*LIN*) and distortion range (THD), respectively. These calculations are applied within the first stage for simplification.

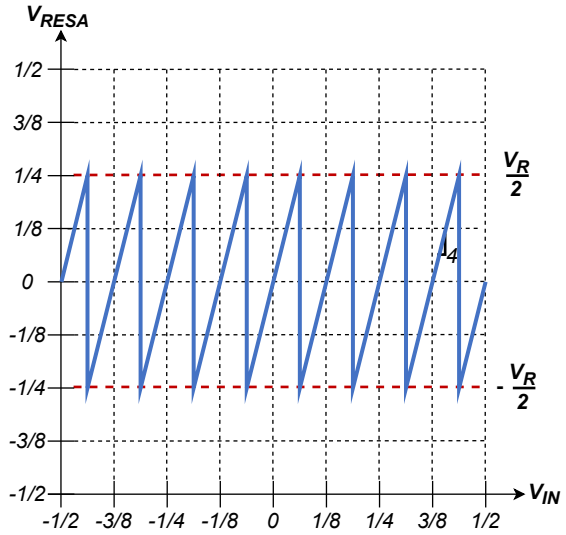
$$LIN = ENOB - n_1 + 1 \tag{2.33}$$

$$20\log_{10}(A_F) - SFDR \leqslant THD \leqslant 20\log_{10}(A_F) - SNDR \tag{2.34}$$

Figure 2.26 presents the residue voltage relative to a 3-bit stage with a 1 V differential input before and after amplification. Before amplification (2.26a), the residue is confined between $\pm 1/2V_{LSB}$ with a unitary slope. After amplification (2.26b), the residue is constrained to $\pm 1/2V_R$ with a margin of $1/4V_R$ left for comparator offset correction and slope 2^{n-1}.



(a) Ideal residue voltage before amplification.



(b) Ideal residue voltage after amplification.

Figure 2.26: Ideal residue voltage for a 3-bit stage.

Figure 2.27 illustrates the voltage progression through a three-stage pipeline, with each stage resolving 3 bits. Initially, the input signal enters stage 1, where the first 3 bits are determined as 100_2. This resolution produces the initial residue voltage, which is then passed to stage 2. In stage 2, the residue is represented as on the voltage scale. After applying a correction by subtracting 10_2, the next 3 bits are resolved as 001_2. Stage 3 follows the same process, resulting in the final 3 bits of 011_2. Combining these 3-bit sets produces a 7-bit word: 0111001_2.

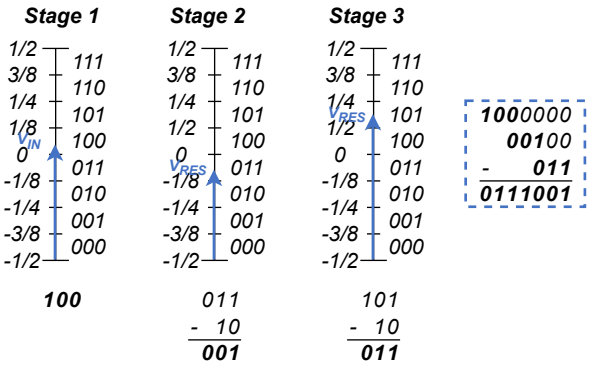


Figure 2.27: Residue voltage along a pipeline with 3-bit stages.

The pipeline architecture is well-suited for applications requiring high resolution and fast sample rates. However, achieving this resolution demands a substantial number of components, which in turn leads to higher power consumption and increased circuit complexity. Additionally, the increased resolution results in greater latency due to the need for additional stages. This latency can be a critical factor in applications where real-time processing is essential, such as in high-speed data acquisition systems and communication devices. Despite these challenges, the architecture supports high throughput by allowing multiple conversions to be processed concurrently, which enhances overall system efficiency and performance [57] [3].

2.3 Residue Amplifier

2.3.1 Basic Concepts

An amplifier is a device that enhances the amplitude of an analog signal using external power while preserving its waveform [13]. Figure 2.28 shows the amplifier symbol.

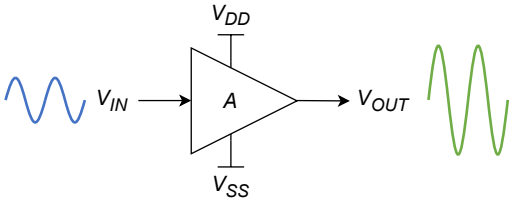
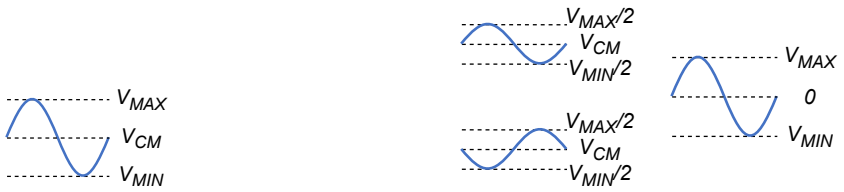


Figure 2.28: Amplifier symbol.

The amplifier can operate with either a single-supply or dual-supply configuration and supports both single-ended (2.29a) or differential signals (2.29b), each with distinct characteristics, as demonstrated in Figure 2.29.



(a) Single-ended signal. (b) Differential signal.

Figure 2.29: Types of signal.

Gain and Bandwidth

The gain of the amplifier, A quantifies the increase between the input signal and output signal, and is given by the ratio of its terminals as specified by Equation (2.35) [13].

$$V_{OUT} = A \cdot V_{IN} \tag{2.35}$$

The ideal transfer function of an amplifier, as depicted in Figure 2.30, shows a perfect linear relationship between input and output. However, in practical applications, this ideal behavior is often compromised due to component mismatches, leading to deviations from the expected performance. These discrepancies can result in nonlinearities and reduced accuracy, impacting the overall effectiveness of the amplifier.

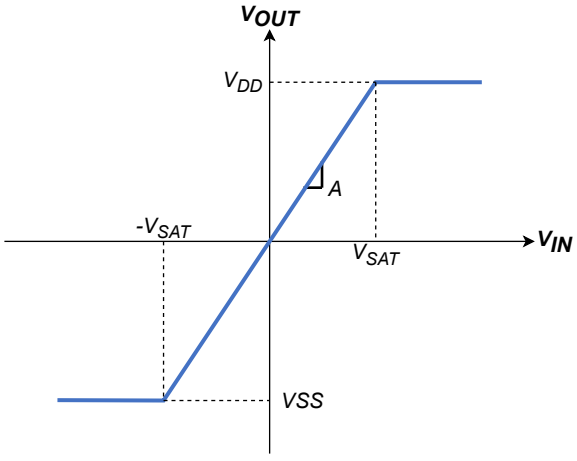


Figure 2.30: Amplifier ideal transfer function.

The bandwidth, BW of an amplifier is defined by the frequency range at which the gain has dropped 3 dB as demonstrated in Figure 2.31 and inferred by Equation (2.36). In essence, it represents the range of frequencies over which the gain is effective [13].

$$BW = f_{-3dB} \tag{2.36}$$

To relate gain and bandwidth, the gain-bandwidth product (GBW), as defined in Equation (2.37), is used. The GBW represents the linear rate of change beyond the cutoff frequency and is typically measured when the amplifier gain is set to unity, as in Figure 2.31.

$$GBW = A \cdot BW \tag{2.37}$$

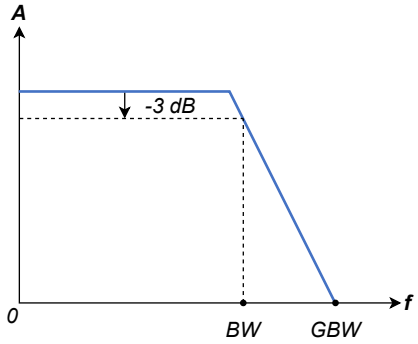


Figure 2.31: Amplifier bandwidth and gain-bandwidth product.

Input and Output Impedance

In the realm of amplifier analysis, it's essential to understand that input impedance characterizes the impedance encountered by the signal source, whereas output impedance signifies the impedance perceived by the load when the input is zero [13]. These are represented by Equations (2.38) and (2.39), respectively.

$$Z_{IN} = \frac{V_{IN}}{I_{IN}} \tag{2.38}$$

$$Z_{OUT} = \left.\frac{V_{OUT}}{I_{OUT}}\right|_{V_{IN}=0} \tag{2.39}$$

Feedback

The amplifier could be integrated into a negative feedback system illustrated in Figure 2.32. In this configuration the feedback signal is in anti-phase with the input signal, expanding the bandwidth and enhancing linearity, through decreased gain [13].

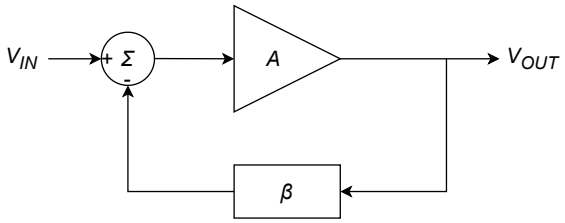


Figure 2.32: Negative feedback system.

The feedback gain, A_F, as expressed by Equation (2.40), changes according to the frequency-dependent characteristics of the amplifier gain. In contrast, the feedback factor, β, stays constant, determining the portion of the output voltage fed back to the input.

$$A_F(s) = \frac{A(s)}{1 + A(s)\beta} \tag{2.40}$$

For negative feedback, adherence to the Barkhausen stability criterion, outlined by Equation (2.41), is sufficient for marginal stability [69].

$$\begin{cases} |A(j\omega)\beta| = 1 \\ \angle A(j\omega)\beta = -\pi \end{cases} \tag{2.41}$$

The feedback bandwidth, BW_F relates to the amplifier bandwidth as in Equation (2.42).

$$BW_F = (1 + A\beta)BW \tag{2.42}$$

The amplifier gain, denoted by A, represents the open-loop gain of the amplifier, reflecting its amplification capability without feedback. In contrast, the feedback gain, represented by A_F, refers to the closed-loop gain, which is the amplifier's gain when feedback is applied to stabilize the output. Additionally, the product $A\beta$, where β is the feedback factor, is known as the loop gain. This product indicates the effectiveness of the feedback loop in controlling the overall gain of the system. These various gains are illustrated in Figure 2.33 for an amplifier integrated within a negative feedback system.

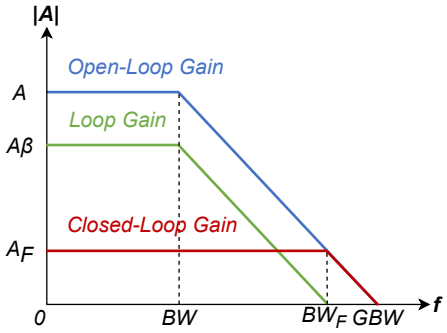


Figure 2.33: Gains of the amplifier in negative feedback.

Magnitude and Phase Margin

A straightforward method for analyzing the stability of an amplifier operating within a negative feedback system involves calculating the magnitude margin and phase margin. These margins are assessed by analyzing the Bode plots, as illustrated in Figure 2.34.

The magnitude margin is identified at the frequency where the phase reaches $-\pi$ radians, as defined by Equation (2.43). Conversely, the phase margin is determined at the frequency where the magnitude drops to 0 dB, as outlined in Equation (2.44). A commonly referenced criterion for stability is a phase margin of at least 60° [69].

$$A_M = -20 \log \left(|A(jw_{\angle A\beta=-\pi})|\beta \right) \tag{2.43}$$

$$\phi_M = \angle A(j\omega_{|A\beta|=0})\beta + \pi \tag{2.44}$$

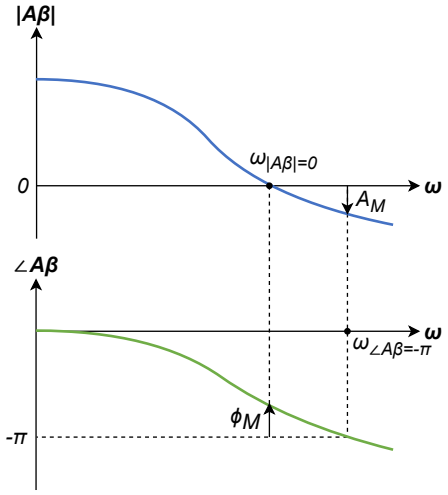


Figure 2.34: Bode magnitude and phase plots.

Slew Rate

The behavior of an amplifier in response to a step input is visually depicted in Figure 2.35. There is a slewing phase, followed by a stabilization phase until the amplifier settles into a steady state within an error band, reaching the settling time, t_s.

The slewing phase is characterized by the slew rate (SR), defined as the maximum rate at which the output changes [19]. This is expressed by Equation (2.45).

$$SR = \frac{dv_{out(t)}}{dt} \tag{2.45}$$

Equation (2.46) provides the settling time for an amplifier in negative feedback, approximated as a first-order system. It is important to note that the time constant, τ, is the reciprocal of βGBW.

$$t_s = -\frac{1}{2\pi}\frac{\ln(\epsilon)}{\beta GBW} \tag{2.46}$$

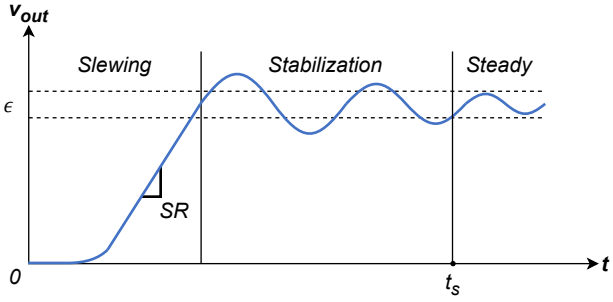


Figure 2.35: Amplifier step response.

Output Voltage Swing

The output voltage swing (OVS) is the range of output voltage where all transistors function within the active region and can be enunciated by Equation (2.47) [19].

$$V_{OUT_{SW}} = VDD - V_{DSAT} - VSS \tag{2.47}$$

Common-Mode and Power Supply Rejection Ratio

The common-mode rejection ratio (CMRR) is a parameter that measures the capability of a differential amplifier to suppress signals that are in phase and present at both inputs [60]. Equation (2.48) extrapolates this value as the ratio of the gain in the differential mode to the gain in the common mode.

$$CMRR = 20\log_{10}\left(\frac{A_{DM}}{A_{CM}}\right) \tag{2.48}$$

The gain in the differential mode is characterized by Equation (2.49), whereas the gain in the common mode is described by Equation (2.50).

$$A_{DM} = \frac{V_{OUT}}{V_{IN}} \tag{2.49}$$

$$A_{CM} = \frac{V_{OUT}}{V_{CM}} \tag{2.50}$$

The power supply rejection ratio (PSRR) is a metric that measures the amplifier's capability to attenuate variations in the power supply. It is calculated as the ratio of the gain concerning the input to the gain concerning the power supplies, with one defined for the positive supply according to Equation (2.51), and one for the negative supply according to Equation (2.52).

$$PSRR^{+} = 20\log_{10}\left(\frac{A_V}{A_{V_{DD}}}\right) \tag{2.51}$$

$$PSRR^{-} = 20\log_{10}\left(\frac{A_V}{A_{V_{SS}}}\right) \tag{2.52}$$

The gain concerning the input, positive supply, and negative supply are given by Equations (2.53), (2.54) and (2.55), respectively.

$$A_V = \frac{V_{OUT}}{V_{IN}} \tag{2.53}$$

$$A_{VDD} = \frac{V_{OUT}}{V_{DD}} \tag{2.54}$$

$$A_{VSS} = \frac{V_{OUT}}{V_{SS}} \tag{2.55}$$

Dissipated Power

The dissipated power, P_d is a function of the amplifier supply consumption and can be expressed using Equation (2.56). This calculation includes both supply voltages, V_{DD} and V_{SS} as well as the common mode voltages, $V_{CM_{IN}}$ and $V_{CM_{OUT}}$.

$$P_d = (V_{DD} \cdot I_{DD} + V_{SS} \cdot I_{SS}) + (V_{CM_{IN}} \cdot I_{CM_{IN}} + V_{CM_{OUT}} \cdot I_{CM_{OUT}}) \tag{2.56}$$

Figure of Merit

The FOM serves as an indicator of an amplifier's performance, and it is quantified by Equation (2.57). It is given by the ratio of the product of GBW and the load capacitance, C_L to the dissipated power.

$$FOM = \frac{GBW \cdot C_L}{P_d} \tag{2.57}$$

Noise

Noise refers to any undesired disturbances that disrupt the integrity of a signal in a system. To measure this quantity is normally used the power spectral density (PSD). The most noticeable noise sources in the complementary-metal-oxide-semiconductor (CMOS) circuits are thermal and flicker noise [6].

Thermal noise arises from the agitation of carriers in a semiconductor due to thermal fluctuations [19]. The PSD of this noise is given by Equation (2.58), where K_B is the Boltzmann constant, T is the temperature and g_m is the transconductance. In addition, γ is a channel size-dependent coefficient.

$$\overline{V_{TN}^2} = \frac{4K_B T\gamma}{g_m}\Delta f \tag{2.58}$$

Flicker noise is caused by impurities in the semiconductor materials and is more pronounced at low frequencies [19]. Its PSD is given by Equation (2.59), where C_{ox} is the oxide capacitance, W is the channel width and L is the channel length. Also, K_F is a conductor constant, and f is the frequency.

$$\overline{V_{FN}^2} = \frac{K_F}{C_{ox}WLf}\Delta f \tag{2.59}$$

Typically, noise performance in amplifiers is expressed as input-referred noise, which is independent of the specific circuit configuration, making it a more useful tool for comparison. It is determined by measuring the output noise and dividing it by the amplifier's gain, as shown in Equation (2.60).

$$\overline{V_{N_{IN}}^2} = \frac{\overline{V_{N_{OUT}}^2}}{A^2} \tag{2.60}$$

2.3.2 Operational Transconductance Amplifier

A fundamental approach for a residue amplifier is an operational transconductance amplifier (OTA), renowned for its high linearity and substantial gain. An OTA is an amplifier that converts an input voltage into an output current. The key parameter here is transconductance, which defines the relationship between the terminals and can be regulated by current. Figure 2.36 illustrates the symbol associated with this configuration.

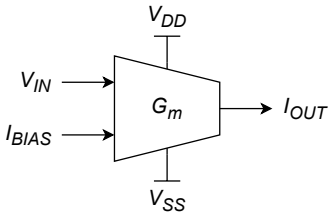
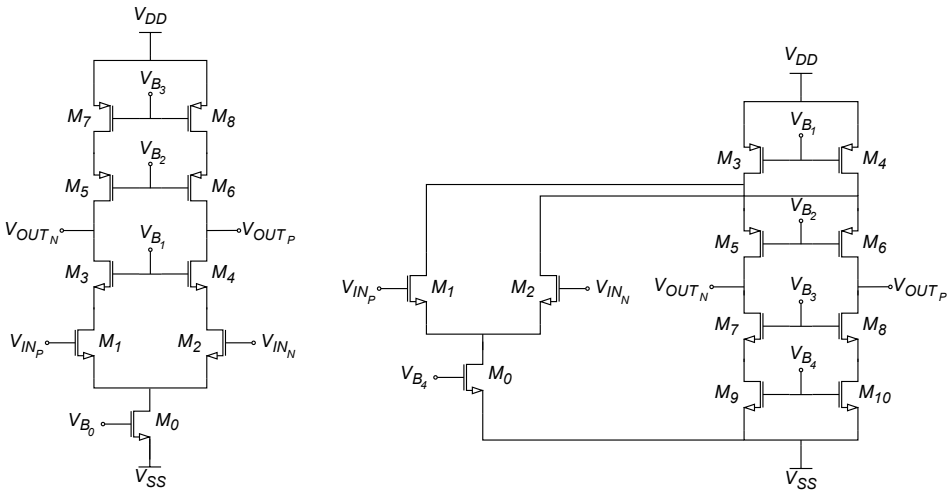


Figure 2.36: OTA symbol.

In the case of an OTA, the voltage gain is given by the product of the amplifier transconductance G_m and the output resistance R_{out}, as expressed in Equation (2.61).

$$A_v = G_m R_{out} \tag{2.61}$$

In Figure 2.37, the schematics of two fully differential cascode amplifiers are presented: the telescopic cascode [78] (2.37a) and the folded cascode [80] (2.37b).



(a) Telescopic [78]. (b) Folded [80].

Figure 2.37: Cascode amplifiers schematics.

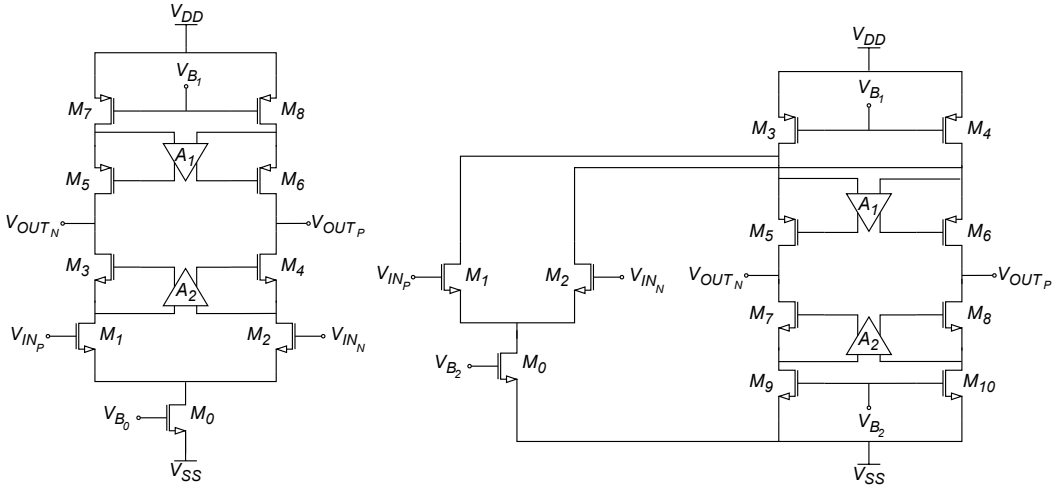
To increase the open-loop gain, one could either increase the cascode transistors or add more stages without cascode devices. However, single-stage amplifiers suffer from limited gain due to restricted output swing, while multi-stage amplifiers face bandwidth limitations due to reduced phase margin. The gain-boosting technique [78], represented in Figure 2.38, is introduced as an effective solution to address these issues. This technique

is straightforward to understand: in 2.38a, the output resistance is given by $R_{out} = r_{ds_1}\frac{g_{m_2}}{g_{ds_2}}$. In contrast, in 2.38b, with the addition of amplifier A, the output resistance increases to $R_{out} = A \cdot r_{ds_1}\frac{g_{m_2}}{g_{ds_2}}$, thereby enhancing the voltage gain by a factor of A.

(a) Cascode. (b) Gain-boosted cascode.

Figure 2.38: Gain-boosting technique [78].

Figure 2.39 presents the gain-boosted architectures: the telescopic configuration [59] in 2.39a and the cascode configuration [54] in 2.39b. Both configurations demonstrate gain-boosting on both sides of the amplifier, resulting in enhanced performance throughout the entire output range.



(a) Telescopic [59]. (b) Folded [54].

Figure 2.39: Gain-boosted amplifiers schematics.

Equations (2.62) and (2.63) provide the low-frequency voltage gain for the gain-boosted telescopic cascode and folded cascode configurations, respectively.

$$A_v = -g_{m_{1,2}}\left[\left(A_1 \cdot r_{ds_{1,2}}\frac{g_{m_{3,4}}}{g_{ds_{3,4}}}\right) \| \left(A_2 \cdot r_{ds_{5,6}}\frac{g_{m_{7,8}}}{g_{ds_{7,8}}}\right)\right] \tag{2.62}$$

$$A_v = -g_{m_{1,2}}\left[\left(A_1 \cdot r_{ds_{1,2}} \| r_{ds_{3,4}}\frac{g_{m_{5,6}}}{g_{ds_{5,6}}}\right) \| \left(A_2 \cdot r_{ds_{9,10}}\frac{g_{m_{7,8}}}{g_{ds_{7,8}}}\right)\right] \tag{2.63}$$

The transient response of this amplifier typically follows the pattern shown in Figure 2.40. It usually begins with a reset phase, during which initial conditions are established, followed by an amplification phase where the signal is processed. The response then stabilizes into a steady state, characterized by two differential outputs.

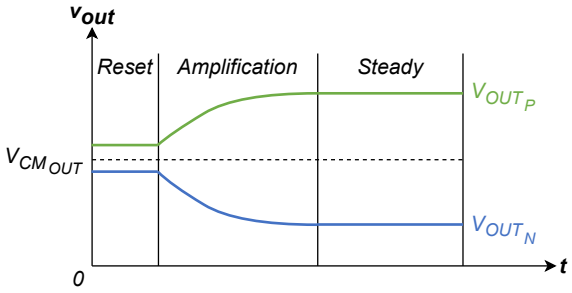


Figure 2.40: OTA transient response.

2.3.3 Dynamic Amplifier

To overcome the power dissipation issues and scaling limitations associated with OTAs, a dynamic amplifier (DA) can be an effective alternative for residue amplification, benefiting from the absence of a static current. However, since they operate in an open-loop configuration, this can result in distortion. This distortion can be mitigated through calibration techniques, which adjust the amplifier's performance to improve accuracy. The corresponding symbol is shown in Figure 2.41.

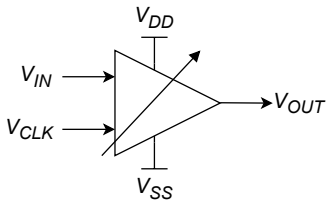


Figure 2.41: Dynamic amplifier symbol.

The conventional dynamic amplifier is discussed in [66], highlighting how the common-mode detection (CMD) facilitates a stable output for the subsequent stage. The corresponding circuit diagram is presented in Figure 2.42. During the reset phase, when the clock (V_{CLK}) is low, both outputs (V_{OUT_P} and V_{OUT_N}) are precharged to V_{DD}. The amplification phase starts when the clock signal rises, leading transistors M_1 and M_2 to discharge the loads (C_L) in proportion to their transconductance. As the outputs approach the optimal common-mode voltage ($V_{CM_{OUT}}$), the CMD generates a stop signal (V_{stop}) that disables the switches, concluding the amplification phase and reaching a steady state.

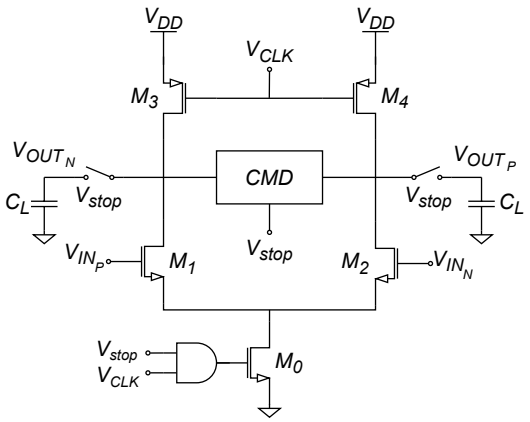


Figure 2.42: Conventional dynamic amplifier schematic [66].

The transient response of the amplifier described earlier is graphically depicted in Figure 2.43 for clearer visualization.

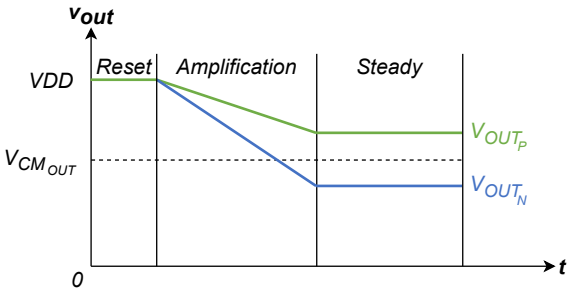


Figure 2.43: Dynamic amplifier transient response [66].

The voltage gain of the amplifier is described by Equation (2.64), where G_m, C_L, and T_A are the transconductance of the input transistors, the load capacitance, and the amplification time, respectively.

$$A_v = \frac{G_m}{C_L} \cdot T_A = \frac{g_{m_1} + g_{m_2}}{2} \cdot \frac{T_A}{C_L} \tag{2.64}$$

However, this gain is nonlinear to the input signal. To address this challenge, the capacitive degeneration linearization (CDL) technique [1] is employed, where transistors operate in the weak inversion saturation region. This technique addresses two types of nonlinearity: compression and expansion.

Compressing nonlinearity (Figure 2.44) occurs when a tail-current source with a high source impedance biases the differential pair. In this setup, the effective transconductance of the differential pair is determined by the smaller transconductance of the two transistors. As the input signal increases, the current source forces one transistor to conduct less current, causing its transconductance to decrease. Consequently, the overall transconductance of the differential pair decreases, resulting in compressive behavior.

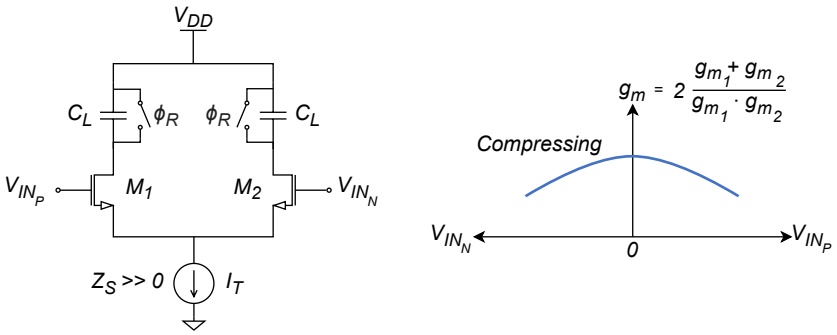


Figure 2.44: Differential pair with tail-current source [1].

Conversely, expanding nonlinearity (Figure 2.45) occurs when the differential pair is connected directly to ground, where the source impedance is zero. In this case, the transistors are dependent and the total transconductance is simply the average of the individual transconductances, with the larger one having a greater influence. As the input signal increases, both transconductances will increase. As a result, the transconductance of the differential pair also rises, leading to an expansion response.

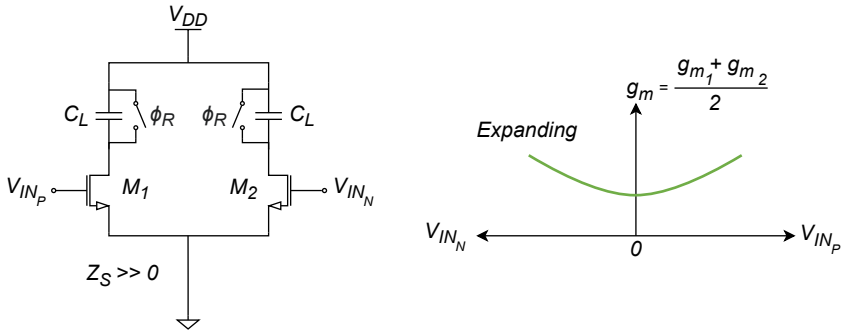


Figure 2.45: Differential pair with ground [1].

Figure 2.46 presents half of an amplifier that uses the CDL technique. The amplifier is degenerated by a capacitor on the source (C_{DEG}) and the load (C_L) is added on the drain forming the small signal-gain as $\frac{C_{DEG}}{C_L}$.

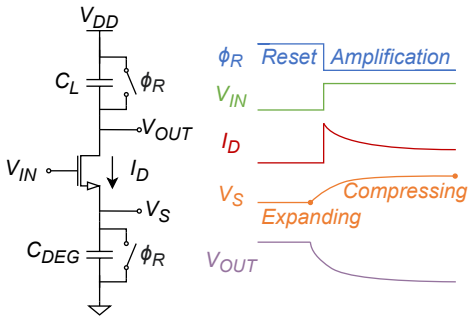


Figure 2.46: Time-domain analysis of a dynamic amplifier half-circuit with CDL [1].

The circuit operates in two phases: reset and amplification. During the reset phase, capacitors C_{DEG} and C_L are charged to ground and the supply voltage, respectively. When

the amplification phase begins and an input step (V_{IN}) is applied, the drain current (I_D) charges the capacitors. Since C_L is smaller than C_{DEG}, the drain voltage (V_{OUT}) increases faster than the source voltage (V_S), resulting in amplification. For higher input signal values, C_{DEG} behaves as a low impedance, minimizing degeneration and making the circuit resemble a differential pair with a grounded source, exhibiting expanding nonlinearity (Figure 2.45). As amplification progresses, the impedance of C_{DEG} increases, causing more degeneration and compressing nonlinearity, similar to a differential pair with a tail-current source (Figure 2.44).

Figure 2.47 shows the dynamic amplifier using the CDL technique [1]. Over time, the amplifier's gain increases but depends on the input amplitude, reflecting nonlinearity. There is a specific point where this nonlinearity transitions from expanding to compressive. At this crossover point, the amplifier's gain becomes independent of the input signal, indicating ideal linearity [1].

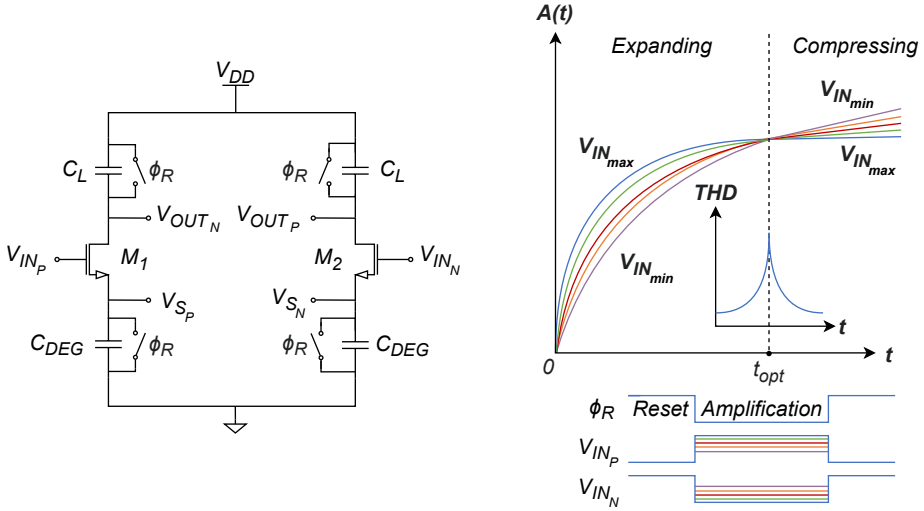


Figure 2.47: Dynamic amplifier with CDL [1].

2.3.4 Ring Amplifier

The ring amplifier presents a promising solution for residue amplification, effectively addressing the challenges of OTA power dissipation and DA distortion. Its architecture, consisting of a cascade of three inverters stabilized by feedback provides significant advantages. Notably, its immunity to output compression allows rail-to-rail output swing in voltage-scaled environments. Furthermore, its slew-based load charging decouples internal power requirements from output load size, enabling high slew rates even with small transistors. As technology advances, the power-delay product decreases linearly, leading to enhanced speed, accuracy, and power efficiency, thus ensuring that performance scales effectively with technological improvements. However, the design requires significant effort due to notable trade-offs between gain, noise, and speed [61].

2.3.4.1 Conventional

The conventional ring amplifier is introduced in [30], where stability is achieved by dividing the second stage into two distinct signal paths, as shown in Figure 2.48. Each path incorporates an offset voltage (V_{OS}) stored by the embedded capacitors C_B. This results in a range of input values where neither output transistor M_{C_P} nor M_{C_N} fully conducts. This range is referred to as the dead zone (V_{DZ}) region and should be large enough, ideally larger than the input amplitude, to stabilize the ring amplifier preventing it from oscillating indefinitely around its common ground, which is half the supply voltage.

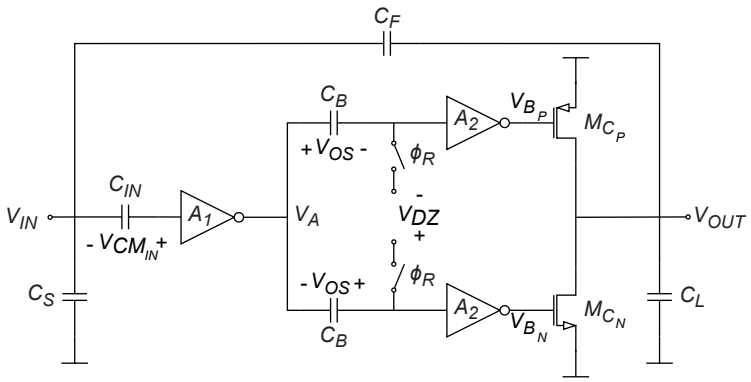


Figure 2.48: Ring amplifier fundamental structure [30].

The transient response of the amplifier is divided into three different phases of ring amplification relative to the dead zone: *slewing* - charging toward the dead zone; *stabilization* - oscillating around the dead zone, and *steady* - locked in the dead zone [21].

During the initial slewing phase, the ring amplifier is presented with a large input amplitude, resulting in maximum overdrive voltage being applied to one of the output transistors. Essentially, this phase behaves similarly to the circuit depicted in Figure 2.49. The first two stages act as comparators, while the output stage functions as current sources. Depending on the input signal, one of the transistors is activated to deliver maximum current to the load.

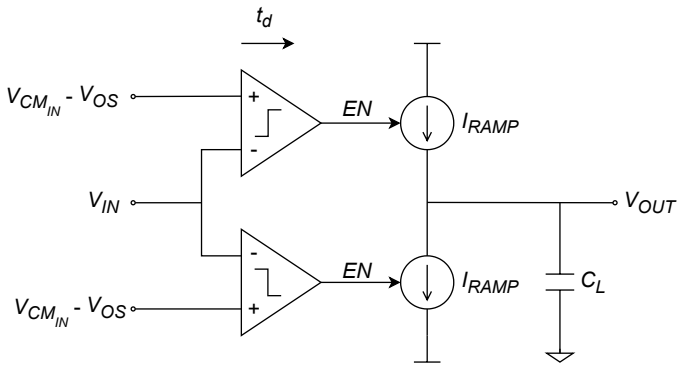


Figure 2.49: Conceptual model of the ring amplifier in slewing phase [30].

Afterward, during the stabilization phase, as the input signal passes the gain of the first stage is shifted to different offset voltages. As the signal moves through the second stage, these shifted signals will manifest differently at the gates of the output transistors. This phase is explained by the feedback mechanism, through which there will be a reduction in the overdrive voltage in each oscillation period, with the peaks becoming progressively smaller as shown in Figure 2.50. As a result, the output current delivered to the load is proportionally reduced, resulting in a lower output amplitude than the previous cycle. When this reduced amplitude is fed back into the input, it further decreases the overdrive voltage, current, and amplitude, continuing in this pattern.

Finally, in the steady state, the amplitude applied to the input drops below the dead zone threshold, and the overdrive becomes significantly small, leading to both transistors turning off. At this stage, the ring amplifier reaches a stable condition.

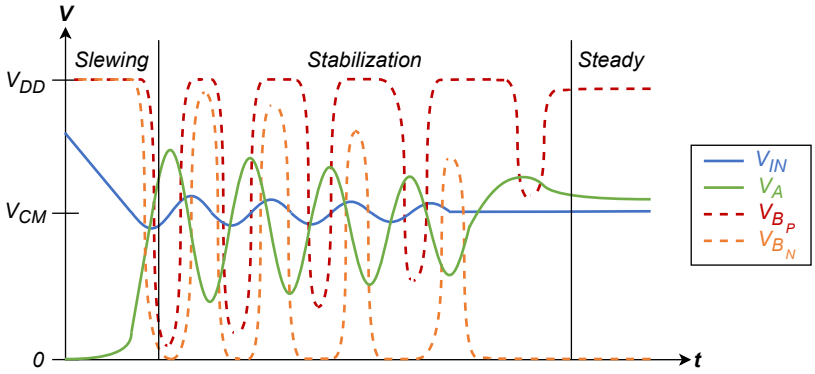


Figure 2.50: Ring amplifier transient response [30].

This behavior can be conceptualized as a dynamic adjustment of the output pole as illustrated in Figure 2.51. As the overdrive progressively reduces, the decrease in output current raises the output resistance R_{out}, which in turn shifts the output pole $f_{p_{out}} = \frac{1}{2\pi}\frac{1}{R_{out}C_L}$ to lower frequencies with each oscillation period. When the ring amplifier locks into the dead zone, R_{out} approaches infinity, and the output pole moves to DC.

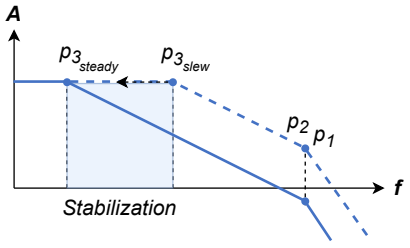


Figure 2.51: Dynamic adjustment of output pole in a ring amplifier [4].

However, the ring amplifier can stabilize across a range of low-frequency output pole positions, possibly stabilizing right at the edge of the dead zone. When this happens, one of the output transistors may continue to conduct a small amount of current and may not turn off completely. Figure 2.52 shows this stable boundary region, called the weak zone.

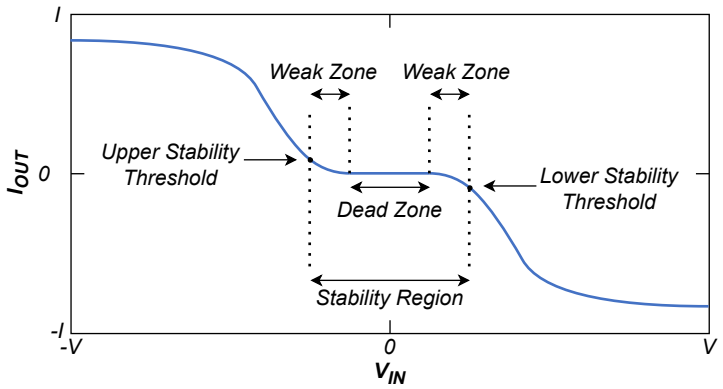


Figure 2.52: Ring amplifier's output current as a function of the input voltage [30].

This implementation can be realized at the transistor level, as depicted in Figure 2.53. The first stage, the primary noise source, comprises transistors M_{1_P} and M_{1_N}. Before the second stage, which is divided into two paths each containing transistors M_{2_P} and M_{2_N}, an offset is generated by capacitors C_B that store voltages from the biasing sources V_{BIAS_H} and V_{BIAS_L}, commanded by the switches M_{SW_P} and M_{SW_N}. This offset sets up the third stage, which includes transistors M_{3_P} and M_{3_N}, operating in the sub-threshold region as the input signal approaches the common-mode, and consequently stabilizing the circuit. The input capacitor C_{IN} stabilizes the input common mode, regardless of the trip point of the first inverter. Collectively, these stages achieve high gain due to their cascaded arrangement.

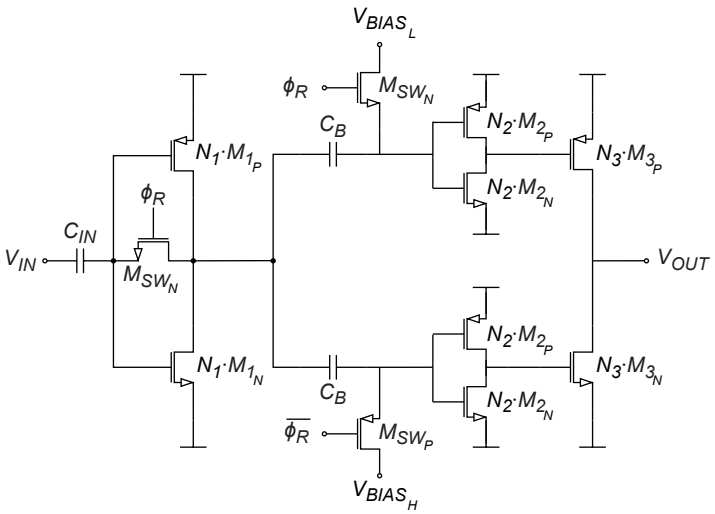


Figure 2.53: Conventional ring amplifier schematic [30].

2.3.4.2 Self-Biased

The conventional ring amplifier relies on external biases, which conflict with a voltage-scaled environment and power efficiency. The self-biased ring amplifier introduced in [50]

is proposed to overcome this issue, making it more robust to process voltage temperature (PVT) variation.

This configuration is set as demonstrated in Figure 2.54. The input stage, which comprises transistors M_{1_P} and M_{1_N}, has been optimized for noise reduction by lowering the voltage supply. This method enhances transconductance, as it is inversely proportional to thermal noise, without necessitating a larger transistor, which would otherwise result in increased power dissipation. This optimization is achieved by incorporating the diode-connected transistor M_{D_N}. The second stage includes transistors M_{2_P} and M_{2_N}, along with a polysilicon resistance R_B, which dynamically sets an offset due to voltage drop caused by the short-circuit current in this stage, allowing it to track the supply voltage and operate across a wider supply range, thereby eliminating the need for an additional inverter. This offset drives the third stage into the sub-threshold region. The third stage, comprising high-threshold transistors M_{3_P} and M_{3_N}, benefits from high output resistance to take advantage of high overdrive voltage, enabling stability across a wider range.

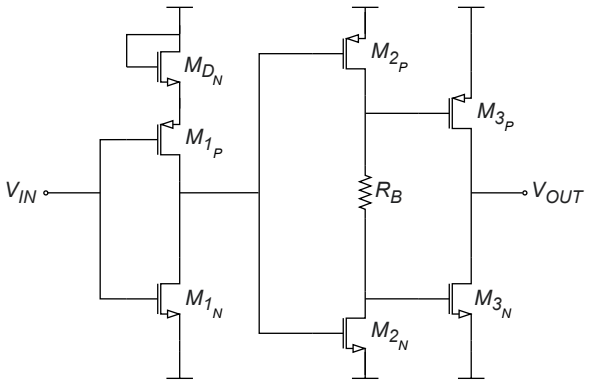


Figure 2.54: Self-biased ring amplifier schematic [50].

2.3.4.3 Bias-Enhanced

A novel architecture that enhances stability at the expense of gain is the bias-enhanced ring amplifier introduced in [8]. Also, this design offers a faster response compared to previous configurations.

For stability the dominant pole f_{p_3} must be significantly smaller than the non-dominant poles f_{p_2} and f_{p_1}, as in Equation (2.65). To do so, Equation (2.66) is respected.

$$f_{p_3} \ll f_{p_2} < f_{p_1} \tag{2.65}$$

$$f_{p_2} \approx f_{p_1} > 2.5GBW \tag{2.66}$$

As illustrated in Figure 2.55, the self-biased configuration shifts the dominant pole to lower frequencies, thereby increasing the phase margin compared to the conventional

architecture. Nevertheless, the positions of the non-dominant poles remain nearly fixed, exerting a greater impact on phase margin and, consequently, on stability. It would be advantageous to push the non-dominant poles to higher frequencies to enhance stability. This can be accomplished through the implementation of a bias-enhanced configuration, although at the expense of reduced gain.

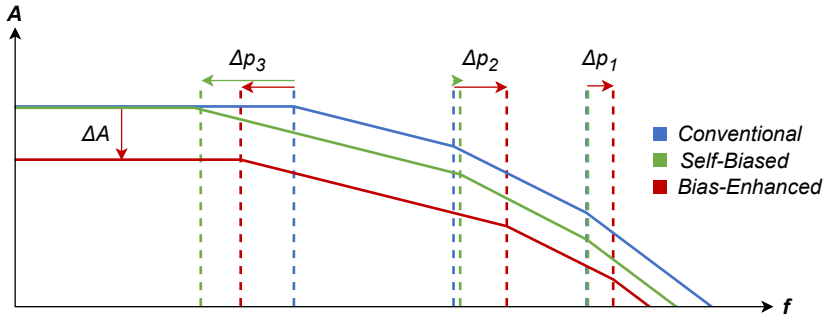


Figure 2.55: Stability comparison for different ring amplifier configurations [8].

This implementation is carried out at the transistor level, as demonstrated in Figure 2.56. The first stage incorporates transistors M_{1_P} and M_{1_N} and a pair of resistors R_A, which raise the bias voltage of the second stage, comprised of transistors M_{2_P} and M_{2_N}. This results in higher overdrive voltage and a larger stability region. The third stage uses high-threshold transistors, M_{3_P} and M_{3_N}, and respond faster.

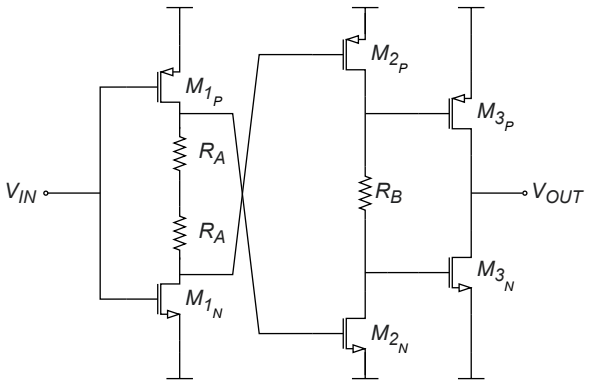


Figure 2.56: Bias-enhanced ring amplifier schematic [8].

2.3.4.4 Dynamically-Biased

The design proposed in [43] is based on the self-biased architecture but replaces the fixed passive resistor with a CMOS resistor, offering several advantages. First, it allows for post-fabrication biasing adjustments, enhancing the system's versatility and flexibility. Second, a CMOS resistor provides faster dynamic stabilization than a fixed resistor, improving speed, and can be powered down during idle to reduce power consumption.

At the transistor level, this configuration is represented as shown in Figure 2.57. The first and third stages are similar to the self-biased design. However, the second stage includes a CMOS resistor formed by transistors M_{B_P} and M_{B_N}.

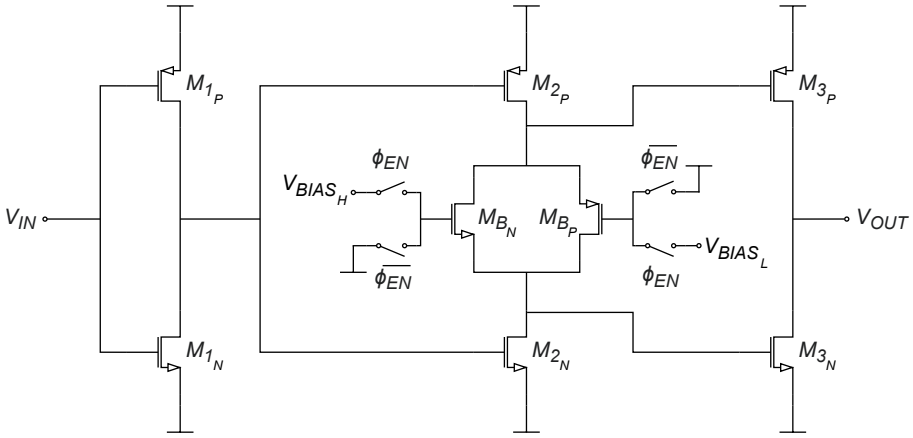


Figure 2.57: Dynamically-biased ring amplifier schematic [43].

2.3.4.5 Dead-Zone-Degenerated

In [41], a novel architecture combines the dynamically biased and bias-enhanced designs to achieve high linearity with gain calibration, high speed, and low power consumption, known as the dead-zone-degenerated ring amplifier.

Figure 2.58 illustrates the linearity comparison of different ring amplifier configurations. It can be observed that there is no significant difference in linearity between the conventional and bias-enhanced configurations, aside from a drop in gain. However, when analyzing the dead-zone-degenerated configuration, a noticeable improvement in linearity is evident.

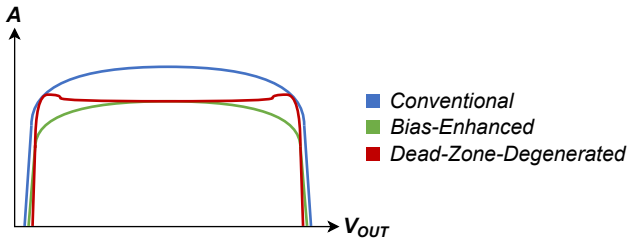


Figure 2.58: Linearity for different configurations of ring amplifiers [41].

This is implemented at the transistor level as represented in Figure 2.59. The first stage, composed by transistors M_{1_P} and M_{1_P}, includes a CMOS resistor formed by transistors M_{A_P} and M_{A_N} to enhance the bias for the second stage. The second stage, which includes transistors M_{2_P} and M_{2_N}, employs a CMOS resistor formed by transistor M_{B_P} and M_{B_N} to establish the offset for the third stage, composed by transistors M_{3_P} and M_{3_N}. Preceding the third stage, the feedback transistors, designated as M_{C_P} and M_{C_N}, improve overall linearity by maintaining a flat open-loop gain and output swing profile. To achieve this,

the transistors adjust the dead-zone voltage asymmetrically when the output approaches the supply voltages, thereby creating an output-dependent offset that the feedback loop compensates for to mitigate errors from finite gain and gain compression.

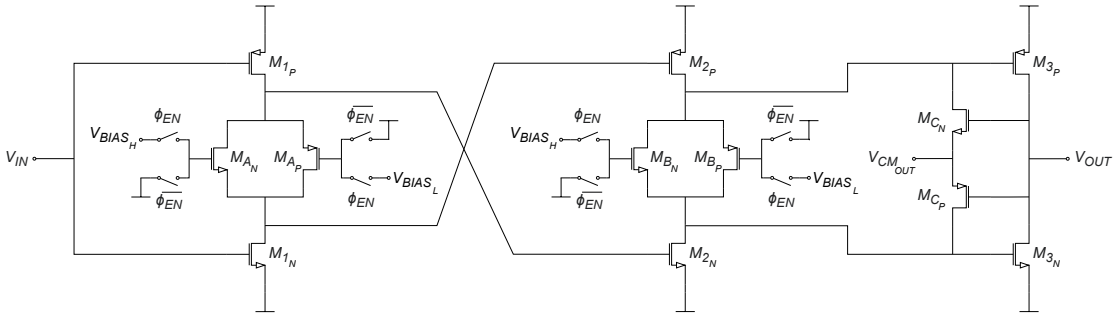


Figure 2.59: Dead-zone-degenerated ring amplifier schematic [41].

2.3.4.6 Critically Damped

To ensure stability, earlier ring amplifier configurations have incorporated the concept of a dead zone voltage to gradually reduce the overdrive voltages in the output stage, shifting the dominant pole to lower frequencies through a feedback mechanism that causes ringing. Hence, this approach comprises both speed and PVT sensitivity. Calibration is used to mitigate these issues without impacting linearity. In [4], the critically damped ring amplifier is introduced as a solution that achieves high linearity and high speed without requiring calibration by achieving stability to push the non-dominant poles to high frequency using $\frac{1}{g_m}$ loading.

Figure 2.60 visually compares the conventional and critically damped ring amplifiers. It highlights an increase in bandwidth, though this results in a gain reduction. Despite this, the new configuration maintains linearity at moderate open-loop gain levels.

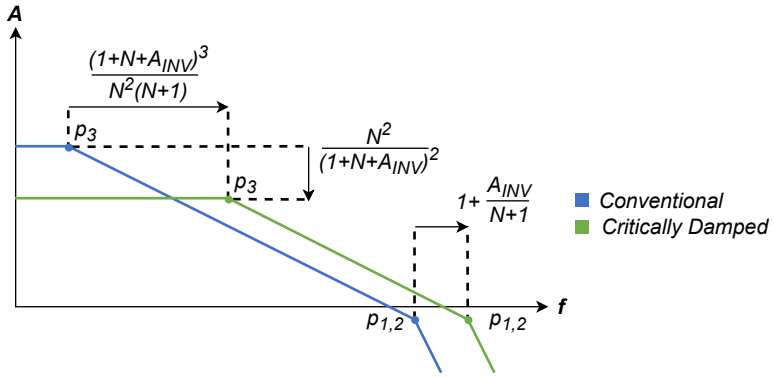


Figure 2.60: Magnitude response for conventional and critically damped ring amplifiers [4].

This implementation can be achieved at the transistor level as represented in Figure 2.61. The first stage consists of an inverter with transistors M_{1_P} and M_{1_N}, and a $\frac{1}{g_m}$ inverter with

transistors $M_{1'_P}$ and $M_{1'_N}$. The second stage follows the same structure but with transistors M_{2_P}, M_{2_N}, $M_{2'_P}$ and $M_{2'_N}$. The third stage consists of a single inverter with transistors M_{3_P} and M_{3_N}. Also, the input and output common modes should be coupled to the trip point of the first inverter.

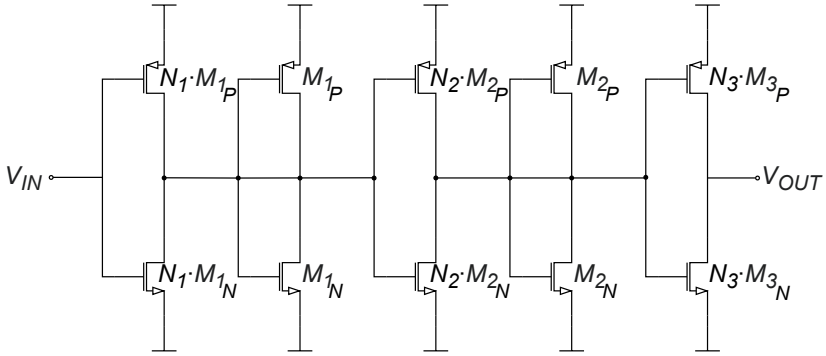


Figure 2.61: Critically damped ring amplifier schematic [4].

2.4 Conclusion

The data extracted from the survey [63] provides an ENOB versus sampling frequency analysis, offering valuable insights for selecting the most suitable ADC architecture for specific application requirements, as illustrated in Figure 2.62. This analysis highlights the trade-offs between speed and resolution across different architectures. It can be concluded that pipeline converters offer an optimal balance for applications demanding high speed and high resolution.

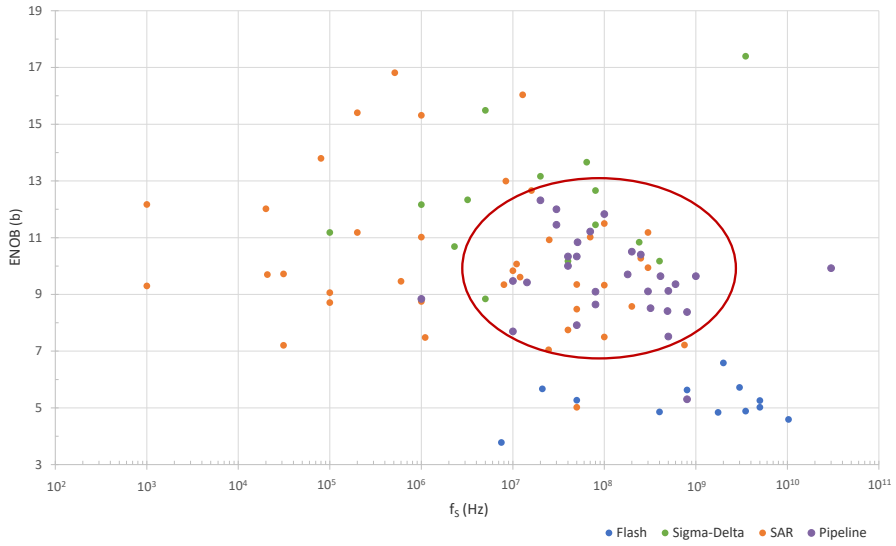


Figure 2.62: ENOB versus sampling frequency for different ADCs configurations.

A preliminary analysis was conducted based on the survey [20], as detailed in Annex A.2 to comprehensively understand the advancements in ring amplifiers for pipeline ADCs. This analysis emphasizes the need for both high speed and high resolution in pipeline ADCs, as illustrated in Figure 2.63, and in their correspondent ring amplifier, as shown in Figure 2.64. The previous ring amplifier configurations are highlighted to enable a comparison between them.

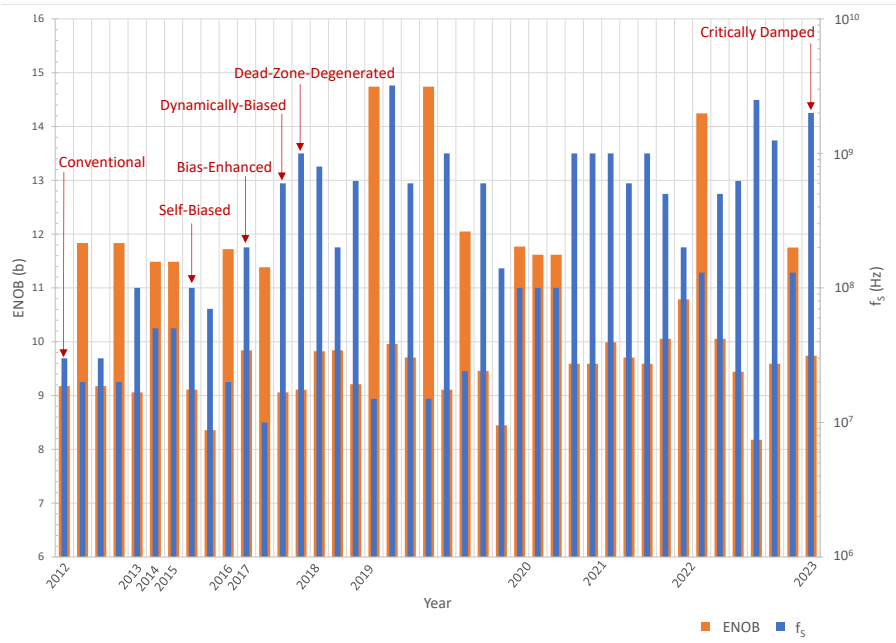


Figure 2.63: ENOB and sampling frequency evolution for pipeline converters.

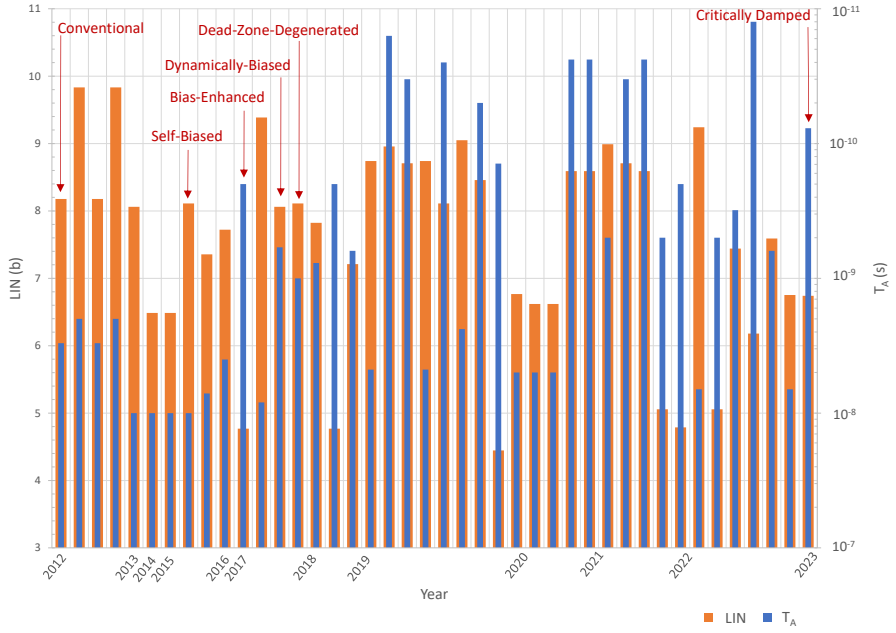


Figure 2.64: Linearity and amplification time evolution for ring amplifiers.

A clear relationship exists between the pipeline ADC and its corresponding ring amplifiers in terms of both speed and resolution across all configurations. Higher ENOB in the converter is associated with improved linearity in the amplifier, while a higher sample rate in the converter correlates with shorter amplification times in the amplifier.

Table 2.1 presents the characteristics of the pipeline converters and ring amplifiers from the previously explored configurations. As the study of ring amplification progresses, a speed improvement is observed in both the converter and its amplifier, but this comes at the cost of increased power dissipation, with resolution requirements remaining nearly constant. The table also reflects advancements in technology and voltage-scaled environments, highlighting the growing demands of the industry.

Table 2.1: Specifications of different pipeline ADCs containing ring amplifiers.

$Reference$	[30]	[50]	[8]	[43]	[41]	[4]
$CMOS(nm)$	180	65	65	28	28	28
$V_{DD}(V)$	1.3	1.2	1.2	0.9	0.9	1.0
$V_{IN}(V_{pp})$	2.2	2.0	1.6	1.6	1.6	–
$N(b)$	10.5	10.5	11.0	12.0	12.0	12.0
$ENOB(b)$	9.2	9.1	9.8	9.1	9.1	9.7
$f_S(Gsps)$	0.03	0.1	0.2	0.6	1.0	2.0
$SNDR(dB)$	57.0	56.6	61.0	56.3	56.6	60.4
$SFDR(dB)$	72.0	64.7	57.5	69.2	73.1	75.8
$P_d(mW)$	2.6	2.5	2.3	14.2	24.8	27.0
$FOM_S(dB)$	154.6	159.7	167.4	159.5	159.6	166.1
$FOM_W(fJ/cs)$	149.8	44.5	12.4	44.3	44.9	15.8
$A_F(V/V)$	2	2	4	2	2	4
$T_A(ns)$	33.3	10.0	5.0	1.7	1.0	0.13
$GBW(GHz)$	0.04	0.15	0.59	0.88	1.5	22.6
$LIN(b)$	8.2	8.1	7.8	8.1	8.1	7.7
$THD(dB)$	-58.5	-54.6	-47.2	-56.7	-58.8	-56.1
$P_d(mW)$	0.09	1.0	0.9	4.9	–	–
$FOM(mV^{-2})$	195.4	58.6	630.5	89.8	–	–

With the completion of this chapter, the fundamental principles of ADCs and amplifiers have been established. Additionally, exploring the evolution of ring amplification has provided insight into its initial and final configurations — the conventional and critically damped architectures — each approached from different perspectives in the following chapter.

3

Ring Amplifier

3.1 Introduction

This chapter investigates both the conventional ring amplifier [31] and the critically damped ring amplifier [5], employing distinct approaches for each. For the conventional ring amplifier, a thorough theoretical analysis is followed by circuit simulations, where only technology scaling is applied to understand the principles of ring amplification. This process also supports the creation of a parameterized testbench, which is subsequently used in the second analysis. For the critically damped ring amplifier, the theoretical analysis is enhanced with a design strategy based on the g_m/I_D model. This leads to two designs: one optimized for maximum gain (open-loop gain) and the other for maximum speed (gain-bandwidth product), with careful attention given to power dissipation in both. Electrical simulations are then conducted to validate the theoretical predictions and assess the accuracy of the design methodology.

3.2 Conventional Ring Amplifier

3.2.1 Theory Analysis

The block diagram of the conventional ring amplifier in Figure 3.1 represents an approximation of the architecture shown in Figure 2.53, intended to streamline the analysis. It illustrates the three simple inverter stages and the corresponding poles positions.

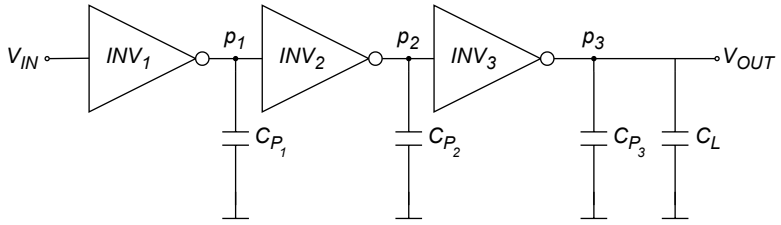


Figure 3.1: Conventional ring amplifier block diagram.

The simple inverter schematic model is presented in Figure 3.2, defined by a N multiplier.

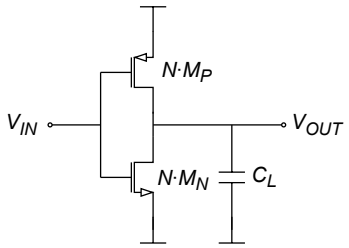


Figure 3.2: Simple inverter schematic.

The small-signal model of the simple inverter in Figure 3.3 enables its AC analysis.

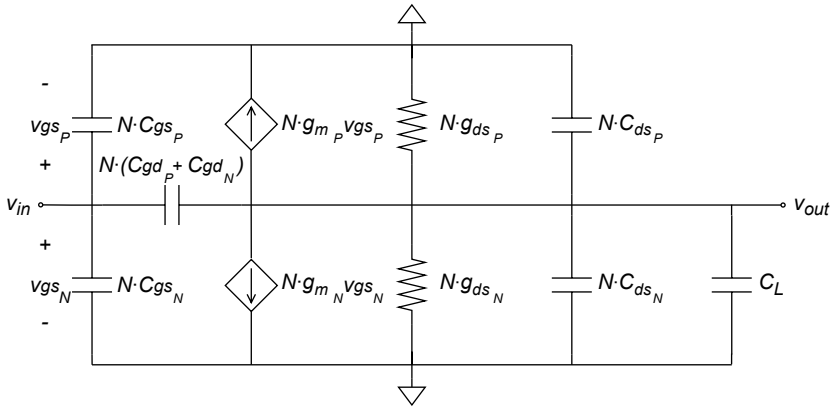


Figure 3.3: Simple inverter small-signal model.

To simplify the analysis, the body effect was neglected ($V_{sb} = 0$), and the notation C_{ds} was used instead of C_{db} to emphasize this. The following assumptions were made, assuming operation in the saturation region:

$$\begin{cases} C_{gs} = C_{gs_P} + C_{gs_N} \\ C_{gd} = C_{gd_P} + C_{gd_N} \\ C_{ds} = C_{ds_P} + C_{ds_N} \\ g_m = g_{m_P} + g_{m_N} \\ g_{ds} = g_{ds_P} + g_{ds_N} \end{cases} . \tag{3.1}$$

By applying Kirchhoff's current law at v_{out}, the subsequent equation was derived:

$$N\left[s \cdot C_{gd}\left(v_{out} - v_{in}\right) + g_m v_{in} + \left(g_{ds} + s \cdot C_{ds}\right) v_{out}\right] + s \cdot C_L v_{out} = 0. \tag{3.2}$$

From this, the inverter transfer function was obtained as:

$$H(s) = \frac{N\left(sC_{gd} - g_m\right)}{s\left[N\left(C_{ds} + C_{gd}\right) + C_L\right] + N g_{ds}}. \tag{3.3}$$

The corresponding parasitic capacitances are given by:

$$C_{P_{in}} = N\left(C_{gd} + C_{gs}\right), \tag{3.4}$$

$$C_{P_{out}} = N\left(C_{ds} + C_{gd}\right). \tag{3.5}$$

For an amplifier consisting of a cascade of inverters, the transfer function can be obtained by multiplying the transfer functions of the three stages. In this calculation, the transfer function of the inverter is applied to the first two stages, with the C_L parameter substituted by the input parasitic capacitance. Thus, the conventional ring amplifier transfer function, obtained with the assistance of *wxMaxima*, is given by:

$$\begin{aligned} H(s) = & \frac{N_1\left(sC_{gd_1} - g_{m_1}\right)}{s\left[N_1\left(C_{ds_1} + C_{gd_1}\right) + N_2\left(C_{gd_2} + C_{gs_2}\right)\right] + N_1 g_{ds_1}} \\ & \cdot \frac{N_2\left(sC_{gd_2} - g_{m_2}\right)}{s\left[N_2\left(C_{ds_2} + C_{gd_2}\right) + N_3\left(C_{gd_3} + C_{gs_3}\right)\right] + N_2 g_{ds_2}} \\ & \cdot \frac{N_3\left(sC_{gd_3} - g_{m_3}\right)}{s\left[N_3\left(C_{ds_3} + C_{gd_3}\right) + C_L\right] + N_3 g_{ds_3}}. \end{aligned} \tag{3.6}$$

DC Gain

The DC gain is determined by setting the frequency to zero, resulting in:

$$A = \frac{g_{m_1}}{g_{ds_1}} \cdot \frac{g_{m_2}}{g_{ds_2}} \cdot \frac{g_{m_3}}{g_{ds_3}}. \tag{3.7}$$

Zeros

To find the positive zeros, the numerator of the transfer function was set to zero, yielding:

$$f_{z_1} = \frac{1}{2\pi}\frac{g_{m_1}}{C_{gd_1}}, \tag{3.8}$$

$$f_{z_2} = \frac{1}{2\pi}\frac{g_{m_2}}{C_{gd_2}}, \tag{3.9}$$

$$f_{z_3} = \frac{1}{2\pi}\frac{g_{m_3}}{C_{gd_3}}. \tag{3.10}$$

Poles

For the negative poles, the denominator of the transfer function was set to zero, giving:

$$f_{p_1} = \frac{1}{2\pi}\frac{N_1 g_{ds_1}}{N_1\left(C_{ds_1} + C_{gd_1}\right) + N_2\left(C_{gd_2} + C_{gs_2}\right)}, \tag{3.11}$$

$$f_{p_2} = \frac{1}{2\pi}\frac{N_2 g_{ds_2}}{N_2\left(C_{ds_2} + C_{gd_2}\right) + N_3\left(C_{gd_3} + C_{gs_3}\right)}, \tag{3.12}$$

$$f_{p_3} = \frac{1}{2\pi}\frac{N_3 g_{ds_3}}{N_3\left(C_{ds_3} + C_{gd_3}\right) + C_L}. \tag{3.13}$$

Gain-Bandwdith Product

The gain-bandwidth product, calculated by multiplying the gain by the output pole in the context of a single-pole approximation, is given by:

$$GBW = \frac{1}{2\pi}\frac{g_{m_1}}{g_{ds_1}} \cdot \frac{g_{m_2}}{g_{ds_2}} \cdot \frac{N_3 g_{m_3}}{N_3\left(C_{ds_3} + C_{gd_3}\right) + C_L}. \tag{3.14}$$

Dissipated Power

The power dissipation of the inverter is given by:

$$P_d = V_{DD} N I_D. \tag{3.15}$$

Therefore, the total power dissipation of the amplifier can be calculated as the sum of the power dissipation for each inverter stage, given by:

$$P_d = V_{DD}\left[N_1 I_{D_1} + N_2 I_{D_2} + N_3 I_{D_3}\right]. \tag{3.16}$$

Input-Referred Noise

The noise in the inverter primarily consists of thermal noise. Given the transistor thermal noise discussed in Chapter 2, the input-referred noise can be calculated by dividing the thermal noise by the inverter gain, as follows:

$$\overline{V_{N_{IN}}^2} = 4K_B T\gamma \frac{N g_{ds}^2}{g_m^3}. \tag{3.17}$$

Thus, the amplifier's input-referred noise can be determined by summing the input-referred noise for each stage, while dividing by the product of the gains of those stages as the number of stages increases, as follows:

$$\overline{V_{N_{IN}}^2} = 4K_B T\gamma \left(\frac{N_1 g_{ds_1}^2}{g_{m_1}^3} + \frac{g_{ds_1}^2}{g_{m_1}^2} \cdot \frac{N_2 g_{ds_2}^2}{g_{m_2}^3} + \frac{g_{ds_1}^2}{g_{m_1}^2} \cdot \frac{g_{ds_2}^2}{g_{m_2}^2} \cdot \frac{N_3 g_{ds_3}^2}{g_{m_3}^3}\right). \tag{3.18}$$

Slew Rate

The slew rate is determined by the saturation current of the final stage, with a focus on the rising transition, as follows:

$$SR = N_3 \frac{\frac{1}{2} k \frac{W_3}{L_3} V_{DSAT_{P3}}^2}{C_L}. \tag{3.19}$$

Output Voltage Swing

The output voltage swing is evaluated at the final stage and is given by:

$$V_{OUT_{SW}} = V_{DD} + V_{DSAT_P} - V_{DSAT_N}. \tag{3.20}$$

Common-Mode Input and Output Voltage

The common-mode voltage for both the input and output was assumed to be half the voltage supply, as follows:

$$V_{CM_{IN}} = V_{CM_{OUT}} = \frac{V_{DD}}{2}. \tag{3.21}$$

3.2.2 Design Method

Regarding the design variables, the three stages are defined using the variables presented in Equation (3.22) for NMOS in Equation (3.23) for PMOS. In this context, K_S is the scaling factor applied to the reference width W (1.2 μm) to determine the NMOS width. This value is multiplied by the width factor k_W to calculate the PMOS width. Also, L represents one of the defined channel lengths from the tables (60, 90, 120, 240, 300, or 600 nm), which are explained in detail in the next section.

$$N_i \cdot \frac{K_{S_i} \cdot W}{L_i}, i = 1, 2, 3 \tag{3.22}$$

$$N_i \cdot \frac{k_{W_i} \cdot K_{S_i} \cdot W}{L_i}, i = 1, 2, 3 \tag{3.23}$$

This design, if it may be called design, for this ring amplifier was simply scale the 180 nm configuration presented in [30] to 65 nm where the following stage parameters resumed in Table 3.1 were used. Additionally, the input capacitor C_{IN} and the embedded capacitors C_B were set at 100 fF.

Table 3.1: Conventional ring amplifier parameters.

$L_1(nm)$	N_1	K_{S_1}	k_{W_1}
90	1	0.3	2.625
$L_2(nm)$	N_2	K_{S_2}	k_{W_3}
60	1	0.1	2.625
$L_3(nm)$	N_3	K_{S_3}	k_{W_3}
120	1	0.1	3

3.2.3 Electrical Simulation

The simulations were performed in *Cadence Virtuoso* using the testbenches provided in Appendix B.2 and employing the P_11_SPRVT and N_BPW_SPRVT transistors from the UMC65SP library. These transistors are characterized by standard performance and a regular voltage threshold, with the N-MOS featuring a bottom p-well, all within the 65 nm CMOS technology, allowing for general operation. The simulations were carried out with a supply voltage of 1.2 V and a room temperature of 27 °C.

The testbench used to evaluate the conventional ring amplifier configuration (Figure 2.53) for transient and FFT simulations, with parameters specified in Table 3.1, comprises a pseudo-differential flip-around MDAC. This is preceded by a multiplexer (implemented in $Verilog-A$, as detailed in Appendix C.1) and followed by a load, as illustrated in Figure 3.4. This setup operates at a clock frequency f_{CLK} of 30 MHz, with a resolution N of 3 bits and a reference voltage V_{REF} of 1 V, resulting in a feedback gain of 4 V/V and a V_{LSB} of 125 mV. The feedback capacitor C_F is 100 fF, the sample capacitor C_S is 300 fF and

the load capacitor C_F is 400 fF. The common-mode voltage at both the input and output is set to 0.6 V, corresponding to half the supply voltage.

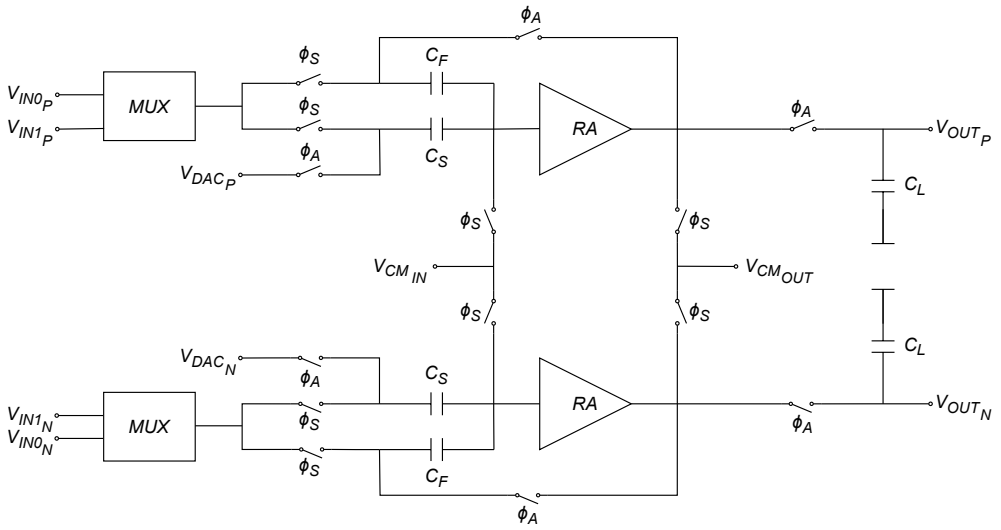


Figure 3.4: Pseudo-differential flip-around MDAC testbench.

Nonoverlapping Phases

The nonoverlapping phases of the flip-around MDAC are defined as shown in Figure 3.4, where a rise and fall time of 10 ps and a nonoverlap time of 33.333 ps can be observed.

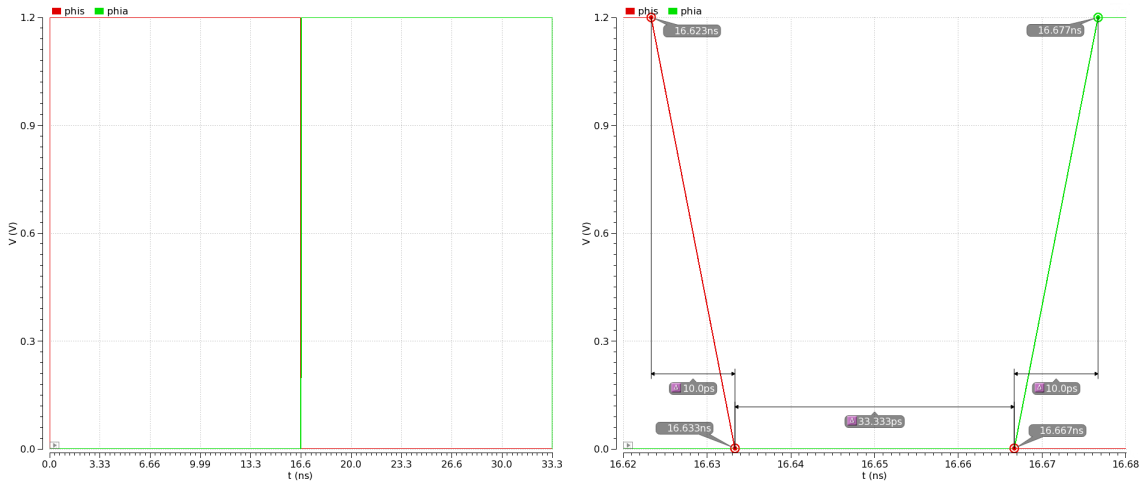


Figure 3.5: Flip-around MDAC nonoverlapping phases.

Multiplexer Control Signal

The multiplexer enables two simulations: one to assess linearity using a ramp signal and another for FFT analysis with a sine signal. To accomplish this, the control signal, as defined in Figure 3.6, is set to 0 V during the ramp phase and 1.2 V during the sine wave phase. Specifically, a few clock cycles are initially provided to stabilize the configuration, followed by the generation of the ramp signal from 266.67 ns to 4.43 μs. Subsequently, the sine wave is generated from 4.43 μs to 8.7 μs. Finally, additional clock cycles are

introduced to ensure the bins are properly filled for an accurate FFT simulation, given a total simulation time of 8.96 μs.

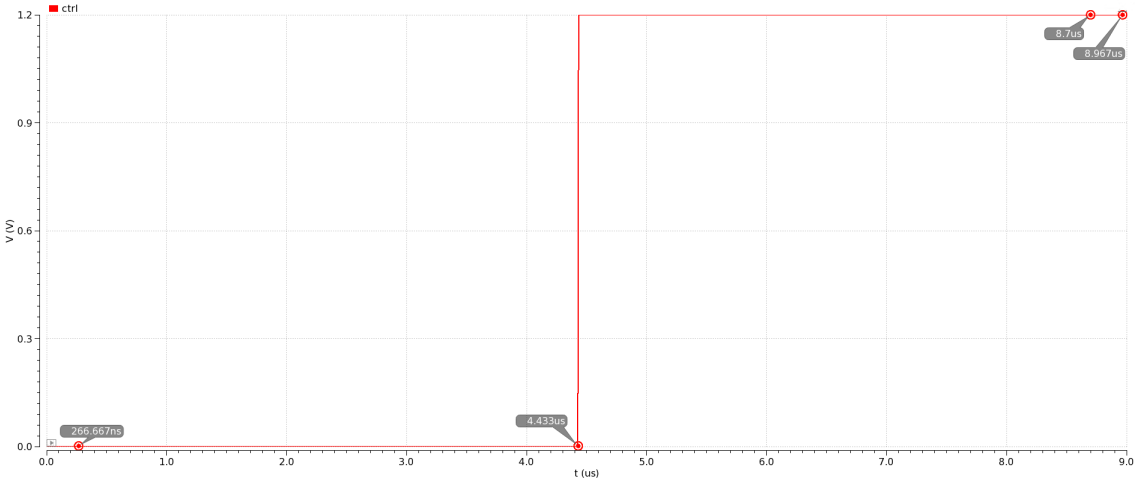


Figure 3.6: Multiplexer control signal.

Common-Mode Input and Output Signals

The common-mode input and output voltages are illustrated in 3.7. As it can be observed both voltages are set at $V_{DD}/2$ (600.0 mV), ensuring that the circuit operates effectively.

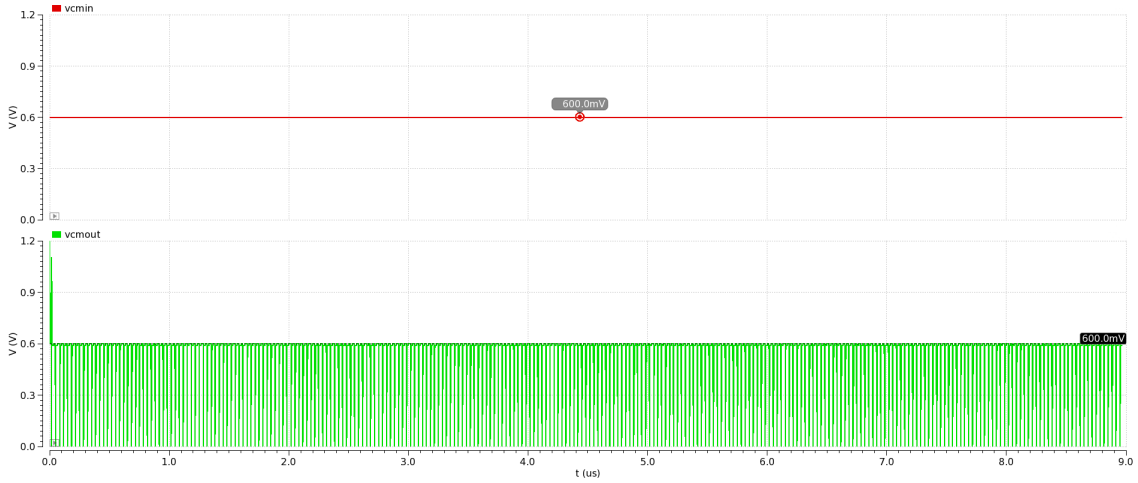
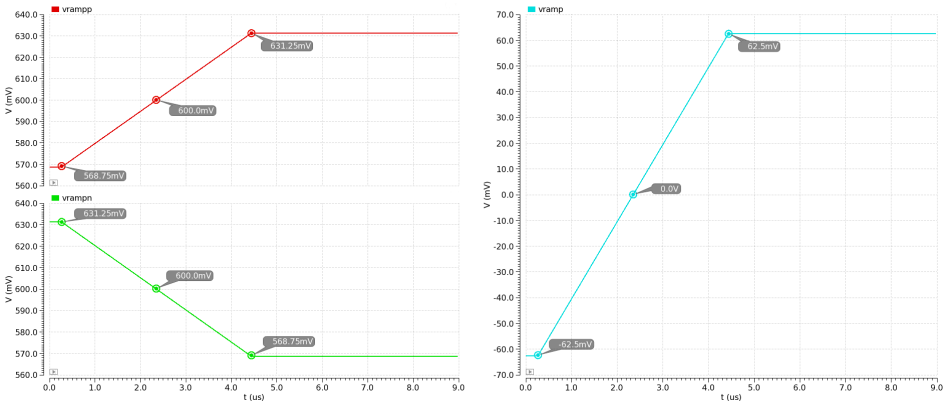


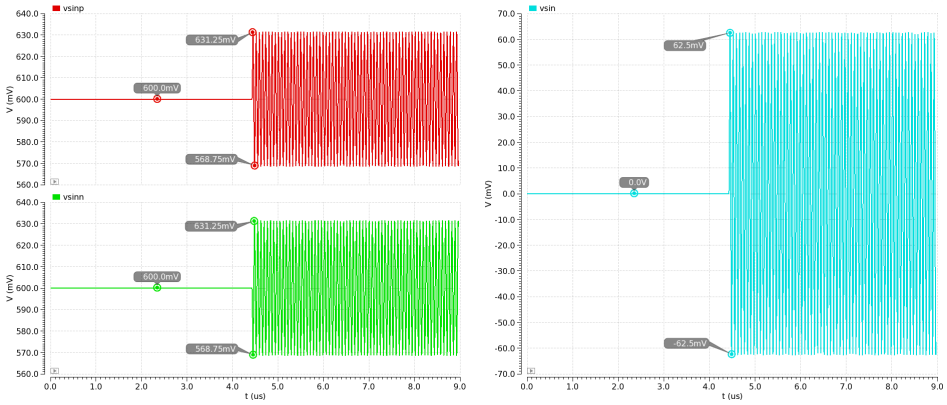
Figure 3.7: Conventional ring amplifier input and output common-mode signals.

Ramp and Sine Signals

The ramp and sine differential signals are centered at ground (0 V) with amplitudes $V_{LSB}/2$ (62.5 mV), each consisting of two symmetric single-ended components centered at V_{CM} (600 mV) and with amplitude $V_{LSB}/4$ (31.25 mV), as shown in Figure 3.8.



(a) Ramp signals.



(b) Sine signals.

Figure 3.8: Single-ended and differential ramp and sine signals.

Nodal Signals

The differential nodal signals between stages are shown in Figure 3.9, where a 70 mV dead zone was considered. The amplification from one stage to the next is evident, confirming that the amplifier is functioning as expected.

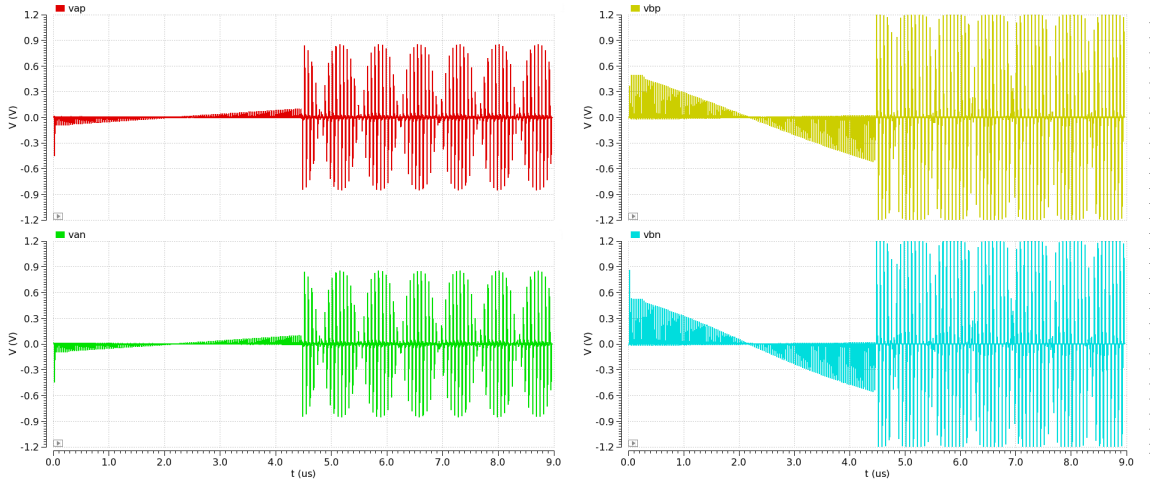


Figure 3.9: Conventional ring amplifier nodal differential signals between stages.

Input and Output Signals

The input and output differential signals of the conventional ring amplifier are shown in Figure 3.10. The feedback gain is calculated as the ratio of the output signal's slope to the input signal's slope, yielding a value of 3.997, which confirms that the amplifier operates within the specified resolution, tough with some distortion.

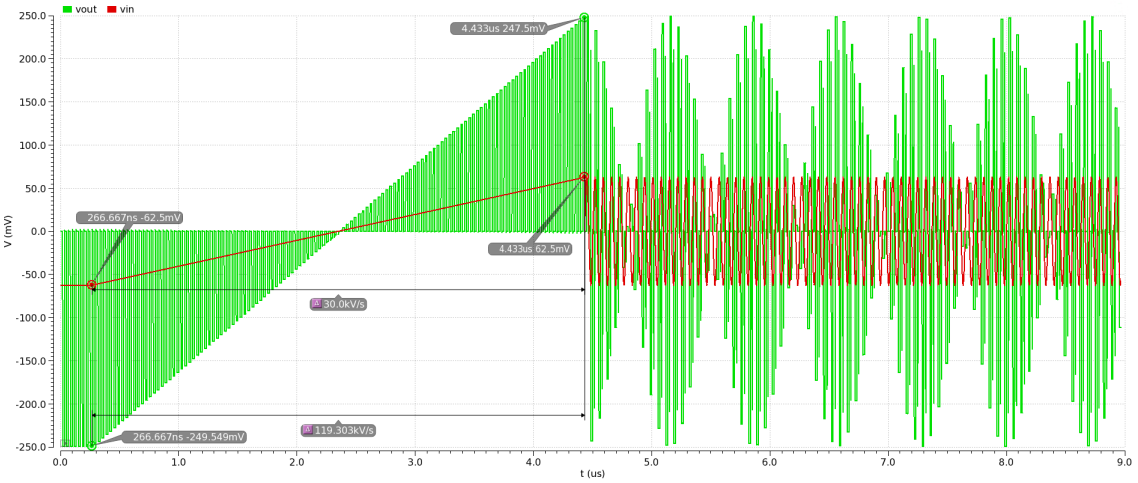


Figure 3.10: Conventional ring amplifier input and output differential signals.

Input and Output Sampled Signals

However, the gain should be calculated as the ratio of the sampled input and output signals for accurate analysis, as illustrated in Figure 3.11. The input signal is sampled during the descending branch of the sampling phase, approximately at half a period, while the output is sampled during the descending phase of the amplification phase, around one period. This process is repeated throughout the simulation duration. The final sampled signals on the ramp are also shown, indicating a gain of 3.993 at that specific point. Additionally, to align both input signals for the ratio calculation, a right shift of 16.667 ns, equivalent to half the clock period, should be applied to the input signal.

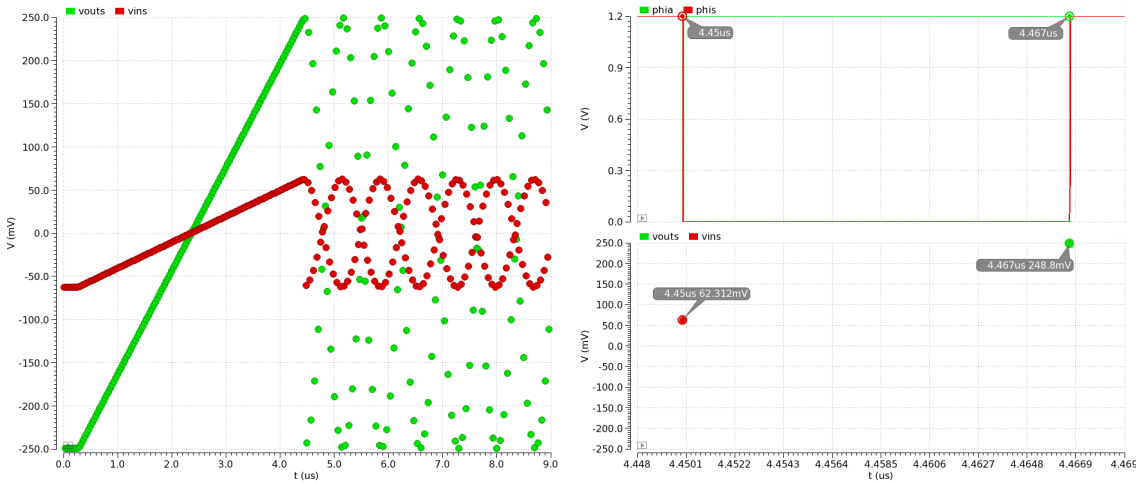


Figure 3.11: Conventional ring amplifier input and output sampled signals.

3.3 Critically Damped Ring Amplifier

3.3.1 Theory Analysis

The block diagram of the critically damped ring amplifier in Figure 3.12, represents an approximation of the damped architecture depicted in Figure 2.61, intended to streamline the analysis. It highlights the two $1/g_m$ loading inverter stages, the output simple inverter stage, and the corresponding poles positions.

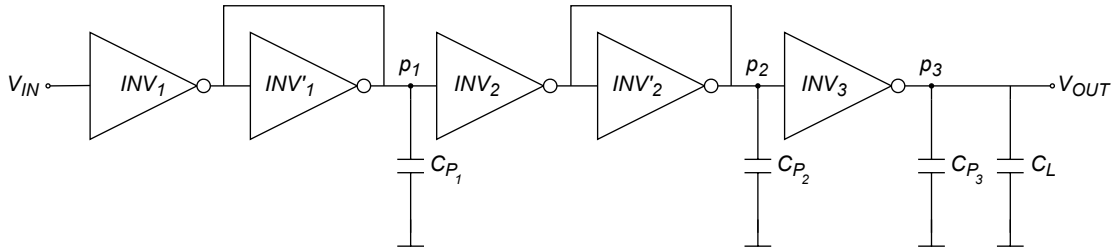


Figure 3.12: Critically damped ring amplifier block diagram.

Figure 3.13 presents the $1/g_m$ loading inverter schematic defined by a $N:1$ multiplier relation.

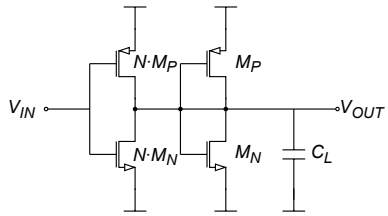


Figure 3.13: $1/g_m$ loading inverter schematic.

The small-signal model of the $1/g_m$ loading inverter in Figure 3.14 enables its AC analysis.

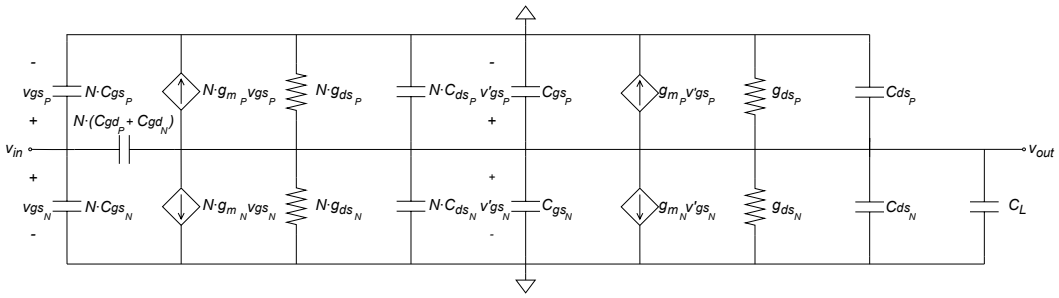


Figure 3.14: $1/g_m$ loading inverter small-signal model.

To simplify the analysis, the body effect was neglected ($V_{sb} = 0$), and the notation C_{ds} was used instead of C_{db} to emphasize this. The following assumptions were made, assuming operation in the saturation region:

$$\begin{cases} C_{gs} = C_{gs_P} + C_{gs_N} \\ C_{gd} = C_{gd_P} + C_{gd_N} \\ C_{ds} = C_{ds_P} + C_{ds_N} \\ g_m = g_{m_P} + g_{m_N} \\ g_{ds} = g_{ds_P} + g_{ds_N} \end{cases}. \tag{3.24}$$

By applying Kirchhof's current law at v_{out}, the subsequent equation was derived:

$$N\left[sC_{gd}\left(v_{out} - v_{in}\right) + g_m v_{in} + \left(g_{ds} + sC_{ds}\right)v_{out}\right] + \left[g_m + g_{ds} + s(C_{gs} + C_{ds} + C_L)\right]v_{out} = 0. \tag{3.25}$$

From this, the $1/g_m$ loading inverter transfer function was obtained as:

$$H(s) = \frac{N\left(sC_{gd} - g_m\right)}{s\left[N\left(C_{ds} + C_{gd}\right) + C_{ds} + C_{gd} + C_L\right] + (N+1)g_{ds} + g_m}. \tag{3.26}$$

The corresponding parasitic capacitances are given by:

$$C_{P_{in}} = N\left(C_{gd} + C_{gs}\right), \tag{3.27}$$

$$C_{P_{out}} = N\left(C_{ds} + C_{gd}\right) + C_{ds} + C_{gs}. \tag{3.28}$$

For an amplifier consisting of a cascade of inverters, the transfer function can be obtained by multiplying the transfer functions of the three stages. In this calculation, the transfer function of the loading inverter is applied to the first two stages, with the C_L parameter substituted by the input parasitic capacitance. For the third stage, the simple inverter transfer function is used. Thus, the critically damped ring amplifier transfer function, obtained with the assistance of *wxMaxima*, is given by:

$$\begin{aligned} H(s) = &\frac{N_1\left(sC_{gd_1} - g_{m_1}\right)}{s\left[N_1\left(C_{ds_1} + C_{gd_1}\right) + C_{ds_1} + C_{gs_1} + N_2\left(C_{gd_2} + C_{gs_2}\right)\right] + (N_1+1)g_{ds_1} + g_{m_1}} \\ &\cdot \frac{N_2\left(sC_{gd_2} - g_{m_2}\right)}{s\left[N_2\left(C_{ds_2} + C_{gd_2}\right) + C_{ds_2} + C_{gs_2} + N_3\left(C_{gd_3} + C_{gs_3}\right)\right] + (N_2+1)g_{ds_2} + g_{m_2}} \\ &\cdot \frac{N_3\left(sC_{gd_3} - g_{m_3}\right)}{s\left[N_3\left(C_{ds_3} + C_{gd_3}\right) + C_L\right] + N_3 g_{ds_3}}. \end{aligned} \tag{3.29}$$

DC Gain

The DC gain is determined by setting the frequency to zero, resulting in:

$$A = \frac{N_1 g_{m_1}}{(N_1+1)g_{ds_1} + g_{m_1}} \cdot \frac{N_2 g_{m_2}}{(N_2+1)g_{ds_2} + g_{m_2}} \cdot \frac{g_{m_3}}{g_{ds_3}}. \tag{3.30}$$

Zeros

To find the positive zeros, the numerator of the transfer function was set to zero, yielding:

$$f_{z_1} = \frac{1}{2\pi}\frac{g_{m_1}}{C_{gd_1}}, \tag{3.31}$$

$$f_{z_2} = \frac{1}{2\pi}\frac{g_{m_2}}{C_{gd_2}}, \tag{3.32}$$

$$f_{z_3} = \frac{1}{2\pi}\frac{g_{m_3}}{C_{gd_3}}. \tag{3.33}$$

Poles

For the negative poles, the denominator of the transfer function was set to zero, giving:

$$f_{p_1} = \frac{1}{2\pi}\frac{(N_1+1)g_{ds_1} + g_{m_1}}{N_1\left(C_{ds_1} + C_{gd_1}\right) + C_{ds_1} + C_{gs_1} + N_2\left(C_{gd_2} + C_{gs_2}\right)}, \tag{3.34}$$

$$f_{p_2} = \frac{1}{2\pi}\frac{(N_2+2)g_{ds_2} + g_{m_2}}{N_2\left(C_{ds_2} + C_{gd_2}\right) + C_{ds_2} + C_{gs_2} + N_3\left(C_{gd_3} + C_{gs_3}\right)}, \tag{3.35}$$

$$f_{p_3} = \frac{1}{2\pi}\frac{N_3 g_{ds_3}}{N_3\left(C_{ds_3} + C_{gd_3}\right) + C_L}. \tag{3.36}$$

Gain-Bandwidth Product

The gain-bandwidth product, calculated by multiplying the gain by the output pole in the context of a single-pole approximation, is given by:

$$GBW = \frac{1}{2\pi}\frac{N_1 g_{m_1}}{(N_1+1)g_{ds_1} + g_{m_1}} \cdot \frac{N_2 g_{m_2}}{(N_2+1)g_{ds_2} + g_{m_2}} \cdot \frac{N_3 g_{m_3}}{N_3\left(C_{ds_3} + C_{gd_3}\right) + C_L}. \tag{3.37}$$

Dissipated Power

The power dissipation of the loading inverter is given by:

$$P_d = V_{DD}(N+1)I_D. \tag{3.38}$$

Therefore, the total power dissipation of the amplifier can be calculated as the sum of the power dissipation for each inverter stage, given by:

$$P_d = V_{DD}\left[(N_1+1)I_{D_1} + (N_2+1)I_{D_2} + N_3 I_{D_3}\right]. \tag{3.39}$$

Input-Referred Noise

The noise in the inverter primarily consists of thermal noise. Given the transistor thermal noise discussed in Chapter 2, the input-referred noise can be calculated by dividing the thermal noise by the inverter gain, as follows:

$$\overline{V_{N_{IN}}^2} = 4K_B T\gamma \frac{(N+1)\left[g_m + (N+1)g_{ds}\right]^2}{N^2 g_m^3}. \tag{3.40}$$

Thus, the amplifier's input-referred noise can be determined by summing the input-referred noise for each stage, while dividing by the product of the gains of those stages as the number of stages increases, as follows:

$$\overline{V_{N_{IN}}^2} = 4K_BT\gamma \left(\frac{(N_1+1)\left[g_{m_1}+(N_1+1)g_{ds_1}\right]^2}{N_1^2 g_{m_1}^3} + \frac{\left[(N_1+1)g_{ds_1}+g_{m_1}\right]^2}{N_1^2 g_{m_1}^2} \cdot \frac{(N_2+1)\left[g_{m_2}+(N_2+1)g_{ds_2}\right]^2}{N_2^2 g_{m_2}^3} + \frac{\left[(N_1+1)g_{ds_1}+g_{m_1}\right]^2}{N_1^2 g_{m_1}^2} \cdot \frac{\left[(N_2+1)g_{ds_2}+g_{m_2}\right]^2}{N_2^2 g_{m_2}^2} \cdot \frac{N_3 g_{ds_3}^2}{g_{m_3}^3} \right). \quad (3.41)$$

Slew Rate

The slew rate is determined by the saturation current of the final stage, with a focus on the rising transition, as follows:

$$SR = N_3 \frac{\frac{1}{2}k\frac{W_3}{L_3}V_{DSAT_{P3}}^2}{C_L}. \quad (3.42)$$

Output Voltage Swing

The output voltage swing is evaluated at the final stage and is given by:

$$V_{OUT_{SW}} = V_{DD} + V_{DSAT_P} - V_{DSAT_N}. \quad (3.43)$$

Common-Mode Input and Output Voltage

The common-mode voltage for both the input and output was assumed to be the trip point voltage of the first stage, as follows:

$$V_{CM_{IN}} = V_{CM_{OUT}} = V_{TP_1}. \quad (3.44)$$

3.3.2 Design Method

The design methodology stands its foundations on the g_m/I_D method [71]. A diode-connected inverter, illustrated in Figure 3.15 and described in Appendix B.1, was analyzed through a series of simulations. In these, the width factor k_W, defined in Equation (3.45) as the ratio of the widths of the NMOS and PMOS transistors in the inverter, was varied from 1 to 10 in increments of 1, with additional values at 2.625 and 3.635. Throughout the simulations, the supply voltage was maintained at a constant 1.2 V, and the NMOS channel width W was set at 1.2 μm, facilitating the extraction of both transistor parameters (I_D, V_{GS}, V_{TH}, V_{DS}, V_{DSAT}, C_{ds}, C_{gd}, C_{gs}, g_{ds}, g_m). This process was repeated across a range of channel lengths (60, 90, 120, 240, 300, and 600 nm), leading to the tables presented in Appendix A.1.

$$W_P = k_W \cdot W, W = 1.2\mu m \tag{3.45}$$

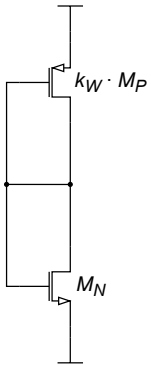


Figure 3.15: Diode-connected inverter schematic.

This methodology facilitated the creation of graphs illustrating the transconductance efficiency of the diode-connected inverter for both NMOS and PMOS transistors, as shown in Figure 3.16 and Figure 3.17, respectively.

Initial observations indicate that as the channel length increases, the gain improves, as shown in the self-gain graph. Conversely, an increase in channel length results in reduced bandwidth, as depicted in the transit frequency graph, along with lower power dissipation, as reflected in the normalized current graph.

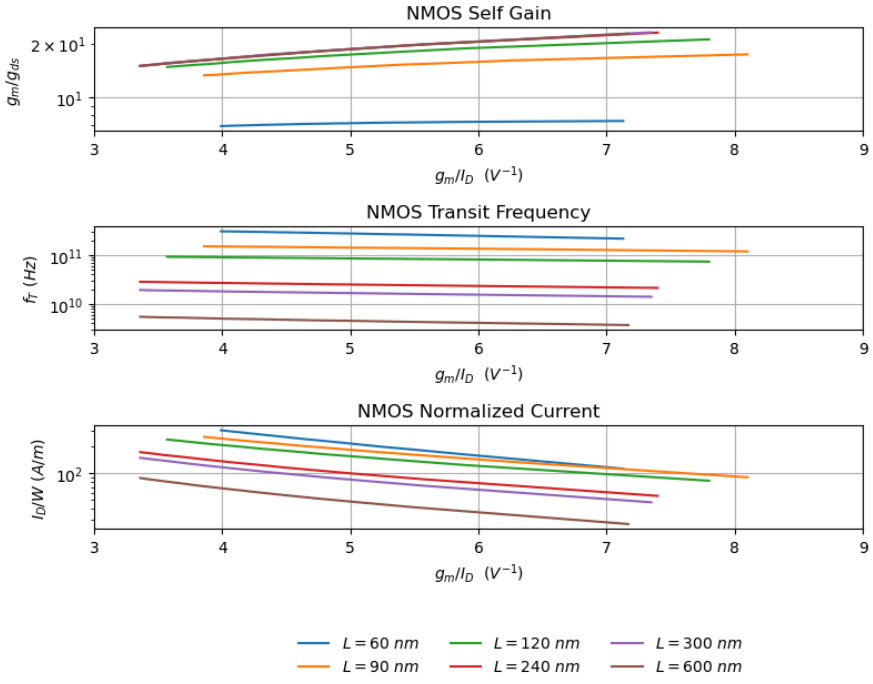


Figure 3.16: Diode-connected inverter NMOS transistor transconductance efficiency.

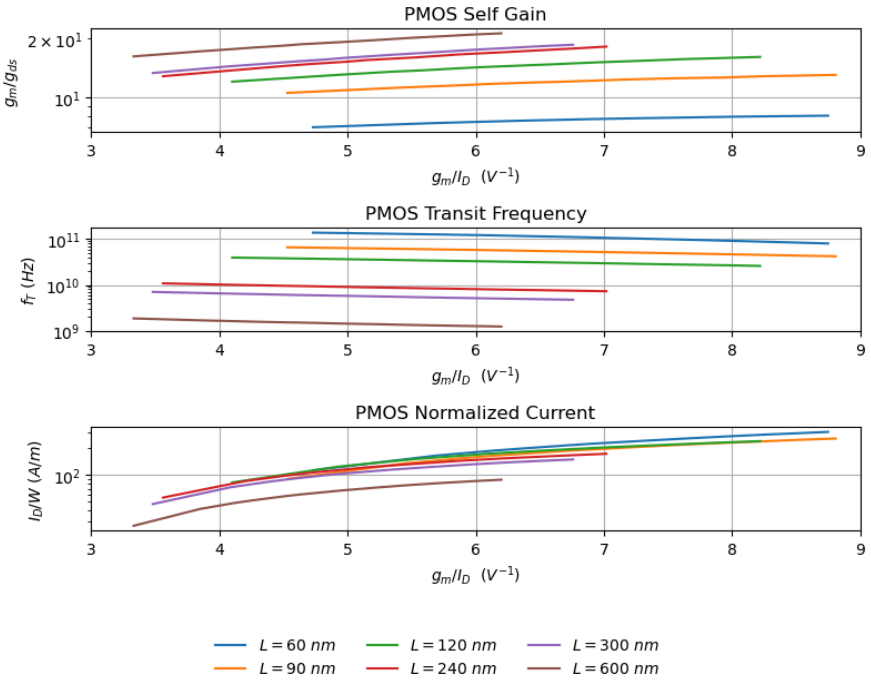


Figure 3.17: Diode-connected inverter PMOS transistor transconductance efficiency.

Another key factor to consider is the inverter trip point voltage, which should be approximately $VDD/2$ (0.6 V) to ensure both transistors operate in the saturation region. This value is highly dependent on the width factor. To better understand this relationship, a graph of trip point voltage versus width factor was generated, as illustrated in Figure 3.18.

This analysis allows for the association of each channel length to a specific width factor, thereby helping to limit the design variables. The corresponding width factors are 2.625 for 60 nm and 90 nm, 3 for 120 nm, 3.625 for 240 nm and 300 nm, and 4 for 600 nm.

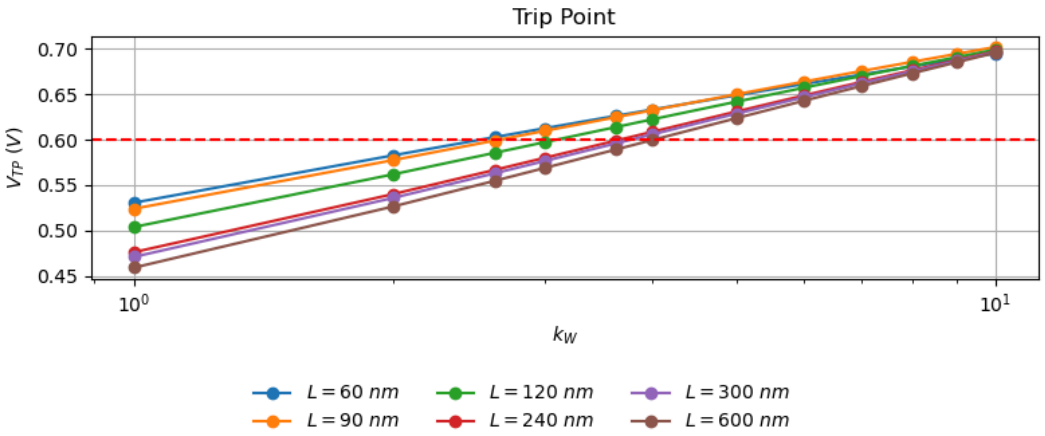


Figure 3.18: Diode-connected inverter trip point voltage.

The design variables for the first two stages are defined by Equations (3.46) and (3.47) for the NMOS and PMOS transistors, respectively, with dots used to distinguish the multi-inverter factors from those of the linear inverter. The final stage is characterized by Equations (3.48) and (3.49) for the NMOS and PMOS transistors, respectively. In this context, N is the stage multiplier, K_S is the scaling factor applied to the reference width W (1.2 μm) to determine the NMOS width, and L represents one of the defined channel lengths from the table (60, 90, 120, 240, 300, or 600 nm). The PMOS width is calculated by multiplying the NMOS width by the width ratio factor k_W.

$$N_i \cdot \frac{K_{S_i} \cdot W}{L_i} : \frac{K_{S_i} \cdot W}{L_i}, i = 1, 2 \tag{3.46}$$

$$N_i \cdot \frac{k_{W_i} \cdot K_{S_i} \cdot W}{L_i} : \frac{k_{W_i} \cdot K_{S_i} \cdot W}{L_i}, i = 1, 2 \tag{3.47}$$

$$N_i \cdot \frac{K_{S_i} \cdot W}{L_i}, i = 3 \tag{3.48}$$

$$N_i \cdot \frac{k_{W_i} \cdot K_{S_i} \cdot W}{L_i}, i = 3 \tag{3.49}$$

Using the tables for each channel length in Appendix A.1 and the theoretical equations provided later, Python graphs (generated with the algorithm in Appendix C.2) visualize key performance metrics — such as DC gain, bandwidth, gain-bandwidth product, power dissipation, input-referred noise, phase margin, and non-dominant pole frequencies — illustrating their interaction with design variables for each stage. In this analysis, the multiplier N and the scaling factor K_S are varied across all stages: N ranges from 1 to 10 in increments of 1, and K_S ranges from 0.1 to 1 in increments of 0.1, both constrained to these ranges for low power consumption.

The graphs for stage 1 are presented in Figures 3.19 and 3.20.

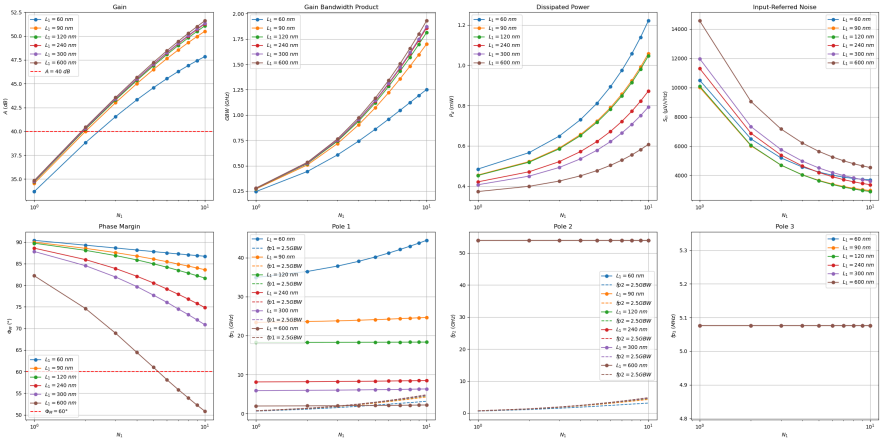


Figure 3.19: Performance metrics variation with the multiplier for stage 1.

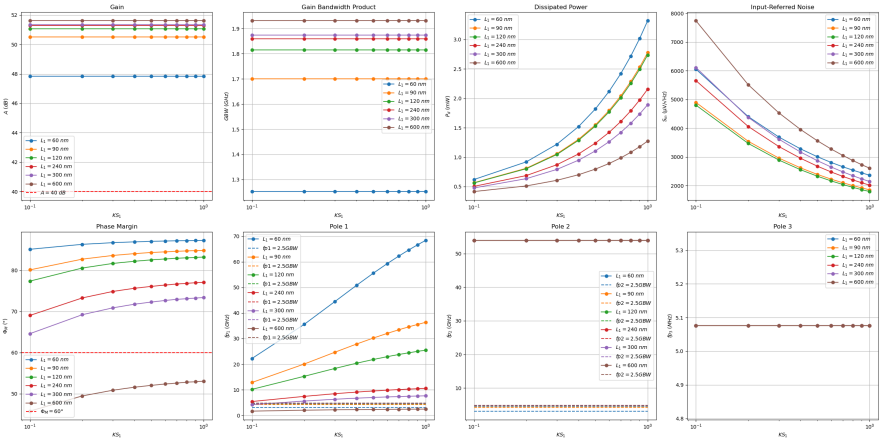


Figure 3.20: Performance metrics variation with the scale factor for stage 1.

The graphs for stage 2 are presented in Figures 3.21 and 3.22.

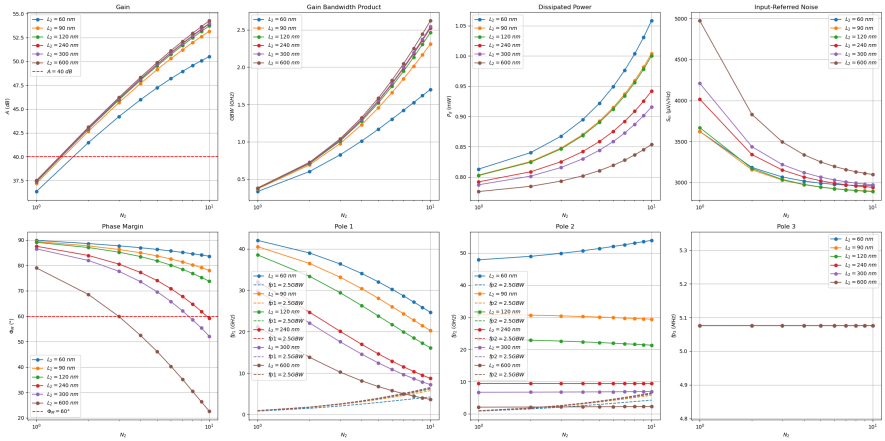


Figure 3.21: Performance metrics variation with the multiplier for stage 2.

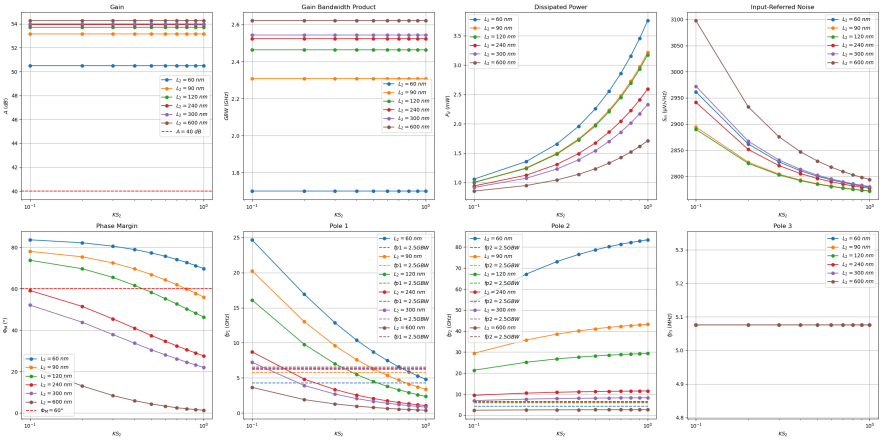


Figure 3.22: Performance metrics variation with the scale factor for stage 2.

The graphs for stage 3 are presented in Figures 3.23 and 3.24.

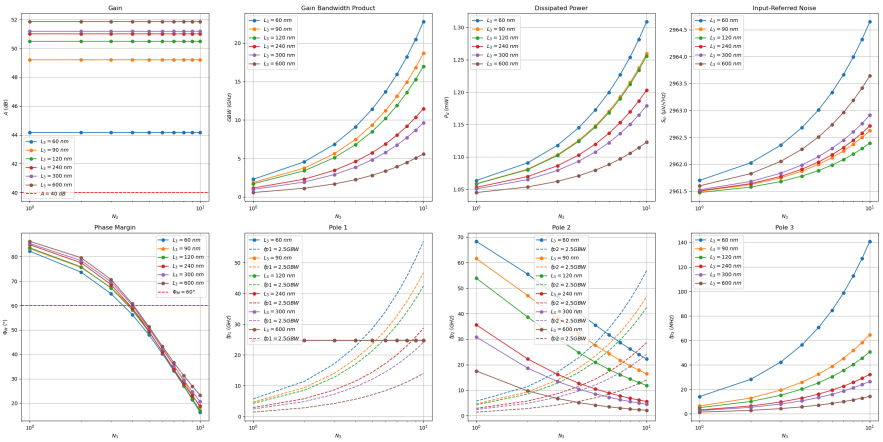


Figure 3.23: Performance metrics variation with the multiplier for stage 3.

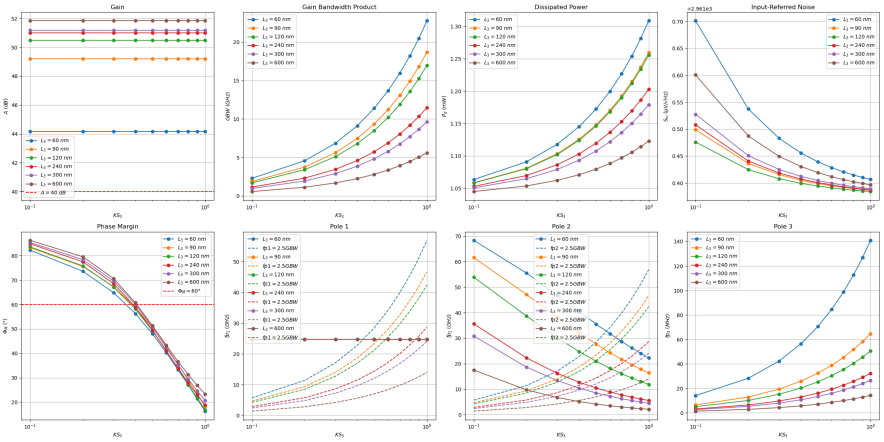


Figure 3.24: Performance metrics variation with the scale factor for stage 3.

Table 3.2 summarizes the findings from these graphs and supports the design by visually representing the relationship between stage variables and amplifier performance metrics, making it easier to evaluate trade-offs.

Table 3.2: Performance metrics associated with design variables. The symbols indicate the type of relationship: ↗ for direct, ↘ for inverse, – for static, and ~ for variable.

	A	GBW	P_d	PM	f_{p_1}	f_{p_2}	f_{p_3}
L_1	↗	↗	↘	~	↘	–	–
N_1	↗	↗	↗	↘	~	–	–
K_{S_1}	–	–	↗	↗	↗	–	–
L_2	↗	↗	↘	~	↘	↘	–
N_2	↗	↗	↗	↘	↘	~	–
K_{S_2}	–	–	↗	~	↘	↗	–
L_3	↗	↘	↘	~	–	↘	↘
N_3	–	↗	↗	↘	–	↘	↗
K_{S_3}	–	↗	↗	~	–	↘	↗

This enables several conclusions to be drawn, including:

- To maximize the DC gain, it is necessary to increase L_1, L_2, L_3, N_1, N_2.
- To maximize the gain-bandwidth product, it is necessary to increase L_1, L_2, N_1, N_2, N_3 and decrease L_3.
- To minimize power dissipation, it is necessary to increase the L_1, L_2, L_3 and decrease N_1, N_2, N_3, K_{S_1}, K_{S_2}, K_{S_3}.
- To achieve stability, by pushing the first non-dominant pole to higher frequencies, it is necessary to decrease L_1, L_2 and increase K_{S_1}.
- To achieve stability, by pushing the second non-dominant pole to higher frequencies, it is necessary to decrease L_1, L_2, N_3.
- To achieve stability, by higher phase margin, it is necessary to decrease N_1, N_2, N_3 and increase K_{S_1}.
- Increasing K_{S_1} is an effective way to reach stability, as it only affects the correspondent pole, leaving the others unaffected, and also increases the phase margin.
- Increasing K_{S_2} and K_{S_3} is not an effective way to reach stability as they impact the adjacent poles differently.
- There is a trade-off between the stability-enhancing tools and performance, affecting both DC gain, gain-bandwidth, and efficiency.

Two design examples are presented: one focused on gain (open-loop gain) and the other on speed (gain-bandwidth product), with both considering efficiency to illustrate how these conclusions can be applied. It is crucial to ensure that both designs meet the following stability criteria: a DC gain of at least 40 dB, a phase margin of at least 60°, and the non-dominant poles should be located at least 2.5 times greater than the gain-bandwidth product.

3.3.2.1 Design Aiming Maximum Gain

The design aimed at achieving maximum gain is presented in Table 3.3, which outlines the process iterations. (**1**) Initially, all stages were assigned maximum lengths, with the largest multipliers applied to the first two stages. However, the phase margin was inadequate. (**2**) As a result, shorter lengths and smaller multipliers were employed to bring the phase margin closer to the target value, leading to lower gain, reduced speed, and increased dissipation. (**3**) An increase in the first stage scale factor was implemented to align it more closely with the phase margin stability target. While this adjustment maintained gain and speed, it also resulted in higher dissipation. Nonetheless, it ultimately proved insufficient. (**4**) Another increase in the first stage scale factor led to even greater dissipation and the stability target was achieved. (**5**) Conversely, the results could not be replicated in simulation, prompting another increase in the first stage scale factor for this setup, yet

the desired phase margin still was not achieved. (**6**) Another adjustment followed, but the target remained unfulfilled. (**7**) Ultimately, the stability target was achieved (60.670°), yielding a gain of 53.603 dB, a GBW of 772.121 MHz, and dissipation of 1.029 mW. The design method estimated a gain of 54.997 dB, a GBW of 807.097 MHz, dissipation of 1.008 mW, a phase margin of 65.794°, and a settling time of 3.632 ns.

Table 3.3: Iterations in design aiming maximum gain.

Iteration	*Stage*	$L(nm)$	N	k_W	K_S	$A(dB)$	$GBW(MHz)$	$P_d(mW)$	$PM(°)$	f_{p_1}/GBW	f_{p_2}/GBW
1	1	600	10	4	0.1	56.707	982.705	0.199	12.276	0.313	1.515
	2	600	10	4	0.1						
	3	600	1	4	0.1						
2	1	300	9	3.625	0.1	54.997	807.097	0.294	48.269	1.520	4.336
	2	300	9	3.635	0.1						
	3	600	1	4	0.1						
3	1	300	9	3.625	0.2	54.997	807.097	0.437	57.498	2.655	4.336
	2	300	9	3.625	0.1						
	3	600	1	4	0.1						
4	1	300	9	3.625	0.3	54.997	807.097	0.578	61.367	3.535	4.336
	2	300	9	3.625	0.1						
	3	600	1	4	0.1						
4	1	300	9	3.625	0.3	53.562	740.422	0.605	55.990	–	–
	2	300	9	3.625	0.1						
	3	600	1	4	0.1						
5	1	300	9	3.625	0.4	53.596	756.273	0.745	58.182	–	–
	2	300	9	3.625	0.1						
	3	600	1	4	0.1						
6	1	300	9	3.625	0.5	53.612	765.833	0.887	59.637	–	–
	2	300	9	3.625	0.1						
	3	600	1	4	0.1						
7	1	300	9	3.625	0.6	53.603	772.121	1.029	60.670	–	–
	2	300	9	3.625	0.1						
	3	600	1	4	0.1						
7	1	300	9	3.625	0.6	54.997	807.097	1.008	65.794	5.288	4.336
	2	300	9	3.625	0.1						
	3	600	1	4	0.1						

The design parameters for achieving maximum gain are summarized in Table 3.4.

Table 3.4: Critically damped ring amplifier parameters for maximum gain.

$L_1(nm)$	N_1	K_{S_1}	k_{W_1}
300	9	0.6	3.625
$L_2(nm)$	N_2	K_{S_2}	k_{W_3}
300	9	0.1	3.625
$L_3(nm)$	N_3	K_{S_3}	k_{W_3}
600	1	0.1	4

3.3.2.2 Design Aiming Maximum Speed

The design focused on achieving maximum speed is detailed in Table 3.5, outlining the process iterations. (**1**) Initially, maximum lengths were assigned to the first two stages, while the last stage received the minimum length, and the largest multipliers were applied to all stages. However, the phase margin remained distant from the desired stability target. (**2**) As a result, minimum lengths were used for the first two stages, and all multipliers were reduced, which brought the phase margin closer to the stability target. (**3**), (**4**) Equivalent to the previous design. (**5**) Similarly, the results could not be replicated in the simulation,

so the maximum value was assigned to the first stage scale factor to maintain gain and speed, but the phase margin stability target was not achieved. (**6**) This led to a reduction in the final multiplier and a proper first stage scale factor, but the target was still not reached. (**7**) Ultimately, after further decreasing the third stage multiplier and a corresponding reduction in the first scale factor, the phase margin was achieved, resulting in a gain of 40.632 dB, a GBW of 5.672 GHz, a dissipation of 2.001 mW, and a phase margin of 60.029°. The design method estimated a gain of 40.652 dB, a GBW of 6.085 GHz, a dissipation of 2.016 mW, a phase margin of 74.094°, and a settling time of 481.817 ps.

Table 3.5: Iterations in design aiming maximum speed.

Iteration	*Stage*	$L(nm)$	N	k_W	K_S	$A(dB)$	$GBW(GHz)$	$P_d(mW)$	$PM(°)$	f_{p_1}/GBW	f_{p_2}/GBW
1	1	600	10	4	0.1	49.027	39.845	0.463	-52.592	0.008	0.042
	2	600	10	4	0.1						
	3	60	10	2.625	0.1						
2	1	60	9	2.625	0.1	40.652	9.123	0.708	52.781	2.546	3.387
	2	60	9	2.625	0.1						
	3	60	6	2.625	0.1						
3	1	60	9	2.625	0.2	40.652	9.123	0.981	58.852	4.057	3.387
	2	60	9	2.625	0.1						
	3	60	6	2.625	0.1						
4	1	60	9	2.625	0.3	40.652	9.123	1.253	61.065	5.058	3.387
	2	60	9	2.625	0.1						
	3	60	6	2.625	0.1						
4	1	60	9	2.625	0.3	40.605	7.522	1.241	43.175	–	–
	2	60	9	2.625	0.1						
	3	60	6	2.625	0.1						
5	1	60	9	2.625	1	40.673	7.945	3.170	47.855	–	–
	2	60	9	2.625	0.1						
	3	60	6	2.625	0.1						
6	1	60	9	2.625	1	40.673	6.912	3.142	54.342	–	–
	2	60	9	2.625	0.1						
	3	60	5	2.625	0.1						
7	1	60	9	2.625	0.6	40.632	5.672	2.001	60.029	–	–
	2	60	9	2.625	0.1						
	3	60	4	2.625	0.1						
7	1	60	9	2.625	0.6	40.652	6.085	2.016	74.094	10.068	6.513
	2	60	9	2.625	0.1						
	3	60	4	2.625	0.1						

The design parameters for achieving maximum speed are summarized in Table 3.6.

Table 3.6: Critically damped ring amplifier parameters for maximum speed.

$L_1(nm)$	N_1	K_{S_1}	k_{W_1}
60	9	0.6	2.625
$L_2(nm)$	N_2	K_{S_2}	k_{W_3}
60	9	0.1	2.625
$L_3(nm)$	N_3	K_{S_3}	k_{W_3}
60	4	0.1	2.625

3.3.3 Electrical Simulation

The simulations were performed in *Cadence Virtuoso* using the testbenches provided in Appendix B.3 and employing the same P_11_SPRVT and N_BPW_SPRVT transistors from the UMC65SP library. The simulations were also carried out with a supply voltage of 1.2 V and a room temperature of 27 °C.

The testbench used to evaluate the critically damped ring amplifier configuration (Figure 2.61) for transient and FFT simulations, with parameters provided in Tables 3.5 and 3.6 for maximum gain and speed, respectively, comprises a pseudo-differential flip-around MDAC with common-mode feedback (CMFB). This is preceded by a multiplexer (implemented in $Verilog-A$, as detailed in Appendix C.1) and followed by a load, as illustrated in Figure 3.25. This setup operates under the same parameters previously described, with a simple modification ensuring that the common-mode input and output are set to the first stage's trip point.

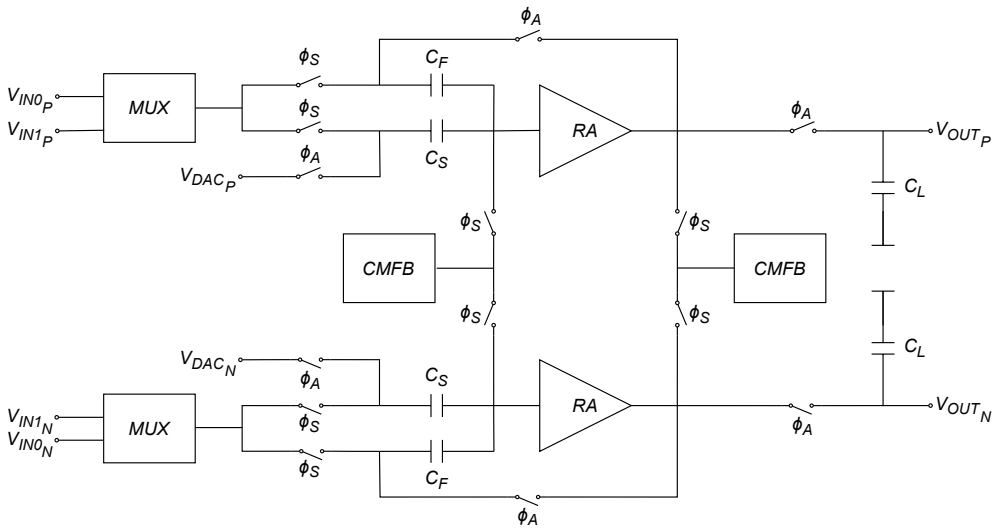


Figure 3.25: Pseudo-differential flip-around MDAC with CMFB.

A new testbench is also introduced to examine frequency response, as depicted in Figure 3.26 it consists of a probe in negative feedback.

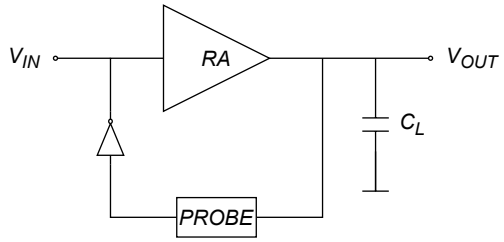


Figure 3.26: Probe in negative feedback.

Common-Mode Input and Output Signals

Both input and output common-mode voltages were set to the first stage trip point V_{TP_1} (597.99 mV), as shown in Figure 3.27, to ensure effective operation.

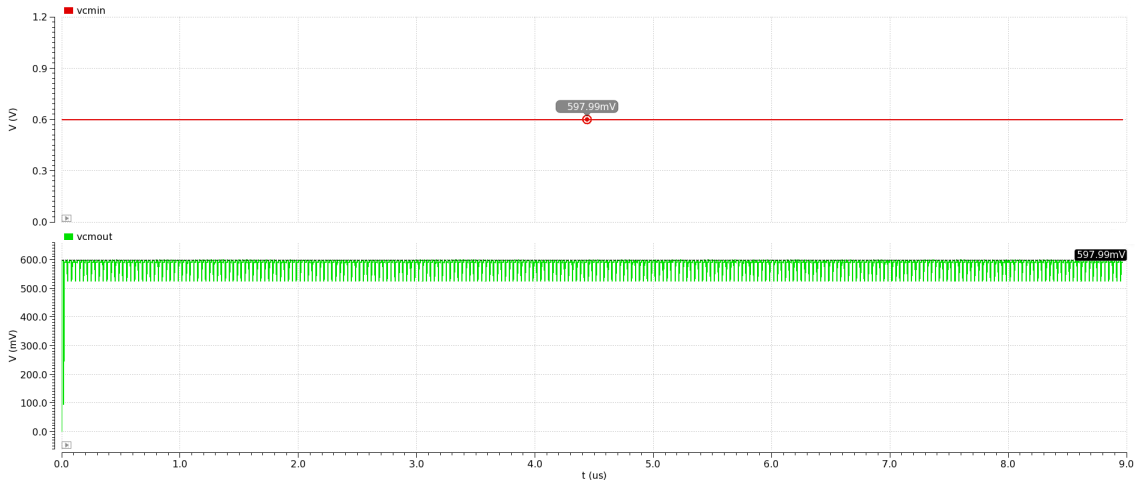


Figure 3.27: Critically damped ring amplifier input and output common-mode signals.

Nodal Signals

The nodal voltages between stages were reduced to a single pair, as shown in Figure 3.28. Additionally, no specific dead zone was required for proper amplification.

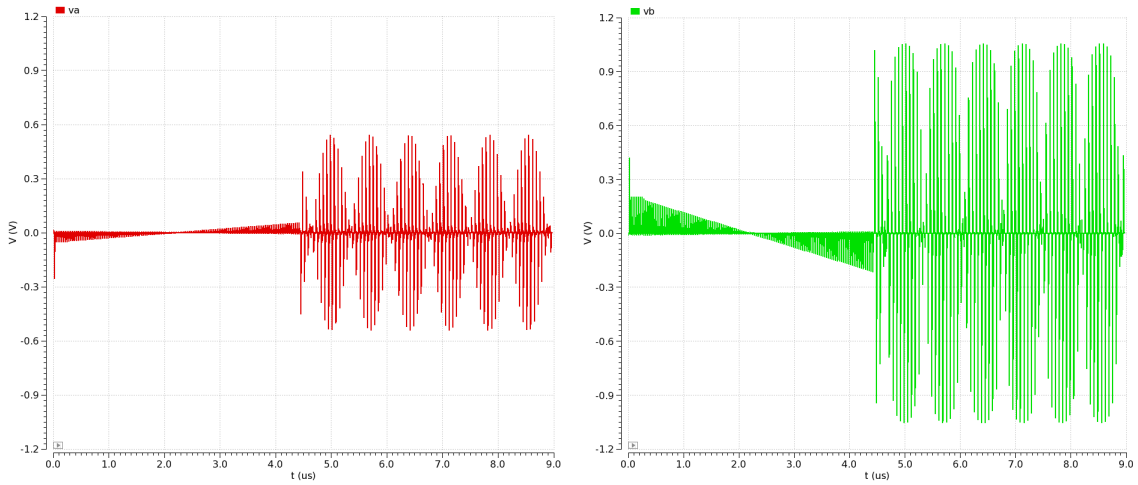


Figure 3.28: Critically damped nodal differential signals between stages.

3.3.3.1 Design Aiming Maximum Gain

Feedback Gain

The input and output differential signals of the critically damped ring amplifier for the maximum gain are shown in Figure 3.29. The feedback gain is calculated as the ratio of the output signal's slope to the input signal's slope, yielding a value of 3.942 V/V, which

confirms that the amplifier operates within the specified resolution, tough with some distortion.

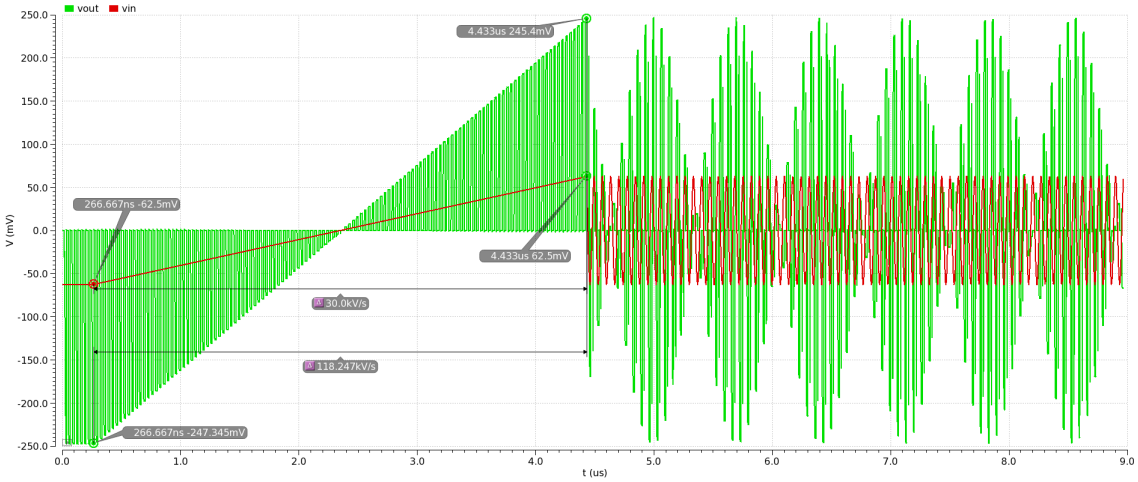


Figure 3.29: Conventional ring amplifier input and output differential signals for maximum gain.

Distortion

This distortion is analyzed in more detail in Figure 3.30. There the deviation between the theoretical and simulated amplified ramps is evident. However, a more effective graph for comparing this deviation is the one showing feedback gain versus input signal. This allows for the approximation of the THD, calculated as the ratio of the gain drop at an input level of 62.5 mV to the maximum gain, resulting in -64.752 dB. From this, the ENOB can be estimated by dividing the absolute value of the THD by 6, yielding approximately 10.792 bits.

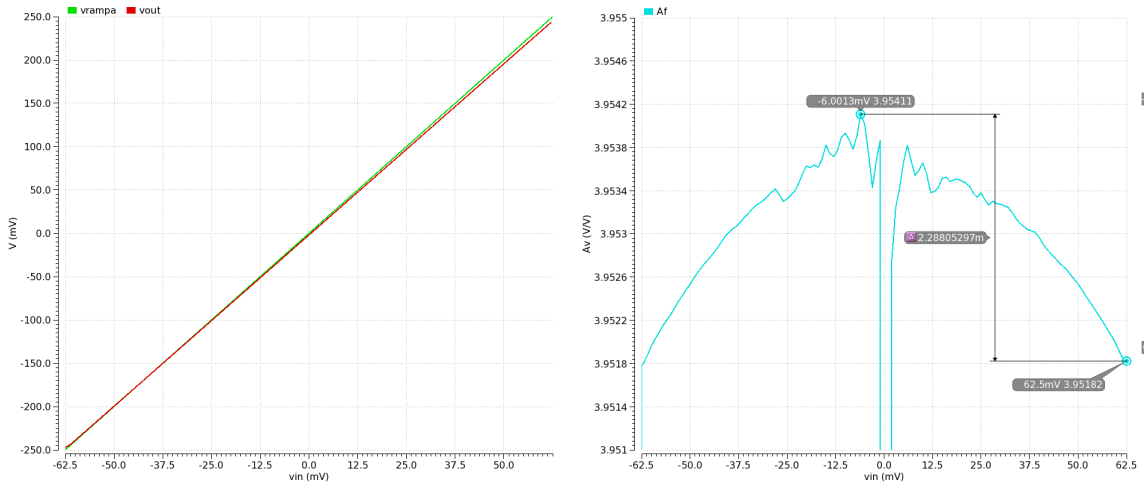


Figure 3.30: Output signal and feedback gain versus input signal for maximum gain.

FFT

However, these values are only rough estimates, so the FFT spectrum of the output signal was generated considering 128 samples, as shown in Figure 3.31. The signal carrier

14.297 MHz is measured at -12.156 dB, while the noise floor is at -103.990 dB and the DC component is at -114.781 dB. The highest spur registers at -79.433 dB and the total distortion harmonics is set at -79.310 dB. The SFDR is calculated by subtracting the noise floor from the signal carrier, resulting in 67.277 dB. Additionally, this analysis yields an ENOB of 13.039 bits, a THD of -67.153 dB, a SNR of 73.840 dB, and a SNDR of 66.383 dB.

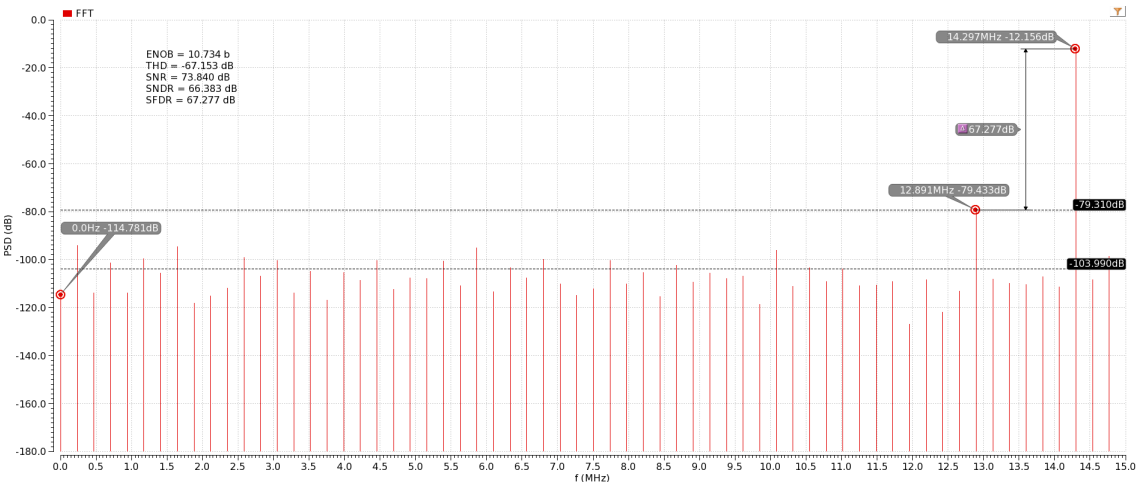


Figure 3.31: FFT spectrum of the output signal for maximum gain.

Settling Time

The settling time is determined from the output signal on the last ramp cycle as illustrated in Figure 3.32 with a 1% error margin. This is calculated by taking 99% of the final value and measuring the time difference between that point and the signal's starting point, yielding 3.512 ns.

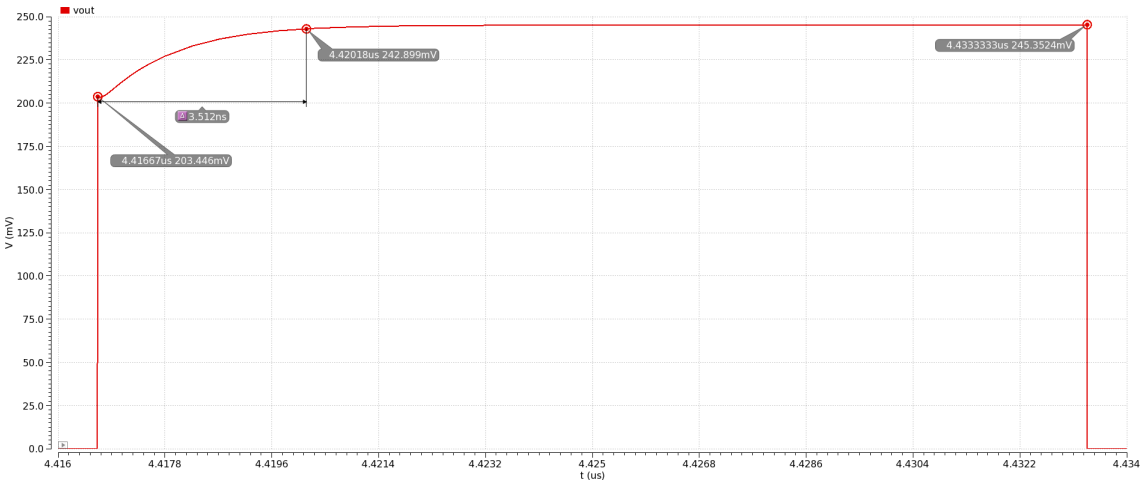


Figure 3.32: Settling time measured from the output signal for maximum gain.

Frequency Response

The frequency response, shown in Figure 3.33, reveals a low-frequency gain of 53.630 dB. The GBW is found to be 772.121 MHz at unity gain, and the phase margin measured at

this frequency corresponds to 60.670°. Additionally, the power dissipation was measured as 1.029 mW.

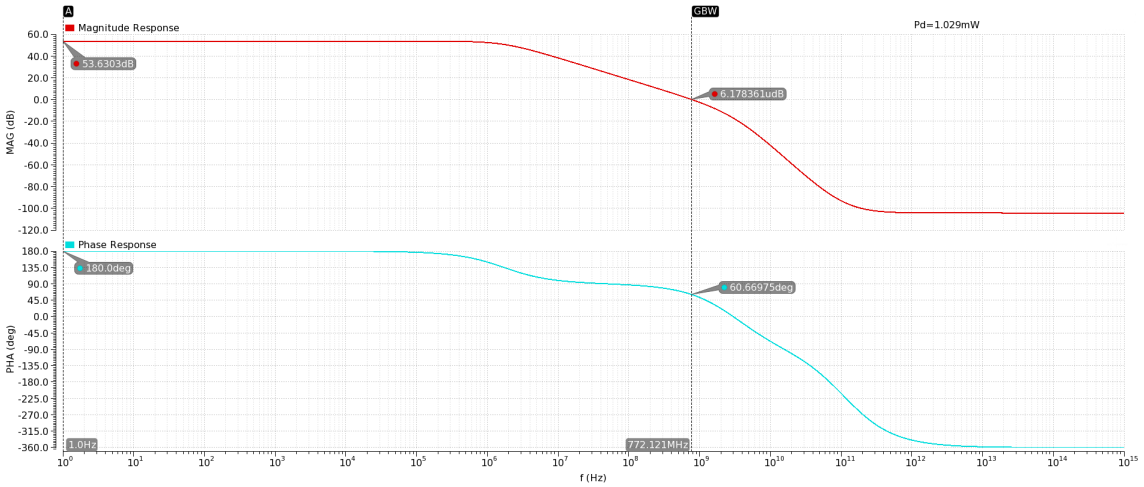


Figure 3.33: Frequency response for maximum gain.

3.3.3.2 Design Aiming Maximum Speed

Feedback Gain

The input and output differential signals of the critically damped ring amplifier for the maximum speed are shown in Figure 3.34. A feedback gain of 3.821 V/V was achieved, indicating proper operation, though some distortion was present.

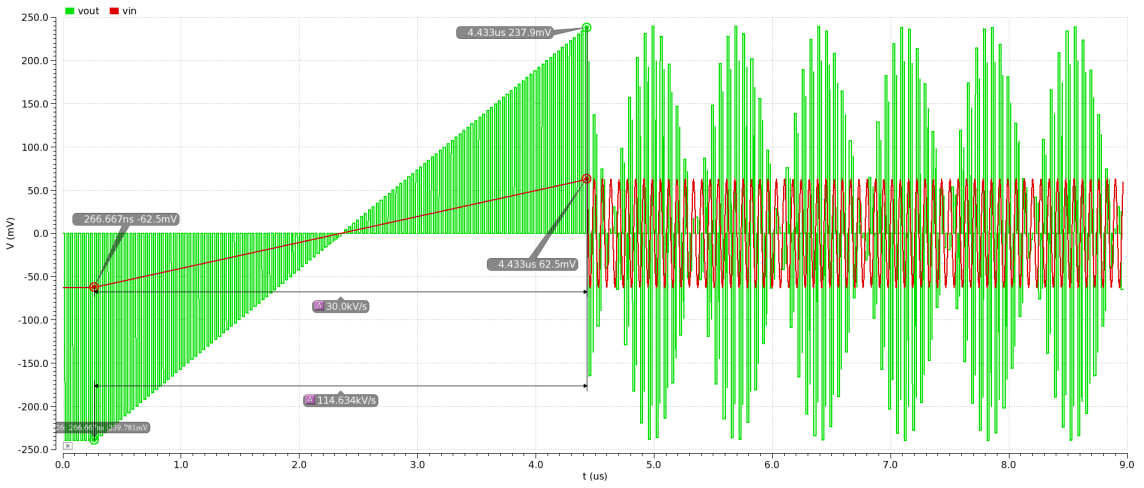


Figure 3.34: Conventional ring amplifier input and output differential signals for maximum speed.

Distortion

This distortion is analyzed in more detail in Figure 3.35. There the deviation between the theoretical and simulated amplified ramps is also evident. The estimated THD is -63.455 dB, corresponding to an ENOB of 10.576 bits.

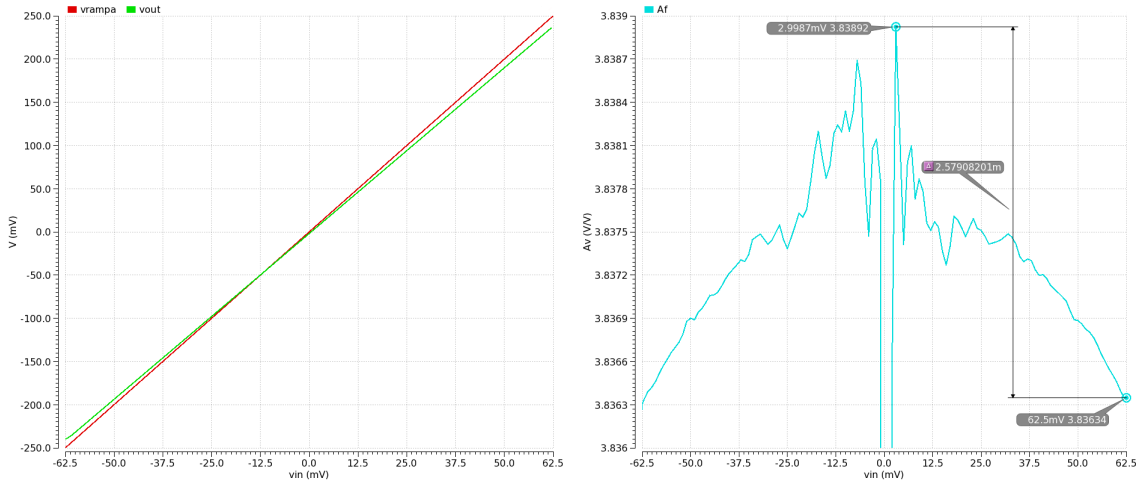


Figure 3.35: Output signal and feedback gain versus input signal for maximum speed.

FFT

This result appears to be worse than the previous design, but as it is only an approximation, no conclusions should be drawn. To gain further insight, the FFT spectrum of the output signal was generated again, as shown in Figure 3.36. The signal carrier 14.297 is measured at -12.403 dB, while the noise floor is at -126.878 dB and the DC component is -138.712 dB. The highest spur registers at -92.872 dB, and the total harmonic distortion is at -92.760 dB. The SFDR is reported as 80.469 dB. Furthermore, this analysis provides an ENOB of 13.059 bits, a THD of -80.357 dB, a SNR of 96.481 dB, and a SNDR of 80.469 dB, indicating that the approximation is not reliable.

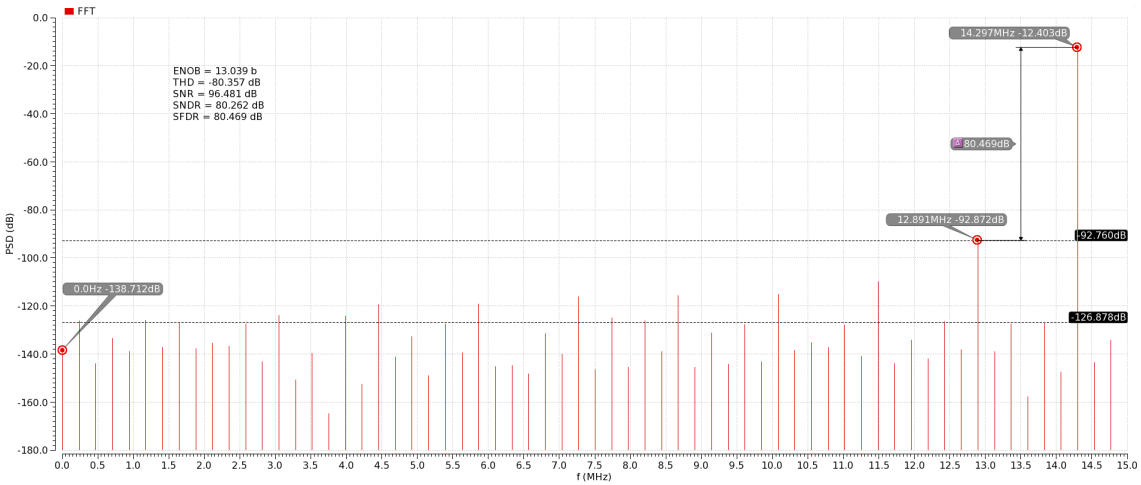


Figure 3.36: FFT spectrum for maximum speed.

Settling Time

The output signal at the last cycle of the ramp is presented in Figure 3.37. For this design, a settling time of 345.335 ps was achieved.

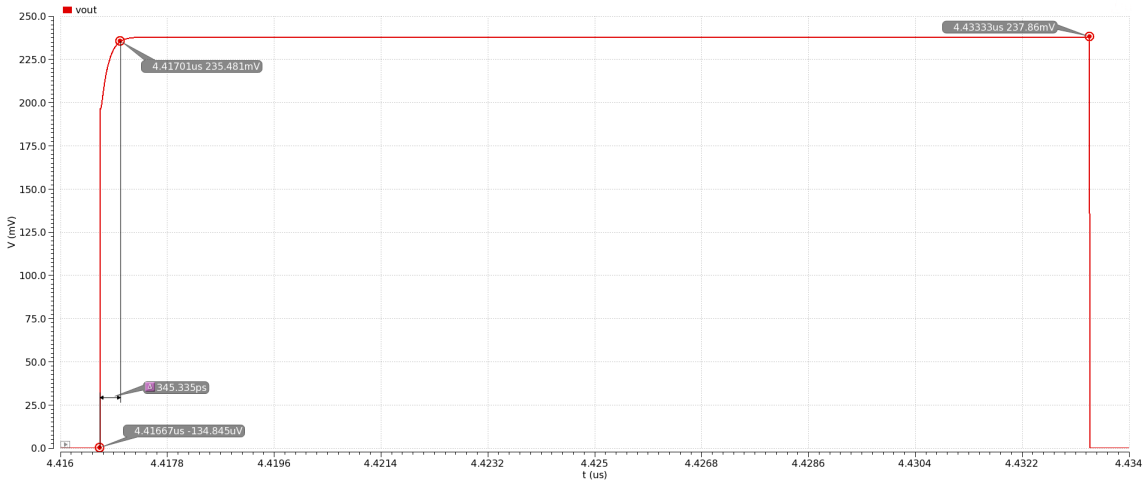


Figure 3.37: Settling time measured from the output signal for maximum speed.

Frequency Response

The frequency response concerning the maximum speed design is represented in Figure 3.38. It is possible to observe a low-frequency gain of 40.632 dB, a GBW of 5.672 GHz and a phase margin of 60.029° associated with a power dissipation of 2.001 mW.

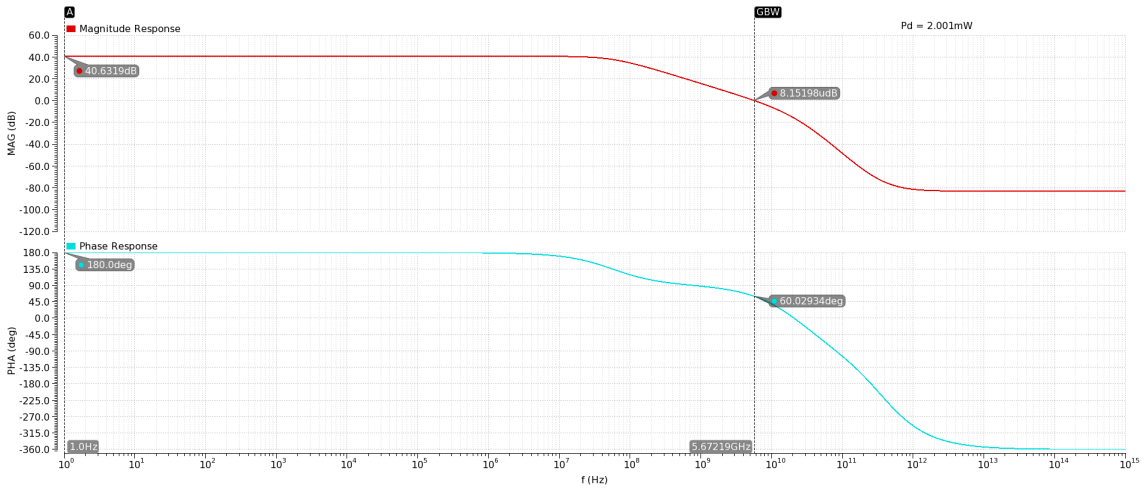


Figure 3.38: Frequency response for maximum speed.

3.4 Conclusion

The residue amplifier was configured using the parameters specified in Table 3.7, where the design method was implemented. However, both the resolution and the feedback factor can be adjusted easily, ensuring the versatility of the design.

Table 3.7: Residue amplifier parameters.

$CMOS(nm)$	$V_{DD}(V)$	$V_{REF}(V)$	$N(b)$	$f_S(Msps)$	$C_L(fF)$	$A_F(V/V)$	$\beta(V/V)$
65	1.2	1	3	30	400	4	0.25

The discrepancies between the design models and the simulations are summarized in Table 3.8 for maximum gain and in Table 3.9 for maximum speed. The relative error remains relatively low for the design focused on maximum gain across all performance metrics. However, the design aimed at maximum speed exhibits a slight error in both phase margin and settling time that may be due to simulation conditions. Overall, the approximation between the design method and the simulations is satisfactory.

Table 3.8: Deviation in efficiency performance metrics for maximum gain.

$Metric$	$Model$	$Simulation$	$\epsilon(\%)$
$A(dB)$	54.997	53.630	2.486
$GBW(MHz)$	807.097	772.121	4.334
$P_d(mW)$	1.008	1.029	2.083
$PM(°)$	65.794	60.670	7.788
$t_s(ns)$	3.632	3.512	3.304

Table 3.9: Deviation in efficiency performance metrics for maximum speed.

$Metric$	$Model$	$Simulation$	$\epsilon(\%)$
$A(dB)$	40.652	40.632	0.049
$GBW(GHz)$	6.085	5.672	6.787
$P_d(mW)$	2.016	2.001	0.744
$PM(°)$	74.094	60.029	18.983
$t_s(ps)$	481.817	345.335	28.327

The dynamic performance metrics for maximum gain are displayed in Table 3.10, while those for maximum speed are shown in Table 3.11. Both designs successfully achieved all requirements, indicating that the work was conducted effectively.

Table 3.10: Dynamic performance metrics in maximum gain design.

$Metric$	$Simulation$
$ENOB(b)$	10.734
$THD(dB)$	-67.153
$SNR(dB)$	73.840
$SNDR(dB)$	66.383
$SFDR(dB)$	67.277

Table 3.11: Dynamic performance metrics in maximum speed design.

Metric	*Simulation*
$ENOB(b)$	13.039
$THD(dB)$	-80.357
$SNR(dB)$	96.481
$SNDR(dB)$	80.262
$SFDR(dB)$	80.469

To visually demonstrate the validity of the design method, the frequency responses from the design and simulation were compared. Figure 3.39 illustrates this comparison for maximum gain, while Figure 3.40 shows it for maximum speed. In both cases, minimal deviations are observed.

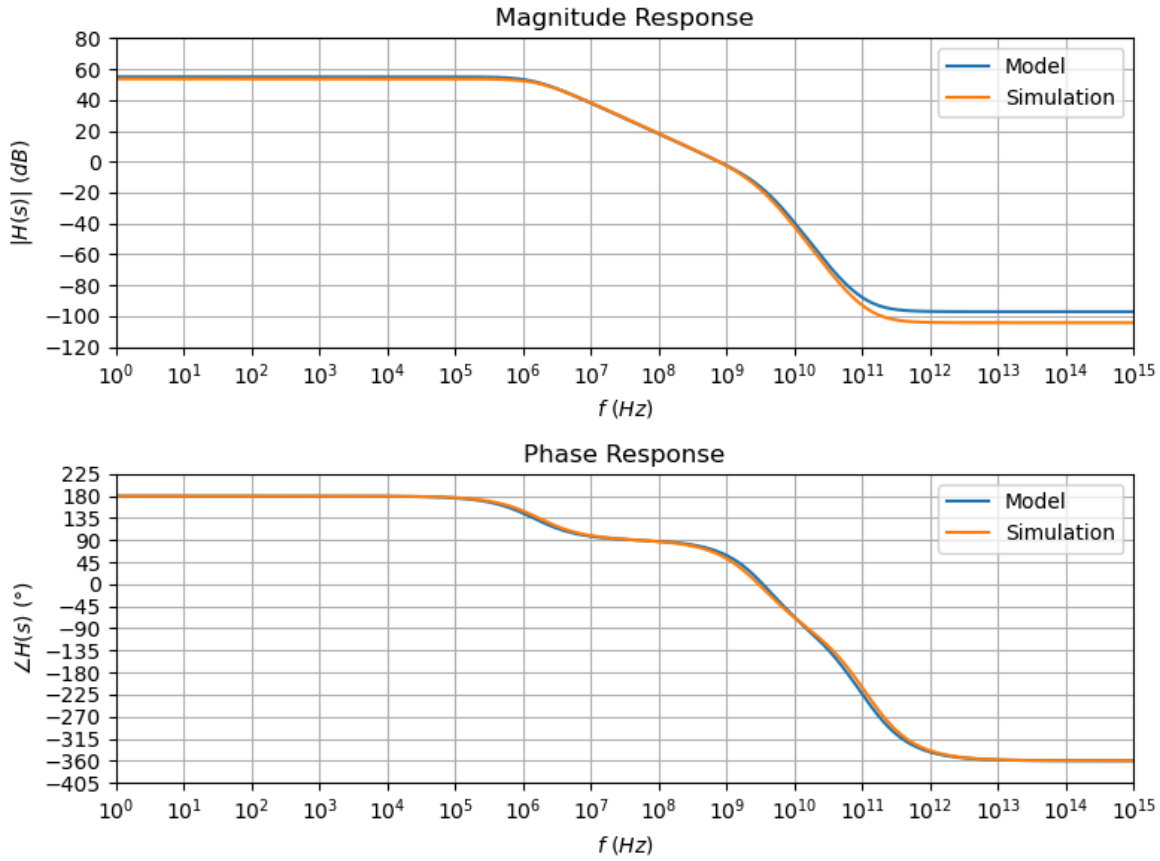


Figure 3.39: Comparison between the designed and simulated frequency response for the maximum gain.

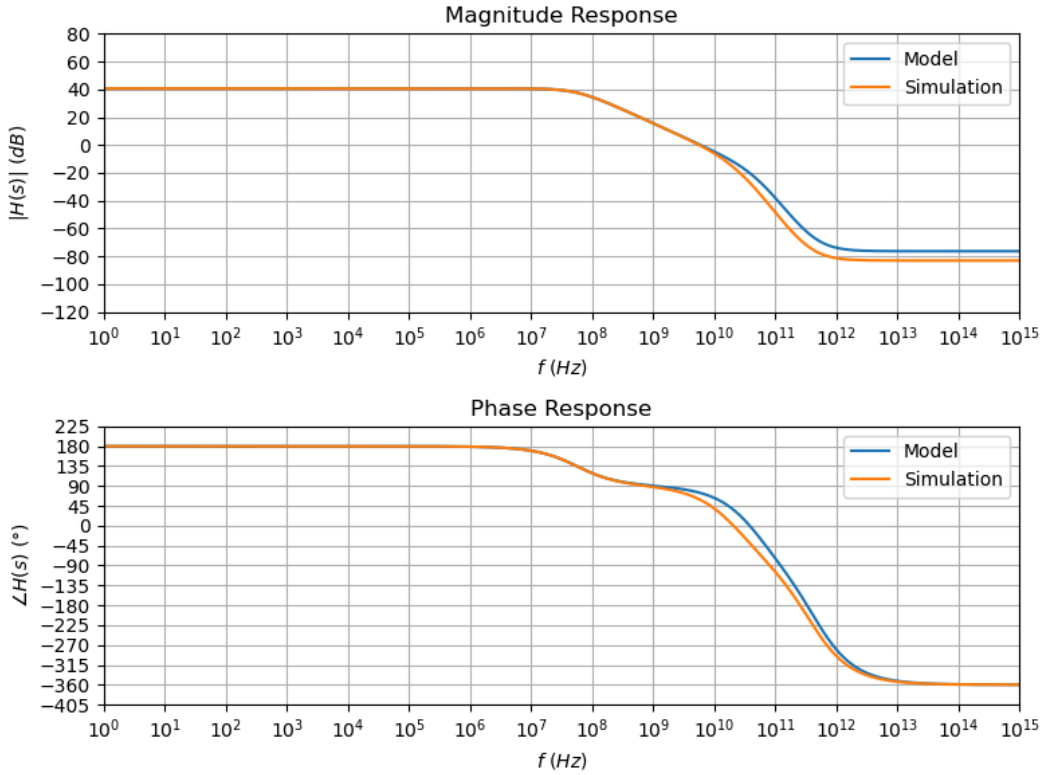


Figure 3.40: Comparison between the designed and simulated frequency response for the maximum speed.

These comparisons strongly suggest that the design method accurately captures the performance characteristics of the critically ramped ring amplifier. The close agreement between theoretical and simulated frequency responses suggests reliable predictions of the behavior of the amplifier. Overall, the results reinforce confidence in the design approach and its relevance in real-world applications. Furthermore, the design successfully fulfills the required specifications, highlighting its practicality.

4

Conclusion

4.1 Contributions

This work provides a thorough survey of the current state of ring amplifiers in pipeline analog-to-digital converters. It includes visual representations, such as graphs and tables, that highlight their evolution over the years. Furthermore, the study examines the most significant configurations, providing explanations and context for their relevance in the field.

This work presents a novel design methodology leveraging computer-assisted techniques based on the g_m/I_D method, enabling designers to explore and optimize various parameters efficiently. This approach streamlines the design process by allowing multiple iterations without extensive simulations, simplifying the integration of residue amplifiers — particularly critically damped ring amplifiers. The straightforward design and implementation of critically damped ring amplifiers make them ideal for this methodology. By bridging theoretical concepts with practical applications, this research enhances the adaptability and efficiency of ring amplifiers for a wide range of System-on-Chip devices.

The design method has been successfully tested in a flip-around configuration with a 3-bit resolution. Nevertheless, it is versatile enough to be adapted to other configurations by simply adjusting the feedback factor associated with the respective architecture. Furthermore, if another resolution is to be studied, it can also be easily changed.

A 65 nm CMOS critically damped ring amplifier was successfully developed, featuring two designs: one optimized for open-loop gain, achieving 53.6 dB, and another focused on optimizing GBW, reaching 5.7 GHz with a correspondent settling time of 345.7 ps. The gain-oriented design delivered an ENOB of 10.7 bits, a THD of -67.1 dB, and consumed 1.0 mW. In contrast, the GBW-oriented design achieved an ENOB of 13.0 bits, a THD of -80.4 dB, and consumed 2.0 mW. Both designs met the specifications and are considered relevant in the current state of the art.

4.2 Limitations

The design method applies to both conventional and critically damped ring amplifiers; however, only the latter were analyzed for comparison with simulations, as the method has been specifically developed for this type. Additionally, it is also important to note that the design variables were maintained within a reasonable range, as excessively increasing their values could negatively impact efficiency.

Another important consideration is that this design is optimized for 65 nm CMOS technology, where all width specifications were successfully accommodated in simulations. However, other technologies may not support these dimensions. Furthermore, fabricating widths smaller than 1 μm presents challenges in the manufacturing process.

4.3 Future Work

Since the testbench was developed in a parametric format, it would be valuable to conduct a comprehensive corner analysis through PVT simulations in future work. This would allow for a more thorough evaluation of the critically damped ring amplifier performance under varying conditions, helping to identify potential vulnerabilities and ensuring robustness in real-world applications. Such simulations would enhance the understanding of how different factors interact, ultimately leading to improved design reliability and performance.

The design strategy has been applied in both academic and corporate environments, assisting students in designing residue amplifiers for their dissertations and being adapted to a recent critically damped configuration by Renesas. An article titled "A Design Strategy Based on the g_m/I_D Method for Critically Damped Ring Amplifiers," which explains the design method, has been submitted to ISCAS 2025.

4.4 Final Thoughts

Initially, there was some deviation regarding the direction of the study. However, as the research progressed, it found its distinct focus, culminating in a successful dissertation that provides an in-depth examination of the state of the art in ring amplifiers. It thoroughly analyzed two specific configurations and developed a novel design methodology that bridges theoretical values with practical applications. This work reinforces the assertion that ring amplifiers are essential in the semiconductor industry and warrant further exploration.

Moreover, the application of diverse tools and knowledge throughout this project highlights a solid understanding of both theoretical concepts and practical implementations gained during the entire master's program. This experience not only deepened my skills but also reinforced my ability to apply what I learned, in real-world scenarios.

Bibliography

[1] M. Akter, K. Makinwa, and K. Bult. "A Capacitively Degenerated 100-dB Linear 20–150 MS/s Dynamic Amplifier". In: *IEEE Journal of Solid-State Circuits* 53.4 (2018), pp. 1115–1126. DOI: 10.1109/JSSC.2017.2778277 (cit. on pp. 31–33).

[2] C. Bae et al. "An 11-bit Ring Amplifier Pipeline ADC with Settling-Time Improvement Scheme". In: *2020 International Conference on Electronics, Information, and Communication*. 2020, pp. 1–4. DOI: 10.1109/ICEIC49074.2020.9051188 (cit. on p. 88).

[3] B. Black. "Analog-to-Digital Converter Architectures and Choices for System Design". In: *Analog Dialogue* 33 (1999). URL: https://www.analog.com/en/analog-dialogue/articles/analog-to-digital-converter-architectures-and-choices (cit. on pp. 11, 14, 17, 21).

[4] Y. Cao et al. "A Single-Channel 12b 2GS/s PVT-Robust Pipelined ADC with Critically Damped Ring Amplifier and Time-Domain Quantizer". In: *2023 IEEE International Solid-State Circuits Conference (ISSCC)*. 2023, pp. 9–11. DOI: 10.1109/ISSCC42615.2023.10067687 (cit. on pp. 35, 40, 41, 43, 88).

[5] Y. Cao et al. "A Single-Channel 12b 2GS/s PVT-Robust Pipelined ADC with Critically Damped Ring Amplifier and Time-Domain Quantizer". In: State Key Laboratory of Analog, Mixed-Signal VLSI, University of Macau, and Instituto Superior Técnico, Universidade de Lisboa. 2023 (cit. on p. 44).

[6] T. Carusone, D. Johns, and K. Martin. *Analog Integrated Circuit Design*. 2nd. Wiley, 2011. ISBN: 978-0-470-77010-8 (cit. on pp. 4, 5, 10, 12–14, 18, 27).

[7] W.-T. Chen et al. "A Pipeline ADC with Latched-Based Ring Amplifiers". In: *2016 IEEE International Symposium on Circuits and Systems*. 2016, pp. 85–88. DOI: 10.1109/ISCAS.2016.7527176 (cit. on p. 88).

[8] Y. Chen et al. "A 200MS/s, 11 bit SAR-Assisted Pipeline ADC with Bias-Enhanced Ring Amplifier". In: *2017 IEEE International Symposium on Circuits and Systems*. 2017, pp. 1–4. DOI: 10.1109/ISCAS.2017.8050244 (cit. on pp. 37, 38, 43, 88).

[9] Y. Chen et al. "A 625MS/s, 12-Bit, SAR Assisted Pipeline ADC with Effective Gain Analysis for Inter-stage Ringamps". In: *IEEE 45th European Solid State Circuits Conference*. 2019, pp. 197–200. DOI: 10.1109/ESSCIRC.2019.8902892 (cit. on p. 88).

[10] Y. Chen et al. "A 800 MS/s, 12-Bit, Ringamp-Based SAR assisted Pipeline ADC with Gain Error Cancellation". In: *2018 IEEE International Symposium on Circuits and Systems*. 2018, pp. 1–4. DOI: 10.1109/ISCAS.2018.8351074 (cit. on p. 88).

[11] Y. Chen et al. "A Time-Interleaved SAR Assisted Pipeline ADC With a Bias-Enhanced Ring Amplifier". In: *IEEE Transactions on Circuits and Systems II: Express Briefs* 65.11 (2018), pp. 1584–1588. DOI: 10.1109/TCSII.2017.2769107 (cit. on p. 88).

[12] N. Çikan and M. Aksoy. "Analog to Digital Converters Performance Evaluation Using Figure of Merits in Industrial Applications". In: *2016 European Modelling Symposium*. 2016, pp. 205–209. DOI: 10.1109/EMS.2016.043 (cit. on p. 9).

[13] E. Coutes. *Amplifiers*. Learnabout Electronics, 2012. URL: https://learnabout-electronics.org/Downloads/amplifiers-module-01.pdf (cit. on pp. 21–23).

[14] *Demystifying Delta-Sigma ADCs*. Maxim Integrated. 2003. URL: http://www.maximintegrated.com/an1870 (cit. on pp. 12, 14).

[15] A. ElShater et al. "A 10-mW 16-b 15-MS/s Two-Step SAR ADC With 95-dB DR Using Dual-Deadzone Ring Amplifier". In: *IEEE Journal of Solid-State Circuits*. Vol. PP. 2019, pp. 1–11. DOI: 10.1109/JSSC.2019.2943935 (cit. on p. 88).

[16] A. ElShater et al. "A 10mW 16b 15MS/s Two-Step SAR ADC with 95dB DR Using Dual-Deadzone Ring-Amplifier". In: *2019 IEEE International Solid-State Circuits Conference*. 2019, pp. 70–72. DOI: 10.1109/ISSCC.2019.8662400 (cit. on p. 88).

[17] L. Fan et al. "A Fast Settling Low Noise Ring Amplifier for High-Speed Pipelined SAR ADCs". In: *2020 IEEE 15th International Conference on Solid-State and Integrated Circuit Technology*. 2020, pp. 1–3. DOI: 10.1109/ICSICT49897.2020.9278374 (cit. on p. 88).

[18] J. Goes et al. "Purely-Digital versus Mixed-Signal Self-Calibration Techniques in High-Resolution Pipeline ADCs". In: *2010 Norchip*. 2010, pp. 1–8. DOI: 10.1109/NORCHIP.2010.5669424 (cit. on p. 19).

[19] P. Gray et al. *Analysis and Design of Analog Integrated Circuits*. 5th. Willey, 2009. ISBN: 978-0-470-24599-6 (cit. on pp. 25–27).

[20] B. Hershberg. *Ringamp Survey 2012-2023*. URL: http://benjamin.hershberg.com/ringamp-survey (cit. on pp. 42, 85).

[21] B. Hershberg. "Ringamp: The Scalable Amplifier We've All Been Waiting For?" In: IEEE CICC. 2020. URL: https://www.youtube.com/watch?v=TtxKnD2BVwc&t=326s (cit. on p. 34).

[22] B. Hershberg and U.-K. Moon. "A 75.9dB-SNDR 2.96mW 29fJ/conv-step Ringamp-Only Pipelined ADC". In: *2013 Symposium on VLSI Circuits*. 2013, pp. C94–C95. URL: https://ieeexplore.ieee.org/abstract/document/6578731 (cit. on p. 88).

[23] B. Hershberg et al. "A 1-MS/s to 1-GS/s Ringamp-Based Pipelined ADC With Fully Dynamic Reference Regulation and Stochastic Scope-on-Chip Background Monitoring in 16 nm". In: *IEEE Journal of Solid-State Circuits* 56.4 (2021), pp. 1227–1240. DOI: 10.1109/JSSC.2020.3044831 (cit. on p. 88).

[24] B. Hershberg et al. "A 1MS/s to 1GS/s Ringamp-Based Pipelined ADC with Fully Dynamic Reference Regulation and Stochastic Scope-on-Chip Background Monitoring in 16nm". In: *2020 IEEE Symposium on VLSI Circuits*. 2020, pp. 1–2. DOI: 10.1109/VLSICircuits18222.2020.9162788 (cit. on p. 88).

[25] B. Hershberg et al. "A 3.2GS/s 10 ENOB 61mW Ringamp ADC in 16nm with Background Monitoring of Distortion". In: *2019 IEEE International Solid-State Circuits Conference*. 2019, pp. 58–60. DOI: 10.1109/ISSCC.2019.8662290 (cit. on p. 88).

[26] B. Hershberg et al. "A 4-GS/s 10-ENOB 75-mW Ringamp ADC in 16-nm CMOS With Background Monitoring of Distortion". In: *IEEE Journal of Solid-State Circuits* 56.8 (2021), pp. 2360–2374. DOI: 10.1109/JSSC.2021.3053893 (cit. on p. 88).

[27] B. Hershberg et al. "A 6-to-600MS/s Fully Dynamic Ringamp Pipelined ADC with Asynchronous Event-Driven Clocking in 16nm". In: *2019 IEEE International Solid-State Circuits Conference*. 2019, pp. 68–70. DOI: 10.1109/ISSCC.2019.8662319 (cit. on p. 88).

[28] B. Hershberg et al. "A 61.5dB SNDR Pipelined ADC Using Simple Highly-Scalable Ring Amplifiers". In: *2012 Symposium on VLSI Circuits*. 2012, pp. 32–33. DOI: 10.1109/VLSIC.2012.6243775 (cit. on p. 88).

[29] B. Hershberg et al. "Asynchronous Event-Driven Clocking and Control in Pipelined ADCs". In: *IEEE Transactions on Circuits and Systems I: Regular Papers* 68.7 (2021), pp. 2813–2826. DOI: 10.1109/TCSI.2021.3077881 (cit. on p. 88).

[30] B. Hershberg et al. "Ring Amplifiers for Switched Capacitor Circuits". In: *IEEE Journal of Solid-State Circuits* 47.12 (2012), pp. 2928–2942. DOI: 10.1109/JSSC.2012.2217865 (cit. on pp. 34–36, 43, 48, 88).

[31] B. Hershberg et al. "Ring Amplifiers for Switched-Capacitor Circuits". In: Oregon State University and Asahi Kasei Microdevices. 2012 (cit. on p. 44).

[32] B. Hershberg et al. "Ring Amplifiers for Switched-Capacitor Circuits". In: *2012 IEEE International Solid-State Circuits Conference*. 2012, pp. 460–462. DOI: 10.1109/ISSCC.2012.6177090 (cit. on p. 88).

[33] T.-C. Hung and T.-H. Kuo. "A 75.3-dB SNDR 24-MS/s Ring Amplifier-Based Pipelined ADC Using Averaging Correlated Level Shifting and Reference Swapping for Reducing Errors From Finite Opamp Gain and Capacitor Mismatch". In: *IEEE Journal of Solid-State Circuits* 54.5 (2019), pp. 1425–1435. DOI: 10.1109/JSSC.2019.2891650 (cit. on p. 88).

[34] T.-C. Hung, J.-C. Wang, and T.-H. Kuo. "A Calibration-Free 71.7dB SNDR 100MS/s 0.7mW Weighted-Averaging Correlated Level Shifting Pipelined SAR ADC with Speed-Enhancement Scheme". In: *2020 IEEE International Solid-State Circuits Conference*. 2020, pp. 256–258. DOI: 10.1109/ISSCC19947.2020.9063055 (cit. on p. 88).

[35] "IEEE Standard for Terminology and Test Methods for Analog-to-Digital Converters". In: *IEEE Std 1241-2010* (2011), pp. 1–139. DOI: 10.1109/IEEESTD.2011.5692956 (cit. on pp. 7, 8).

[36] *INL/DNL Measurements for High-Speed Analog-to-Digital Converters (ADCs)*. Maxim Integrated. 2001. URL: http://www.maximintegrated.com/an283 (cit. on p. 7).

[37] M. Kinyua and E. Soenen. "A 72.6 dB SNDR 14b 100 MSPS Ring Amplifier Based Pipelined SAR ADC with Dynamic Deadzone Control in 16 nm CMOS". In: *2020 IEEE Custom Integrated Circuits Conference* (2020), pp. 1–4. DOI: 10.1109/CICC48029.2020.9075921 (cit. on p. 88).

[38] J. Lagos et al. "A 1-GS/s, 12-b, Single-Channel Pipelined ADC With Dead-Zone-Degenerated Ring Amplifiers". In: *IEEE Journal of Solid-State Circuits* 54.3 (2019), pp. 646–658. DOI: 10.1109/JSSC.2018.2889680 (cit. on p. 88).

[39] J. Lagos et al. "A 10.0 ENOB, 6.2 fJ/conv.-step, 500 MS/s Ringamp-Based Pipelined-SAR ADC with Background Calibration and Dynamic Reference Regulation in 16nm CMOS". In: *2021 Symposium on VLSI Circuits*. 2021, pp. 1–2. DOI: 10.23919/VLSICircuits52068.2021.9492354 (cit. on p. 88).

[40] J. Lagos et al. "A 10.1-ENOB, 6.2-fJ/conv.-step, 500-MS/s, Ringamp-Based Pipelined-SAR ADC With Background Calibration and Dynamic Reference Regulation in 16-nm CMOS". In: *IEEE Journal of Solid-State Circuits* 57.4 (2022), pp. 1112–1124. DOI: 10.1109/JSSC.2021.3133829 (cit. on p. 88).

[41] J. Lagos et al. "A 1Gsps, 12-bit, Single-Channel Pipelined ADC with Dead-Zone-Degenerated Ring Amplifiers". In: *2018 IEEE Custom Integrated Circuits Conference*. 2018, pp. 1–4. DOI: 10.1109/CICC.2018.8357056 (cit. on pp. 39, 40, 43, 88).

[42] J. Lagos et al. "A Single-Channel, 600-MS/s, 12-b, Ringamp-Based Pipelined ADC in 28-nm CMOS". In: *IEEE Journal of Solid-State Circuits* 54.2 (2019), pp. 403–416. DOI: 10.1109/JSSC.2018.2879923 (cit. on p. 88).

[43] J. Lagos et al. "A Single-Channel, 600Msps, 12bit, Ringamp-Based Pipelined ADC in 28nm CMOS". In: *2017 Symposium on VLSI Circuits*. 2017, pp. C96–C97. DOI: 10.23919/VLSIC.2017.8008561 (cit. on pp. 38, 39, 43, 88).

[44] J. Lan et al. "A 2.5-GS/s Time-Interleaved SAR-Assisted Ringamp-Based Pipelined ADC with Digital Background Calibration". In: *2022 IEEE International Symposium on Circuits and Systems*. 2022, pp. 2655–2659. DOI: 10.1109/ISCAS48785.2022.9937296 (cit. on p. 88).

[45] J. Lan et al. "A Single-Channel 1.25-GS/s 11-bit Pipelined ADC with Robust Floating-Powered Ring Amplifier and First-Order Gain Error Calibration". In: *2022 IEEE 65th International Midwest Symposium on Circuits and Systems*. 2022, pp. 1–5. DOI: 10.1109/MWSCAS54063.2022.9859478 (cit. on p. 88).

[46] J. Lan et al. "Effective Gain Analysis and Statistic Based Calibration for Ring Amplifier With Robustness to PVT Variation". In: *IEEE Transactions on Circuits and Systems II: Express Briefs* 69.2 (2022), pp. 304–308. DOI: 10.1109/TCSII.2021.3093571 (cit. on p. 88).

[47] D. Lee et al. "Fat Tree Encoder Design for Ultra-High Speed Flash A/D Converters". In: *The 2002 45th Midwest Symposium on Circuits and Systems*. Vol. 2. 2002, p. II. DOI: 10.1109/MWSCAS.2002.1186804 (cit. on p. 10).

[48] S. Leuenberger et al. "A 74.33 dB SNDR 20 MSPS 2.74 mW Pipelined ADC using a Dynamic Deadzone Ring Amplifier". In: *2017 IEEE Custom Integrated Circuits Conference*. 2017, pp. 1–4. DOI: 10.1109/CICC.2017.7993697 (cit. on p. 88).

[49] Y. Lim and M. Flynn. "A 1 mW 71.5 dB SNDR 50 MS/s 13 bit Fully Differential Ring Amplifier Based SAR-Assisted Pipeline ADC". In: *IEEE Journal of Solid-State Circuits* 50.12 (2015), pp. 2901–2911. DOI: 10.1109/JSSC.2015.2463094 (cit. on p. 88).

[50] Y. Lim and M. Flynn. "A 100 MS/s, 10.5 Bit, 2.46 mW Comparator-Less Pipeline ADC Using Self-Biased Ring Amplifiers". In: *IEEE Journal of Solid-State Circuits* 50.10 (2015), pp. 2331–2341. DOI: 10.1109/JSSC.2015.2453332 (cit. on pp. 36, 37, 43, 88).

[51] Y. Lim and M. Flynn. "A 100MS/s 10.5b 2.46mW Comparator-Less Pipeline ADC Using Self-Biased Ring Amplifiers". In: *2014 IEEE International Solid-State Circuits Conference Digest of Technical Papers*. 2014, pp. 202–203. DOI: 10.1109/ISSCC.2014.6757400 (cit. on p. 88).

[52] Y. Lim and M. Flynn. "A 1mW 71.5dB SNDR 50MS/S 13b Fully Differential Ring-Amplifier-Based SAR-Assisted Pipeline ADC". In: *2015 IEEE International Solid-State Circuits Conference - Digest of Technical Papers*. 2015, pp. 1–3. DOI: 10.1109/ISSCC.2015.7063124 (cit. on p. 88).

[53] Y. Lim and M. Flynn. "A Calibration-Free 2.3 mW 73.2 dB SNDR 15b 100 MS/s Four-Stage Fully Differential Ring Amplifier Based SAR-Assisted Pipeline ADC". In: *2017 Symposium on VLSI Circuits*. 2017, pp. C98–C99. DOI: 10.23919/VLSIC.2017.8008562 (cit. on p. 88).

[54] X. Liu and J. McDonald. "Design of Single-Stage Folded-Cascode Gain Boost Amplifier for 14bit 12.5Ms/S Pipelined Analog-to-Digital-Converter". In: *2012 10th IEEE International Conference on Semiconductor Electronics (ICSE)*. 2012, pp. 622–626. DOI: 10.1109/SMElec.2012.6417222 (cit. on p. 29).

[55] J. Lourenço. *The NOVAthesis LaTeX Template User's Manual*. NOVA University Lisbon. 2021. URL: https://github.com/joaomlourenco/novathesis/raw/master/template.pdf (cit. on p. ii).

[56] K. Lundberg. "Analog-to-Digital Converter Testing". 2002. URL: https://www.mit.edu/~klund/papers/UNP_A2Dtest.pdf (cit. on pp. 5, 6).

[57] J. Marques. *Low Latency, Low Power, Precision Analog-to-Digital Converters*. Renesas. 2021. URL: https://www.renesas.com/us/en/document/whp/low-latency-low-power-precision-adcs (cit. on pp. 9, 11, 12, 14, 15, 17, 19, 21).

[58] J. Marques. *The Evolution of SAR ADCs for High Sampling Rate Applications*. Renesas. 2021. URL: https://www.renesas.com/us/en/document/whp/evolution-sar-adcs-high-sampling-rate-applications (cit. on pp. 15, 16).

[59] F. Matter, M. Ibrahim, and K. Shehata. "CMOS Single-Stage Fully Differential Telescopic Cascode OTA with Gain Boosted Technique for 14Bit 100MSps Pipelined ADC". In: *2015 World Congress on Information Technology and Computer Applications (WCITCA)*. 2015, pp. 1–6. DOI: 10.1109/WCITCA.2015.7367042 (cit. on p. 29).

[60] *Measuring Common-Mode and Power Supply Rejection Ratio*. Renesas. 2022. URL: https://www.renesas.com/us/en/document/apn/isl28006-measuring-common-mode-and-power-supply-rejection-ratio (cit. on p. 26).

[61] K. Megawer et al. "A Systematic Design Methodology for Class-AB-Style Ring Amplifiers". In: *IEEE Transactions on Circuits and Systems II: Express Briefs* 65.9 (2018), pp. 1169–1173. DOI: 10.1109/TCSII.2018.2815705 (cit. on p. 33).

[62] J. Muhlestein et al. "A Multi-Path Ring Amplifier with Dynamic Biasing". In: *2017 IEEE International Symposium on Circuits and Systems*. 2017, pp. 1–4. DOI: 10.1109/ISCAS.2017.8050677 (cit. on p. 88).

[63] B. Murmann. *ADC Performance Survey 1997-2023*. URL: https://github.com/bmurmann/ADC-survey (cit. on p. 41).

[64] *Operation of a SAR-ADC Based on Charge Redistribution*. Renesas. 2020. URL: https://www.renesas.com/us/en/document/apn/r14an0001-operation-sar-adc-based-charge-redistribution-rev100 (cit. on p. 16).

[65] J.-S. Park et al. "A 12.1 fJ/Conv.-Step 12b 140 MS/s 28-nm CMOS Pipelined SAR ADC Based on Energy-Efficient Switching and Shared Ring Amplifier". In: *IEEE Transactions on Circuits and Systems II: Express Briefs* 66.7 (2019), pp. 1119–1123. DOI: 10.1109/TCSII.2018.2880288 (cit. on p. 88).

[66] Y. Park et al. "An 11-b 100-MS/s Fully Dynamic Pipelined ADC Using a High-Linearity Dynamic Amplifier". In: *IEEE Journal of Solid-State Circuits* 55.9 (2020), pp. 2468–2477. DOI: 10.1109/JSSC.2020.2987684 (cit. on pp. 30, 31).

[67] C. Pearson. *High Speed, Analog to Digital Converter Basics*. Texas Instruments. 2009. URL: https://www.ti.com/lit/pdf/slaa510 (cit. on p. 8).

[68] M. Pelgrom. *Analog-to-Digital Conversion*. 4th. Springer, 2022. ISBN: 978-3-030-90808-9 (cit. on p. 18).

[69] B. Razavi. *Design of Analog CMOS Integrated Circuits*. 2nd. McGraw-Hill, 2017. ISBN: 978-0-07-252493-2 (cit. on p. 24).

[70] M. Rodrigues. "Técnicas Avançadas de Auto-Calibração de ADC's de Elevado Ritmo-de-Conversão utilizando Ruído Gaussiano". Faculdade de Ciências e Tecnologias, University Nova de Lisboa, 2007 (cit. on p. 19).

[71] F. Silveira, D. Flandre, and P. Jespers. "A g_m/I_D Based Methodology for the Design of CMOS Analog Circuits and its Application to the Synthesis of a Silicon-on-Insulator Micropower OTA". In: *IEEE Journal of Solid-State Circuits* 31.9 (1996), pp. 1314–1319. DOI: 10.1109/4.535416 (cit. on p. 56).

[72] *Understanding Flash ADCs*. Maxim Integrated. 2014. URL: http://www.maximintegrated.com/en/an810 (cit. on p. 9).

[73] *Understanding Pipelined ADCs*. Maxim Integrated. 2001. URL: http://www.maximintegrated.com/an1023 (cit. on pp. 17, 19).

[74] *Understanding SAR ADCs: Their Architecture and Comparison with Other ADCs*. Maxim Integrated. 2001. URL: http://www.maximintegrated.com/an1080 (cit. on pp. 15, 16).

[75] J.-C. Wang, T.-C. Hung, and T.-H. Kuo. "A Calibration-Free 14-b 0.7-mW 100-MS/s Pipelined-SAR ADC Using a Weighted-Averaging Correlated Level Shifting Technique". In: *IEEE Journal of Solid-State Circuits* 55.12 (2020), pp. 3271–3280. DOI: 10.1109/JSSC.2020.3015863 (cit. on p. 88).

[76] J.-C. Wang and T.-H. Kuo. "A 0.82mW 14b 130MS/S Pipelined-SAR ADC With a Distributed Averaging Correlated Level Shifting (DACLS) Ringamp and Bypass-Window Backend". In: *2022 IEEE International Solid-State Circuits Conference*. Vol. 65. 2022, pp. 162–164. DOI: 10.1109/ISSCC42614.2022.9731546 (cit. on p. 88).

[77] J.-C. Wang and T.-H. Kuo. "A 72-dB SNDR 130-MS/s 0.8-mW Pipelined-SAR ADC Using a Distributed Averaging Correlated Level Shifting Ring Amplifier". In: *IEEE Journal of Solid-State Circuits* 57 (2022), pp. 3794–3803. URL: https://api.semanticscholar.org/CorpusID:251852747 (cit. on p. 88).

[78] J. Yang and Z. Li. "Design and Error Analysis of an OTA for High-Speed Pipeline ADC". In: *2013 5th IEEE International Symposium on Microwave, Antenna, Propagation and EMC Technologies for Wireless Communications*. 2013, pp. 679–683. DOI: 10.1109/MAPE.2013.6689932 (cit. on pp. 28, 29).

[79] M. Zhan et al. "A 0.004mm2 200MS/S Pipelined SAR ADC with kT/C Noise Cancellation and Robust Ring-Amp". In: *2022 IEEE International Solid-State Circuits Conference*. Vol. 65. 2022, pp. 164–166. DOI: 10.1109/ISSCC42614.2022.9731599 (cit. on p. 88).

[80] Z. Zhaodong and L. Zhiqun. "A 7-bit 16-MS/s Low-Power CMOS Pipeline ADC". In: *2011 IEEE 13th International Conference on Communication Technology*. 2011, pp. 1082–1085. DOI: 10.1109/ICCT.2011.6158048 (cit. on p. 28).

A

Data Tables

This appendix presents the data tables associated with the diode-connected inverter used in the critically damped ring amplifier design, specifying the parameters for both NMOS and PMOS transistors across different channel lengths. Additionally, it includes a table that features the development of ring amplification technology in pipeline ADCs, extracted from [20], which is analyzed and visually represented to leverage the review on the state-of-the-art.

A.1 Diode-Connected Inverter Parameters for Different Lengths

Table A.1: NMOS diode-connected inverter parameters L = 60 nm.

k	$I_d(A)$	$V_{gs}(V)$	$V_{th}(V)$	$V_{ds}(V)$	$V_{dsat}(V)$	$C_{ds}(F)$	$C_{gd}(F)$	$C_{gs}(F)$	$g_{ds}(S)$	$g_m(S)$	$g_m/I_d(V)$	g_m/g_{ds}	$f_t(Hz)$	$I_d/W(A/m)$
1	1.36E-04	5.31E-01	4.17E-01	5.31E-01	1.35E-01	3.08E-19	2.90E-16	7.19E-16	1.31E-04	9.70E-04	7.13E+00	7.41E+00	2.15E+11	1.13E+02
2	1.99E-04	5.83E-01	4.14E-01	5.83E-01	1.61E-01	2.45E-19	2.89E-16	7.44E-16	1.59E-04	1.16E-03	5.85E+00	7.32E+00	2.49E+11	1.66E+02
2.625	2.27E-04	6.03E-01	4.12E-01	6.03E-01	1.72E-01	2.23E-19	2.88E-16	7.51E-16	1.69E-04	1.23E-03	5.44E+00	7.27E+00	2.61E+11	1.89E+02
3	2.40E-04	6.12E-01	4.12E-01	6.12E-01	1.77E-01	2.12E-19	2.88E-16	7.54E-16	1.74E-04	1.26E-03	5.25E+00	7.25E+00	2.66E+11	2.00E+02
3.625	2.60E-04	6.26E-01	4.11E-01	6.26E-01	1.84E-01	1.98E-19	2.88E-16	7.58E-16	1.81E-04	1.30E-03	5.01E+00	7.20E+00	2.74E+11	2.17E+02
4	2.71E-04	6.33E-01	4.11E-01	6.33E-01	1.87E-01	1.91E-19	2.88E-16	7.60E-16	1.84E-04	1.32E-03	4.89E+00	7.18E+00	2.77E+11	2.26E+02
5	2.95E-04	6.49E-01	4.10E-01	6.49E-01	1.95E-01	1.76E-19	2.87E-16	7.64E-16	1.92E-04	1.37E-03	4.64E+00	7.13E+00	2.85E+11	2.45E+02
6	3.14E-04	6.61E-01	4.09E-01	6.61E-01	2.02E-01	1.65E-19	2.87E-16	7.67E-16	1.97E-04	1.40E-03	4.45E+00	7.08E+00	2.90E+11	2.62E+02
7	3.31E-04	6.72E-01	4.08E-01	6.72E-01	2.07E-01	1.56E-19	2.87E-16	7.69E-16	2.02E-04	1.42E-03	4.30E+00	7.04E+00	2.95E+11	2.76E+02
8	3.45E-04	6.80E-01	4.08E-01	6.80E-01	2.12E-01	1.48E-19	2.87E-16	7.70E-16	2.06E-04	1.44E-03	4.18E+00	7.00E+00	2.98E+11	2.88E+02
9	3.58E-04	6.88E-01	4.07E-01	6.88E-01	2.16E-01	1.42E-19	2.87E-16	7.72E-16	2.09E-04	1.46E-03	4.08E+00	6.97E+00	3.01E+11	2.98E+02
10	3.69E-04	6.95E-01	4.07E-01	6.95E-01	2.19E-01	1.36E-19	2.87E-16	7.73E-16	2.12E-04	1.47E-03	3.99E+00	6.94E+00	3.04E+11	3.08E+02

Table A.2: PMOS diode-connected inverter parameters L = 60 nm.

k	$I_d(A)$	$V_{gs}(V)$	$V_{th}(V)$	$V_{ds}(V)$	$V_{dsat}(V)$	$C_{ds}(F)$	$C_{gd}(F)$	$C_{gs}(F)$	$g_{ds}(S)$	$g_m(S)$	$g_m/I_d(V)$	g_m/g_{ds}	$f_t(Hz)$	$I_d/W(A/m)$
1	-1.36E-04	-6.69E-01	-3.74E-01	-6.69E-01	-2.46E-01	1.61E-19	2.39E-16	7.34E-16	9.21E-05	6.43E-04	4.73E+00	6.98E+00	1.39E+11	1.13E+02
2	-1.99E-04	-6.17E-01	-3.77E-01	-6.17E-01	-2.11E-01	4.15E-19	4.73E-16	1.42E-15	1.54E-04	1.13E-03	5.69E+00	7.34E+00	1.27E+11	1.66E+02
2.625	-2.27E-04	-5.97E-01	-3.78E-01	-5.97E-01	-1.97E-01	5.97E-19	6.21E-16	1.84E-15	1.86E-04	1.39E-03	6.14E+00	7.48E+00	1.21E+11	1.89E+02
3	-2.40E-04	-5.88E-01	-3.79E-01	-5.88E-01	-1.90E-01	7.12E-19	7.09E-16	2.08E-15	2.03E-04	1.53E-03	6.37E+00	7.55E+00	1.17E+11	2.00E+02
3.625	-2.60E-04	-5.74E-01	-3.80E-01	-5.74E-01	-1.81E-01	9.12E-19	8.57E-16	2.49E-15	2.29E-04	1.75E-03	6.72E+00	7.64E+00	1.12E+11	2.17E+02
4	-2.71E-04	-5.67E-01	-3.81E-01	-5.67E-01	-1.77E-01	1.04E-18	9.46E-16	2.73E-15	2.43E-04	1.87E-03	6.90E+00	7.69E+00	1.09E+11	2.26E+02
5	-2.95E-04	-5.51E-01	-3.82E-01	-5.51E-01	-1.66E-01	1.38E-18	1.18E-15	3.36E-15	2.77E-04	2.16E-03	7.33E+00	7.78E+00	1.02E+11	2.45E+02
6	-3.14E-04	-5.39E-01	-3.83E-01	-5.39E-01	-1.58E-01	1.73E-18	1.42E-15	3.99E-15	3.08E-04	2.42E-03	7.70E+00	7.85E+00	9.65E+10	2.62E+02
7	-3.31E-04	-5.28E-01	-3.84E-01	-5.28E-01	-1.52E-01	2.10E-18	1.66E-15	4.60E-15	3.35E-04	2.65E-03	8.01E+00	7.91E+00	9.17E+10	2.76E+02
8	-3.45E-04	-5.20E-01	-3.85E-01	-5.20E-01	-1.46E-01	2.47E-18	1.90E-15	5.20E-15	3.60E-04	2.86E-03	8.28E+00	7.95E+00	8.75E+10	2.88E+02
9	-3.58E-04	-5.12E-01	-3.86E-01	-5.12E-01	-1.41E-01	2.86E-18	2.14E-15	5.80E-15	3.83E-04	3.05E-03	8.53E+00	7.98E+00	8.38E+10	2.98E+02
10	-3.69E-04	-5.05E-01	-3.86E-01	-5.05E-01	-1.37E-01	3.24E-18	2.38E-15	6.39E-15	4.04E-04	3.23E-03	8.75E+00	8.01E+00	8.05E+10	3.08E+02

Table A.3: NMOS diode-connected inverter parameters L = 90 nm.

k	$I_d(A)$	$V_{gs}(V)$	$V_{th}(V)$	$V_{ds}(V)$	$V_{dsat}(V)$	$C_{ds}(F)$	$C_{gd}(F)$	$C_{gs}(F)$	$g_{ds}(S)$	$g_m(S)$	$g_m/I_d(V)$	g_m/g_{ds}	$f_t(Hz)$	$I_d/W(A/m)$
1	1.08E-04	5.24E-01	3.41E-01	5.24E-01	1.49E-01	6.82E-19	3.03E-16	1.18E-15	5.05E-05	8.77E-04	8.10E+00	1.74E+01	1.18E+11	9.02E+01
2	1.62E-04	5.77E-01	3.41E-01	5.77E-01	1.80E-01	5.21E-19	3.02E-16	1.22E-15	6.31E-05	1.01E-03	6.26E+00	1.61E+01	1.32E+11	1.35E+02
2.625	1.86E-04	5.99E-01	3.40E-01	5.99E-01	1.93E-01	4.63E-19	3.02E-16	1.24E-15	6.81E-05	1.06E-03	5.69E+00	1.55E+01	1.36E+11	1.55E+02
3	1.98E-04	6.10E-01	3.40E-01	6.10E-01	1.99E-01	4.37E-19	3.02E-16	1.24E-15	7.05E-05	1.08E-03	5.44E+00	1.53E+01	1.38E+11	1.65E+02
3.625	2.15E-04	6.25E-01	3.40E-01	6.25E-01	2.07E-01	4.01E-19	3.02E-16	1.25E-15	7.38E-05	1.10E-03	5.12E+00	1.49E+01	1.40E+11	1.79E+02
4	2.24E-04	6.32E-01	3.40E-01	6.32E-01	2.11E-01	3.84E-19	3.02E-16	1.25E-15	7.55E-05	1.11E-03	4.96E+00	1.47E+01	1.41E+11	1.87E+02
5	2.45E-04	6.50E-01	3.40E-01	6.50E-01	2.21E-01	3.46E-19	3.02E-16	1.26E-15	7.93E-05	1.14E-03	4.64E+00	1.43E+01	1.44E+11	2.04E+02
6	2.62E-04	6.64E-01	3.40E-01	6.64E-01	2.29E-01	3.17E-19	3.02E-16	1.26E-15	8.23E-05	1.16E-03	4.40E+00	1.40E+01	1.46E+11	2.19E+02
7	2.77E-04	6.75E-01	3.40E-01	6.75E-01	2.35E-01	2.95E-19	3.02E-16	1.27E-15	8.48E-05	1.17E-03	4.22E+00	1.38E+01	1.47E+11	2.31E+02
8	2.90E-04	6.85E-01	3.40E-01	6.85E-01	2.41E-01	2.77E-19	3.02E-16	1.27E-15	8.69E-05	1.18E-03	4.08E+00	1.36E+01	1.48E+11	2.41E+02
9	3.01E-04	6.94E-01	3.40E-01	6.94E-01	2.46E-01	2.61E-19	3.02E-16	1.27E-15	8.87E-05	1.19E-03	3.96E+00	1.34E+01	1.49E+11	2.51E+02
10	3.10E-04	7.02E-01	3.40E-01	7.02E-01	2.50E-01	2.48E-19	3.02E-16	1.28E-15	9.02E-05	1.20E-03	3.86E+00	1.33E+01	1.50E+11	2.59E+02

Table A.4: PMOS diode-connected inverter parameters L = 90 nm.

k	$I_d(A)$	$V_{gs}(V)$	$V_{th}(V)$	$V_{ds}(V)$	$V_{dsat}(V)$	$C_{ds}(F)$	$C_{gd}(F)$	$C_{gs}(F)$	$g_{ds}(S)$	$g_m(S)$	$g_m/I_d(V)$	g_m/g_{ds}	$f_t(Hz)$	$I_d/W(A/m)$
1	-1.08E-04	-6.76E-01	-3.48E-01	-6.76E-01	-2.80E-01	3.03E-19	2.51E-16	1.17E-15	4.66E-05	4.90E-04	4.53E+00	1.05E+01	6.64E+10	9.02E+01
2	-1.62E-04	-6.23E-01	-3.49E-01	-6.23E-01	-2.40E-01	8.20E-19	4.98E-16	2.29E-15	7.86E-05	8.81E-04	5.44E+00	1.12E+01	6.13E+10	1.35E+02
2.625	-1.86E-04	-6.01E-01	-3.50E-01	-6.01E-01	-2.25E-01	1.21E-18	6.52E-16	2.97E-15	9.50E-05	1.09E-03	5.89E+00	1.15E+01	5.87E+10	1.55E+02
3	-1.98E-04	-5.90E-01	-3.51E-01	-5.90E-01	-2.17E-01	1.45E-18	7.45E-16	3.37E-15	1.04E-04	1.21E-03	6.13E+00	1.17E+01	5.73E+10	1.65E+02
3.625	-2.15E-04	-5.75E-01	-3.52E-01	-5.75E-01	-2.06E-01	1.89E-18	8.99E-16	4.03E-15	1.17E-04	1.40E-03	6.49E+00	1.19E+01	5.52E+10	1.79E+02
4	-2.24E-04	-5.68E-01	-3.52E-01	-5.68E-01	-2.01E-01	2.16E-18	9.92E-16	4.42E-15	1.25E-04	1.50E-03	6.69E+00	1.20E+01	5.40E+10	1.87E+02
5	-2.45E-04	-5.50E-01	-3.53E-01	-5.50E-01	-1.88E-01	2.93E-18	1.24E-15	5.44E-15	1.43E-04	1.76E-03	7.16E+00	1.23E+01	5.13E+10	2.04E+02
6	-2.62E-04	-5.36E-01	-3.53E-01	-5.36E-01	-1.79E-01	3.74E-18	1.49E-15	6.45E-15	1.59E-04	1.99E-03	7.57E+00	1.25E+01	4.90E+10	2.19E+02
7	-2.77E-04	-5.25E-01	-3.54E-01	-5.25E-01	-1.71E-01	4.58E-18	1.73E-15	7.44E-15	1.74E-04	2.20E-03	7.93E+00	1.26E+01	4.70E+10	2.31E+02
8	-2.90E-04	-5.15E-01	-3.54E-01	-5.15E-01	-1.64E-01	5.44E-18	1.98E-15	8.41E-15	1.87E-04	2.39E-03	8.25E+00	1.28E+01	4.52E+10	2.41E+02
9	-3.01E-04	-5.06E-01	-3.54E-01	-5.06E-01	-1.58E-01	6.33E-18	2.23E-15	9.37E-15	2.00E-04	2.57E-03	8.54E+00	1.29E+01	4.36E+10	2.51E+02
10	-3.10E-04	-4.98E-01	-3.55E-01	-4.98E-01	-1.53E-01	7.24E-18	2.48E-15	1.03E-14	2.11E-04	2.73E-03	8.81E+00	1.30E+01	4.22E+10	2.59E+02

Table A.5: NMOS diode-connected inverter parameters L = 120 nm.

k	$I_d(A)$	$V_{gs}(V)$	$V_{th}(V)$	$V_{ds}(V)$	$V_{dsat}(V)$	$C_{ds}(F)$	$C_{gd}(F)$	$C_{gs}(F)$	$g_{ds}(S)$	$g_m(S)$	$g_m/I_d(V)$	g_m/g_{ds}	$f_t(Hz)$	$I_d/W(A/m)$
1	9.89E-05	5.04E-01	2.83E-01	5.04E-01	1.67E-01	1.04E-18	3.10E-16	1.68E-15	3.65E-05	7.71E-04	7.80E+00	2.11E+01	7.31E+10	8.24E+01
2	1.49E-04	5.62E-01	2.83E-01	5.62E-01	2.03E-01	7.34E-19	3.10E-16	1.73E-15	4.70E-05	8.82E-04	5.90E+00	1.88E+01	8.11E+10	1.24E+02
2.625	1.72E-04	5.86E-01	2.83E-01	5.86E-01	2.18E-01	6.30E-19	3.10E-16	1.75E-15	5.12E-05	9.17E-04	5.34E+00	1.79E+01	8.35E+10	1.43E+02
3	1.83E-04	5.97E-01	2.83E-01	5.97E-01	2.25E-01	5.83E-19	3.11E-16	1.76E-15	5.32E-05	9.33E-04	5.09E+00	1.75E+01	8.43E+10	1.53E+02
3.625	2.00E-04	6.14E-01	2.83E-01	6.14E-01	2.34E-01	5.20E-19	3.11E-16	1.76E-15	5.61E-05	9.53E-04	4.77E+00	1.70E+01	8.60E+10	1.66E+02
4	2.08E-04	6.22E-01	2.83E-01	6.22E-01	2.40E-01	4.89E-19	3.11E-16	1.77E-15	5.76E-05	9.63E-04	4.62E+00	1.67E+01	8.66E+10	1.74E+02
5	2.28E-04	6.42E-01	2.83E-01	6.42E-01	2.51E-01	4.24E-19	3.11E-16	1.78E-15	6.08E-05	9.84E-04	4.31E+00	1.62E+01	8.80E+10	1.90E+02
6	2.45E-04	6.57E-01	2.83E-01	6.57E-01	2.60E-01	3.76E-19	3.11E-16	1.79E-15	6.34E-05	1.00E-03	4.09E+00	1.58E+01	8.89E+10	2.04E+02
7	2.59E-04	6.70E-01	2.83E-01	6.70E-01	2.68E-01	3.39E-19	3.11E-16	1.79E-15	6.55E-05	1.01E-03	3.91E+00	1.54E+01	9.00E+10	2.15E+02
8	2.70E-04	6.81E-01	2.83E-01	6.81E-01	2.74E-01	3.08E-19	3.11E-16	1.79E-15	6.73E-05	1.02E-03	3.78E+00	1.52E+01	9.08E+10	2.25E+02
9	2.81E-04	6.91E-01	2.83E-01	6.91E-01	2.80E-01	2.84E-19	3.11E-16	1.80E-15	6.89E-05	1.03E-03	3.67E+00	1.50E+01	9.11E+10	2.34E+02
10	2.90E-04	6.99E-01	2.83E-01	6.99E-01	2.84E-01	2.63E-19	3.11E-16	1.80E-15	7.02E-05	1.04E-03	3.57E+00	1.48E+01	9.17E+10	2.42E+02

Table A.6: PMOS diode-connected inverter parameters L = 120 nm.

k	$I_d(A)$	$V_{gs}(V)$	$V_{th}(V)$	$V_{ds}(V)$	$V_{dsat}(V)$	$C_{ds}(F)$	$C_{gd}(F)$	$C_{gs}(F)$	$g_{ds}(S)$	$g_m(S)$	$g_m/I_d(V)$	g_m/g_{ds}	$f_t(Hz)$	$I_d/W(A/m)$
1	-9.89E-05	-6.96E-01	-3.25E-01	-6.96E-01	-3.17E-01	3.75E-19	2.65E-16	1.63E-15	3.39E-05	4.05E-04	4.10E+00	1.20E+01	3.96E+10	8.24E+01
2	-1.49E-04	-6.38E-01	-3.26E-01	-6.38E-01	-2.73E-01	1.09E-18	5.25E-16	3.19E-15	5.66E-05	7.35E-04	4.92E+00	1.30E+01	3.67E+10	1.24E+02
2.625	-1.72E-04	-6.14E-01	-3.26E-01	-6.14E-01	-2.55E-01	1.65E-18	6.88E-16	4.15E-15	6.82E-05	9.17E-04	5.34E+00	1.35E+01	3.52E+10	1.43E+02
3	-1.83E-04	-6.03E-01	-3.27E-01	-6.03E-01	-2.47E-01	2.01E-18	7.85E-16	4.71E-15	7.44E-05	1.02E-03	5.56E+00	1.37E+01	3.44E+10	1.53E+02
3.625	-2.00E-04	-5.86E-01	-3.27E-01	-5.86E-01	-2.34E-01	2.66E-18	9.48E-16	5.64E-15	8.39E-05	1.18E-03	5.90E+00	1.41E+01	3.33E+10	1.66E+02
4	-2.08E-04	-5.78E-01	-3.27E-01	-5.78E-01	-2.28E-01	3.07E-18	1.05E-15	6.19E-15	8.91E-05	1.27E-03	6.09E+00	1.43E+01	3.27E+10	1.74E+02
5	-2.28E-04	-5.58E-01	-3.28E-01	-5.58E-01	-2.14E-01	4.24E-18	1.31E-15	7.65E-15	1.02E-04	1.50E-03	6.55E+00	1.47E+01	3.11E+10	1.90E+02
6	-2.45E-04	-5.43E-01	-3.28E-01	-5.43E-01	-2.03E-01	5.49E-18	1.56E-15	9.07E-15	1.13E-04	1.70E-03	6.96E+00	1.51E+01	2.99E+10	2.04E+02
7	-2.59E-04	-5.30E-01	-3.28E-01	-5.30E-01	-1.93E-01	6.81E-18	1.82E-15	1.05E-14	1.23E-04	1.89E-03	7.32E+00	1.54E+01	2.87E+10	2.15E+02
8	-2.70E-04	-5.19E-01	-3.28E-01	-5.19E-01	-1.86E-01	8.19E-18	2.08E-15	1.19E-14	1.32E-04	2.07E-03	7.65E+00	1.57E+01	2.77E+10	2.25E+02
9	-2.81E-04	-5.09E-01	-3.29E-01	-5.09E-01	-1.79E-01	9.61E-18	2.34E-15	1.32E-14	1.40E-04	2.23E-03	7.95E+00	1.59E+01	2.69E+10	2.34E+02
10	-2.90E-04	-5.01E-01	-3.29E-01	-5.01E-01	-1.73E-01	1.11E-17	2.60E-15	1.46E-14	1.48E-04	2.39E-03	8.22E+00	1.61E+01	2.60E+10	2.42E+02

Table A.7: NMOS diode-connected inverter parameters L = 240 nm.

k	$I_d(A)$	$V_{gs}(V)$	$V_{th}(V)$	$V_{ds}(V)$	$V_{dsat}(V)$	$C_{ds}(F)$	$C_{gd}(F)$	$C_{gs}(F)$	$g_{ds}(S)$	$g_m(S)$	$g_m/I_d(V)$	g_m/g_{ds}	$f_t(Hz)$	$I_d/W(A/m)$
1	6.66E-05	4.76E-01	1.88E-01	4.76E-01	2.00E-01	1.83E-18	3.31E-16	3.70E-15	2.14E-05	4.92E-04	7.40E+00	2.30E+01	2.12E+10	5.55E+01
2	1.02E-04	5.40E-01	1.88E-01	5.40E-01	2.47E-01	9.74E-19	3.32E-16	3.80E-15	2.89E-05	5.72E-04	5.59E+00	1.98E+01	2.39E+10	8.53E+01
2.625	1.19E-04	5.67E-01	1.89E-01	5.67E-01	2.67E-01	6.99E-19	3.33E-16	3.83E-15	3.20E-05	5.98E-04	5.04E+00	1.87E+01	2.48E+10	9.89E+01
3	1.27E-04	5.80E-01	1.89E-01	5.80E-01	2.76E-01	5.77E-19	3.33E-16	3.85E-15	3.35E-05	6.11E-04	4.80E+00	1.82E+01	2.52E+10	1.06E+02
3.625	1.39E-04	5.99E-01	1.89E-01	5.99E-01	2.90E-01	4.18E-19	3.33E-16	3.87E-15	3.57E-05	6.27E-04	4.50E+00	1.76E+01	2.58E+10	1.16E+02
4	1.46E-04	6.09E-01	1.89E-01	6.09E-01	2.97E-01	3.42E-19	3.33E-16	3.87E-15	3.68E-05	6.35E-04	4.35E+00	1.73E+01	2.61E+10	1.22E+02
5	1.61E-04	6.31E-01	1.89E-01	6.31E-01	3.12E-01	1.85E-19	3.34E-16	3.89E-15	3.93E-05	6.53E-04	4.06E+00	1.66E+01	2.67E+10	1.34E+02
6	1.74E-04	6.49E-01	1.89E-01	6.49E-01	3.25E-01	7.24E-20	3.34E-16	3.91E-15	4.13E-05	6.67E-04	3.84E+00	1.61E+01	2.71E+10	1.45E+02
7	1.84E-04	6.64E-01	1.89E-01	6.64E-01	3.36E-01	1.31E-20	3.34E-16	3.92E-15	4.30E-05	6.78E-04	3.68E+00	1.58E+01	2.75E+10	1.54E+02
8	1.93E-04	6.76E-01	1.89E-01	6.76E-01	3.45E-01	8.05E-20	3.34E-16	3.93E-15	4.44E-05	6.87E-04	3.55E+00	1.55E+01	2.78E+10	1.61E+02
9	2.02E-04	6.87E-01	1.89E-01	6.87E-01	3.52E-01	1.35E-19	3.34E-16	3.93E-15	4.56E-05	6.94E-04	3.45E+00	1.52E+01	2.81E+10	1.68E+02
10	2.09E-04	6.97E-01	1.90E-01	6.97E-01	3.59E-01	1.80E-19	3.34E-16	3.94E-15	4.66E-05	7.01E-04	3.36E+00	1.50E+01	2.83E+10	1.74E+02

Table A.8: PMOS diode-connected inverter parameters L = 240 nm.

k	$I_d(A)$	$V_{gs}(V)$	$V_{th}(V)$	$V_{ds}(V)$	$V_{dsat}(V)$	$C_{ds}(F)$	$C_{gd}(F)$	$C_{gs}(F)$	$g_{ds}(S)$	$g_m(S)$	$g_m/I_d(V)$	g_m/g_{ds}	$f_t(Hz)$	$I_d/W(A/m)$
1	-6.65E-05	-7.24E-01	-2.76E-01	-7.24E-01	-3.82E-01	5.00E-19	2.93E-16	3.50E-15	1.85E-05	2.37E-04	3.56E+00	1.28E+01	1.08E+10	5.55E+01
2	-1.02E-04	-6.60E-01	-2.75E-01	-6.60E-01	-3.31E-01	1.82E-18	5.83E-16	6.88E-15	3.08E-05	4.29E-04	4.20E+00	1.39E+01	9.93E+09	8.53E+01
2.625	-1.19E-04	-6.33E-01	-2.75E-01	-6.33E-01	-3.10E-01	2.94E-18	7.64E-16	8.96E-15	3.71E-05	5.37E-04	4.53E+00	1.45E+01	9.54E+09	9.89E+01
3	-1.27E-04	-6.20E-01	-2.75E-01	-6.20E-01	-3.00E-01	3.70E-18	8.72E-16	1.02E-14	4.05E-05	5.98E-04	4.71E+00	1.48E+01	9.34E+09	1.06E+02
3.625	-1.39E-04	-6.01E-01	-2.75E-01	-6.01E-01	-2.86E-01	5.11E-18	1.05E-15	1.22E-14	4.57E-05	6.95E-04	4.99E+00	1.52E+01	9.04E+09	1.16E+02
4	-1.46E-04	-5.91E-01	-2.75E-01	-5.91E-01	-2.78E-01	6.02E-18	1.16E-15	1.35E-14	4.86E-05	7.51E-04	5.14E+00	1.55E+01	8.85E+09	1.22E+02
5	-1.61E-04	-5.69E-01	-2.75E-01	-5.69E-01	-2.61E-01	8.68E-18	1.45E-15	1.67E-14	5.55E-05	8.91E-04	5.53E+00	1.60E+01	8.49E+09	1.34E+02
6	-1.74E-04	-5.51E-01	-2.75E-01	-5.51E-01	-2.47E-01	1.16E-17	1.73E-15	1.98E-14	6.16E-05	1.02E-03	5.88E+00	1.66E+01	8.20E+09	1.45E+02
7	-1.84E-04	-5.36E-01	-2.75E-01	-5.36E-01	-2.36E-01	1.48E-17	2.02E-15	2.30E-14	6.71E-05	1.14E-03	6.20E+00	1.70E+01	7.90E+09	1.54E+02
8	-1.93E-04	-5.24E-01	-2.75E-01	-5.24E-01	-2.27E-01	1.82E-17	2.31E-15	2.60E-14	7.20E-05	1.26E-03	6.49E+00	1.74E+01	7.69E+09	1.61E+02
9	-2.02E-04	-5.13E-01	-2.75E-01	-5.13E-01	-2.18E-01	2.18E-17	2.59E-15	2.91E-14	7.65E-05	1.36E-03	6.77E+00	1.78E+01	7.46E+09	1.68E+02
10	-2.09E-04	-5.03E-01	-2.75E-01	-5.03E-01	-2.11E-01	2.55E-17	2.87E-15	3.21E-14	8.07E-05	1.47E-03	7.02E+00	1.82E+01	7.27E+09	1.74E+02

Table A.9: NMOS diode-connected inverter parameters L = 300 nm.

k	$I_d(A)$	$V_{gs}(V)$	$V_{th}(V)$	$V_{ds}(V)$	$V_{dsat}(V)$	$C_{ds}(F)$	$C_{gd}(F)$	$C_{gs}(F)$	$g_{ds}(S)$	$g_m(S)$	$g_m/I_d(V)$	g_m/g_{ds}	$f_t(Hz)$	$I_d/W(A/m)$
1	5.63E-05	4.71E-01	1.67E-01	4.71E-01	2.08E-01	1.96E-18	3.41E-16	4.72E-15	1.80E-05	4.14E-04	7.35E+00	2.31E+01	1.40E+10	4.69E+01
2	8.69E-05	5.36E-01	1.68E-01	5.36E-01	2.57E-01	8.66E-19	3.43E-16	4.84E-15	2.44E-05	4.84E-04	5.57E+00	1.98E+01	1.59E+10	7.24E+01
2.625	1.01E-04	5.63E-01	1.68E-01	5.63E-01	2.78E-01	5.19E-19	3.43E-16	4.88E-15	2.72E-05	5.08E-04	5.03E+00	1.87E+01	1.66E+10	8.42E+01
3	1.08E-04	5.77E-01	1.68E-01	5.77E-01	2.88E-01	3.66E-19	3.43E-16	4.90E-15	2.85E-05	5.19E-04	4.79E+00	1.82E+01	1.69E+10	9.03E+01
3.625	1.19E-04	5.96E-01	1.68E-01	5.96E-01	3.02E-01	1.69E-19	3.44E-16	4.92E-15	3.04E-05	5.35E-04	4.49E+00	1.76E+01	1.73E+10	9.93E+01
4	1.25E-04	6.06E-01	1.68E-01	6.06E-01	3.10E-01	7.47E-20	3.44E-16	4.93E-15	3.14E-05	5.42E-04	4.34E+00	1.73E+01	1.75E+10	1.04E+02
5	1.38E-04	6.28E-01	1.68E-01	6.28E-01	3.27E-01	1.18E-19	3.44E-16	4.96E-15	3.37E-05	5.59E-04	4.05E+00	1.66E+01	1.79E+10	1.15E+02
6	1.49E-04	6.47E-01	1.69E-01	6.47E-01	3.41E-01	2.56E-19	3.44E-16	4.97E-15	3.54E-05	5.72E-04	3.84E+00	1.61E+01	1.83E+10	1.24E+02
7	1.58E-04	6.62E-01	1.69E-01	6.62E-01	3.52E-01	3.59E-19	3.45E-16	4.99E-15	3.69E-05	5.82E-04	3.68E+00	1.58E+01	1.86E+10	1.32E+02
8	1.66E-04	6.75E-01	1.69E-01	6.75E-01	3.62E-01	4.41E-19	3.45E-16	5.00E-15	3.82E-05	5.91E-04	3.55E+00	1.55E+01	1.88E+10	1.39E+02
9	1.74E-04	6.86E-01	1.69E-01	6.86E-01	3.71E-01	5.06E-19	3.45E-16	5.01E-15	3.93E-05	5.98E-04	3.44E+00	1.52E+01	1.90E+10	1.45E+02
10	1.80E-04	6.96E-01	1.69E-01	6.96E-01	3.78E-01	5.60E-19	3.45E-16	5.01E-15	4.03E-05	6.04E-04	3.36E+00	1.50E+01	1.92E+10	1.50E+02

Table A.10: PMOS diode-connected inverter parameters L = 300 nm.

k	$I_d(A)$	$V_{gs}(V)$	$V_{th}(V)$	$V_{ds}(V)$	$V_{dsat}(V)$	$C_{ds}(F)$	$C_{gd}(F)$	$C_{gs}(F)$	$g_{ds}(S)$	$g_m(S)$	$g_m/I_d(V)$	g_m/g_{ds}	$f_t(Hz)$	$I_d/W(A/m)$
1	-5.63E-05	-7.29E-01	-2.64E-01	-7.29E-01	-3.96E-01	5.12E-19	3.04E-16	4.44E-15	1.48E-05	1.96E-04	3.48E+00	1.33E+01	7.02E+09	4.69E+01
2	-8.69E-05	-6.64E-01	-2.63E-01	-6.64E-01	-3.44E-01	2.07E-18	6.04E-16	8.75E-15	2.47E-05	3.55E-04	4.08E+00	1.44E+01	6.45E+09	7.24E+01
2.625	-1.01E-04	-6.37E-01	-2.63E-01	-6.37E-01	-3.22E-01	3.43E-18	7.91E-16	1.14E-14	2.97E-05	4.44E-04	4.39E+00	1.49E+01	6.20E+09	8.42E+01
3	-1.08E-04	-6.23E-01	-2.63E-01	-6.23E-01	-3.12E-01	4.37E-18	9.03E-16	1.30E-14	3.25E-05	4.94E-04	4.56E+00	1.52E+01	6.05E+09	9.03E+01
3.625	-1.19E-04	-6.04E-01	-2.62E-01	-6.04E-01	-2.97E-01	6.11E-18	1.09E-15	1.56E-14	3.67E-05	5.75E-04	4.82E+00	1.56E+01	5.87E+09	9.93E+01
4	-1.25E-04	-5.94E-01	-2.62E-01	-5.94E-01	-2.89E-01	7.25E-18	1.20E-15	1.71E-14	3.91E-05	6.21E-04	4.97E+00	1.59E+01	5.78E+09	1.04E+02
5	-1.38E-04	-5.72E-01	-2.62E-01	-5.72E-01	-2.71E-01	1.06E-17	1.50E-15	2.12E-14	4.47E-05	7.37E-04	5.34E+00	1.65E+01	5.53E+09	1.15E+02
6	-1.49E-04	-5.53E-01	-2.62E-01	-5.53E-01	-2.57E-01	1.44E-17	1.80E-15	2.53E-14	4.97E-05	8.45E-04	5.67E+00	1.70E+01	5.31E+09	1.24E+02
7	-1.58E-04	-5.38E-01	-2.62E-01	-5.38E-01	-2.46E-01	1.84E-17	2.09E-15	2.93E-14	5.42E-05	9.46E-04	5.97E+00	1.75E+01	5.14E+09	1.32E+02
8	-1.66E-04	-5.25E-01	-2.62E-01	-5.25E-01	-2.36E-01	2.28E-17	2.39E-15	3.32E-14	5.82E-05	1.04E-03	6.25E+00	1.79E+01	4.99E+09	1.39E+02
9	-1.74E-04	-5.14E-01	-2.62E-01	-5.14E-01	-2.27E-01	2.75E-17	2.68E-15	3.71E-14	6.19E-05	1.13E-03	6.52E+00	1.83E+01	4.85E+09	1.45E+02
10	-1.80E-04	-5.04E-01	-2.62E-01	-5.04E-01	-2.20E-01	3.23E-17	2.97E-15	4.10E-14	6.54E-05	1.22E-03	6.76E+00	1.86E+01	4.73E+09	1.50E+02

Table A.11: NMOS diode-connected inverter parameters L = 600 nm.

k	$I_d(A)$	$V_{gs}(V)$	$V_{th}(V)$	$V_{ds}(V)$	$V_{dsat}(V)$	$C_{ds}(F)$	$C_{gd}(F)$	$C_{gs}(F)$	$g_{ds}(S)$	$g_m(S)$	$g_m/I_d(V)$	g_m/g_{ds}	$f_t(Hz)$	$I_d/W(A/m)$
1	3.19E-05	4.59E-01	1.19E-01	4.59E-01	2.24E-01	1.18E-18	3.90E-16	9.84E-15	1.01E-05	2.29E-04	7.17E+00	2.27E+01	3.70E+09	2.66E+01
2	4.96E-05	5.27E-01	1.19E-01	5.27E-01	2.79E-01	8.82E-19	3.92E-16	1.01E-14	1.39E-05	2.72E-04	5.49E+00	1.96E+01	4.29E+09	4.13E+01
2.625	5.80E-05	5.55E-01	1.19E-01	5.55E-01	3.03E-01	1.50E-18	3.93E-16	1.01E-14	1.56E-05	2.88E-04	4.97E+00	1.85E+01	4.53E+09	4.83E+01
3	6.23E-05	5.69E-01	1.20E-01	5.69E-01	3.14E-01	1.77E-18	3.93E-16	1.02E-14	1.64E-05	2.96E-04	4.74E+00	1.81E+01	4.61E+09	5.19E+01
3.625	6.87E-05	5.89E-01	1.20E-01	5.89E-01	3.31E-01	2.10E-18	3.94E-16	1.02E-14	1.76E-05	3.06E-04	4.45E+00	1.74E+01	4.77E+09	5.73E+01
4	7.22E-05	6.00E-01	1.20E-01	6.00E-01	3.40E-01	2.26E-18	3.94E-16	1.02E-14	1.82E-05	3.11E-04	4.31E+00	1.71E+01	4.86E+09	6.02E+01
5	8.02E-05	6.23E-01	1.20E-01	6.23E-01	3.60E-01	2.58E-18	3.94E-16	1.03E-14	1.96E-05	3.23E-04	4.03E+00	1.65E+01	4.99E+09	6.68E+01
6	8.69E-05	6.43E-01	1.20E-01	6.43E-01	3.76E-01	2.79E-18	3.95E-16	1.03E-14	2.07E-05	3.32E-04	3.82E+00	1.61E+01	5.14E+09	7.24E+01
7	9.27E-05	6.59E-01	1.20E-01	6.59E-01	3.90E-01	2.95E-18	3.95E-16	1.03E-14	2.16E-05	3.40E-04	3.67E+00	1.57E+01	5.25E+09	7.73E+01
8	9.78E-05	6.73E-01	1.20E-01	6.73E-01	4.02E-01	3.08E-18	3.96E-16	1.04E-14	2.25E-05	3.46E-04	3.54E+00	1.54E+01	5.30E+09	8.15E+01
9	1.02E-04	6.85E-01	1.20E-01	6.85E-01	4.13E-01	3.17E-18	3.96E-16	1.04E-14	2.32E-05	3.52E-04	3.44E+00	1.52E+01	5.38E+09	8.52E+01
10	1.06E-04	6.95E-01	1.21E-01	6.95E-01	4.22E-01	3.25E-18	3.96E-16	1.04E-14	2.38E-05	3.57E-04	3.36E+00	1.50E+01	5.46E+09	8.85E+01

Table A.12: PMOS diode-connected inverter parameters L = 600 nm.

k	$I_d(A)$	$V_{gs}(V)$	$V_{th}(V)$	$V_{ds}(V)$	$V_{dsat}(V)$	$C_{ds}(F)$	$C_{gd}(F)$	$C_{gs}(F)$	$g_{ds}(S)$	$g_m(S)$	$g_m/I_d(V)$	g_m/g_{ds}	$f_t(Hz)$	$I_d/W(A/m)$
1	-3.19E-05	-7.41E-01	-2.34E-01	-7.41E-01	-4.29E-01	2.85E-19	3.56E-16	9.15E-15	6.59E-06	1.06E-04	3.33E+00	1.62E+01	1.85E+09	2.66E+01
2	-4.96E-05	-6.73E-01	-2.32E-01	-6.73E-01	-3.73E-01	2.71E-18	7.08E-16	1.81E-14	1.11E-05	1.91E-04	3.85E+00	1.72E+01	1.68E+09	4.13E+01
2.625	-5.80E-05	-6.45E-01	-2.31E-01	-6.45E-01	-3.50E-01	5.07E-18	9.27E-16	2.36E-14	1.35E-05	2.39E-04	4.12E+00	1.77E+01	1.61E+09	4.83E+01
3	-6.23E-05	-6.31E-01	-2.31E-01	-6.31E-01	-3.38E-01	6.75E-18	1.06E-15	2.69E-14	1.48E-05	2.66E-04	4.27E+00	1.80E+01	1.57E+09	5.19E+01
3.625	-6.87E-05	-6.11E-01	-2.31E-01	-6.11E-01	-3.22E-01	9.98E-18	1.27E-15	3.23E-14	1.68E-05	3.09E-04	4.50E+00	1.84E+01	1.52E+09	5.73E+01
4	-7.22E-05	-6.00E-01	-2.31E-01	-6.00E-01	-3.14E-01	1.21E-17	1.41E-15	3.55E-14	1.79E-05	3.34E-04	4.63E+00	1.87E+01	1.50E+09	6.02E+01
5	-8.02E-05	-5.77E-01	-2.30E-01	-5.77E-01	-2.95E-01	1.86E-17	1.75E-15	4.41E-14	2.06E-05	3.97E-04	4.95E+00	1.92E+01	1.43E+09	6.68E+01
6	-8.69E-05	-5.57E-01	-2.30E-01	-5.57E-01	-2.80E-01	2.61E-17	2.10E-15	5.25E-14	2.31E-05	4.55E-04	5.24E+00	1.97E+01	1.38E+09	7.24E+01
7	-9.27E-05	-5.41E-01	-2.30E-01	-5.41E-01	-2.67E-01	3.44E-17	2.44E-15	6.09E-14	2.53E-05	5.10E-04	5.50E+00	2.02E+01	1.33E+09	7.73E+01
8	-9.78E-05	-5.27E-01	-2.30E-01	-5.27E-01	-2.57E-01	4.35E-17	2.78E-15	6.92E-14	2.73E-05	5.62E-04	5.75E+00	2.06E+01	1.29E+09	8.15E+01
9	-1.02E-04	-5.15E-01	-2.29E-01	-5.15E-01	-2.47E-01	5.32E-17	3.13E-15	7.74E-14	2.92E-05	6.12E-04	5.98E+00	2.10E+01	1.26E+09	8.52E+01
10	-1.06E-04	-5.05E-01	-2.29E-01	-5.05E-01	-2.39E-01	6.35E-17	3.47E-15	8.55E-14	3.09E-05	6.59E-04	6.20E+00	2.13E+01	1.23E+09	8.85E+01

A.2 Overview of Ring Amplifiers in Pipeline ADCs

Table A.13: Characteristics of pipeline ADCs and their ring amplifiers.

Reference	[32]	[30]	[30]	[28]	[22]	[51]	[52]	[49]	[50]	[7]	[48]	[8]	[62]	[53]	[43]	[41]	[10]	[11]	[9]	[16]	[25]	[27]	[15]
CMOS(nm)	1.80E-07	1.80E-07	1.80E-07	1.80E-07	1.80E-07	6.50E-08	6.50E-08	6.50E-08	6.50E-08	9.00E-08	1.86E-07	6.50E-08	1.80E-07	4.00E-08	2.80E-08	2.80E-08	2.80E-08	6.50E-08	2.80E-08	1.80E-04	1.60E-08	1.60E-08	1.80E-04
$A(m^2)$	1.98E-06	5.00E-07	1.98E-06	5.00E-07	1.98E-06	9.70E-08	5.40E-08	5.40E-08	9.70E-08	-	1.21E-06	2.15E-07	3.06E-06	6.80E-08	6.20E-07	5.40E-07	1.75E-07	2.15E-07	1.20E-07	1.82E-06	1.94E-07	3.70E-08	1.82E-06
$V_{DD}(V)$	1.3	1.3	1.3	1.3	1.2	1.2	1.2	1.2	1.2	1.0	1.2	1.2	1.7	1.1	2.2	1.6	1.2	1.2	1.0	1.8	0.9	0.9	1.8
$V_{FS}(V_{pp})$	2.5	2.2	2.5	2.2	2.4	2.0	2.4	2.4	2.0	-	2.4	1.6	-	2.2	1.6	1.6	1.6	-	1.4	6.6	1.5	1.5	6.6
N(b)	15.0	10.5	15.0	10.5	15.0	10.5	13.0	13.0	10.5	10.0	15.0	11.0	13.0	15.0	12.0	12.0	12.0	11.0	12.0	16.0	13.0	11.0	16.0
$f_1(b)$	7.0	10.0	7.0	10.0	7.0	10.0	1.0	1.0	10.0	8.0	2.0	1.0	6.0	1.0	10.0	10.0	1.0	1.0	1.0	1.0	9.0	7.0	1.0
$n_1(b)$	3.0	1.5	3.0	1.5	3.0	1.5	6.0	6.0	1.5	1.5	5.0	2.5	3.0	7.0	1.5	1.5	2.5	2.5	2.5	9.0	1.5	1.5	9.0
$f_2(b)$	-	-	-	-	-	-	1.0	1.0	-	1.0	3.0	1.0	-	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
$n_2(b)$	-	-	-	-	-	-	8.0	8.0	-	2.0	3.0	9.0	-	9.0	2.0	2.0	2.0	9.0	2.0	13.0	4.5	4.5	13.0
$f_3(b)$	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1.0	-	1.0	-	-	-	-
$n_3(b)$	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	9.0	-	9.0	-	-	-	-
R(b)	1.0	0.5	1.0	0.5	1.0	0.5	1.0	1.0	0.5	0.5	1.0	0.5	1.0	1.0	0.5	0.5	0.5	0.5	0.5	3.0	0.5	0.5	3.0
$fs(sps)$	2.00E+07	3.00E+07	2.00E+07	3.00E+07	2.00E+07	1.00E+08	5.00E+07	5.00E+07	1.00E+08	7.00E+07	2.00E+07	2.00E+08	1.00E+07	1.00E+08	6.00E+08	1.00E+09	8.00E+08	2.00E+08	6.25E+08	1.50E+07	3.20E+09	6.00E+08	1.50E+07
ENOB(b)	11.8	9.2	11.8	9.2	11.8	9.1	11.5	11.5	9.1	8.4	11.7	6.8	11.4	-	9.1	9.1	9.8	6.8	9.2	14.7	10.0	9.7	14.7
SNDR(dB)	73.0	57.0	73.0	57.0	73.0	56.3	70.9	70.9	56.6	52.1	72.3	42.5	70.3	-	56.3	56.6	60.9	42.5	57.2	90.5	61.7	60.2	90.5
SFDR(dB)	85.0	72.0	85.0	72.0	79.0	67.6	84.6	84.6	64.7	63.2	78.1	57.5	80.7	-	69.2	73.1	72.0	57.5	67.8	-	72.3	78.3	-
$P_d(W)$	5.10E-03	2.60E-03	5.10E-03	2.60E-03	2.96E-03	2.46E-03	1.00E-03	1.00E-03	2.46E-03	3.65E-03	2.72E-03	2.28E-03	7.68E-03	2.30E-03	1.42E-02	2.48E-02	2.02E-02	2.28E-03	1.32E-02	9.82E-03	6.13E-02	6.00E-03	9.82E-03
$FOM_s(dB)$	165.9	154.6	165.9	154.6	168.3	159.4	174.9	174.9	159.7	151.9	167.9	148.9	158.4	-	139.5	139.6	163.9	148.9	160.9	179.3	165.9	167.2	179.3
$FOM_w(fJ/cs)$	6.99E-14	1.50E-13	6.99E-14	1.50E-13	4.05E-14	4.61E-14	6.98E-15	4.45E-14	1.59E-13	4.06E-14	1.04E-13	2.87E-13	-	3.20E-14	4.43E-14	4.49E-14	2.79E-14	1.03E-13	3.57E-14	2.39E-14	1.93E-14	1.20E-14	2.39E-14
$C_1(F)$	2.00E-13	2.00E-13	2.00E-13	2.00E-13	2.00E-13	2.00E-13	2.00E-13	2.00E-13	-	2.50E-13	-	-	3.20E-14	2.50E-13	2.50E-13	2.50E-13	2.00E-13	2.50E-13	2.00E-13	3.20E-14	2.00E-13	6.40E-11	3.20E-14
$C_f(F)$	8.00E-13	4.00E-13	8.00E-13	4.00E-13	8.00E-13	4.00E-13	6.40E-12	6.40E-12	4.00E-13	-	4.00E-12	1.00E-12	-	2.05E-12	5.00E-13	5.00E-13	8.00E-13	1.00E-12	8.00E-13	2.05E-12	4.00E-13	1.28E-10	2.05E-12
$A_V(V/V)$	4	2	4	2	4	2	32	32	2	2	16	4	64	4	2	2	4	4	4	64	2	2	64
$T_A(s)$	5.00E-08	3.33E-08	5.00E-08	3.33E-08	5.00E-08	1.00E-08	1.00E-08	1.00E-08	1.00E-08	1.43E-08	2.50E-08	5.00E-09	1.20E-08	1.00E-08	1.67E-09	1.00E-09	1.25E-09	5.00E-09	1.60E-09	2.10E-08	6.25E-10	3.00E-10	2.10E-08
$GBW(Hz)$	5.86E+07	4.40E+07	5.86E+07	4.40E+07	5.86E+07	1.47E+08	2.35E+09	1.47E+08	1.03E+08	4.69E+08	5.91E+08	2.44E+08	4.69E+09	8.80E+08	1.47E+09	2.35E+09	5.86E+08	1.83E+09	2.23E+09	2.35E+09	4.89E+09	2.23E+09
$LIN(b)$	9.8	8.2	9.8	8.2	9.8	8.1	6.5	6.5	8.1	7.4	7.7	4.8	9.4	-	8.1	8.1	7.8	4.8	7.2	8.7	9.0	8.7	8.7
$PSRR(dB)$	9.8	3.7	9.8	3.7	10.5	4.4	28.5	28.5	4.4	6.0	22.5	10.5	7.4	35.3	6.9	6.9	12.0	10.5	12.0	31.0	7.4	7.4	31.0
$CMRR(dB)$	12.0	6.0	12.0	6.0	12.0	6.0	30.1	30.1	6.0	6.0	24.1	12.0	12.0	36.1	6.0	6.0	12.0	12.0	12.0	36.1	6.0	6.0	36.1
$THD^-(dB)$	-61.0	-51.0	-61.0	-51.0	-61.0	-50.3	-40.8	-40.8	-50.6	-46.0	-48.2	-30.5	-58.3	-	-50.3	-50.6	-48.9	-30.5	-45.2	-	-55.7	-54.2	-
$THD^+(dB)$	-73.0	-66.0	-73.0	-66.0	-67.0	-61.6	-54.5	-54.5	-58.7	-57.1	-54.0	-45.5	-68.7	-	-63.2	-67.1	-60.0	-45.5	-55.8	-	-67.3	-72.3	-
$THD(dB)$	-67.0	-58.5	-67.0	-58.5	-64.0	-55.9	-47.6	-47.6	-54.6	-51.6	-51.1	-38.0	-63.5	-	-56.7	-58.8	-54.4	-38.0	-50.5	-54.4	-61.5	-63.2	-54.4
$P_d(W)$	9.00E-05	9.00E-05	9.00E-05	9.00E-05	3.98E-04	1.00E-03	3.66E-04	3.66E-04	1.00E-03	-	9.10E-04	9.30E-04	-	-	4.90E-03	-	1.03E-02	9.30E-04	8.77E-03	2.70E-03	-	2.98E-03	2.70E-03
$FOM(V^{-2})$	4.34E+00	1.95E-01	5.21E-01	1.95E-01	1.18E-01	5.86E-02	4.10E+01	4.10E+01	5.86E-02	-	2.06E+00	6.30E-01	-	-	8.97E-02	-	1.82E-01	6.30E-01	1.67E-01	1.69E+00	-	2.42E+02	1.69E+00
Year	2012	2012	2012	2012	2013	2014	2015	2015	2015	2016	2017	2017	2017	2017	2017	2018	2018	2018	2019	2019	2019	2019	2019
Reference	[38]	[33]	[42]	[65]	[37]	[2]	[177]	[34]	[75]	[24]	[123]	[26]	[29]	[29]	[39]	[79]	[76]	[40]	[46]	[44]	[45]	[77]	[4]
CMOS(nm)	2.80E-08	9.00E-08	2.80E-08	2.80E-08	1.60E-08	6.50E-08	4.00E-08	2.80E-08	2.80E-08	1.60E-08	1.60E-08	1.60E-08	1.60E-08	1.60E-08	1.60E-08	2.80E-08	2.80E-08	1.60E-08	2.80E-08	6.50E-08	2.80E-08	2.80E-08	2.80E-08
$A(m^2)$	5.40E-07	3.68E-07	6.21E-04	2.02E-07	1.02E-07	1.88E-06	-	1.80E-08	1.80E-08	9.50E-05	9.50E-08	1.94E-04	3.70E-08	9.50E-08	8.40E-09	4.00E-09	1.30E-08	8.40E-09	1.20E-07	1.73E-06	-	1.30E-08	3.40E-08
$V_{DD}(V)$	0.9	1.1	0.9	0.8	0.8	1.2	1.3	1.1	1.1	0.9	0.9	0.9	0.9	0.9	0.9	0.9	1.1	0.9	1.0	1.0	0.9	1.1	1.0
$V_{FS}(V_{pp})$	1.6	2.2	1.6	1.4	2.9	1.0	-	-	-	1.6	1.6	-	1.5	1.6	1.8	-	-	1.8	1.4	-	0.8	2.2	-
N(b)	12.0	16.0	12.0	12.0	14.0	11.0	12.0	14.0	14.0	11.0	11.0	13.0	11.0	11.0	11.5	16.0	14.0	11.5	12.0	12.5	11.0	14.0	12.0
$f_1(b)$	10.0	4.0	10.0	1.0	1.0	9.0	1.0	1.0	1.0	7.0	7.0	9.0	7.0	7.0	1.0	1.0	1.0	1.0	1.0	1.0	5.0	1.0	1.0
$n_1(b)$	1.5	3.5	1.5	5.0	6.0	1.5	6.0	6.0	1.5	1.5	1.5	1.5	1.5	1.5	6.0	6.0	6.0	2.5	2.5	2.5	3.0	6.0	3.0
$f_2(b)$	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	-	1.0	3.0
$n_2(b)$	2.0	4.0	2.0	8.0	9.0	2.0	7.0	9.0	9.0	4.5	4.5	4.5	4.5	4.5	8.0	10.0	9.0	8.0	2.0	2.0	-	9.0	4.0
$f_3(b)$	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1.0	1.0	-	-	-
$n_3(b)$	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	9.0	9.0	-	-	-
R(b)	0.5	0.5	0.5	1.0	1.0	0.5	1.0	1.0	1.0	0.5	0.5	0.5	0.5	0.5	1.0	1.0	1.0	1.0	0.5	0.5	1.0	1.0	1.0
$fs(sps)$	1.00E+09	2.40E+07	6.00E+08	1.40E+08	1.00E+08	1.00E+08	2.00E+08	1.00E+08	1.00E+08	1.01E+09	1.00E+09	1.00E+09	6.00E+08	1.00E+09	5.00E+08	2.00E+08	1.30E+08	5.00E+08	6.25E+08	2.50E+09	1.25E+09	1.30E+08	2.00E+09
ENOB(b)	9.1	12.0	9.5	8.4	11.8	-	-	11.6	11.6	9.6	9.6	10.0	9.7	9.6	10.1	10.8	14.2	10.1	9.4	8.2	9.6	11.8	9.7
SNDR(dB)	56.6	74.3	58.7	52.6	72.6	-	-	71.7	71.7	59.5	59.5	61.9	60.2	59.5	62.3	66.7	87.5	62.3	58.6	51.0	59.5	72.5	60.4
SFDR(dB)	73.1	85.5	72.4	65.7	86.5	-	-	85.1	87.2	75.9	75.9	75.2	78.3	75.9	75.5	87.2	72.5	75.5	70.7	68.0	73.5	87.5	75.8
$P_d(W)$	2.48E-02	5.10E-03	1.45E-02	1.10E-03	2.50E-03	1.86E-02	-	7.00E-04	7.00E-04	1.09E-02	1.09E-02	6.00E-03	1.09E-02	3.30E-03	1.09E-02	3.30E-03	1.30E-03	8.20E-04	3.30E-03	1.32E-02	4.18E-01	2.85E-02	8.20E-04	2.70E-02
$FOM_s(dB)$	159.6	168.0	161.9	160.6	175.6	94.3	-	180.2	180.2	166.1	166.1	166.1	167.2	166.1	171.1	175.6	196.5	171.1	162.3	145.8	162.9	181.5	166.1
$FOM_w(fJ/cs)$	4.39E-14	5.01E-14	3.44E-14	2.25E-14	7.17E-15	-	-	2.23E-15	2.23E-15	1.41E-14	1.41E-14	1.41E-14	1.20E-14	1.41E-14	6.20E-15	3.68E-15	3.25E-16	6.20E-15	3.04E-14	5.77E-13	2.96E-14	1.83E-15	1.58E-14
$C_1(F)$	2.00E-13	-	2.00E-13	1.00E-14	1.00E-13	-	-	-	-	2.56E-13	2.56E-13	2.00E-13	6.40E-14	2.56E-13	-	8.00E-15	4.80E-14	-	2.00E-13	2.00E-10	1.28E-13	4.80E-14	1.40E-14
$C_f(F)$	4.00E-13	-	4.00E-13	1.60E-13	-	2.00E-13	-	-	-	5.12E-13	5.12E-13	4.00E-13	1.28E-13	5.12E-13	-	5.12E-13	1.54E-12	-	8.00E-13	8.00E-10	5.12E-13	1.54E-12	5.60E-13
$A_V(V/V)$	2	8	2	16	32	2	32	32	32	2	2	2	2	2	32	64	32	32	4	4	4	32	4
$T_A(s)$	4.00E-10	4.17E-08	2.00E-10	7.14E-09	2.00E-08	1.50E-09	7.30E-10	2.00E-08	2.00E-08	4.20E-10	4.20E-10	2.00E-09	3.00E-10	4.20E-10	2.00E-09	5.00E-09	1.54E-08	0.000000002	3.20E-09	8.00E-10	1.60E-09	1.54E-08	1.3E-10
$GBW(Hz)$	3.66E+09	1.41E+08	7.33E+09	1.64E+09	1.17E+09	9.77E+08	3.21E+10	1.17E+09	1.17E+09	3.49E+09	3.49E+09	7.33E+08	4.89E+09	3.49E+09	1.17E+10	9.38E+09	1.52E+09	1.17E+10	9.16E+08	3.66E+09	1.83E+09	1.52E+09	2.26E+10
$LIN(b)$	8.1	9.0	8.5	4.4	6.8	-	-	6.6	6.6	8.6	8.6	9.0	8.7	8.6	5.1	4.8	9.2	5.1	7.4	6.2	7.6	6.8	6.7
$PSRR(dB)$	7.4	17.2	7.4	26.0	32.6	4.4	28.1	29.3	29.3	7.4	6.9	6.9	7.4	7.4	31.5	37.5	29.3	31.5	12.0	12.0	13.0	29.3	12.0
$CMRR(dB)$	6.0	18.1	6.0	24.1	30.1	6.0	30.1	30.1	30.1	6.0	6.0	6.0	6.0	6.0	30.1	36.1	30.1	30.1	12.0	12.0	12.0	30.1	12.0
$THD^-(dB)$	-50.6	-56.2	-52.7	-28.5	-42.5	-	-	-41.6	-41.6	-53.5	-53.5	-55.9	-54.2	-53.5	-32.2	-30.6	-57.4	-32.2	-46.6	-39.0	-47.5	-42.4	-48.4
$THD^+(dB)$	-67.1	-67.4	-66.4	-41.6	-56.4	-	-	-55.0	-57.1	-69.9	-69.9	-69.9	-72.3	-69.9	-45.4	-51.1	-42.4	-45.4	-58.7	-56.0	-61.5	-57.4	-63.8
$THD(dB)$	-58.8	-61.8	-59.5	-35.1	-49.4	6.0	30.1	-48.3	-49.3	-61.7	-61.7	-62.5	-63.2	-61.7	-38.8	-40.8	-49.9	-38.8	-52.6	-47.5	-54.5	-49.9	-56.1
$P_d(W)$	-	-	4.90E-03	1.70E-04	1.05E-03	-	1.16E-03	2.30E-04	2.30E-04	-	-	-	2.58E-03	-	-	-	-	-	3.32E-03	3.72E-02	7.40E-03	2.90E-04	-
$FOM(V^{-2})$	-	-	4.90E-01	1.55E+00	-	-	-	-	-	-	-	-	2.42E-01	-	-	-	-	-	2.21E-01	7.88E+01	1.27E-01	8.07E+00	-
Year	2019	2019	2019	2019	2020	2020	2020	2020	2020	2020	2021	2021	2021	2021	2021	2022	2022	2022	2022	2022	2022	2022	2023

B

Cadence Virtuoso Testbenches

This appendix contains the *Cadence Virtuoso* testbenches utilized in the electrical simulations, including those for the diode-connected inverter, as well as the ones for conventional and critically damped ring amplifiers.

B.1 Diode-Connected Inverter

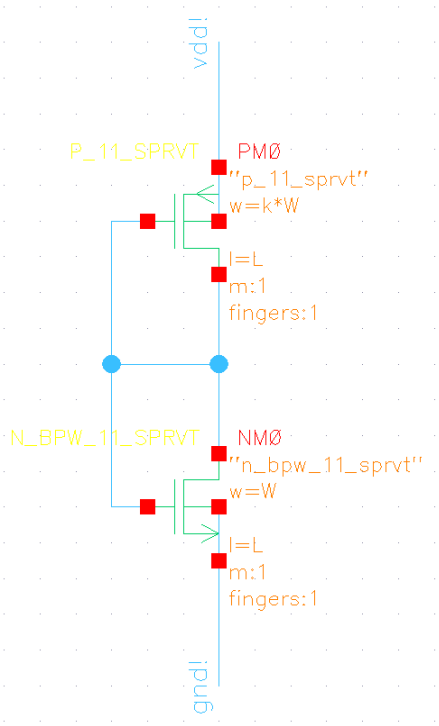


Figure B.1: Diode-connected inverter testbench.

B.2 Conventional Ring Amplifier

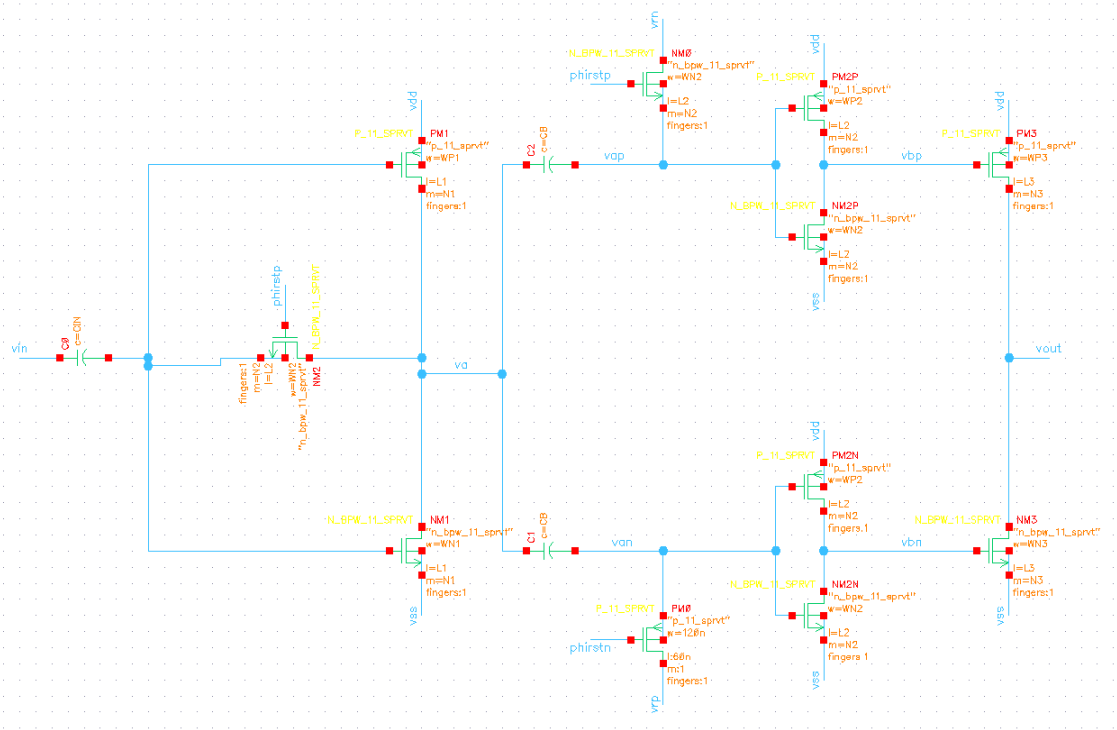


Figure B.2: Conventional ring amplifier testbench.

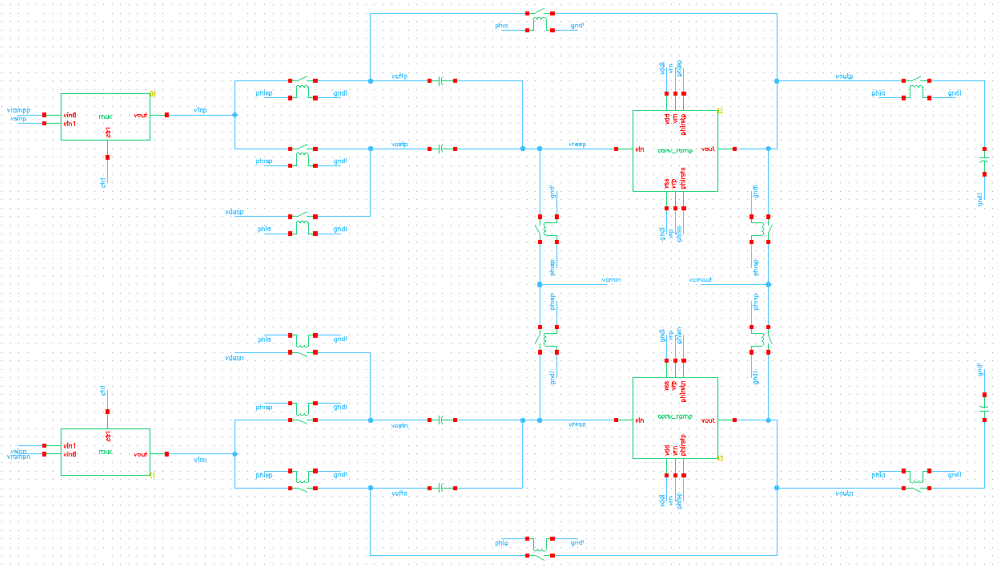


Figure B.3: Pseudo-differential flip-around MDAC testbench.

B.3 Critically Damped Ring Amplifier

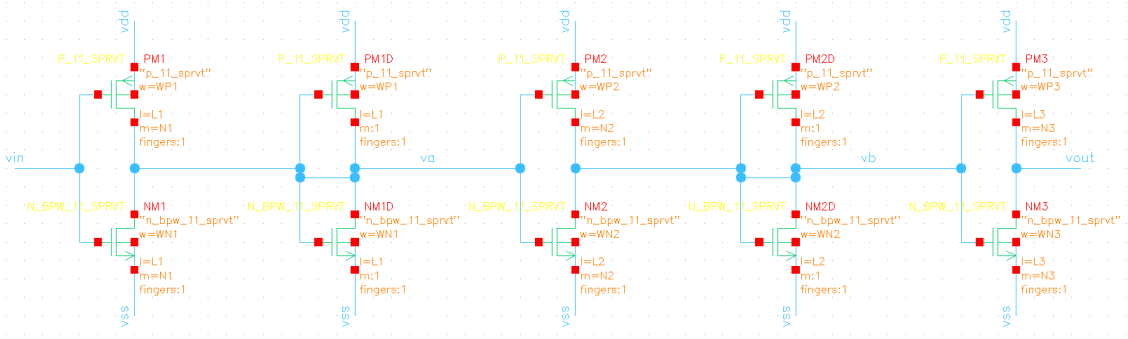


Figure B.4: Critically damped ring amplifier testbench.

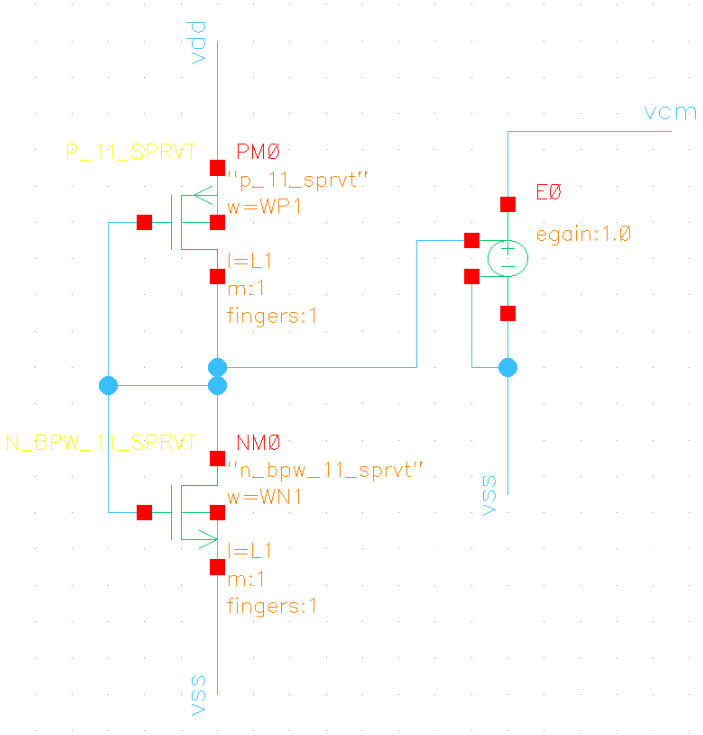


Figure B.5: CMFB testbench.

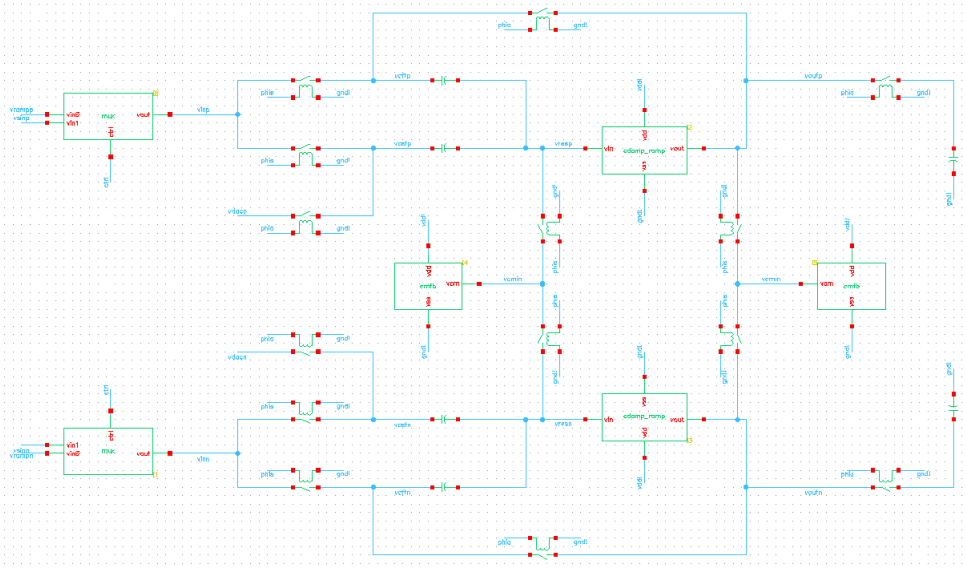


Figure B.6: Pseudo-differential flip-around MDAC with CMFB testbench.

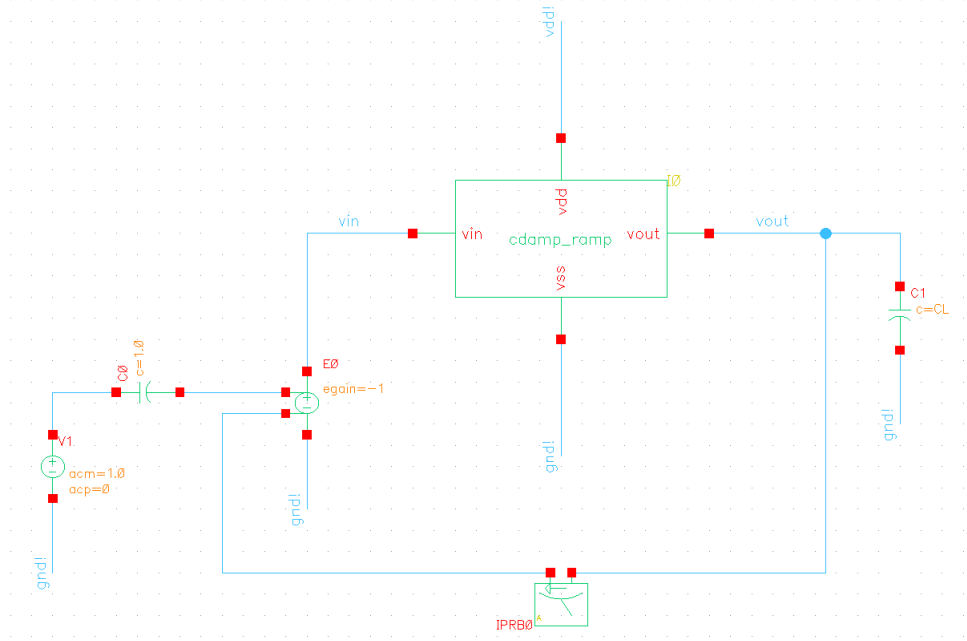


Figure B.7: Probe in negative feedback testbench.

C

Code Algorithms

This appendix provides the code algorithms, including the *Verilog-A* code for the multiplexer and the *Python* code for the analysis and design algorithm.

C.1 *Verilog-A* Devices

Listing C.1: Multiplexer.

```
// VerilogA for ramps, mux, veriloga

`include "constants.vams"
`include "disciplines.vams"

module mux(vin0, vin1, ctrl, vout);
    input vin0, vin1, ctrl;
    output vout;
    electrical vin0, vin1, ctrl, vout;
    real val;
    analog begin
        @ (initial_step) begin
            if (V(ctrl) < 0) begin
                $display("Error: ctrl < 0.\n");
                $finish;
            end
        end
        if (V(ctrl) > 0.5) begin
            val = V(vin1);
        end
        else begin
            val = V(vin0);
        end
        V(vout) <+ slew(val);
    end
endmodule
```

C.2 *Python* Algorithm

Listing C.2: Analysis and design algorithm.

```
## Packages and Files

### Packages
import pandas as pd # type: ignore
import matplotlib.pyplot as plt # type: ignore
import numpy as np # type: ignore

from tabulate import tabulate # type: ignore

### Read Files
def read_file(transistors_params, l_value, sizing_ratios, k_value,
    cols_indexes, cols_names):
    """
    Read file

    Parameters:
    - transistors_params (dict): Dictionary of file paths keyed by lengths
    - l_value (int): Transistor length
    - sizing_ratios (dict): Dictionary of sizing ratios keyed by column
        indexes
    - k_value (float): Sizing ratio
    - cols_indexes (array int): Columns indexes
    - cols_names (array str): Columns names

    Returns:
    - data (df): Specified data
    """
    # Validate inputs
    if l_value not in transistors_params:
        raise ValueError(f"Error: Transistor length {l_value} nm not found in
            file paths.")

    if k_value is not None:
        if sizing_ratios is None or k_value not in sizing_ratios.values():
            raise ValueError(f"Error: Sizing ratio {k_value} not found in
                sizing_ratios.")
        row_idx = list(sizing_ratios.keys())[list(sizing_ratios.values()).
            index(k_value)]
    else:
        row_idx = None

    # Select file path by length
    trans_param = transistors_params[l_value]

    # Read the CSV file directly
    try:
```

```
42          df = pd.read_csv(trans_param, usecols=cols_indexes, names=cols_names,
                header=None)
43      except Exception as e:
44          raise ValueError(f"Error reading CSV file: {e}")
45
46      # Get the specified row for the given k_value if provided
47      if row_idx is not None:
48          try:
49              data = df.iloc[row_idx]
50          except IndexError as e:
51              raise IndexError(f"Error accessing row index: {e}")
52      else:
53          data = df
54
55      return data
56
57  ####################################################
58
59  ## Diode Inverter Efficiency
60
61  ### Plot Transconductance Efficiency
62  def plot_inv_transconductance_efficiency_diff_lengths(transistors_params,
        sizing_ratios=None, k_value=None):
63      """
64      Plot inverter transconductance efficiency for different lengths
65
66      Parameters:
67      - transistors_params (dict): Dictionary of file paths keyed by lengths
68      - sizing_ratios (dict, optional): Dictionary of sizing ratios keyed by
            column indexes
69      - k_value (float, optional): Sizing ratio
70      """
71      # Define constants for column indices and names
72      nmos_gmid_index = 11
73      nmos_gmgds_ft_idwl_indexes = [12, 13, 14]
74
75      pmos_gmid_index = 25
76      pmos_gmgds_ft_idwl_indexes = [26, 27, 28]
77
78      gmid_name = '$g_m/I_D$'
79      gmgds_ft_idwl_names = ['$g_m/g_{ds}$', '$f_T$', '$I_D/W$']
80
81      _ = 'Transconductance Efficiency'
82      gmgds_ft_idwl_descriptions = ['Self Gain', 'Transit Frequency', '
            Normalized Current']
83
84      gmid_unit = ' $(V^{-1})$'
85      gmgds_ft_idwl_units = ['', '$(Hz)$', '$(A/m)$']
86
87      # Number of graphs to plot
```

```
88      num_gphs = len(gmgds_ft_idwl_names)
89
90      # Create subplots for NMOS and PMOS metrics side by side
91      fig, axes = plt.subplots(nrows=num_gphs, ncols=2, figsize=(16, 2 *
            num_gphs))
92
93      # Store lines and labels for legend
94      nmos_lines = []
95      pmos_lines = []
96      nmos_labels = []
97      pmos_labels = []
98
99      for l_value, trans_param in transistors_params.items():
100
101         for j in range(num_gphs):
102             # NMOS
103             nmos_metric_index = nmos_gmgds_ft_idwl_indexes[j]
104             nmos_metric_name = gmgds_ft_idwl_names[j]
105             nmos_metric_description = gmgds_ft_idwl_descriptions[j]
106             nmos_metric_unit = gmgds_ft_idwl_units[j]
107
108             # Read the CSV file and extract NMOS metrics
109             nmos_df = read_file(transistors_params, l_value, sizing_ratios,
                    k_value, [nmos_gmid_index, nmos_metric_index], [gmid_name,
                    nmos_metric_name])
110
111             # Plot NMOS metrics against gm/ID
112             nmos_line, = axes[j, 0].plot(nmos_df[gmid_name], nmos_df[
                    nmos_metric_name], label=f'$L = {l_value}$ $nm')
113
114             # Only store labels once
115             if j == 2:
116                 nmos_lines.append(nmos_line)
117                 nmos_labels.append(f'$L = {l_value}$ $nm$')
118
119             # Set NMOS plot titles, labels, grid and limits
120             axes[j, 0].set_title(f'NMOS {nmos_metric_description}')
121             axes[j, 0].set_xlabel(f'{gmid_name} {gmid_unit}')
122             axes[j, 0].set_ylabel(f'{nmos_metric_name} {nmos_metric_unit}')
123             axes[j, 0].set_yscale('log')
124             axes[j, 0].grid(True)
125             axes[j, 0].set_xlim([3, 9])
126
127             # PMOS
128             pmos_metric_index = pmos_gmgds_ft_idwl_indexes[j]
129             pmos_metric_name = gmgds_ft_idwl_names[j]
130             pmos_metric_description = gmgds_ft_idwl_descriptions[j]
131             pmos_metric_unit = gmgds_ft_idwl_units[j]
132
133             # Read the CSV file and extract PMOS metrics
```

```
134             pmos_df = read_file(transistors_params, l_value, sizing_ratios,
                    k_value, [pmos_gmid_index, pmos_metric_index], [gmid_name,
                    pmos_metric_name])
135
136             # Plot PMOS metrics against gm/ID
137             pmos_line, = axes[j, 1].plot(pmos_df[gmid_name], pmos_df[
                    pmos_metric_name])
138
139             # Only store labels once
140             if j == 2:
141                 pmos_lines.append(pmos_line)
142                 pmos_labels.append(f'$L = {l_value}$ $nm$')
143
144             # Set PMOS plot titles, labels, grid and limits
145             axes[j, 1].set_title(f'PMOS {pmos_metric_description}')
146             axes[j, 1].set_xlabel(f'{gmid_name} {gmid_unit}')
147             axes[j, 1].set_ylabel(f'{pmos_metric_name} {pmos_metric_unit}')
148             axes[j, 1].set_yscale('log')
149             axes[j, 1].grid(True)
150             axes[j, 1].set_xlim([3, 9])
151
152             # Customize y-axis ticks to avoid intermediate labels
153             axes[j, 0].yaxis.set_major_locator(plt.LogLocator(base=10.0))
154             axes[j, 1].yaxis.set_major_locator(plt.LogLocator(base=10.0))
155
156     # Adjust subplot layout to prevent overlap
157     plt.tight_layout(rect=[0, 0.03, 1, 0.95])
158
159     # Create legends for NMOS and PMOS
160     fig.legend(nmos_lines, nmos_labels, loc='upper center', bbox_to_anchor
            =(0.3, 0), ncol=3, fontsize=10, frameon=False, bbox_transform=plt.gcf()
            .transFigure)
161     fig.legend(pmos_lines, pmos_labels, loc='upper center', bbox_to_anchor
            =(0.75, 0), ncol=3, fontsize=10, frameon=False, bbox_transform=plt.gcf
            ().transFigure)
162
163     # Display the plot
164     plt.show()
165
166 ### Plot Trip Point
167 def plot_inv_trip_point_diff_lengths(transistors_params):
168     """
169     Plot inverter trip point for different lengths
170
171     Parameters:
172     - transistors_params (dict): Dictionary containing file paths keyed by
            lengths
173     """
174     # Define constants for column indices and names
175     k_idx = 0
```

```
176     vgs_idx = 2
177 
178     k_name = '$k_W$'
179     vgs_name = '$V_{TP}$'
180 
181     _ = 'Sizing Ratio'
182     vgs_name_ = 'Trip Point Voltage against Width Ratio Factor'
183     k_unit = ''
184     vgs_unit = '$(V)$'
185 
186     # Create single plot
187     fig, ax = plt.subplots(figsize=(8, 3))
188 
189     # Ensure x-axis is set to log scale
190     ax.set_xscale('log')
191 
192     for l_value, trans_param in transistors_params.items():
193 
194         # Read CSV file and extract Vgs and k columns
195         df = read_file(transistors_params, l_value, None, None, [k_idx,
                vgs_idx], [k_name, vgs_name])
196 
197         vgs_mV = df[vgs_name]
198 
199         # Plot Vgs vs. k for the current L value
200         ax.semilogx(df[k_name], vgs_mV, marker='o', label=f'$L = {l_value}$
                $nm$')
201 
202     # Set plot title, labels, and grid
203     ax.set_title(f'\n{vgs_name_}', fontsize=12)  # Add newline for space
204     ax.set_xlabel(f'{k_name} {k_unit}', fontsize=10)
205     ax.set_ylabel(f'{vgs_name} {vgs_unit}', fontsize=10)
206     ax.axhline(y=0.6, color='r', linestyle='--')
207     ax.grid(True)
208 
209     # Adjust subplot layout to prevent overlap
210     plt.tight_layout()
211 
212     # Create a single horizontal legend just above the last subplot
213     plt.legend(loc='upper center', bbox_to_anchor=(0.5, 0), ncol=3, fontsize
            =10, frameon=False, bbox_transform=plt.gcf().transFigure)
214 
215     # Display the plot
216     plt.show()
217 
218 ##################################################
219 
220 ## Ring Amplifier Perfomance Close Loop
221 
222 ### Calculate Ramp Perfomance Close Loop
```

```
223 def calc_ramp_performance_cl(VDD, N, CF):
224     """
225     Calculate ring amplifier perfomance in close loop
226 
227     Parameters:
228     - N (int): Resolution
229     - CF (float): Feedback cpacitance
230 
231     Returns:
232     - Af (float): Feedback gain
233     - B (float): Feedback factor
234     - CS, CL (float): Source and load capacitances
235 
236     """
237     # Determine feedback gain (V/V)
238     Af = 2 ** (N - 1)
239 
240     # Determine the source and load capacitances (F)
241     CS = CF * (Af - 1)
242     CL = Af * CF
243 
244     # Determine the feedback factor           *** considering the flip-around
            ***
245     β  = CF / (CS + CF)
246 
247     # Determine the input and output common mode voltages (V)
248     Vcmin = VDD / 2
249     Vcmout = VDD / 2
250 
251     return Af, β, CS, CL, Vcmin, Vcmout
252 
253 ### Print Ramp Perfomance Close Loop
254 def print_ramp_perfomance_cl(Af, β , CF, CS, CL, VDD, Vcmin, Vcmout):
255     """
256     Print design characteristics
257 
258     Parameters:
259     - Af (float): Feedback gain
260     - β  (float): Feedback factor
261     - CF, CS, CL (float): Feedback, sample and load capacitances
262     - VDD (float): Suplly voltage
263     - Vcmin, Vcmout (float): Input and output common mode voltages
264     """
265     # Define header
266     headers = ["Parameter", "Value"]
267 
268     # Define data
269     data = [
270         ["Af (V/V)", f"{Af:.3f}"],
271         ["β (F/F)", f"{β :.3f}"],
```

```
272         ["VDD (V)", f"{VDD:.1f}"],
273         ["Vcmin (V)", f"{Vcmin:.3f}"],
274         ["Vcmout (V)", f"{Vcmout:.3f}"],
275         ["CF (fF)", f"{CF * 1e15:.3f}"],
276         ["CS (fF)", f"{CS * 1e15:.3f}"],
277         ["CL (fF)", f"{CL * 1e15:.3f}"],
278     ]
279 
280     # Create table
281     table = tabulate(data, headers, tablefmt="grid")
282 
283     # Print table
284     print(table)
285 
286 ######################################################
287 
288 ## Ring Amplifier Perfomance Open Loop
289 
290 ### Transfer Function
291 def ramp_transfer_function(ramp, s, N_1, Cds_1, Cgd_1, Cgs_1, gds_1, gm_1, N_2
        , Cds_2, Cgd_2, Cgs_2, gds_2, gm_2, N_3, Cds_3, Cgd_3, Cgs_3, gds_3, gm_3,
        CL):
292     """
293     Calculates the transfer function for the specified ring amplifier
294 
295     Parameters:
296     - ramp (str): Ringamp
297     - s (list float): Complex frequency
298     - N_1, Cds_1, Cgd_1, Cgs_1, gds_1, gm_1 (float): Parameters stage 1
299     - N_2, Cds_2, Cgd_2, Cgs_2, gds_2, gm_2 (float): Parameters stage 2
300     - N_3, Cds_3, Cgd_3, Cgs_3, gds_3, gm_3 (float): Parameters stage 3
301     - CL (float): Load capacitance
302 
303     Returns:
304     - H (list float): Transfer function
305     """
306     # Calculate inverters transfer functions by selecting the ringamp
307     if ramp == "Conventional":
308         inv_1 = ( N_1 * ( Cgd_1 * s - gm_1 ) ) / ( ( N_1 * ( Cds_1 + Cgd_1 ) +
                N_2 * ( Cgd_2 + Cgs_2 ) ) * s + N_1 * gds_1 )
309         inv_2 = ( N_2 * ( Cgd_2 * s - gm_2 ) ) / ( ( N_2 * ( Cds_2 + Cgd_2 ) +
                N_3 * ( Cgd_3 + Cgs_3 ) ) * s + N_2 * gds_2 )
310 
311     elif ramp == "Critically Damped":
312         inv_1 = ( N_1 * ( Cgd_1 * s - gm_1 ) ) / ( ( N_1 * ( Cds_1 + Cgd_1 ) +
                Cds_1 + Cgs_1 + N_2 * ( Cgd_2 + Cgs_2 ) ) * s + ( N_1 + 1 ) *
               gds_1 + gm_1 )
313         inv_2 = ( N_2 * ( Cgd_2 * s - gm_2 ) ) / ( ( N_2 * ( Cds_2 + Cgd_2 ) +
                Cds_2 + Cgs_2 + N_3 * ( Cgd_3 + Cgs_3 ) ) * s + ( N_2 + 1 ) *
               gds_2 + gm_2 )
```

```
314     inv_3 = ( N_3 * ( Cgd_3 * s - gm_3 ) ) / ( ( N_3 * (Cds_3 + Cgd_3 ) + CL )
            * s + N_3 * gds_3 )
315
316     # Calculate ringamp transfer function
317     H = inv_1 * inv_2 * inv_3
318
319     return H
320
321 ### Gain
322 def ramp_gain(ramp,
323                 N_1, gds_1, gm_1,
324                 N_2, gds_2, gm_2,
325                 gds_3, gm_3):
326     """
327     Calculates the gain magnitude for the specified ring amplifier
328
329     Parameters:
330     - ramp (str): Ringamp
331     - N_1, gds_1, gm_1 (float): Parameters stage 1
332     - N_2, gds_2, gm_2 (float): Parameters stage 2
333     - gds_3, gm_3 (float): Parameters stage 3
334     - CL (float): Load capacitance
335
336     Returns:
337      - A (float): Gain
338     """
339     # Calculates inverters gains function for the selected ringamp
340     if ramp == "Conventional":
341         A_1 = - gm_1 / gds_1
342         A_2 = - gm_2 / gds_2
343
344     elif ramp == "Critically Damped":
345         A_1 = - ( N_1 * gm_1 ) / ( ( N_1 + 1 ) * gds_1 + gm_1 )
346         A_2 = - ( N_2 * gm_2 ) / ( ( N_2 + 1 ) * gds_2 + gm_2 )
347
348     A_3 = - gm_3  / gds_3
349
350     # Calculates ringamp gain in dB
351     A = 20 * np.log10( np.abs( A_1 * A_2 * A_3 ) )
352
353     return A
354
355 ### Zeros
356 def ramp_freqs_zeros(Cgd_1, gm_1,
357                      Cgd_2, gm_2,
358                      Cgd_3, gm_3):
359     """
360     Calculates the frequencies of zeros for both ring amplifiers
361
362     Parameters:
```

```
363     - Cgd_1, gm_1 (float): Parameters stage 1
364     - Cgd_2, gm_2 (float): Parameters stage 2
365     - Cgd_3, gm_3 (float): Parameters stage 3
366 
367     Returns:
368     - f_zeros (list floats): Frequencies of zeros
369     """
370     # Calculates ringamp zeros in rad/s
371     s_z_1 = gm_1 / Cgd_1
372     s_z_2 = gm_2 / Cgd_2
373     s_z_3 = gm_3 / Cgd_3
374 
375     # Add zeros to array
376     s_z = np.array( [s_z_1, s_z_2, s_z_3] )
377 
378     # Convert ringamp zeros to Hz
379     f_z = np.abs( s_z / ( 2 * np.pi ) )
380 
381     return f_z
382 
383 ### Poles
384 def ramp_freqs_poles(ramp, N_1, Cds_1, Cgd_1, Cgs_1, gds_1, gm_1, N_2, Cds_2,
        Cgd_2, Cgs_2, gds_2, gm_2, N_3, Cds_3, Cgd_3, Cgs_3, gds_3, CL):
385     """
386     Calculates the frequencies of poles for the specified ring amplifier
387 
388     Parameters:
389     - ramp (str): Ringamp
390     - N_1, Cds_1, Cgd_1, gds_1, gm_1 (float): Parameters stage 1
391     - N_2, Cds_2, Cgd_2, Cgs_2, gds_2, gm_2 (float): Parameters stage 2
392     - N_3, Cds_3, Cgd_3, Cgs_3, gds_3 (float): Parameters stage 3
393     - CL (float): Load capacitance
394 
395     Returns:
396     - f_p (list float): Frequencies of poles (Hz)
397     """
398     # Calculates ringamp poles in rad/s for the selected ringamp
399     if ramp == "Conventional":
400         s_p_1 = - ( N_1 * gds_1) / ( N_1 * ( Cds_1 + Cgd_1) + N_2 * ( Cgd_2 +
                Cgs_2 ) )
401         s_p_2 = - ( N_2 * gds_1) / ( N_2 * ( Cds_2 + Cgd_2) + N_3 * ( Cgd_2 +
                Cgs_2 ) )
402 
403     elif ramp == "Critically Damped":
404         s_p_1 = - ( ( N_1 + 1 ) * gds_1 + gm_1 ) / ( N_1 * ( Cds_1 + Cgd_1) +
                Cds_1 + Cgs_1 + N_2 * ( Cgd_2 + Cgs_2 ) )
405         s_p_2 = - ( ( N_2 + 1 ) * gds_2 + gm_2 ) / ( N_2 * ( Cds_2 + Cgd_2) +
                Cds_2 + Cgs_2 + N_3 * ( Cgd_3 + Cgs_3 ) )
406 
407     s_p_3 = - ( N_3 * gds_3) / ( N_3 * ( Cds_3 + Cgd_3 ) + CL )
```

```
408
409     # Add poles to array
410     s_p = np.array( [s_p_1, s_p_2, s_p_3] )
411
412     # Convert ringamp poles to Hz
413     f_p = np.abs( s_p / ( 2 * np.pi ) )
414
415     return f_p
416
417 ### Bandwidth
418 def ramp_bandwidth(N_3, Cds_3, Cgd_3, gds_3, CL):
419     """
420     Calculates the bandwidth for both ring amplifiers
421
422     Parameters:
423     - N_3, Cds_3, Cgd_3, gds_3 (float): Parameters stage 3
424     - CL (float): Load capacitance
425
426     Returns:
427     - BW (float): Bandwidth (Hz)
428     """
429     # Calculates ringamp bandwidth in rad/s
430     s_p_3 = - ( N_3 * gds_3 ) / ( N_3 * ( Cds_3 + Cgd_3 ) + CL)
431
432     # Convert ringamp bandwidth to Hz
433     BW = np.abs( s_p_3 / ( 2 * np.pi ) )
434
435     return BW
436
437 ### Gain-Bandwidth Product
438 def ramp_gain_bandwidth_product(ramp, N_1, gds_1, gm_1, N_2, gds_2, gm_2, N_3,
      Cds_3, Cgd_3, gm_3, CL):
439     """
440     Calculates the gain-bandwidth product for the specified ring amplifier
441
442     Parameters:
443     - ramp (str): Ringamp
444     - N_1, gds_1, gm_1 (float): Parameters stage 1
445     - N_2, gds_2, gm_2 (float): Parameters stage 2
446     - N_3, Cds_3, Cgs_3, gds_3, gm_3 (float): Parameters stage 3
447     - CL (float): Load capacitance
448
449     Returns:
450     - GBW (float): Gain bandwidth product
451     """
452     # Calculates the gain bandwidth product for the selected ringamp
453     if ramp == "Conventional":
454         A_BW = ( gm_1 * gm_2 * N_3 * gm_3) / ( gds_1 * gds_2 * ( N_3 * ( Cds_3
                + Cgd_3 ) + CL ) )
455
```

```
456     elif ramp == "Critically Damped":
457         A_BW = ( N_1 * gm_1 * N_2 * gm_2 * N_3 * gm_3) / ( ( ( N_1 + 1 ) *
                gds_1 + gm_1 ) * ( ( N_2 + 1 ) * gds_2 + gm_2 ) * ( N_3 * ( Cds_3 +
                 Cgd_3 ) + CL ) )
458
459     # Convert ringamp gain bandwidth product to Hz
460     GBW = A_BW / ( 2 * np.pi )
461
462     return GBW
463
464 ### Settling Time
465 def ramp_settling_time(ϵ, β, GBW):
466     """
467     Calculates the settling time for both ring amplifiers
468
469     Parameters:
470     - GBW (float): Gain bandwidth product
471     - ϵ (float): Error
472     - β (float): Feedback factor
473
474     Returns:
475      - ts (float): Settling time
476     """
477     # Calculates ringamp settling time in s
478     ts = -np.log(ϵ) / (β * GBW * 2 * np.pi)
479
480     return ts
481
482 ### Slew Rate
483 def ramp_slew_rate(Id_3, CL):
484     """
485     Calculates the settling time for both ring amplifiers
486
487     Parameters:
488     - Id_3 (float): Drain current stage 3
489     - CL (float): Load capacitance
490
491     Returns:
492      - SR (float): Slew rate
493     """
494     # Calculates ringamp slew rate in V/s
495     SR = Id_3 / CL
496
497     return SR
498
499 ### Output Voltage Swing
500 def ramp_output_voltage_swing(VDD, Vdsat_p_3, Vdsat_n_3):
501     """
502     Calculates the output voltage swing for both ring amplifiers
503
```

```
504     Parameters:
505     - VDD (float): Supply voltage
506     - Vdsat_p_3 (float): Saturation voltage PMOS stage 3
507     - Vdsat_n_3 (float): Saturation voltage NMOS stage 3
508
509     Returns:
510      - Vos (float): Output voltage swing
511     """
512     Vos = VDD + Vdsat_p_3 - Vdsat_n_3
513
514     return Vos
515
516 ### Dissipated Power
517 def ramp_dissipated_power(ramp, N_1, Id_1, N_2, Id_2, N_3, Id_3, VDD):
518     """
519     Calculates the dissipated power for the specified ring amplifier
520
521     Parameters:
522     - ramp (str): Ringamp
523     - N_1 (int), Id_1 (float): Parameters stage 1
524     - N_2 (int), Id_2 (float): Parameters stage 2
525     - N_3 (int), Id_3 (float): Parameters stage 3
526     - VDD (float): Voltage supply
527
528     Returns:
529      - Pd (float): Dissipated power
530     """
531     # Calculates inverters dissipated power for the selected ringamp
532     if ramp == "Conventional":
533         Pd_1 = VDD * N_1 * Id_1
534         Pd_2 = VDD * N_2 * Id_2
535
536     elif ramp == "Critically Damped":
537         Pd_1 = VDD * ( N_1 + 1 ) * Id_1
538         Pd_2 = VDD * ( N_2 + 1 ) * Id_2
539
540     Pd_3 = VDD * N_3 * Id_3
541
542     # Calculates ringamp dissipated power
543     Pd = Pd_1 + Pd_2 + Pd_3
544
545     return Pd
546
547 ### Input-Referred Noise
548 def ramp_input_referred_noise(ramp, N_1, gds_1, gm_1, N_2, gds_2, gm_2, N_3,
        gds_3, gm_3, T, kB, γ, Δf):
549     """
550     Calculates the input referred noise for both ring amplifiers
551     Parameters:
552     - N_1, gm_1 (float): Parameters stage 1
```

```
553     - T (float): Temperature
554     - kB (float): Boltzmann constant
555     - γ (float): Channel length coefficiet
556     - Δf (float): Band
557 
558     Returns:
559      - S_in (float): Input-referred noise
560     """
561     # Calculates inverters input-referred noise for the selected ringamp
562     if ramp == "Conventional":
563         Vnin_1 = 4 * kB * T * γ * ( N_1 * gds_1 ** 2 ) / ( gm_1 ** 3 )
564         Vnin_2 = 4 * kB * T * γ * ( N_2 * gds_1 ** 2 * gds_2 ** 2 ) / ( gm_1
                ** 2 * gm_2 ** 3 )
565         Vnin_3 = 4 * kB * T * γ * ( N_3 * gds_1 ** 2 * gds_2 **2 * gds_3 ** 2
                ) / ( gm_1 ** 2 * gm_2 ** 2 * gm_3 ** 3 )
566 
567     elif ramp == "Critically Damped":
568         Vnin_1 = 4 * kB * T * γ * ( ( N_1 + 1 ) * ( ( N_1 + 1 ) * gds_1 + gm_1
                 ) ** 2 ) / ( N_1 ** 2 * gm_1 ** 3 )
569         Vnin_2 = 4 * kB * T * γ * ( ( ( N_1 + 1 ) * gds_1 + gm_1 ) ** 2 * (
                N_2 + 1 ) * ( ( N_2 + 1 ) * gds_2 + gm_2 ) ** 2 ) / ( N_1 ** 2 *
                gm_1 ** 2 * N_2 ** 2 * gm_2 ** 3 )
570         Vnin_3 = 4 * kB * T * γ * ( ( ( N_1 + 1 ) * gds_1 + gm_1 ) ** 2 * ( (
                N_2 + 1 ) * gds_2 + gm_2 ) ** 2 * N_3 * gds_3 ** 2) / ( N_1 ** 2 *
                gm_1 ** 2 * N_2 ** 2 * gm_2 ** 2 * gm_3 ** 3 )
571 
572     # Calculates ringamp input-referred noise in root spectral density
573     Sin =  np.sqrt( ( Vnin_1 + Vnin_2 + Vnin_3 ) * Δf )
574 
575     return Sin
576 
577 ### Phase Margin
578 def ramp_phase_margin(ramp, freqs, N_1, Cds_1, Cgd_1, Cgs_1, gds_1, gm_1, N_2,
        Cds_2, Cgd_2, Cgs_2, gds_2, gm_2, N_3, Cds_3, Cgd_3, Cgs_3, gds_3, gm_3,
        CL):
579     """
580     Calculates the phase margin for the specified ring amplifier
581 
582     Parameters:
583     - ramp (str): Ringamp
584     - freqs (list of float): Frequencies at which to evaluate the transfer
            function
585     - N_1, Cds_1, Cgd_1, Cgs_1, gds_1, gm_1 (float): Parameters stage 1
586     - N_2, Cds_2, Cgd_2, Cgs_2, gds_2, gm_2 (float): Parameters stage 2
587     - N_3, Cds_3, Cgd_3, Cgs_3, gds_3, gm_3 (float): Parameters stage 3
588     - CL (float): Load capacitance
589 
590     Returns:
591     - phase_margin (float): Phase margin
592     """
```

```
593     # Define complex frequency
594     s = 1j * 2 * np.pi * freqs
595 
596     # Calculate the transfer function over the given frequencies
597     H = ramp_transfer_function(ramp, s, N_1, Cds_1, Cgd_1, Cgs_1, gds_1, gm_1,
            N_2, Cds_2, Cgd_2, Cgs_2, gds_2, gm_2, N_3, Cds_3, Cgd_3, Cgs_3, gds_3
            , gm_3, CL)
598 
599     # Module transfer function
600     mod_H = np.abs(H)
601 
602     # Find the gain crossover frequency (where |H(jω)| = 1)
603     crossover_index = np.argmin(np.abs(mod_H - 1))
604 
605     # Calculate the phase at the gain crossover frequency
606     phase_H = np.angle(H[crossover_index], deg=True)
607 
608     # Phase margin is 0 + phase at the gain crossover frequency
609     phase_margin = 0 + phase_H
610 
611     return phase_margin
612 
613 ### Calculate Ramp Perfomance Open Loop
614 def calc_ramp_performance_ol(ramp, N_1, Id_1, gm_1, gds_1, N_2, Id_2, gm_2,
        gds_2, N_3, Id_3, Cds_3, Cgd_3, gm_3, gds_3, Vdsat_p_3, Vdsat_n_3, VDD, CL,
        T, kB, γ, ϵ, β, Δf):
615     """
616     Calculate the specified ring amplifier performance in open loop
617 
618     Parameters:
619     - ramp (str): Ringamp
620     - N_1, Cds_1, Cgd_1, gds_1, gm_1 (float): Parameters stage 1
621     - N_2, Cds_2, Cgd_2, Cgs_2, gds_2, gm_2 (float): Parameters stage 2
622     - N_3, Cds_3, Cgd_3, Cgs_3, gds_3, gm_3 (float): Parameters stage 3
623     - VDD (float): Voltage supply
624     - CL (float): Load capacitance
625     - T (float): Temperature
626     - kB (float): Boltzmann constant
627     - γ (float): Channel length coefficiet
628     - ϵ (float): Error
629     - β (float): Feedback factor
630     - Δf (float): Band
631 
632     Returns:
633     - A (float): Gain
634     - BW (float): Bandwidth
635     - GBW (float): Gain bandwidth product
636     - ts (float): Settling time
637     - SR (float): Slew rate
638     - Pd (float): Dissipated power
```

```
639     - Sin (float): Input-referred noise
640     """
641     # Determine the DC gain (dB)
642     A = ramp_gain(ramp, N_1, gds_1, gm_1, N_2, gds_2, gm_2, gds_3, gm_3)
643 
644     # Determine the bandwidth (Hz)
645     BW = ramp_bandwidth(N_3, Cds_3, Cgd_3, gds_3, CL)
646 
647     # Determine the gain bandwidth product (Hz)
648     GBW = ramp_gain_bandwidth_product(ramp, N_1, gds_1, gm_1, N_2, gds_2, gm_2
            , N_3, Cds_3, Cgd_3, gm_3, CL)
649 
650     # Determine the settling time (s)
651     ts = ramp_settling_time(ϵ, β, GBW)
652 
653     # Determine the output voltage swing (V)
654     Vos = ramp_output_voltage_swing(VDD, Vdsat_p_3, Vdsat_n_3)
655 
656     # Determine the slew rate (V/S)
657     SR = ramp_slew_rate(Id_3, CL)
658 
659     # Determine the dissipated power (W)
660     Pd = ramp_dissipated_power(ramp, N_1, Id_1, N_2, Id_2, N_3, Id_3, VDD)
661 
662     # Determine the input-refrerred noise (V/√Hz)
663     Sin = ramp_input_referred_noise(ramp, N_1, gds_1, gm_1, N_2, gds_2, gm_2,
            N_3, gds_3, gm_3, T, kB, γ, Δf)
664 
665     return A, BW, GBW, ts, Vos, SR, Pd, Sin
666 
667 ### Print Ramp Perfomance Open Loop
668 def print_ramp_performances(ramp_1, A_1, BW_1, GBW_1, ts_1, Vos_1, SR_1, Pd_1,
        Sin_1, PM_1, ramp_2, A_2, BW_2, GBW_2, ts_2, Vos_2, SR_2, Pd_2, Sin_2,
       PM_2):
669     """
670     Print the performance of two specified ring amplifiers side by side in a
            table format
671 
672     Parameters:
673     - ramp_1 (str): Ringamp 1
674     - A_1, BW_1, GBW_1, ts_1, Vos_1, SR_1, Pd_1, Sin_1 (float): Figures of
            merit ramp 1
675     - ramp_2 (str): Ringamp 2
676     - A_2, BW_2, GBW_2, ts_2, Vos_2, SR_2, Pd_2, Sin_2 (float): Figures of
            merit ramp 2
677     """
678     # Define headers
679     headers = ["", ramp_1, ramp_2]
680 
681     # Define data
```

```
682     data = [
683         ["A (dB)", f"{A_1:.3f}", f"{A_2:.3f}"],
684         ["BW (MHz)", f"{BW_1 * 1e-6:.3f}", f"{BW_2 * 1e-6:.3f}"],
685         ["GBW (MHz)", f"{GBW_1 * 1e-6:.3f}", f"{GBW_2 * 1e-6:.3f}"],
686         ["ts (ps)", f"{ts_1 * 1e12:.3f}", f"{ts_2 * 1e12:.3f}"],
687         ["SR (V/μs)", f"{SR_1 * 1e-6:.3f}", f"{SR_2 * 1e-6:.3f}"],
688         ["Vos (mV)", f"{Vos_1 * 1e3:.3f}", f"{Vos_2 * 1e3:.3f}"],
689         ["Pd (mW)", f"{Pd_1 * 1e3:.3f}", f"{Pd_2 * 1e3:.3f}"],
690         ["Sin (μV/√Hz)", f"{Sin_1 * 1e6 :.3f}", f"{Sin_2 * 1e6:.3f}"],
691         ["PM (°)", f"{PM_1:.3f}", f"{PM_2:.3f}"],
692     ]
693 
694     # Create table
695     table = tabulate(data, headers, tablefmt="grid")
696 
697     # Print table
698     print(table)
699 
700 ### Calculate Stage Perfomance Open Loop
701 def calc_stage_perfomance_ol(ramp, stage, N_1, Id_1, Cds_1, Cgd_1, Cgs_1,
        gds_1, gm_1, N_2, Id_2, Cds_2, Cgd_2, Cgs_2, gds_2, gm_2, N_3, Id_3, Cds_3,
         Cgd_3, Cgs_3, gds_3, gm_3, VDD, CL, kB, T, γ, Δf):
702     """
703     Calculates the gain magnitude for the specified ring amplifier stage
704 
705     Parameters:
706     - ramp (str): Ringamp
707     - stage (str): Stage
708     - N_1, Id_1, Cds_1, Cgd_1, Cgs_1, gds_1, gm_1 (float): Parameters stage 1
709     - N_2, Id_2, Cds_2, Cgd_2, Cgs_2, gds_2, gm_2 (float): Parameters stage 2
710     - N_3, Id_3, Cds_3, Cgd_3, Cgs_3, gds_3, gm_3 (float): Parameters stage 3
711     - VDD (float): Supply voltage
712     - CL (float): Load capacitance
713     - kB (float): Boltzmann constant
714     - T (float): Temperature
715     - γ (float): Length coefficient
716     - Δf (float): Band
717 
718     Returns:
719      - A (float): Gain
720      - P_d (float): Dissipated power
721      - S_in (foat): Input referred noise
722      - f_p (float): Frequency pole
723      - f_z (float): Frequency zero
724     """
725     # By selecting stage and ringamp calculate gain, dissipated power, input-
            referred noise pole and zero frequencies
726     if stage == '1':
727 
728         if ramp == "Conventional":
```

```
729             G = - gm_1 / gds_1
730             Pd = VDD * N_1 * Id_1
731             Vnin = 4 * kB * T * γ * ( N_1 * gds_1 ** 2 ) / ( gm_1 ** 3 )
732             sp = - ( N_1 * gds_1) / ( N_1 * ( Cds_1 + Cgd_1) + N_2 * ( Cgd_2 +
                    Cgs_2 ) )
733
734         elif ramp == "Critically Damped":
735             G = - ( N_1 * gm_1 ) / ( ( N_1 + 1 ) * gds_1 + gm_1 )
736             Pd = VDD * ( N_1 + 1 ) * Id_1
737             Vnin = 4 * kB * T * γ * ( ( N_1 + 1 ) * ( ( N_1 + 1 ) * gds_1 +
                    gm_1 ) ** 2 ) / ( N_1 ** 2 * gm_1 ** 3 )
738             sp = - ( ( N_1 + 1 ) * gds_1 + gm_1 ) / ( N_1 * ( Cds_1 + Cgd_1) +
                    Cds_1 + Cgs_1 + N_2 * ( Cgd_2 + Cgs_2 ) )
739
740         sz = gm_1 / Cgd_1
741
742     elif stage == '2':
743
744         if ramp == "Conventional":
745             G = - gm_2 / gds_2
746             Pd = VDD * N_2 * Id_2
747             Vnin = 4 * kB * T * γ * ( N_2 * gds_1 ** 2 * gds_2 ** 2 ) / ( gm_1
                    ** 2 * gm_2 ** 3 )
748             sp = - ( N_2 * gds_2) / ( N_2 * ( Cds_2 + Cgd_2) + N_3 * ( Cgd_3 +
                    Cgs_3 ) )
749
750         elif ramp == "Critically Damped":
751             G = - ( N_2 * gm_2 ) / ( ( N_2 + 1 ) * gds_2 + gm_2 )
752             Pd = VDD * ( N_2 + 1 ) * Id_2
753             Vnin = 4 * kB * T * γ * ( ( ( N_1 + 1 ) * gds_1 + gm_1 ) ** 2 * (
                    N_2 + 1 ) * ( ( N_2 + 1 ) * gds_2 + gm_2 ) ** 2 ) / ( N_1 ** 2
                    * gm_1 ** 2 * N_2 ** 2 * gm_2 ** 3 )
754             sp = - ( ( N_2 + 1 ) * gds_2 + gm_2 ) / ( N_2 * ( Cds_2 + Cgd_2) +
                    Cds_2 + Cgs_2 + N_3 * ( Cgd_3 + Cgs_3 ) )
755
756         sz = gm_2 / Cgd_2
757
758     elif stage == '3':
759
760         G = - gm_3 / gds_3
761         Pd = VDD * N_3 * Id_3
762         sp = - ( N_3 * gds_3) / ( N_3 * ( Cds_3 + Cgd_3 ) + CL )
763         sz = gm_3 / Cgd_3
764
765         if ramp == "Conventional":
766             Vnin = 4 * kB * T * γ * ( N_3 * gds_1 ** 2 * gds_2 **2 * gds_3 **
                    2 ) / ( gm_1 ** 2 * gm_2 ** 2 * gm_3 ** 3 )
767
768         elif ramp == "Critically Damped":
```

```
769             Vnin = 4 * kB * T * γ * ( ( ( N_1 + 1 ) * gds_1 + gm_1 ) ** 2 * (
                    ( N_2 + 1 ) * gds_2 + gm_2 ) ** 2 * N_3 * gds_3 ** 2) / ( N_1
                    ** 2 * gm_1 ** 2 * N_2 ** 2 * gm_2 ** 2 * gm_3 ** 3 )
770
771     # Calculate stage gain in dB
772     A = 20 * np.log10( np.abs( G ) )
773
774     # Calculate stage input-referred noise in V/√Hz
775     Sin = np.sqrt( Vnin * Δf  )
776
777     # Calculate stage pole in Hz
778     fp = np.abs( sp / ( 2 * np.pi ) )
779
780     # Calculate stage zeri in Hz
781     fz = np.abs( sz / ( 2 * np.pi ) )
782
783     return A, Pd, Sin, fp, fz
784
785 ### Print Stage Perfomance Open Loop
786 def print_stages_performances_ol(ramp_1, ramp_2, A1_ramp_1, Pd1_ramp_1,
        Sin1_ramp_1, fp1_ramp_1, fz1_ramp_1, A1_ramp_2, Pd1_ramp_2, Sin1_ramp_2,
        fp1_ramp_2, fz1_ramp_2, A2_ramp_1, Pd2_ramp_1, Sin2_ramp_1, fp2_ramp_1,
        fz2_ramp_1, A2_ramp_2, Pd2_ramp_2, Sin2_ramp_2, fp2_ramp_2, fz2_ramp_2,
        A3_ramp_1, Pd3_ramp_1, Sin3_ramp_1, fp3_ramp_1, fz3_ramp_1, A3_ramp_2,
        Pd3_ramp_2, Sin3_ramp_2, fp3_ramp_2, fz3_ramp_2):
787     """
788     Print stages perfomance in open loop
789
790     Parameters:
791         ramp_1 (str): Ringamp 1
792         ramp_2 (str): Ringamp 2
793         A1_ramp_1, Pd1_ramp_1, Sin1_ramp_1, fp1_ramp_1, fz1_ramp_1 (float):
                Parameters stage 1 ramp 1
794         A1_ramp_2, Pd1_ramp_2, Sin1_ramp_2, fp1_ramp_2, fz1_ramp_2 (float):
                Parameters stage 1 ramp 2
795         A2_ramp_1, Pd2_ramp_1, Sin2_ramp_1, fp2_ramp_1, fz2_ramp_1 (float):
                Parameters stage 2 ramp 1
796         A2_ramp_2, Pd2_ramp_2, Sin2_ramp_2, fp2_ramp_2, fz2_ramp_2 (float):
                Parameters stage 2 ramp 2
797         A3_ramp_1, Pd3_ramp_1, Sin3_ramp_1, fp3_ramp_1, fz3_ramp_1 (float):
                Parameters stage 3 ramp 1
798         A3_ramp_2, Pd3_ramp_2, Sin3_ramp_2, fp3_ramp_2, fz3_ramp_2 (float):
                Parameters stage 3 ramp 2
799
800     """
801     # Define header
802     headers = ["", f"{ramp_1}", f"{ramp_2}", "", f"{ramp_1}", f"{ramp_2}", "",
            f"{ramp_1}", f"{ramp_2}"]
803
804     # Define data
```

```
805     data = [
806         ["A$_1$ (dB)", f"{A1_ramp_1:.3f}", f"{A1_ramp_2:.3f}", "A$_2$ (dB)", f"{
                A2_ramp_1:.3f}", f"{A2_ramp_2:.3f}", "A$_3$ (dB)", f"{A3_ramp_1:.3f}",
                 f"{A3_ramp_2:.3f}"],
807         ["Pd$_1$ ($\mu$W)", f"{Pd1_ramp_1 * 1e6:.3f}", f"{Pd1_ramp_2 * 1e6:.3f}", "Pd
                $_2$ (mW)", f"{Pd2_ramp_1 * 1e6:.3f}", f"{Pd2_ramp_2 * 1e6:.3f}", "Pd$_3$
                 (mW)", f"{Pd3_ramp_1 * 1e6:.3f}", f"{Pd3_ramp_2 * 1e6:.3f}"],
808         ["Sin$_1$ ($\mu$V/$\sqrt{\text{Hz}}$)", f"{Sin1_ramp_1 * 1e6:.3f}", f"{Sin1_ramp_2 * 1e6:.3f
                }", "Sin$_2$ ($\mu$V/$\sqrt{\text{Hz}}$)", f"{Sin2_ramp_1 * 1e6:.3f}", f"{Sin2_ramp_2 * 1
                e6:.3f}", "Sin$_3$ ($\mu$V/$\sqrt{\text{Hz}}$)", f"{Sin3_ramp_1 * 1e6:.3f}", f"{
                Sin3_ramp_2 * 1e6:.3f}"],
809         ["fp$_1$ (GHz)", f"{fp1_ramp_1 * 1e-9:.3f}", f"{fp1_ramp_2 * 1e-9:.3f}",
                "fp$_2$ (GHz)", f"{fp2_ramp_1 * 1e-9:.3f}", f"{fp2_ramp_2 * 1e-9:.3f}"
                , "fp$_3$ (MHz)", f"{fp3_ramp_1 * 1e-6:.3f}", f"{fp3_ramp_2 * 1e-6:.3f
                }"],
810         ["fz$_1$ (GHz)", f"{fz1_ramp_1 * 1e-9:.3f}", f"{fz1_ramp_2 * 1e-9:.3f}",
                "fz$_2$ (GHz)", f"{fz2_ramp_1 * 1e-9:.3f}", f"{fz2_ramp_2 * 1e-9:.3f}"
                , "fz$_3$ (GHz)", f"{fz3_ramp_1 * 1e-9:.3f}", f"{fz3_ramp_2 * 1e-9:.3f
                }"]
811     ]
812 
813     # Create table
814     table = tabulate(data, headers, tablefmt="grid")
815 
816     # Print table
817     print(table)
818 
819 ### Calculate Target Poles
820 def calc_target_poles(fp1, fp2, GBW):
821     """
822     Calculates the ratio between frequencies of poles 1 and 2 and gain
            bandwidth product
823 
824     Parameters:
825     - fp1 (float): Frequency pole 1
826     - fp2 (float): Frequency pole 2
827     - GBW (float): Gain bandwidth product
828 
829     Returns:
830     - fp1_gbw (float): Ratio between frequency pole 1 and gain bandwidth
            product
831     - fp2_gbw (float): Ratio between frequency pole 2 and gain bandwidth
            product
832     """
833 
834     fp1_gbw = fp1 / GBW
835 
836     fp2_gbw = fp2 / GBW
837 
838     return fp1_gbw, fp2_gbw
```

```
839
840 ### Print Target Poles
841 def print_target_poles(ramp_1, fp1_gbw_1, fp2_gbw_1, ramp_2, fp1_gbw_2,
        fp2_gbw_2):
842     """
843     Prints the ratio between frequencies of poles 1 and 2 and gain bandwidth
            product
844
845     Parameters:
846         ramp_1 (str): Ringamp 1
847         fp1_gbw_1 (float): Ratio between frequency pole 1 and gain bandwidth
                product for ramp 1
848         fp2_gbw_1 (float): Ratio between frequency pole 2 and gain bandwidth
                product for ramp 1
849         ramp_2 (str): Ringamp 2
850         fp1_gbw_2 (float): Ratio between frequency pole 1 and gain bandwidth
                product for ramp 2
851         fp2_gbw_2 (float): Ratio between frequency pole 2 and gain bandwidth
                product for ramp 2
852     """
853     # Define headers
854     headers = ["", ramp_1, ramp_2]
855
856     # Define data
857     data = [
858         ["fp1/gbw", f"{fp1_gbw_1:.3f}", f"{fp1_gbw_2:.3f}"],
859         ["fp2/gbw", f"{fp2_gbw_1:.3f}", f"{fp2_gbw_2:.3f}"]
860     ]
861
862     # Create table
863     table = tabulate(data, headers, tablefmt="grid")
864
865     # Print table
866     print(table)
867
868 ### Calculate Sample Frequency
869 def calc_sample_freq(D, ts):
870     """
871     Calculates sample frequency
872
873     Parameters:
874     - D (float): Duty cycle
875     - ts (float): Settling time
876
877     Returns:
878     - fs (float): Sample frequency
879     """
880     fs = D / ts
881
882     return fs
```

```
883
884 ### Print Sample Frequency
885 def print_sample_freq(ramp_1, fs_1, ramp_2, fs_2):
886     """
887     Prints the ratio between frequencies of poles 1 and 2 and gain bandwidth
            product
888
889     Parameters:
890         ramp_1 (str): Ringamp 1
891         fs_1 (float): Sample frequency ringamp 1
892         ramp_2 (str): Ringamp 2
893         fs_2 (float): Sample frequency ringamp 2
894     """
895     # Define headers
896     headers = ["", ramp_1, ramp_2]
897
898     # Define data
899     data = [
900         ["fclk (MHz)", f"{fs_1 * 1e-6:.1f}", f"{fs_2 * 1e-6:.1f}"]
901     ]
902
903     # Create table
904     table = tabulate(data, headers, tablefmt="grid")
905
906     # Print table
907     print(table)
908
909 ######################################################
910
911 ## Ring Amplifier Frequency Response Open Loop
912
913 ### Calculate Frequency Response Open Loop
914 def calc_ramp_frequency_response_ol(ramp, freqs, N_1, Cds_1, Cgd_1, Cgs_1,
        gds_1, gm_1, N_2, Cds_2, Cgd_2, Cgs_2, gds_2, gm_2, N_3, Cds_3, Cgd_3,
        Cgs_3, gds_3, gm_3, CL):
915     """
916     Calculate the specified ring amplifier frequency response in open loop
917
918     Parameters:
919     - ramp (str): Ringamp
920     - freqs (float): Frequencies
921     - N_1, Cds_1, Cgd_1, Cgs_1, gds_1, gm_1 (float): Parameters stage 1
922     - N_2, Cds_2, Cgd_2, Cgs_2, gds_2, gm_2 (float): Parameters stage 2
923     - N_3, Cds_3, Cgd_3, Cgs_3, gds_3, gm_3 (float): Parameters stage 3
924     - CL (float): Load capacitance
925
926     Returns:
927     - mag: Magnitude
928     - phase: Phase
929     - freq: Frequency
```

```
930     """
931     # Define complex frequency
932     s = 1j * 2 * np.pi * freqs
933 
934     # Calculate transfer function
935     H = ramp_transfer_function(ramp, s, N_1, Cds_1, Cgd_1, Cgs_1, gds_1, gm_1,
            N_2, Cds_2, Cgd_2, Cgs_2, gds_2, gm_2, N_3, Cds_3, Cgd_3, Cgs_3, gds_3
            , gm_3, CL)
936 
937     # Calculate transfer function magnitude in dB
938     mag = 20 * np.log10(np.abs(H))
939 
940     # Calculate transfer function phase in °
941     phase = np.angle(H, deg=True)
942 
943     # Unwrap the phase in rad
944     unwrapped_phase = np.unwrap(np.deg2rad(phase))
945 
946     # Convert the phase back to °
947     unwrapped_phase = np.rad2deg(unwrapped_phase)
948 
949     # Map the phase
950     mapped_unwrapped_phase = np.where(unwrapped_phase > 360, unwrapped_phase -
            720, unwrapped_phase)
951     mapped_unwrapped_phase = np.where(mapped_unwrapped_phase < -360,
            mapped_unwrapped_phase + 720, mapped_unwrapped_phase)
952 
953     return mag, mapped_unwrapped_phase, freqs
954 
955 # Print Frequency Response Open Loop
956 def plot_frequency_responses_ol(ramp_1, mag_1, phase_1, freq_1, ramp_2=None,
        mag_2=None, phase_2=None, freq_2=None):
957     """
958     Plot the frequency response in open loop for one or two ramps
959 
960     Parameters:
961     - ramp_1 (str): Ringamp 1
962     - mag_1, phase_1, freq_1 (list of float): Magnitude, phase and frequency
            for ringamp 1
963     - ramp_2 (str, optional): Ringamp 2
964     - mag_2, phase_2, freq_2 (list of float, optional): Magnitude, phase and
            frequency for ringamp 2
965     """
966     plt.figure(figsize=(8, 6))
967 
968     # Plot magnitude response
969     plt.subplot(2, 1, 1)
970     plt.semilogx(freq_1, mag_1, label=ramp_1)
971     if mag_2 is not None and freq_2 is not None:
972         plt.semilogx(freq_2, mag_2, label=ramp_2)
```

```
973     plt.title('Magnitude Response', fontsize=12)
974     plt.xlabel('$f$ $(Hz)$')
975     plt.ylabel('$|H(s)|$ $(dB)$')
976     plt.grid(True)
977     plt.xticks([10**i for i in range(-1, 16)])
978     plt.yticks(range(-100, 100, 20))
979     plt.xlim(1e0, 1e15)
980     plt.ylim(-100, 80)
981     plt.legend()
982 
983     # Plot phase response
984     plt.subplot(2, 1, 2)
985     plt.semilogx(freq_1, phase_1, label=ramp_1)
986     if phase_2 is not None and freq_2 is not None:
987         plt.semilogx(freq_2, phase_2, label=ramp_2)
988     plt.title('Phase Response', fontsize=12)
989     plt.xlabel('$f$ $(Hz)$')
990     plt.ylabel('$∠H(s)$ $(°)$')
991     plt.grid(True)
992     plt.xticks([10**i for i in range(-1, 16)])
993     plt.yticks(range(-450, 270, 45))
994     plt.xlim(1e0, 1e15)
995     plt.ylim(-405, 225)
996     plt.legend()
997 
998     # Adjust subplot layout to prevent overlap
999     plt.tight_layout()
1000 
1001    # Display the plot
1002    plt.show()
1003 
1004 # Compare Frequency Response Open Loop
1005 def comp_frequency_responses_ol(mag, phase, freq, design):
1006     """
1007     Plot the theoretical and practical frequency responses in open loop
1008 
1009     Parameters:
1010     - mag, phase, freq (list of float): Magnitude, phase, and frequency for
            the theoretical frequency response
1011     - design (str): File path containing the practical frequency response
1012     """
1013     # Read the practical data from the CSV file using column indexes
1014     practical_data = pd.read_csv(design, usecols=[0, 1, 2], header=None)
1015     freq_practical = practical_data[0]
1016     mag_practical = practical_data[1]
1017     phase_practical = practical_data[2]
1018 
1019     plt.figure(figsize=(8, 6))
1020 
1021     # Plot magnitude response
```

```
    plt.subplot(2, 1, 1)
    plt.semilogx(freq, mag, label='Model')
    plt.semilogx(freq_practical, mag_practical, label='Simulation')
    plt.title('Magnitude Response', fontsize=12)
    plt.xlabel('$f$ $(Hz)$')
    plt.ylabel('$|H(s)|$ $(dB)$')
    plt.grid(True)
    plt.xticks([10**i for i in range(-1, 16)])
    plt.yticks(range(-120, 100, 20))
    plt.xlim(1e0, 1e15)
    plt.ylim(-120, 80)
    plt.legend()

    # Plot phase response
    plt.subplot(2, 1, 2)
    plt.semilogx(freq, phase, label='Model')
    plt.semilogx(freq_practical, phase_practical, label='Simulation')
    plt.title('Phase Response', fontsize=12)
    plt.xlabel('$f$ $(Hz)$')
    plt.ylabel('$∠H(s)$ $(°)$')
    plt.grid(True)
    plt.xticks([10**i for i in range(-1, 16)])
    plt.yticks(range(-450, 270, 45))
    plt.xlim(1e0, 1e15)
    plt.ylim(-405, 225)
    plt.legend()

    # Adjust subplot layout to prevent overlap
    plt.tight_layout()

    # Display the plot
    plt.show()

# Compare Frequency Respons Open Loop
def comp_freq_resps_ol(designs, mag_theoretical_A, phase_theoretical_A,
    freq_theoretical_A, mag_theoretical_GBW, phase_theoretical_GBW,
    freq_theoretical_GBW):
    """
    Plot the theoretical and practical frequency responses in open loop for
        both designs

    Parameters:
    - designs: file containing practical designs
    - mag_theoretical_A, phase_theoretical_A, freq_theoretical_A (list of
        float): Magnitude, phase, and frequency for the theoretical frequency
        response A desgn
    - mag_theoretical_GBW, phase_theoretical_GBE, freq_theoretical_GBE (list
        of float): Magnitude, phase, and frequency for the theoretical
        frequency response GBW desgn
    """
```

```
1065    # Read the practical data from the CSV file using column indexes
1066    practical_data_A = pd.read_csv(designs['A'], usecols=[0, 1, 2], header=
            None)
1067    freq_practical_A = practical_data_A[0]
1068    mag_practical_A = practical_data_A[1]
1069    phase_practical_A = practical_data_A[2]
1070
1071    practical_data_GBW = pd.read_csv(designs['GBW'], usecols=[0, 1, 2], header
            =None)
1072    freq_practical_GBW = practical_data_GBW[0]
1073    mag_practical_GBW = practical_data_GBW[1]
1074    phase_practical_GBW = practical_data_GBW[2]
1075
1076    plt.figure(figsize=(8, 6))
1077
1078    # Plot magnitude response
1079    plt.subplot(2, 1, 1)
1080    plt.semilogx(freq_theoretical_A, mag_theoretical_A, label='Model Maximum
            Gain', color='red', linestyle='-')
1081    plt.semilogx(freq_practical_A, mag_practical_A, label='Simulation Maximum
            Gain', color='red', linestyle='--')
1082    plt.semilogx(freq_theoretical_GBW, mag_theoretical_GBW, label='Model
            Maximum Speed', color='blue', linestyle='-')
1083    plt.semilogx(freq_practical_GBW, mag_practical_GBW, label='Simulation
            Maximum Speed', color='blue', linestyle='--')
1084    plt.title('Magnitude Response', fontsize=12)
1085    plt.xlabel('$f$ $(Hz)$')
1086    plt.ylabel('$|H(s)|$ $(dB)$')
1087    plt.grid(True)
1088    plt.xticks([10**i for i in range(-1, 16)])
1089    plt.yticks(range(-120, 100, 20))
1090    plt.xlim(1e0, 1e15)
1091    plt.ylim(-120, 80)
1092    plt.legend()
1093
1094    # Plot phase response
1095    plt.subplot(2, 1, 2)
1096    plt.semilogx(freq_theoretical_A, phase_theoretical_A, label='Model Maximum
             Gain', color='red', linestyle='-')
1097    plt.semilogx(freq_practical_A, phase_practical_A, label='Simulation
            Maximum Gain', color='red', linestyle='--')
1098    plt.semilogx(freq_theoretical_GBW, phase_theoretical_GBW, label='Model
            Maximum Speed', color='blue', linestyle='-')
1099    plt.semilogx(freq_practical_GBW, phase_practical_GBW, label='Simulation
            Maximum Speed', color='blue', linestyle='--')
1100    plt.title('Phase Response', fontsize=12)
1101    plt.xlabel('$f$ $(Hz)$')
1102    plt.ylabel('$∠H(s)$ $(^\circ)$')
1103    plt.grid(True)
1104    plt.xticks([10**i for i in range(-1, 16)])
```

```
    plt.yticks(range(-450, 270, 45))
    plt.xlim(1e0, 1e15)
    plt.ylim(-405, 225)
    plt.legend()

    # Adjust subplot layout to prevent overlap
    plt.tight_layout()

    # Display the plot
    plt.show()

##################################################

## Ring Amplifier Parameters

### Print Ramp Parameters
def print_ramp_params(N_1, kW_1, KS_1, N_2, kW_2, KS_2, N_3, kW_3, KS_3):
    """
    Print ring amplifier cmos specifications

    Parameters:
    - N_1, kW_1, KS_1 (float): Parameters stage 1
    - N_2, kW_2, KS_2 (float): Parameters stage 2
    - N_3, kW_3, KS_3 (float): Parameters stage 3
    """
    # Create a list of dictionaries to store the parameters data
    ramp_params = [
        {"Stage": "1", "N": N_1, "kW": kW_1, "KS": KS_1},
        {"Stage": "2", "N": N_2, "kW": kW_2, "KS": KS_2},
        {"Stage": "3", "N": N_3, "kW": kW_3, "KS": KS_3},
    ]

    # Print the table header for parameters
    print('Ramp Parameters\n')
    print(f"{'Stage':<10} {'N':<10} {'kW':<10} {'KS':<10}")
    print("-" * 36)
    for row in ramp_params:
        print(f"{row['Stage']:<10} {row['N']:<10} {row['kW']:<10} {row['KS
            ']:<10.1f}")
    print("-" * 36)

### Print Transistor Parameters
```

```
1146 def print_trans_params(L_1, Vgs_n_1, Vth_n_1, Vds_n_1, Vdsat_n_1, Id_n_1,
     Cds_n_1, Cgd_n_1, Cgs_n_1, gds_n_1, gm_n_1, W_n_1, Id_n_1_m, Cds_n_1_m,
     Cgd_n_1_m, Cgs_n_1_m, gds_n_1_m, gm_n_1_m, W_n_1_m, Vgs_p_1, Vth_p_1,
     Vds_p_1, Vdsat_p_1, Id_p_1, Cds_p_1, Cgd_p_1, Cgs_p_1, gds_p_1, gm_p_1,
     W_p_1, Id_p_1_m, Cds_p_1_m, Cgd_p_1_m, Cgs_p_1_m, gds_p_1_m, gm_p_1_m,
     W_p_1_m, L_2, Vgs_n_2, Vth_n_2, Vds_n_2, Vdsat_n_2, Id_n_2, Cds_n_2,
     Cgd_n_2, Cgs_n_2, gds_n_2, gm_n_2, W_n_2, Id_n_2_m, Cds_n_2_m, Cgd_n_2_m,
     Cgs_n_2_m, gds_n_2_m, gm_n_2_m, W_n_2_m, Vgs_p_2, Vth_p_2, Vds_p_2,
     Vdsat_p_2, Id_p_2, Cds_p_2, Cgd_p_2, Cgs_p_2, gds_p_2, gm_p_2, W_p_2,
     Id_p_2_m, Cds_p_2_m, Cgd_p_2_m, Cgs_p_2_m, gds_p_2_m, gm_p_2_m, W_p_2_m,
     L_3, Vgs_n_3, Vth_n_3, Vds_n_3, Vdsat_n_3, Id_n_3_m, Cds_n_3_m, Cgd_n_3_m,
     Cgs_n_3_m, gds_n_3_m, gm_n_3_m, W_n_3_m, Vgs_p_3, Vth_p_3, Vds_p_3,
     Vdsat_p_3, Id_p_3_m, Cds_p_3_m, Cgd_p_3_m, Cgs_p_3_m, gds_p_3_m, gm_p_3_m,
     W_p_3_m ):
1147     """
1148     Print transistors parameters
1149
1150     Parameters:
1151     - L_1, Vgs_n_1, Vth_n_1, Vds_n_1, Vdsat_n_1, Id_n_1, Cds_n_1, Cgd_n_1,
           Cgs_n_1, gds_n_1, gm_n_1, W_n_1, Id_n_1_m, Cds_n_1_m, Cgd_n_1_m,
           Cgs_n_1_m, gds_n_1_m, gm_n_1_m, W_n_1_m, Vgs_p_1, Vth_p_1, Vds_p_1,
           Vdsat_p_1,
1152       Id_p_1, Cds_p_1, Cgd_p_1, Cgs_p_1, gds_p_1, gm_p_1, W_p_1, Id_p_1_m,
             Cds_p_1_m, Cgd_p_1_m, Cgs_p_1_m, gds_p_1_m, gm_p_1_m, W_p_1_m (float)
             : Parameters stage 1
1153     - L_2, Vgs_n_2, Vth_n_2, Vds_n_2, Vdsat_n_2, Id_n_2, Cds_n_2, Cgd_n_2,
           Cgs_n_2, gds_n_2, gm_n_2, W_n_2, Id_n_2_m, Cds_n_2_m, Cgd_n_2_m,
           Cgs_n_2_m, gds_n_2_m, gm_n_2_m, W_n_2_m, Vgs_p_2, Vth_p_2, Vds_p_2,
           Vdsat_p_2,
1154       Id_p_2, Cds_p_2, Cgd_p_2, Cgs_p_2, gds_p_2, gm_p_2, W_p_2, Id_p_2_m,
             Cds_p_2_m, Cgd_p_2_m, Cgs_p_2_m, gds_p_2_m, gm_p_2_m, W_p_2_m (float)
             : Parameters stage 2
1155     - L_3, Vgs_n_3, Vth_n_3, Vds_n_3, Vdsat_n_3, Id_n_3_m, Cds_n_3_m,
           Cgd_n_3_m, Cgs_n_3_m, gds_n_3_m, gm_n_3_m, W_n_3_m, Vgs_p_3, Vth_p_3,
           Vds_p_3, Vdsat_p_3,
1156       Id_p_3_m, Cds_p_3_m, Cgd_p_3_m, Cgs_p_3_m, gds_p_3_m, gm_p_3_m, W_p_3_m
             (float): Parameters stage 3
1157     """
1158     # Create a list of dictionaries to store the characteristics data
1159
1160     trans_params = [
1161         {"Transistor": "NM1",  "Id (μA)": Id_n_1_m*1e6, "Vgs (mV)": Vgs_n_1*1
               e3, "Vth (mV)": Vth_n_1*1e3, "Vds (mV)": Vds_n_1*1e3, "Vdsat (mV)":
                Vdsat_n_1*1e3, "Cds (zF)": Cds_n_1_m*1e21, "Cgd (aF)": Cgd_n_1_m*1
               e18, "Cgs (aF)": Cgs_n_1_m*1e18, "gds (μS)": gds_n_1_m*1e6, "gm (μS
               )": gm_n_1_m*1e6, "W (nm)": W_n_1_m*1e9, "L (nm)": L_1},
```

```
1162        {"Transistor": "PM1",  "Id (μA)": Id_p_1_m*1e6, "Vgs (mV)": Vgs_p_1*1
                e3, "Vth (mV)": Vth_p_1*1e3, "Vds (mV)": Vds_p_1*1e3, "Vdsat (mV)":
                 Vdsat_p_1*1e3, "Cds (zF)": Cds_p_1_m*1e21, "Cgd (aF)": Cgd_p_1_m*1
                e18, "Cgs (aF)": Cgs_p_1_m*1e18, "gds (μS)": gds_p_1_m*1e6, "gm (μS
                )": gm_p_1_m*1e6, "W (nm)": W_p_1_m*1e9, "L (nm)": L_1},
1163        {"Transistor": "NM1D", "Id (μA)":   Id_n_1*1e6, "Vgs (mV)": Vgs_n_1*1
                e3, "Vth (mV)": Vth_n_1*1e3, "Vds (mV)": Vds_n_1*1e3, "Vdsat (mV)":
                 Vdsat_n_1*1e3, "Cds (zF)":   Cds_n_1*1e21, "Cgd (aF)":   Cgd_n_1*1
                e18, "Cgs (aF)":   Cgs_n_1*1e18, "gds (μS)":   gds_n_1*1e6, "gm (μS
                )":   gm_n_1*1e6, "W (nm)":   W_n_1*1e9, "L (nm)": L_1},
1164        {"Transistor": "PM1D", "Id (μA)":   Id_p_1*1e6, "Vgs (mV)": Vgs_p_1*1
                e3, "Vth (mV)": Vth_p_1*1e3, "Vds (mV)": Vds_p_1*1e3, "Vdsat (mV)":
                 Vdsat_p_1*1e3, "Cds (zF)":   Cds_p_1*1e21, "Cgd (aF)":   Cgd_p_1*1
                e18, "Cgs (aF)":   Cgs_p_1*1e18, "gds (μS)":   gds_p_1*1e6, "gm (μS
                )":   gm_p_1*1e6, "W (nm)":   W_p_1*1e9, "L (nm)": L_1},
1165        {"Transistor": "NM2",  "Id (μA)": Id_n_2_m*1e6, "Vgs (mV)": Vgs_n_2*1
                e3, "Vth (mV)": Vth_n_2*1e3, "Vds (mV)": Vds_n_2*1e3, "Vdsat (mV)":
                 Vdsat_n_2*1e3, "Cds (zF)": Cds_n_2_m*1e21, "Cgd (aF)": Cgd_n_2_m*1
                e18, "Cgs (aF)": Cgs_n_2_m*1e18, "gds (μS)": gds_n_2_m*1e6, "gm (μS
                )": gm_n_2_m*1e6, "W (nm)": W_n_2_m*1e9, "L (nm)": L_2},
1166        {"Transistor": "PM2",  "Id (μA)": Id_p_2_m*1e6, "Vgs (mV)": Vgs_p_2*1
                e3, "Vth (mV)": Vth_p_2*1e3, "Vds (mV)": Vds_p_2*1e3, "Vdsat (mV)":
                 Vdsat_p_2*1e3, "Cds (zF)": Cds_p_2_m*1e21, "Cgd (aF)": Cgd_p_2_m*1
                e18, "Cgs (aF)": Cgs_p_2_m*1e18, "gds (μS)": gds_p_2_m*1e6, "gm (μS
                )": gm_p_2_m*1e6, "W (nm)": W_p_2_m*1e9, "L (nm)": L_2},
1167        {"Transistor": "NM2D", "Id (μA)":   Id_n_2*1e6, "Vgs (mV)": Vgs_n_2*1
                e3, "Vth (mV)": Vth_n_2*1e3, "Vds (mV)": Vds_n_2*1e3, "Vdsat (mV)":
                 Vdsat_n_2*1e3, "Cds (zF)":   Cds_n_2*1e21, "Cgd (aF)":   Cgd_n_2*1
                e18, "Cgs (aF)":   Cgs_n_2*1e18, "gds (μS)":   gds_n_2*1e6, "gm (μS
                )":   gm_n_2*1e6, "W (nm)":   W_n_2*1e9, "L (nm)": L_2},
1168        {"Transistor": "PM2D", "Id (μA)":   Id_p_2*1e6, "Vgs (mV)": Vgs_p_2*1
                e3, "Vth (mV)": Vth_p_2*1e3, "Vds (mV)": Vds_p_2*1e3, "Vdsat (mV)":
                 Vdsat_p_2*1e3, "Cds (zF)":   Cds_p_2*1e21, "Cgd (aF)":   Cgd_p_2*1
                e18, "Cgs (aF)":   Cgs_p_2*1e18, "gds (μS)":   gds_p_2*1e6, "gm (μS
                )":   gm_p_2*1e6, "W (nm)":   W_p_2*1e9, "L (nm)": L_2},
1169        {"Transistor": "NM3",  "Id (μA)": Id_n_3_m*1e6, "Vgs (mV)": Vgs_n_3*1
                e3, "Vth (mV)": Vth_n_3*1e3, "Vds (mV)": Vds_n_3*1e3, "Vdsat (mV)":
                 Vdsat_n_3*1e3, "Cds (zF)": Cds_n_3_m*1e21, "Cgd (aF)": Cgd_n_3_m*1
                e18, "Cgs (aF)": Cgs_n_3_m*1e18, "gds (μS)": gds_n_3_m*1e6, "gm (μS
                )": gm_n_3_m*1e6, "W (nm)": W_n_3_m*1e9, "L (nm)": L_3},
1170        {"Transistor": "PM3",  "Id (μA)": Id_p_3_m*1e6, "Vgs (mV)": Vgs_p_3*1
                e3, "Vth (mV)": Vth_p_3*1e3, "Vds (mV)": Vds_p_3*1e3, "Vdsat (mV)":
                 Vdsat_p_3*1e3, "Cds (zF)": Cds_p_3_m*1e21, "Cgd (aF)": Cgd_p_3_m*1
                e18, "Cgs (aF)": Cgs_p_3_m*1e18, "gds (μS)": gds_p_3_m*1e6, "gm (μS
                )": gm_p_3_m*1e6, "W (nm)": W_p_3_m*1e9, "L (nm)": L_3},
1171    ]
1172
1173    # Print the table header for NMOS characteristics
1174    print('Transistors Parameters\n')
```

```
1175     header = f"{'Transistor':<10} {'Id (μA)':<10} {'Vgs (mV)':<10} {'Vth (mV)
             ':<10} {'Vds (mV)':<10} {'Vdsat (mV)':<12} {'Cds (zF)':<10} {'Cgd (aF)
             ':<10} {'Cgs (aF)':<10} {'gds (μS)':<10} {'gm (μS)':<10} {'W (nm)':<10}
              {'L (nm)':<10}"
1176     print(header)
1177     print("-" * len(header))
1178     for row in trans_params:
1179         print(f"{row['Transistor']:<10} {row['Id (μA)']:<10.3f} {row['Vgs (mV)
                 ']:<10.3f} {row['Vth (mV)']:<10.3f} {row['Vds (mV)']:<10.3f} {row['
                 Vdsat (mV)']:<12.3f} {row['Cds (zF)']:<10.3f} {row['Cgd (aF)
                 ']:<10.3f} {row['Cgs (aF)']:<10.3f} {row['gds (μS)']:<10.3f} {row['
                 gm (μS)']:<10.3f} {row['W (nm)']:<10.0f} {row['L (nm)']:<10.0f}")
1180     print("-" * len(header))
1181
1182 ### Get Width Ratio
1183 def get_width_ratio(length):
1184     """
1185     Gets width ratio based in length
1186
1187     Parameters:
1188     - length (float): Length
1189
1190     Returns:
1191     - kW (float): Width ratio
1192     """
1193     if length == 60 or length == 90:
1194         kW = 2.625
1195     elif length == 120:
1196         kW = 3
1197     elif length == 240 or length == 300:
1198         kW = 3.625
1199     elif length == 600:
1200         kW = 4
1201
1202     return kW
1203
1204 ### Get Ramp Parameters
1205 def get_ramp_params(design):
1206     """
1207     Gets ramp parameters based on design target
1208
1209     Parameters:
1210     - design (str): Design target
1211
1212     Returns:
1213     - N_1, L_1, kW_1, KS_1 (float): Parameters stage 1
1214     - N_2, L_2, kW_2, KS_2 (float): Parameters stage 2
1215     - N_3, L_3, kW_3, KS_3 (float): Parameters stage 3
1216     """
1217     N_1 = 9
```

```
1218     N_2 = 9
1219
1220     KS_1 = 0.6
1221     KS_2 = 0.1
1222     KS_3 = 0.1
1223
1224     if design == 'A':
1225         L_1 = 300
1226         L_2 = 300
1227         L_3 = 600
1228         N_3 = 1
1229     elif design == 'GBW':
1230         L_1 = 60
1231         L_2 = 60
1232         L_3 = 60
1233         N_3 = 4
1234
1235     kW_1 = get_width_ratio(L_1)
1236     kW_2 = get_width_ratio(L_2)
1237     kW_3 = get_width_ratio(L_3)
1238
1239     return N_1, L_1, kW_1, KS_1, N_2, L_2, kW_2, KS_2, N_3, L_3, kW_3, KS_3
1240
1241 ### Get Ramp Characteristics
1242 def get_ramp_chrts(L_1, kW_1, KS_1, L_2, kW_2, KS_2, L_3, kW_3, KS_3):
1243     """
1244     Gets ramp characteristics
1245
1246     Parameters:
1247     - L_1, kW_1, KS_1: Parameters stage 1
1248     - L_2, kW_2, KS_2: Parameters stage 2
1249     - L_3, kW_3, KS_3: Parameters stage 3
1250
1251     Returns:
1252     - Cds_1, Cgd_1, Cgs_1, gds_1, gm_1: Characteristics stage 1
1253     - Cds_2, Cgd_2, Cgs_2, gds_2, gm_2: Characteristics stage 2
1254     - Cds_3, Cgd_3, Cgs_3, gds_3, gm_3: Characteristics stage 3
1255     """
1256     # Parameters
1257     cols_idx_nmos = [ [1, 6, 7, 8, 9, 10], [2, 3, 4, 5] ]
1258     cols_nms_nmos = [ ['Id_n', 'Cds_n', 'Cgd_n', 'Cgs_n', 'gds_n', 'gm_n'], ['
         Vgs_n', 'Vth_n', 'Vds_n', 'Vdsat_n'] ]
1259
1260     cols_idx_pmos = [ [15, 20, 21, 22, 23, 24], [16, 17, 18, 19] ]
1261     cols_nms_pmos = [ ['Id_p', 'Cds_p', 'Cgd_p', 'Cgs_p', 'gds_p', 'gm_p'], ['
         Vgs_p', 'Vth_p', 'Vds_p', 'Vdsat_p'] ]
1262
1263     # Define linear parameters and scale them
```

```
1264     _, Cds_n_1, Cgd_n_1, Cgs_n_1, gds_n_1, gm_n_1 = KS_1 * read_file(
             transistors_params, L_1, width_ratios, kW_1, cols_idx_nmos[0],
             cols_nms_nmos[0])
1265     _, Cds_n_2, Cgd_n_2, Cgs_n_2, gds_n_2, gm_n_2 = KS_2 * read_file(
             transistors_params, L_2, width_ratios, kW_2, cols_idx_nmos[0],
             cols_nms_nmos[0])
1266     _, Cds_n_3, Cgd_n_3, Cgs_n_3, gds_n_3, gm_n_3 = KS_3 * read_file(
             transistors_params, L_3, width_ratios, kW_3, cols_idx_nmos[0],
             cols_nms_nmos[0])
1267
1268     _, Cds_p_1, Cgd_p_1, Cgs_p_1, gds_p_1, gm_p_1 = KS_1 * read_file(
             transistors_params, L_1, width_ratios, kW_1, cols_idx_pmos[0],
             cols_nms_pmos[0])
1269     _, Cds_p_2, Cgd_p_2, Cgs_p_2, gds_p_2, gm_p_2 = KS_2 * read_file(
             transistors_params, L_2, width_ratios, kW_2, cols_idx_pmos[0],
             cols_nms_pmos[0])
1270     _, Cds_p_3, Cgd_p_3, Cgs_p_3, gds_p_3, gm_p_3 = KS_3 * read_file(
             transistors_params, L_3, width_ratios, kW_3, cols_idx_pmos[0],
             cols_nms_pmos[0])
1271
1272     # Define total parameters
1273     Cds_1 = Cds_p_1 + Cds_n_1
1274     Cgd_1 = Cgd_p_1 + Cgd_n_1
1275     Cgs_1 = Cgs_p_1 + Cgs_n_1
1276     gds_1 = gds_p_1 + gds_n_1
1277     gm_1  = gm_p_1 + gm_n_1
1278
1279     Cds_2 = Cds_p_2 + Cds_n_2
1280     Cgd_2 = Cgd_p_2 + Cgd_n_2
1281     Cgs_2 = Cgs_p_2 + Cgs_n_2
1282     gds_2 = gds_p_2 + gds_n_2
1283     gm_2  = gm_p_2 + gm_n_2
1284
1285     Cds_3 = Cds_p_3 + Cds_n_3
1286     Cgd_3 = Cgd_p_3 + Cgd_n_3
1287     Cgs_3 = Cgs_p_3 + Cgs_n_3
1288     gds_3 = gds_p_3 + gds_n_3
1289     gm_3  = gm_p_3 + gm_n_3
1290
1291     return Cds_1, Cgd_1, Cgs_1, gds_1, gm_1, Cds_2, Cgd_2, Cgs_2, gds_2, gm_2,
              Cds_3, Cgd_3, Cgs_3, gds_3, gm_3
1292
1293 ######################################################
1294
1295 ## Ring Amplifier FOM Different Lengths, Multipliers and Stage Factors
1296
1297 ### Different Multipiers
```

```
def plot_foms_diff_multipliers_lengths(transistors_params, width_ratios, ramp,
     stage, multipliers, freqs, cols_idx_nmos, cols_nms_nmos, cols_idx_pmos,
    cols_nms_pmos, N_1, KS_1, Id_1, Cds_1, Cgd_1, Cgs_1, gds_1, gm_1, N_2, KS_2
    , Id_2, Cds_2, Cgd_2, Cgs_2, gds_2, gm_2, N_3, KS_3, Id_3, Cds_3, Cgd_3,
    Cgs_3, gds_3, gm_3, VDD, CL, T, kB, γ, Δf):
    """
    Plots gain, bandwidth, gain bandwidth product, dissipated power, input-
        referred noise, phase margin, pole 1 and pole 2 for different
        multipliers and lengths

    Parameters:
    - transistors_params (dic): Dictionary of file paths keyed by lengths
    - width_ratios (dic): Dictionary of width ratios keyed by column indexes
    - ramp (str): Ringamp
    - stage (str): Stage
    - multipliers (array float): Multipliers
    - freqs (array float): Frequencies
    - cols_idx_nmos (array int): Columns indexes NMOS
    - cols_nms_nmos (array int): Columns names NMOS
    - cols_idx_pmos (array int): Columns indexes PMOS
    - cols_nms_pmos (array int): Columns names PMOS
    - N_1, KS_1, Id_1, Cds_1, Cgd_1, Cgs_1, gds_1, gm_1 (float): Parameters
        stage 1
    - N_1, KS_1, Id_1, Cds_1, Cgd_1, Cgs_1, gds_1, gm_1 (float): Parameters
        stage 1
    - N_1, KS_1, Id_1, Cds_1, Cgd_1, Cgs_1, gds_1, gm_1 (float): Parameters
        stage 1
    - VDD (float): Supply voltage
    - CL (float): Load capacitance
    - T (float): Temperature
    - kB (float): Boltzmann constant
    - γ (float): Channel length coefficient
    - Δf (float): Band

    Returns:
    - Plots of Gain, Pole, Dissipated Power, Noise, Bandwidth, GBW, Phase
        Margin, and Poles
    """
    gains = {L: [] for L in transistors_params.keys()}
    bandwidths = {L: [] for L in transistors_params.keys()}
    gain_bandwidth_products = {L: [] for L in transistors_params.keys()}
    dissipated_powers = {L: [] for L in transistors_params.keys()}
    input_referred_noises = {L: [] for L in transistors_params.keys()}
    phase_margins = {L: [] for L in transistors_params.keys()}
    pole_1_frequencies = {L: [] for L in transistors_params.keys()}
    pole_2_frequencies = {L: [] for L in transistors_params.keys()}
    pole_1_2_targets = {L: [] for L in transistors_params.keys()}

    # Iterates over lengths
    for L in transistors_params.keys():
```

```
1338
1339        # Get the width ratio correspondent to lenght
1340        kw = get_width_ratio(L)
1341
1342        # Iterates over multipliers
1343        for N in multipliers:
1344
1345            # Determine N values based on the stage
1346
1347            if stage == '1':
1348                N_1 = N
1349                Id_n_1, Cds_n_1, Cgd_n_1, Cgs_n_1, gds_n_1, gm_n_1 = KS_1 *
                        read_file(transistors_params, L, width_ratios, kw,
                        cols_idx_nmos[0], cols_nms_nmos[0])
1350                _, Cds_p_1, Cgd_p_1, Cgs_p_1, gds_p_1, gm_p_1 = KS_1 *
                        read_file(transistors_params, L, width_ratios, kw,
                        cols_idx_pmos[0], cols_nms_pmos[0])
1351                Id_1 = Id_n_1
1352                Cds_1 = Cds_n_1 + Cds_p_1
1353                Cgd_1 = Cgd_n_1 + Cgd_p_1
1354                Cgs_1 = Cgs_n_1 + Cgs_p_1
1355                gds_1 = gds_n_1 + gds_p_1
1356                gm_1 = gm_n_1 + gm_p_1
1357
1358            elif stage == '2':
1359                N_2 = N
1360                Id_n_2, Cds_n_2, Cgd_n_2, Cgs_n_2, gds_n_2, gm_n_2 = KS_2 *
                        read_file(transistors_params, L, width_ratios, kw,
                        cols_idx_nmos[0], cols_nms_nmos[0])
1361                _, Cds_p_2, Cgd_p_2, Cgs_p_2, gds_p_2, gm_p_2 = KS_2 *
                        read_file(transistors_params, L, width_ratios, kw,
                        cols_idx_pmos[0], cols_nms_pmos[0])
1362                Id_2 = Id_n_2
1363                Cds_2 = Cds_n_2 + Cds_p_2
1364                Cgd_2 = Cgd_n_2 + Cgd_p_2
1365                Cgs_2 = Cgs_n_2 + Cgs_p_2
1366                gds_2 = gds_n_2 + gds_p_2
1367                gm_2 = gm_n_2 + gm_p_2
1368
1369            elif stage == '3':
1370                N_3 = N
1371                Id_n_3, Cds_n_3, Cgd_n_3, Cgs_n_3, gds_n_3, gm_n_3 = KS_3 *
                        read_file(transistors_params, L, width_ratios, kw,
                        cols_idx_nmos[0], cols_nms_nmos[0])
1372                _, Cds_p_3, Cgd_p_3, Cgs_p_3, gds_p_3, gm_p_3 = KS_3 *
                        read_file(transistors_params, L, width_ratios, kw,
                        cols_idx_pmos[0], cols_nms_pmos[0])
1373                Id_3 = Id_n_3
1374                Cds_3 = Cds_n_3 + Cds_p_3
1375                Cgd_3 = Cgd_n_3 + Cgd_p_3
```

```
                Cgs_3 = Cgs_n_3 + Cgs_p_3
                gds_3 = gds_n_3 + gds_p_3
                gm_3 = gm_n_3 + gm_p_3

            # Calculates the FOMs
            A = ramp_gain(ramp, N_1, gds_1, gm_1, N_2, gds_2, gm_2, gds_3,
                gm_3)
            BW = ramp_bandwidth(N_3, Cds_3, Cgd_3, gds_3, CL) * 1e-6
            GBW = ramp_gain_bandwidth_product(ramp, N_1, gds_1, gm_1, N_2,
                gds_2, gm_2, N_3, Cds_3, Cgd_3, gm_3, CL) * 1e-9
            Pd = ramp_dissipated_power(ramp, N_1, Id_1, N_2, Id_2, N_3, Id_3,
                VDD) * 1e3
            Sin = ramp_input_referred_noise(ramp, N_1, gds_1, gm_1, N_2, gds_2
                , gm_2, N_3, gds_3, gm_3, T, kB, γ, Δf) * 1e6
            PM = ramp_phase_margin(ramp, freqs, N_1, Cds_1, Cgd_1, Cgs_1,
                gds_1, gm_1, N_2, Cds_2, Cgd_2, Cgs_2, gds_2, gm_2, N_3, Cds_3,
                 Cgd_3, Cgs_3, gds_3, gm_3, CL)
            fp = ramp_freqs_poles(ramp, N_1, Cds_1, Cgd_1, Cgs_1, gds_1, gm_1,
                 N_2, Cds_2, Cgd_2, Cgs_2, gds_2, gm_2, N_3, Cds_3, Cgd_3,
                Cgs_3, gds_3, CL)
            fp1, fp2 = fp[0] * 1e-9, fp[1] * 1e-9
            tp12 = 2.5 * GBW

            # Append the FOMs on the dictionary by length
            gains[L].append(A)
            bandwidths[L].append(BW)
            gain_bandwidth_products[L].append(GBW)
            dissipated_powers[L].append(Pd)
            input_referred_noises[L].append(Sin)
            phase_margins[L].append(PM)
            pole_1_frequencies[L].append(fp1)
            pole_2_frequencies[L].append(fp2)
            pole_1_2_targets[L].append(tp12)

    # Plotting
    plt.figure(figsize=(24, 12))

    plt.subplot(2, 4, 1)
    for L in transistors_params.keys():
        plt.plot(multipliers, gains[L], marker='o', label=f'$L_{stage}={L}$
            $nm$')
    plt.axhline(y=40, color='r', linestyle='--', label=f'$A = 40$ $dB$')
    plt.title(f'Gain')
    plt.grid(True)
    plt.xlabel(f'$N_{stage}$')
    plt.ylabel(f'$A$ $(dB)$')
    plt.xscale('log')
    plt.legend()

    plt.subplot(2, 4, 2)
```

```
    for L in transistors_params.keys():
        plt.plot(multipliers, gain_bandwidth_products[L], marker='o', label=f'
            $L_{stage}={L}$ $nm$')
    plt.title(f'Gain Bandwidth Product')
    plt.grid(True)
    plt.xlabel(f'$N_{stage}$')
    plt.ylabel(f'$GBW$ $(GHz)$')
    plt.xscale('log')
    plt.legend()

    plt.subplot(2, 4, 3)
    for L in transistors_params.keys():
        plt.plot(multipliers, dissipated_powers[L], marker='o', label=f'$L_{
            stage}={L}$ $nm$')
    plt.title(f'Dissipated Power')
    plt.grid(True)
    plt.xlabel(f'$N_{stage}$')
    plt.ylabel(f'$P_d$ $(mW)$')
    plt.xscale('log')
    plt.legend()

    plt.subplot(2, 4, 4)
    for L in transistors_params.keys():
        plt.plot(multipliers, input_referred_noises[L], marker='o', label=f'
            $L_{stage}={L}$ $nm$')
    plt.title(f'Input-Referred Noise')
    plt.grid(True)
    plt.xlabel(f'$N_{stage}$')
    plt.ylabel('$S_{in}$ ($\mu$V/$\sqrt{}$Hz)')
    plt.xscale('log')
    plt.legend()

    plt.subplot(2, 4, 5)
    for L in transistors_params.keys():
        plt.plot(multipliers, phase_margins[L], marker='o', label=f'$L_{stage
            }={L}$ $nm$')
    plt.axhline(y=60, color='r', linestyle='--', label=f'$\phi_M=60°$')
    plt.title(f'Phase Margin')
    plt.grid(True)
    plt.xlabel(f'$N_{stage}$')
    plt.ylabel('$\phi_M$ $(°)$')
    plt.xscale('log')
    plt.legend()

    plt.subplot(2, 4, 6)
    for L in transistors_params.keys():
        plt.plot(multipliers, pole_1_frequencies[L], marker='o', label=f'$L_{
            stage}={L}$ $nm$')
        plt.plot(multipliers, pole_1_2_targets[L], linestyle='--', color=plt.
            gca().lines[-1].get_color(), label=f'$fp1=2.5GBW$')
```

```
1461     plt.title(f'Pole 1')
1462     plt.grid(True)
1463     plt.xlabel(f'$N_{stage}$')
1464     plt.ylabel(f'$fp_1$ $(GHz)$')
1465     plt.xscale('log')
1466     plt.legend()
1467 
1468     plt.subplot(2, 4, 7)
1469     for L in transistors_params.keys():
1470         plt.plot(multipliers, pole_2_frequencies[L], marker='o', label=f'$L_{
                stage}={L}$ $nm$')
1471         plt.plot(multipliers, pole_1_2_targets[L], linestyle='--', color=plt.
                gca().lines[-1].get_color(), label=f'$fp2=2.5GBW$')
1472     plt.title(f'Pole 2')
1473     plt.grid(True)
1474     plt.xlabel(f'$N_{stage}$')
1475     plt.ylabel(f'$fp_2$ $(GHz)$')
1476     plt.xscale('log')
1477     plt.legend()
1478 
1479     plt.subplot(2, 4, 8)
1480     for L in transistors_params.keys():
1481         plt.plot(multipliers, bandwidths[L], marker='o', label=f'$L_{stage}={L
                }$ $nm$')
1482     plt.title(f'Pole 3')
1483     plt.grid(True)
1484     plt.xlabel(f'$N_{stage}$')
1485     plt.ylabel(f'$fp_3$ $(MHz)$')
1486     plt.xscale('log')
1487     plt.legend()
1488 
1489     plt.tight_layout()
1490     plt.show()
1491 
1492 ### Different Stage Factors
1493 def plot_foms_diff_factors_lengths(transistors_params, width_ratios, ramp,
        stage, stage_factors, freqs, cols_idx_nmos, cols_nms_nmos, cols_idx_pmos,
        cols_nms_pmos, N_1, KS_1, Id_1, Cds_1, Cgd_1, Cgs_1, gds_1, gm_1, N_2, KS_2
        , Id_2, Cds_2, Cgd_2, Cgs_2, gds_2, gm_2, N_3, KS_3, Id_3, Cds_3, Cgd_3,
        Cgs_3, gds_3, gm_3, VDD, CL, T, kB, γ, Δf):
1494     """
1495     Plots gain, bandwidth, gain bandwidth product, dissipated power, input-
             referred noise, phase margin, pole 1 and pole 2 for different
             multipliers and lengths
1496 
1497     Parameters:
1498     - transistors_params (dic): Dictionary of file paths keyed by lengths
1499     - width_ratios (dic): Dictionary of width ratios keyed by column indexes
1500     - ramp (str): Ringamp
1501     - stage (str): Stage
```

```
    - multipliers (array float): Multipliers
    - freqs (array float): Frequencies
    - cols_idx_nmos (array int): Columns indexes NMOS
    - cols_nms_nmos (array int): Columns names NMOS
    - cols_idx_pmos (array int): Columns indexes PMOS
    - cols_nms_pmos (array int): Columns names PMOS
    - N_1, KS_1, Id_1, Cds_1, Cgd_1, Cgs_1, gds_1, gm_1 (float): Parameters
        stage 1
    - N_1, KS_1, Id_1, Cds_1, Cgd_1, Cgs_1, gds_1, gm_1 (float): Parameters
        stage 1
    - N_1, KS_1, Id_1, Cds_1, Cgd_1, Cgs_1, gds_1, gm_1 (float): Parameters
        stage 1
    - VDD (float): Supply voltage
    - CL (float): Load capacitance
    - T (float): Temperature
    - kB (float): Boltzmann constant
    - γ (float): Channel length coefficient
    - Δf (float): Band

    Returns:
    - Plots of Gain, Pole, Dissipated Power, Noise, Bandwidth, GBW, Phase
        Margin, and Poles
    """
    gains = {L: [] for L in transistors_params.keys()}
    bandwidths = {L: [] for L in transistors_params.keys()}
    gain_bandwidth_products = {L: [] for L in transistors_params.keys()}
    dissipated_powers = {L: [] for L in transistors_params.keys()}
    input_referred_noises = {L: [] for L in transistors_params.keys()}
    phase_margins = {L: [] for L in transistors_params.keys()}
    pole_1_frequencies = {L: [] for L in transistors_params.keys()}
    pole_2_frequencies = {L: [] for L in transistors_params.keys()}
    pole_1_2_targets = {L: [] for L in transistors_params.keys()}

     # Iterates over lengths
    for L in transistors_params.keys():

        # Get the width ratio correspondent to lenght
        kw = get_width_ratio(L)

        # Iterates over multipliers
        for KS in stage_factors:

            # Determine KS values based on the stage
            if stage == '1':
                KS_1 = KS
                Id_n_1, Cds_n_1, Cgd_n_1, Cgs_n_1, gds_n_1, gm_n_1 = KS_1 *
                    read_file(transistors_params, L, width_ratios, kw,
                    cols_idx_nmos[0], cols_nms_nmos[0])
```

```
1544                _, Cds_p_1, Cgd_p_1, Cgs_p_1, gds_p_1, gm_p_1 = KS_1 *
                        read_file(transistors_params, L, width_ratios, kw,
                        cols_idx_pmos[0], cols_nms_pmos[0])
1545                Id_1 = Id_n_1
1546                Cds_1 = Cds_n_1 + Cds_p_1
1547                Cgd_1 = Cgd_n_1 + Cgd_p_1
1548                Cgs_1 = Cgs_n_1 + Cgs_p_1
1549                gds_1 = gds_n_1 + gds_p_1
1550                gm_1 = gm_n_1 + gm_p_1
1551            elif stage == '2':
1552                KS_2 = KS
1553                Id_n_2, Cds_n_2, Cgd_n_2, Cgs_n_2, gds_n_2, gm_n_2 = KS_2 *
                        read_file(transistors_params, L, width_ratios, kw,
                        cols_idx_nmos[0], cols_nms_nmos[0])
1554                _, Cds_p_2, Cgd_p_2, Cgs_p_2, gds_p_2, gm_p_2 = KS_2 *
                        read_file(transistors_params, L, width_ratios, kw,
                        cols_idx_pmos[0], cols_nms_pmos[0])
1555                Id_2 = Id_n_2
1556                Cds_2 = Cds_n_2 + Cds_p_2
1557                Cgd_2 = Cgd_n_2 + Cgd_p_2
1558                Cgs_2 = Cgs_n_2 + Cgs_p_2
1559                gds_2 = gds_n_2 + gds_p_2
1560                gm_2 = gm_n_2 + gm_p_2
1561            elif stage == '3':
1562                KS_3 = KS
1563                Id_n_3, Cds_n_3, Cgd_n_3, Cgs_n_3, gds_n_3, gm_n_3 = KS_3 *
                        read_file(transistors_params, L, width_ratios, kw,
                        cols_idx_nmos[0], cols_nms_nmos[0])
1564                _, Cds_p_3, Cgd_p_3, Cgs_p_3, gds_p_3, gm_p_3 = KS_3 *
                        read_file(transistors_params, L, width_ratios, kw,
                        cols_idx_pmos[0], cols_nms_pmos[0])
1565                Id_3 = Id_n_3
1566                Cds_3 = Cds_n_3 + Cds_p_3
1567                Cgd_3 = Cgd_n_3 + Cgd_p_3
1568                Cgs_3 = Cgs_n_3 + Cgs_p_3
1569                gds_3 = gds_n_3 + gds_p_3
1570                gm_3 = gm_n_3 + gm_p_3
1571
1572            A = ramp_gain(ramp, N_1, gds_1, gm_1, N_2, gds_2, gm_2, gds_3,
                    gm_3)
1573            BW = ramp_bandwidth(N_3, Cds_3, Cgd_3, gds_3, CL) * 1e-6
1574            GBW = ramp_gain_bandwidth_product(ramp, N_1, gds_1, gm_1, N_2,
                    gds_2, gm_2, N_3, Cds_3, Cgd_3, gm_3, CL) * 1e-9
1575            Pd = ramp_dissipated_power(ramp, N_1, Id_1, N_2, Id_2, N_3, Id_3,
                    VDD) * 1e3
1576            Sin = ramp_input_referred_noise(ramp, N_1, gds_1, gm_1, N_2, gds_2
                    , gm_2, N_3, gds_3, gm_3, T, kB, γ, Δf) * 1e6
1577            PM = ramp_phase_margin(ramp, freqs, N_1, Cds_1, Cgd_1, Cgs_1,
                    gds_1, gm_1, N_2, Cds_2, Cgd_2, Cgs_2, gds_2, gm_2, N_3, Cds_3,
                     Cgd_3, Cgs_3, gds_3, gm_3, CL)
```

```
            fp = ramp_freqs_poles(ramp, N_1, Cds_1, Cgd_1, Cgs_1, gds_1, gm_1,
                 N_2, Cds_2, Cgd_2, Cgs_2, gds_2, gm_2, N_3, Cds_3, Cgd_3,
                Cgs_3, gds_3, CL)
            fp1, fp2 = fp[0] * 1e-9, fp[1] * 1e-9
            tp12 = 2.5 * GBW

            gains[L].append(A)
            bandwidths[L].append(BW)
            gain_bandwidth_products[L].append(GBW)
            dissipated_powers[L].append(Pd)
            input_referred_noises[L].append(Sin)
            phase_margins[L].append(PM)
            pole_1_frequencies[L].append(fp1)
            pole_2_frequencies[L].append(fp2)
            pole_1_2_targets[L].append(tp12)

    # Plotting
    plt.figure(figsize=(24, 12))

    plt.subplot(2, 4, 1)
    for L in transistors_params.keys():
        plt.plot(stage_factors, gains[L], marker='o', label=f'$L_{stage}={L}$
            $nm$')
    plt.axhline(y=40, color='r', linestyle='--', label=f'$A = 40$ $dB$')
    plt.title(f'Gain')
    plt.grid(True)
    plt.xlabel(f'$KS_{stage}$')
    plt.ylabel(f'$A$ $(dB)$')
    plt.xscale('log')
    plt.legend()

    plt.subplot(2, 4, 2)
    for L in transistors_params.keys():
        plt.plot(stage_factors, gain_bandwidth_products[L], marker='o', label=
            f'$L_{stage}={L}$ $nm$')
    plt.title(f'Gain Bandwidth Product')
    plt.grid(True)
    plt.xlabel(f'$KS_{stage}$')
    plt.ylabel(f'$GBW$ $(GHz)$')
    plt.xscale('log')
    plt.legend()

    plt.subplot(2, 4, 3)
    for L in transistors_params.keys():
        plt.plot(stage_factors, dissipated_powers[L], marker='o', label=f'$L_{
            stage}={L}$ $nm$')
    plt.title(f'Dissipated Power')
    plt.grid(True)
    plt.xlabel(f'$KS_{stage}$')
    plt.ylabel(f'$P_d$ $(mW)$')
```

```
1623    plt.xscale('log')
1624    plt.legend()
1625
1626    plt.subplot(2, 4, 4)
1627    for L in transistors_params.keys():
1628        plt.plot(stage_factors, input_referred_noises[L], marker='o', label=f'
            $L_{stage}={L}$ $nm$')
1629    plt.title(f'Input-Referred Noise')
1630    plt.grid(True)
1631    plt.xlabel(f'$KS_{stage}$')
1632    plt.ylabel('$S_{in}$ ($\mu$V/$\sqrt{\text{Hz}}$)')
1633    plt.xscale('log')
1634    plt.legend()
1635
1636    plt.subplot(2, 4, 5)
1637    for L in transistors_params.keys():
1638        plt.plot(stage_factors, phase_margins[L], marker='o', label=f'$L_{
            stage}={L}$ $nm$')
1639    plt.axhline(y=60, color='r', linestyle='--', label=f'$\phi_M=60°$')
1640    plt.title(f'Phase Margin')
1641    plt.grid(True)
1642    plt.xlabel(f'$KS_{stage}$')
1643    plt.ylabel('$\phi_M$ $(°)$')
1644    plt.xscale('log')
1645    plt.legend()
1646
1647    plt.subplot(2, 4, 6)
1648    for L in transistors_params.keys():
1649        plt.plot(stage_factors, pole_1_frequencies[L], marker='o', label=f'$L_
            {stage}={L}$ $nm$')
1650        plt.plot(stage_factors, pole_1_2_targets[L], linestyle='--', color=plt
            .gca().lines[-1].get_color(), label=f'$fp1=2.5GBW$')
1651    plt.title(f'Pole 1')
1652    plt.grid(True)
1653    plt.xlabel(f'$KS_{stage}$')
1654    plt.ylabel(f'$fp_1$ $(GHz)$')
1655    plt.xscale('log')
1656    plt.legend()
1657
1658    plt.subplot(2, 4, 7)
1659    for L in transistors_params.keys():
1660        plt.plot(stage_factors, pole_2_frequencies[L], marker='o', label=f'$L_
            {stage}={L}$ $nm$')
1661        plt.plot(stage_factors, pole_1_2_targets[L], linestyle='--', color=plt
            .gca().lines[-1].get_color(), label=f'$fp2=2.5GBW$')
1662    plt.title(f'Pole 2')
1663    plt.grid(True)
1664    plt.xlabel(f'$KS_{stage}$')
1665    plt.ylabel(f'$fp_2$ $(GHz)$')
1666    plt.xscale('log')
```

```
    plt.legend()

    plt.subplot(2, 4, 8)
    for L in transistors_params.keys():
        plt.plot(stage_factors, bandwidths[L], marker='o', label=f'$L_{stage
            }={L}$ $nm$')
    plt.title(f'Pole 3')
    plt.grid(True)
    plt.xlabel(f'$KS_{stage}$')
    plt.ylabel(f'$fp_3$ $(MHz)$')
    plt.xscale('log')
    plt.legend()

    plt.tight_layout()
    plt.show()

##################################################
##################################################

## Test

# Define amplifers
ramp_1 = 'Conventional'
ramp_2 = 'Critically Damped'

# Define stages
stage_1 = '1'
stage_2 = '2'
stage_3 = '3'

# Transistors parameters associated with lengths
transistors_params = {
    60: 'mosfet_vdd_1.2v_w_1.2um_l_60nm.csv',
    90: 'mosfet_vdd_1.2v_w_1.2um_l_90nm.csv',
    120: 'mosfet_vdd_1.2v_w_1.2um_l_120nm.csv',
    240: 'mosfet_vdd_1.2v_w_1.2um_l_240nm.csv',
    300: 'mosfet_vdd_1.2v_w_1.2um_l_300nm.csv',
    600: 'mosfet_vdd_1.2v_w_1.2um_l_600nm.csv'
}

# Sizing ratios associated with column indexes
width_ratios = {
     0: 1,
     1: 2,
     2: 2.625,
     3: 3,
     4: 3.625,
     5: 4,
     6: 5,
     7: 6,
```

```
    8: 7,
    9: 8,
    10: 9,
    11: 10
}

# Designs for gain, gain bandwidth product and dissipated power
designs = {
    'A': 'cd_ramp_a.csv',
    'GBW': 'cd_ramp_gbw.csv'
}

# Define frequencies (Hz)
freqs = np.logspace(0, 15, 10000)

# Define band (Hz)
Δf = 1e12 - 1e6

# Define multipliers
multipliers = np.linspace(1, 10, 10)

# Define stage factors
stage_factors = np.linspace(0.1, 1, 10)

# Define lengths (nm)
lengths = [60, 90, 120, 240, 300, 600]

# Define duty cycle
D = 0.499

# Define feedback capacitance (F)
CF = 100e-15

# Define supply voltages (V)
VDD = 1.2

# Define target gain (dB)
A = 40

# Define error
ϵ = 1e-2

# Define resolution (b)
N = 3

# Define temperature (K)
T = 27 + 213.15

# Define Boltzmann constant (J/K)
kB = 1.38e-23
```

```
1766
1767 # Define channel length coefficent
1768 γ = 1
1769
1770 # Define table width (m)
1771 W_tb = 1.2e-6
1772
1773 # Define the multipliers                *** variables ***
1774 N_1 = 1      # A -> 9     GBW -> 9
1775 N_2 = 1      # A -> 9     GBW -> 9
1776 N_3 = 1      # A -> 1     GBW -> 4
1777
1778 # Define the lengths                    *** variables ***
1779 L_1 = 90     # A -> 300   GBW -> 60
1780 L_2 = 60     # A -> 300   GBW -> 60
1781 L_3 = 120    # A -> 600   GBW -> 60
1782
1783 # Define the sizing ratios
1784 kW_1 = get_width_ratio(L_1)
1785 kW_2 = get_width_ratio(L_2)
1786 kW_3 = get_width_ratio(L_3)
1787
1788 # Define the design factors             *** variables ***
1789 KS_1 = 0.3   # A -> 0.6   GBW -> 0.6
1790 KS_2 = 0.1   # A -> 0.1   GBW -> 0.1
1791 KS_3 = 0.1   # A -> 0.1   GBW -> 0.1
1792
1793 # Determine the channel widths
1794 W_n_1 = KS_1 * W_tb
1795 W_n_2 = KS_2 * W_tb
1796 W_n_3 = KS_3 * W_tb
1797
1798 W_p_1 = kW_1 * W_n_1
1799 W_p_2 = kW_2 * W_n_2
1800 W_p_3 = kW_3 * W_n_3
1801
1802 # Parameters
1803 cols_idx_nmos = [ [1, 6, 7, 8, 9, 10], [2, 3, 4, 5] ]
1804 cols_nms_nmos = [ ['Id_n', 'Cds_n', 'Cgd_n', 'Cgs_n', 'gds_n', 'gm_n'], ['
        Vgs_n', 'Vth_n', 'Vds_n', 'Vdsat_n'] ]
1805
1806 cols_idx_pmos = [ [15, 20, 21, 22, 23, 24], [16, 17, 18, 19] ]
1807 cols_nms_pmos = [ ['Id_p', 'Cds_p', 'Cgd_p', 'Cgs_p', 'gds_p', 'gm_p'], ['
        Vgs_p', 'Vth_p', 'Vds_p', 'Vdsat_p'] ]
1808
1809 # Define linear parameters and scale them
1810 Id_n_1, Cds_n_1, Cgd_n_1, Cgs_n_1, gds_n_1, gm_n_1 = KS_1 * read_file(
        transistors_params, L_1, width_ratios, kW_1, cols_idx_nmos[0],
        cols_nms_nmos[0])
```

```
1811 Id_n_2, Cds_n_2, Cgd_n_2, Cgs_n_2, gds_n_2, gm_n_2 = KS_2 * read_file(
        transistors_params, L_2, width_ratios, kW_2, cols_idx_nmos[0],
        cols_nms_nmos[0])
1812 Id_n_3, Cds_n_3, Cgd_n_3, Cgs_n_3, gds_n_3, gm_n_3 = KS_3 * read_file(
        transistors_params, L_3, width_ratios, kW_3, cols_idx_nmos[0],
        cols_nms_nmos[0])
1813
1814 Id_p_1, Cds_p_1, Cgd_p_1, Cgs_p_1, gds_p_1, gm_p_1 = KS_1 * read_file(
        transistors_params, L_1, width_ratios, kW_1, cols_idx_pmos[0],
        cols_nms_pmos[0])
1815 Id_p_2, Cds_p_2, Cgd_p_2, Cgs_p_2, gds_p_2, gm_p_2 = KS_2 * read_file(
        transistors_params, L_2, width_ratios, kW_2, cols_idx_pmos[0],
        cols_nms_pmos[0])
1816 Id_p_3, Cds_p_3, Cgd_p_3, Cgs_p_3, gds_p_3, gm_p_3 = KS_3 * read_file(
        transistors_params, L_3, width_ratios, kW_3, cols_idx_pmos[0],
        cols_nms_pmos[0])
1817
1818 # Define multiplied parameters
1819 Id_n_1_m  = N_1 * Id_n_1
1820 Cds_n_1_m = N_1 * Cds_n_1
1821 Cgd_n_1_m = N_1 * Cgd_n_1
1822 Cgs_n_1_m = N_1 * Cgs_n_1
1823 gds_n_1_m = N_1 * gds_n_1
1824 gm_n_1_m  = N_1 * gm_n_1
1825 W_n_1_m   = N_1 * W_n_1
1826
1827 Id_p_1_m  = N_1 * Id_p_1
1828 Cds_p_1_m = N_1 * Cds_p_1
1829 Cgd_p_1_m = N_1 * Cgd_p_1
1830 Cgs_p_1_m = N_1 * Cgs_p_1
1831 gds_p_1_m = N_1 * gds_p_1
1832 gm_p_1_m  = N_1 * gm_p_1
1833 W_p_1_m   = N_1 * W_p_1
1834
1835 Id_n_2_m  = N_2 * Id_n_2
1836 Cds_n_2_m = N_2 * Cds_n_2
1837 Cgd_n_2_m = N_2 * Cgd_n_2
1838 Cgs_n_2_m = N_2 * Cgs_n_2
1839 gds_n_2_m = N_2 * gds_n_2
1840 gm_n_2_m  = N_2 * gm_n_2
1841 W_n_2_m   = N_2 * W_n_2
1842
1843 Id_p_2_m  = N_2 * Id_p_2
1844 Cds_p_2_m = N_2 * Cds_p_2
1845 Cgd_p_2_m = N_2 * Cgd_p_2
1846 Cgs_p_2_m = N_2 * Cgs_p_2
1847 gds_p_2_m = N_2 * gds_p_2
1848 gm_p_2_m  = N_2 * gm_p_2
1849 W_p_2_m   = N_2 * W_p_2
1850
```

```
Id_n_3_m  = N_3 * Id_n_3
Cds_n_3_m = N_3 * Cds_n_3
Cgd_n_3_m = N_3 * Cgd_n_3
Cgs_n_3_m = N_3 * Cgs_n_3
gds_n_3_m = N_3 * gds_n_3
gm_n_3_m  = N_3 * gm_n_3
W_n_3_m   = N_3 * W_n_3

Id_p_3_m  = N_3 * Id_p_3
Cds_p_3_m = N_3 * Cds_p_3
Cgd_p_3_m = N_3 * Cgd_p_3
Cgs_p_3_m = N_3 * Cgs_p_3
gds_p_3_m = N_3 * gds_p_3
gm_p_3_m  = N_3 * gm_p_3
W_p_3_m   = N_3 * W_p_3

# Define voltages
Vgs_n_1, Vth_n_1, Vds_n_1, Vdsat_n_1 = read_file(transistors_params, L_1,
    width_ratios, kW_1, cols_idx_nmos[1], cols_nms_nmos[1])
Vgs_n_2, Vth_n_2, Vds_n_2, Vdsat_n_2 = read_file(transistors_params, L_2,
    width_ratios, kW_2, cols_idx_nmos[1], cols_nms_nmos[1])
Vgs_n_3, Vth_n_3, Vds_n_3, Vdsat_n_3 = read_file(transistors_params, L_3,
    width_ratios, kW_3, cols_idx_nmos[1], cols_nms_nmos[1])

Vgs_p_1, Vth_p_1, Vds_p_1, Vdsat_p_1 = read_file(transistors_params, L_1,
    width_ratios, kW_1, cols_idx_pmos[1], cols_nms_pmos[1])
Vgs_p_2, Vth_p_2, Vds_p_2, Vdsat_p_2 = read_file(transistors_params, L_2,
    width_ratios, kW_2, cols_idx_pmos[1], cols_nms_pmos[1])
Vgs_p_3, Vth_p_3, Vds_p_3, Vdsat_p_3 = read_file(transistors_params, L_3,
    width_ratios, kW_3, cols_idx_pmos[1], cols_nms_pmos[1])

# Define total parameters
Id_1  = Id_n_1
Cds_1 = Cds_p_1 + Cds_n_1
Cgd_1 = Cgd_p_1 + Cgd_n_1
Cgs_1 = Cgs_p_1 + Cgs_n_1
gds_1 = gds_p_1 + gds_n_1
gm_1  = gm_p_1 + gm_n_1

Id_2  = Id_n_2
Cds_2 = Cds_p_2 + Cds_n_2
Cgd_2 = Cgd_p_2 + Cgd_n_2
Cgs_2 = Cgs_p_2 + Cgs_n_2
gds_2 = gds_p_2 + gds_n_2
gm_2  = gm_p_2 + gm_n_2

Id_3  = Id_n_3
Cds_3 = Cds_p_3 + Cds_n_3
Cgd_3 = Cgd_p_3 + Cgd_n_3
Cgs_3 = Cgs_p_3 + Cgs_n_3
```

```
gds_3 = gds_p_3 + gds_n_3
gm_3  = gm_p_3 + gm_n_3



# Inverter transconductance efficiency
plot_inv_transconductance_efficiency_diff_lengths(transistors_params)
# Inverter trip point
plot_inv_trip_point_diff_lengths(transistors_params)

# Ramp perfomance close loop
Af, β, CS, CL, Vcmin, Vcmout= calc_ramp_performance_cl(VDD, N, CF)
print_ramp_perfomance_cl(Af, β, CF, CS, CL, VDD, Vcmin, Vcmout)
# Stage 1
plot_foms_diff_multipliers_lengths(transistors_params, width_ratios, ramp_2,
    stage_1, multipliers, freqs, cols_idx_nmos, cols_nms_nmos, cols_idx_pmos,
    cols_nms_pmos, N_1, KS_1, Id_1, Cds_1, Cgd_1, Cgs_1, gds_1, gm_1, N_2, KS_2
    , Id_2, Cds_2, Cgd_2, Cgs_2, gds_2, gm_2, N_3, KS_3, Id_3, Cds_3, Cgd_3,
    Cgs_3, gds_3, gm_3, VDD, CL, T, kB, γ, Δf)
plot_foms_diff_factors_lengths(transistors_params, width_ratios, ramp_2,
    stage_1, stage_factors, freqs, cols_idx_nmos, cols_nms_nmos, cols_idx_pmos,
     cols_nms_pmos, N_1, KS_1, Id_1, Cds_1, Cgd_1, Cgs_1, gds_1, gm_1, N_2,
    KS_2, Id_2, Cds_2, Cgd_2, Cgs_2, gds_2, gm_2, N_3, KS_3, Id_3, Cds_3, Cgd_3
    , Cgs_3, gds_3, gm_3, VDD, CL, T, kB, γ, Δf)
# Stage 2
plot_foms_diff_multipliers_lengths(transistors_params, width_ratios, ramp_2,
    stage_2, multipliers, freqs, cols_idx_nmos, cols_nms_nmos, cols_idx_pmos,
    cols_nms_pmos, N_1, KS_1, Id_1, Cds_1, Cgd_1, Cgs_1, gds_1, gm_1, N_2, KS_2
    , Id_2, Cds_2, Cgd_2, Cgs_2, gds_2, gm_2, N_3, KS_3, Id_3, Cds_3, Cgd_3,
    Cgs_3, gds_3, gm_3, VDD, CL, T, kB, γ, Δf)
plot_foms_diff_factors_lengths(transistors_params, width_ratios, ramp_2,
    stage_2, stage_factors, freqs, cols_idx_nmos, cols_nms_nmos, cols_idx_pmos,
     cols_nms_pmos, N_1, KS_1, Id_1, Cds_1, Cgd_1, Cgs_1, gds_1, gm_1, N_2,
    KS_2, Id_2, Cds_2, Cgd_2, Cgs_2, gds_2, gm_2, N_3, KS_3, Id_3, Cds_3, Cgd_3
    , Cgs_3, gds_3, gm_3, VDD, CL, T, kB, γ, Δf)
# Stage 3
plot_foms_diff_multipliers_lengths(transistors_params, width_ratios, ramp_2,
    stage_3, multipliers, freqs, cols_idx_nmos, cols_nms_nmos, cols_idx_pmos,
    cols_nms_pmos, N_1, KS_1, Id_1, Cds_1, Cgd_1, Cgs_1, gds_1, gm_1, N_2, KS_2
    , Id_2, Cds_2, Cgd_2, Cgs_2, gds_2, gm_2, N_3, KS_3, Id_3, Cds_3, Cgd_3,
    Cgs_3, gds_3, gm_3, VDD, CL, T, kB, γ, Δf)
plot_foms_diff_factors_lengths(transistors_params, width_ratios, ramp_2,
    stage_3, stage_factors, freqs, cols_idx_nmos, cols_nms_nmos, cols_idx_pmos,
     cols_nms_pmos, N_1, KS_1, Id_1, Cds_1, Cgd_1, Cgs_1, gds_1, gm_1, N_2,
    KS_2, Id_2, Cds_2, Cgd_2, Cgs_2, gds_2, gm_2, N_3, KS_3, Id_3, Cds_3, Cgd_3
    , Cgs_3, gds_3, gm_3, VDD, CL, T, kB, γ, Δf)

# Ramp parameters
print_ramp_params(N_1, kW_1, KS_1,N_2, kW_2, KS_2, N_3, kW_3, KS_3)
# Transistors parameters
```

```
1921 print_trans_params(L_1, Vgs_n_1, Vth_n_1, Vds_n_1, Vdsat_n_1, Id_n_1, Cds_n_1,
      Cgd_n_1, Cgs_n_1, gds_n_1, gm_n_1, W_n_1, Id_n_1_m, Cds_n_1_m, Cgd_n_1_m,
     Cgs_n_1_m, gds_n_1_m, gm_n_1_m, W_n_1_m, Vgs_p_1, Vth_p_1, Vds_p_1,
     Vdsat_p_1, Id_p_1, Cds_p_1, Cgd_p_1, Cgs_p_1, gds_p_1, gm_p_1, W_p_1,
     Id_p_1_m, Cds_p_1_m, Cgd_p_1_m, Cgs_p_1_m, gds_p_1_m, gm_p_1_m, W_p_1_m,
     L_2, Vgs_n_2, Vth_n_2, Vds_n_2, Vdsat_n_2, Id_n_2, Cds_n_2, Cgd_n_2,
     Cgs_n_2, gds_n_2, gm_n_2, W_n_2, Id_n_2_m, Cds_n_2_m, Cgd_n_2_m, Cgs_n_2_m,
      gds_n_2_m, gm_n_2_m, W_n_2_m, Vgs_p_2, Vth_p_2, Vds_p_2, Vdsat_p_2, Id_p_2
     , Cds_p_2, Cgd_p_2, Cgs_p_2, gds_p_2, gm_p_2, W_p_2, Id_p_2_m, Cds_p_2_m,
     Cgd_p_2_m, Cgs_p_2_m, gds_p_2_m, gm_p_2_m, W_p_2_m, L_3, Vgs_n_3, Vth_n_3,
     Vds_n_3, Vdsat_n_3, Id_n_3_m, Cds_n_3_m, Cgd_n_3_m, Cgs_n_3_m, gds_n_3_m,
     gm_n_3_m, W_n_3_m, Vgs_p_3, Vth_p_3, Vds_p_3, Vdsat_p_3, Id_p_3_m,
     Cds_p_3_m, Cgd_p_3_m, Cgs_p_3_m, gds_p_3_m, gm_p_3_m, W_p_3_m )
1922
1923 # Ramps perfomance open loop
1924 A_c_ramp, BW_c_ramp, GBW_c_ramp, ts_c_ramp, Vos_c_ramp, SR_c_ramp, Pd_c_ramp,
     Sin_c_ramp = calc_ramp_performance_ol(ramp_1, N_1, Id_1, gm_1, gds_1, N_2,
     Id_2, gm_2, gds_2, N_3, Id_3, Cds_3, Cgd_3, gm_3, gds_3, Vdsat_p_3,
     Vdsat_n_3, VDD, CL, T, kB, γ, ϵ, β, Δf)
1925 A_cd_ramp, BW_cd_ramp, GBW_cd_ramp, ts_cd_ramp, Vos_cd_ramp, SR_cd_ramp,
     Pd_cd_ramp, Sin_cd_ramp = calc_ramp_performance_ol(ramp_2, N_1, Id_1, gm_1,
      gds_1, N_2, Id_2, gm_2, gds_2, N_3, Id_3, Cds_3, Cgd_3, gm_3, gds_3,
     Vdsat_p_3, Vdsat_n_3, VDD, CL, T, kB, γ, ϵ, β, Δf)
1926 PM_1 = ramp_phase_margin(ramp_1, freqs, N_1, Cds_1, Cgd_1, Cgs_1, gds_1, gm_1,
      N_2, Cds_2, Cgd_2, Cgs_2, gds_2, gm_2, N_3, Cds_3, Cgd_3, Cgs_3, gds_3,
     gm_3, CL)
1927 PM_2 = ramp_phase_margin(ramp_2, freqs, N_1, Cds_1, Cgd_1, Cgs_1, gds_1, gm_1,
      N_2, Cds_2, Cgd_2, Cgs_2, gds_2, gm_2, N_3, Cds_3, Cgd_3, Cgs_3, gds_3,
     gm_3, CL)
1928 print_ramp_performances(ramp_1, A_c_ramp, BW_c_ramp, GBW_c_ramp, ts_c_ramp,
     Vos_c_ramp, SR_c_ramp, Pd_c_ramp, Sin_c_ramp, PM_1, ramp_2, A_cd_ramp,
     BW_cd_ramp, GBW_cd_ramp, ts_cd_ramp, Vos_cd_ramp, SR_cd_ramp, Pd_cd_ramp,
     Sin_cd_ramp, PM_2)
1929
1930 # Stages Perfomance in open loop
1931 A1_c_ramp, Pd1_c_ramp, Sin1_c_ramp, fp1_c_ramp, fz1_c_ramp =
     calc_stage_perfomance_ol(ramp_1, stage_1, N_1, Id_1, Cds_1, Cgd_1, Cgs_1,
     gds_1, gm_1, N_2, Id_2, Cds_2, Cgd_2, Cgs_2, gds_2, gm_2, N_3, Id_3, Cds_3,
      Cgd_3, Cgs_3, gds_3, gm_3, VDD, CL, kB, T, γ, Δf)
1932 A1_cd_ramp, Pd1_cd_ramp, Sin1_cd_ramp, fp1_cd_ramp, fz1_cd_ramp =
     calc_stage_perfomance_ol(ramp_2, stage_1, N_1, Id_1, Cds_1, Cgd_1, Cgs_1,
     gds_1, gm_1, N_2, Id_2, Cds_2, Cgd_2, Cgs_2, gds_2, gm_2, N_3, Id_3, Cds_3,
      Cgd_3, Cgs_3, gds_3, gm_3, VDD, CL, kB, T, γ, Δf)
1933 A2_c_ramp, Pd2_c_ramp, Sin2_c_ramp, fp2_c_ramp, fz2_c_ramp =
     calc_stage_perfomance_ol(ramp_1, stage_2, N_1, Id_1, Cds_1, Cgd_1, Cgs_1,
     gds_1, gm_1, N_2, Id_2, Cds_2, Cgd_2, Cgs_2, gds_2, gm_2, N_3, Id_3, Cds_3,
      Cgd_3, Cgs_3, gds_3, gm_3, VDD, CL, kB, T, γ, Δf)
```

```
A2_cd_ramp, Pd2_cd_ramp, Sin2_cd_ramp, fp2_cd_ramp, fz2_cd_ramp =
    calc_stage_perfomance_ol(ramp_2, stage_2, N_1, Id_1, Cds_1, Cgd_1, Cgs_1,
    gds_1, gm_1, N_2, Id_2, Cds_2, Cgd_2, Cgs_2, gds_2, gm_2, N_3, Id_3, Cds_3,
     Cgd_3, Cgs_3, gds_3, gm_3, VDD, CL, kB, T, γ, Δf)
A3_c_ramp, Pd3_c_ramp, Sin3_c_ramp, fp3_c_ramp, fz3_c_ramp =
    calc_stage_perfomance_ol(ramp_1, stage_3, N_1, Id_1, Cds_1, Cgd_1, Cgs_1,
    gds_1, gm_1, N_2, Id_2, Cds_2, Cgd_2, Cgs_2, gds_2, gm_2, N_3, Id_3, Cds_3,
     Cgd_3, Cgs_3, gds_3, gm_3, VDD, CL, kB, T, γ, Δf)
A3_cd_ramp, Pd3_cd_ramp, Sin3_cd_ramp, fp3_cd_ramp, fz3_cd_ramp =
    calc_stage_perfomance_ol(ramp_2, stage_3, N_1, Id_1, Cds_1, Cgd_1, Cgs_1,
    gds_1, gm_1, N_2, Id_2, Cds_2, Cgd_2, Cgs_2, gds_2, gm_2, N_3, Id_3, Cds_3,
     Cgd_3, Cgs_3, gds_3, gm_3, VDD, CL, kB, T, γ, Δf)
print_stages_performances_ol(ramp_1, ramp_2, A1_c_ramp, Pd1_c_ramp,
    Sin1_c_ramp, fp1_c_ramp, fz1_c_ramp, A1_cd_ramp, Pd1_cd_ramp, Sin1_cd_ramp,
     fp1_cd_ramp, fz1_cd_ramp, A2_c_ramp, Pd2_c_ramp, Sin2_c_ramp, fp2_c_ramp,
    fz2_c_ramp, A2_cd_ramp, Pd2_cd_ramp, Sin2_cd_ramp, fp2_cd_ramp, fz2_cd_ramp
    , A3_c_ramp, Pd3_c_ramp, Sin3_c_ramp, fp3_c_ramp, fz3_c_ramp, A3_cd_ramp,
    Pd3_cd_ramp, Sin3_cd_ramp, fp3_cd_ramp, fz3_cd_ramp)

# Target poles
fp1_gbw_1, fp2_gbw_1 = calc_target_poles(fp1_c_ramp, fp2_c_ramp, GBW_c_ramp)
fp1_gbw_2, fp2_gbw_2 = calc_target_poles(fp1_cd_ramp, fp2_cd_ramp, GBW_cd_ramp
    )
print_target_poles(ramp_1, fp1_gbw_1, fp2_gbw_1, ramp_2, fp1_gbw_2, fp2_gbw_2)

# Clock frequency
fs_c_ramp = calc_sample_freq(D, ts_c_ramp)
fs_cd_ramp = calc_sample_freq(D, ts_cd_ramp)
print_sample_freq(ramp_1, fs_c_ramp, ramp_2, fs_cd_ramp)

# Ramps frequency response in open loop
mag_c_ramp, phase_c_ramp, freq_c_ramp = calc_ramp_frequency_response_ol(ramp_1
    , freqs, N_1, Cds_1, Cgd_1, Cgs_1, gds_1, gm_1, N_2, Cds_2, Cgd_2, Cgs_2,
    gds_2, gm_2, N_3, Cds_3, Cgd_3, Cgs_3, gds_3, gm_3, CL)
mag_cd_ramp, phase_cd_ramp, freq_cd_ramp = calc_ramp_frequency_response_ol(
    ramp_2, freqs, N_1, Cds_1, Cgd_1, Cgs_1, gds_1, gm_1, N_2, Cds_2, Cgd_2,
    Cgs_2, gds_2, gm_2, N_3, Cds_3, Cgd_3, Cgs_3, gds_3, gm_3, CL)
plot_frequency_responses_ol(ramp_1, mag_c_ramp, phase_c_ramp, freq_c_ramp,
    ramp_2, mag_cd_ramp, phase_cd_ramp, freq_cd_ramp)

# Compare theory and simulation
comp_frequency_responses_ol(mag_cd_ramp, phase_cd_ramp, freq_cd_ramp, designs[
    'A'])
N_1_A, L_1_A, kW_1_A, KS_1_A, N_2_A, L_2_A, kW_2_A, KS_2_A, N_3_A, L_3_A,
    kW_3_A, KS_3_A = get_ramp_params('A')
Cds_1_A, Cgd_1_A, Cgs_1_A, gds_1_A, gm_1_A, Cds_2_A, Cgd_2_A, Cgs_2_A, gds_2_A
    , gm_2_A, Cds_3_A, Cgd_3_A, Cgs_3_A, gds_3_A, gm_3_A = get_ramp_chrts(L_1_A
    , kW_1_A, KS_1_A, L_2_A, kW_2_A, KS_2_A, L_3_A, kW_3_A, KS_3_A)
```

```
1958 mag_A, phase_A, freq_A = calc_ramp_frequency_response_ol(ramp_2, freqs, N_1_A,
         Cds_1_A, Cgd_1_A, Cgs_1_A, gds_1_A, gm_1_A, N_2_A, Cds_2_A, Cgd_2_A,
        Cgs_2_A, gds_2_A, gm_2_A, N_3_A, Cds_3_A, Cgd_3_A, Cgs_3_A, gds_3_A, gm_3_A
        , CL)
1959
1960 N_1_GBW, L_1_GBW, kW_1_GBW, KS_1_GBW, N_2_GBW, L_2_GBW, kW_2_GBW, KS_2_GBW,
        N_3_GBW, L_3_GBW, kW_3_GBW, KS_3_GBW = get_ramp_params('GBW')
1961 Cds_1_GBW, Cgd_1_GBW, Cgs_1_GBW, gds_1_GBW, gm_1_GBW, Cds_2_GBW, Cgd_2_GBW,
        Cgs_2_GBW, gds_2_GBW, gm_2_GBW, Cds_3_GBW, Cgd_3_GBW, Cgs_3_GBW, gds_3_GBW,
         gm_3_GBW = get_ramp_chrts(L_1_GBW, kW_1_GBW, KS_1_GBW, L_2_GBW, kW_2_GBW,
        KS_2_GBW, L_3_GBW, kW_3_GBW, KS_3_GBW)
1962 mag_GBW, phase_GBW, freq_GBW = calc_ramp_frequency_response_ol(ramp_2, freqs,
        N_1_GBW, Cds_1_GBW, Cgd_1_GBW, Cgs_1_GBW, gds_1_GBW, gm_1_GBW, N_2_GBW,
        Cds_2_GBW, Cgd_2_GBW, Cgs_2_GBW, gds_2_GBW, gm_2_GBW, N_3_GBW, Cds_3_GBW,
        Cgd_3_GBW, Cgs_3_GBW, gds_3_GBW, gm_3_GBW, CL)
1963
1964 comp_freq_resps_ol(designs, mag_A, phase_A, freq_A, mag_GBW, phase_GBW,
        freq_GBW)
```