



Trabajo de Fin de Máster
"Máster Universitario en Microelectrónica: Diseño y Aplicaciones de Sistemas Micro/Nanométricos"

Design and Implementation of an Asynchronous SAR Converter in 28 nm Technology for RADAR Systems

Juan Manuel Rodríguez Sánchez
Tutores: Juan Antonio Leñero Bardallo
Pablo Fernández Peramo
julio de 2024

Agradecimientos

En memoria de mi abuela, quien me enseñó todo lo que sé.

En primer lugar, quiero expresar mi profundo agradecimiento a todas las personas que me han motivado a seguir adelante durante este año. A mis compañeros de la PA32, quienes me hacían olvidar mis preocupaciones al sumergirme en discusiones sobre cómo arreglar el mundo, y a aquellos que, sin ser parte de la PA32, siempre alegraban mi día sin faltar a ningún desayuno.

A Rocío, Óscar y Ángel, por esas interminables reuniones en las que intentaba explicarles lo que estaba haciendo, esperando que pudieran ayudarme.

A mis tutores, Juan Antonio y Pablo, por estar siempre disponibles para lo que necesitara, a pesar de sus apretadas agendas. A Juan Antonio, en particular, por darme esta oportunidad y proporcionarme un trabajo que mantenía mi mente ocupada. A Pablo, por ser como ese hermano mayor con quien podía discutir cualquier inquietud.

Finalmente, a mi familia, sin la cual no estaría aquí y que ha sido mi pilar fundamental en los momentos más difíciles.

A todos ustedes, mi más sincero y profundo agradecimiento.

Abstract

This thesis addresses the design and optimization of high-speed Successive Approximation Register (SAR) Analog-to-Digital Converters (ADCs) tailored for high-frequency RADAR applications. In the face of the increasing demand for advanced data conversion systems, the focus is placed on achieving a meticulous balance between speed and resolution.

The research delves into comparator architectures, emphasizing latch-based and clocked designs, to enhance ADC performance. A MATLAB-based simulation model is developed to predict error probabilities stemming from metastability and noise, providing a robust framework for optimizing comparator speed and resolution. Empirical methods for characterizing noise in SAR ADCs are investigated using Cadence Virtuoso simulations, offering insights into precise converter design adjustments.

This thesis also includes a comprehensive analysis of SAR ADC and its hybrid architectures, addressing the challenges of high-speed data conversion. Studying its applicability to CMOS scaling.

The findings are structured to provide a holistic understanding of SAR ADC design, offering practical design guidelines and methodologies for future advancements in high-speed, high-resolution data conversion systems.

Contents

Abstract ii

List of Figures v

1 Introduction 1
- 1.1 Objectives and Scope of the Thesis . . . 6
- 1.2 Structure of the Thesis . . . 8

2 Fundamentals of Sampling Theory and Noise 10
- 2.1 Sampling Aspects . . . 10
  - 2.1.1 Shannon–Nyquist Theorem . . . 12
- 2.2 Quantization . . . 13
- 2.3 Digital-to-Analog Conversion . . . 14
- 2.4 Noise . . . 16
  - 2.4.1 Thermal Noise . . . 16
  - 2.4.2 Quantization Noise . . . 17
  - 2.4.3 Jitter Noise . . . 17
  - 2.4.4 Signal to Quantization Noise Ratio (SQNR) . . . 18
  - 2.4.5 Signal-to-Noise-and-Distortion Ratio (SNDR) . . . 20
  - 2.4.6 Differential Noise . . . 21

3 Analog-to-Digital Conversion 23
- 3.1 Successive Approximation: The Algorithm . . . 26
- 3.2 Hybrid SAR Architectures . . . 28
- 3.3 Synchronous and Asynchronous . . . 30

4 Latch-Based Comparators 35
- 4.1 Latches . . . 35
- 4.2 Strong-ARM . . . 39
- 4.3 Double-Tail comparator . . . 42
- 4.4 Triple-Tail Feedforward Comparator . . . 44

5 Case Study and Probability Model 48
- 5.1 Probability of Error . . . 50
  - 5.1.1 Metastability Only Model . . . 50
    - 5.1.1.1 Metastability Margins . . . 50
    - 5.1.1.2 Probability Mass Function . . . 52

5.1.2 Noise Only Model 54
5.1.3 Combined model of metastability and noise 57
5.2 Obtaining the Input-Referred Noise of the Comparator 59
5.3 Route Map for Designers: Steps to Design Good Comparators for RADAR Applications 63

6 Conclusion and Future Lines **66**
6.1 Conclusions 66
6.2 Future Lines 67

7 MATLAB Codes **69**
7.1 Metastability margins 69
7.2 Probability of metastability 71
7.3 Probability mass function due to metastability 73
7.4 Probability mass function due to noise 76

Bibliography **81**

List of Figures

1.1 Data rate trends over time for various communication technologies [1]. . . . 1
1.2 Basic dual down-conversion receiver 3
1.3 Scaling trends in CMOS technology, showing SNDR versus conversion rate for different technology nodes [12]. 4
1.4 Pipeline SAR ADC architecture where the conversion process is divided into three stages [12]. 5
1.5 Time-Interleaved (TI) SAR ADC architecture where multiple SAR ADCs are interleaved in time, each sampling the input signal at different intervals to effectively increase the overall sampling rate [15]. 6

2.1 Illustration of the sampling process [15]. 11
2.2 Sampling of three time-continuous signals at 20 MS/s [15]. 12
2.3 Effect of sampling frequency on signal aliasing [15]. 12
2.4 Comparison of unary and binary representation methods in digital-to-analog conversion. The unary series uses identical elements, while the binary series uses an exponential series with base 2 [15]. 15
2.5 Differential sampling for a SAR ADC [20]. The figure shows how the differential input V_{id} is sampled by two Track and Hold (T&H) circuits, one for each side of the differential signal. The differential voltage V_{xd} is then compared by the comparator, and the digital output D_{out} is produced by the logic circuitry. This differential configuration enhances noise immunity and linearity while reducing even-order harmonics. 22

3.1 Flow diagram of the SAR algorithm. 27
3.2 Circuit implementation and timing diagram of the SAR algorithm. 28
3.3 Aperture plot showing SNDR versus input frequency for various ADC architectures. 29
3.4 Timing diagrams comparing the synchronous and asynchronous topologies [17] 30
3.5 Total regeneration time and average of easy decisions as a function of ADC input voltage 33
3.6 Comparison of P_{meta} for asynchronous and synchronous architectures versus N_{DC} with $N_{Fix} = 8$ 34

4.1 Equivalent model of NMOS latch [32] 36
4.2 Comparison of time taken for latch and regenerative comparator with latch to reach to logic level 38

4.3 Strong-ARM comparator and waveforms in 28 nm technology [33]. The reset (R), first integration (I_1), second integration (I_2) and latch (L) phases are also represented . . . 40
4.4 Kickback noise paths [35] . . . 41
4.5 Double-Tail comparator and waveforms in 28 nm technology [33] . . . 42
4.6 Triple-Latch feedforward comparator [33] . . . 44
4.7 Waveforms of the Triple-Tail comparator [33] . . . 45
4.8 Comparator delay comparison between Strong-ARM, Double-Tail, Triple-Tail, and Triple-Tail with feedforward [33] . . . 47

5.1 Margin of metastability for 3-bit case . . . 50
5.2 Probability of metastability and Δy for each DAC value computed for $B = 7$, $N_{DC} = 75$, and $N_{Fix} = 8$ . . . 51
5.3 Comparison of computed probability of metastability using MATLAB function *fzero* and bisection against the theoretical expression. Computed for $B = 7$ and $N_{Fix} = 8$ . . . 52
5.4 Probability of metastability for each bit (a), and probability mass function (PMF) computed for $B = 7$, $N_{DC} = 75$, and $N_{Fix} = 8$ (b) . . . 53
5.5 Simulated cumulative distribution function due to thermal noise . . . 54
5.6 Probability mass function due to noise only (a), and zoomed region of PMF (b) computed for $B = 7$ and ENOB $= 6$ . . . 55
5.7 Convolved PMF of metastability and noise (a), and zoomed region of PMF computed for $B = 7$, $N_{DC} = 75$, $N_{Fix} = 8$, and ENOB $= 6$ (b) . . . 56
5.8 Probability of Code Error greater than X computed for $B = 7$, $N_{Fix} = 8$, and ENOB $= 6$ . . . 57
5.9 Probability of error versus sampling frequency for comparator time constants $\tau = 10$ ps and 3 ps. . . . 58
5.10 Example of transient simulation with noise for $v_{\text{in}} = 0.25$ mV, illustrating the decision errors when the input is reasonably low. . . . 59
5.11 Adjusted CDF . . . 60
5.12 Feedforward Triple-Tail schematic . . . 61
5.13 Illustrative position of observation time for PAC as indicated in [41] . . . 61

CHAPTER 1

Introduction

The accelerating growth of technological advancement demands efficient and precise data conversion systems, particularly Analog-to-Digital Converters (ADCs) and Digital-to-Analog Converters (DACs). These converters serve as the critical link between the analog and digital realms, enabling the seamless integration of digital systems with real-world analog signals. ADCs, in particular, play a crucial role in high-frequency applications, converting analog signals into digital data for further processing and analysis.

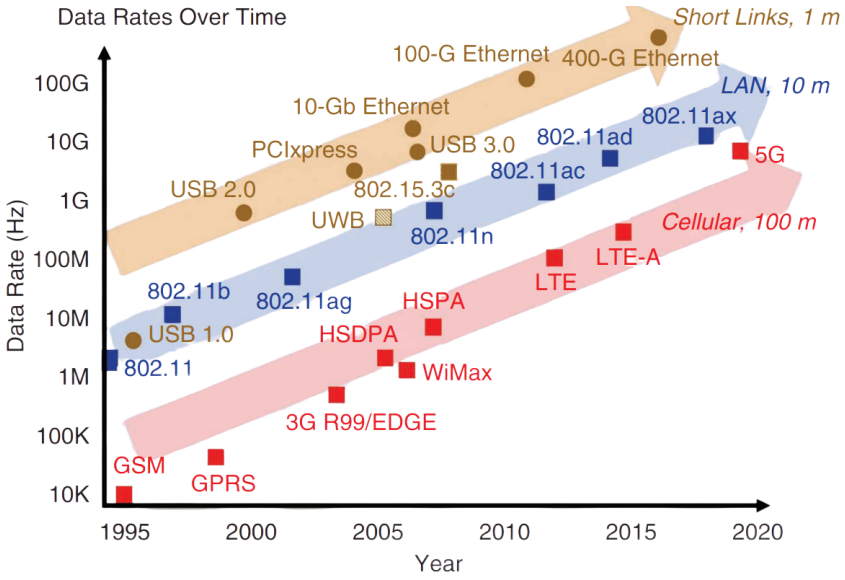


Figure 1.1: Data rate trends over time for various communication technologies [1].

Figure 1.1 illustrates the trends in data rates over time for different communication technologies. As illustrated, there has been a significant increase in data rates from the

early days of GSM and GPRS to the current high-speed technologies like 100-Gb Ethernet and 5G mobile networks. GSM (Global System for Mobile Communications) was one of the first digital cellular technologies, providing improved voice quality and higher capacity compared to analog systems. GPRS (General Packet Radio Service) enhanced GSM by offering packet-switched data services, enabling mobile internet access and multimedia messaging. This trend underscores the growing demand for higher-speed data conversion systems like ADCs to support modern communication systems' increasing data rates. Consequently, developing advanced ADC architectures is crucial to keep pace with rapid communication technology advancements.

ADCs and DACs find applications across a wide range of fields. In the communications sector, they are essential for wireless systems, modems, and broadcasting signal processing [2]. In the medical technology domain, ADCs are employed to convert analog signals from sensors into digital images in MRI and CT scanners, thereby enabling advanced diagnostic capabilities [3]. In industrial automation, these converters are used to monitor and control processes through precise data conversion from sensors, enhancing efficiency and reliability in manufacturing [4]. Furthermore, in the automotive industry, ADCs are integral components of RADAR and LiDAR systems, which are pivotal for the development of autonomous driving technologies [5].

In the particular case of RADAR systems, ADCs convert reflected signals from the RADAR sensor into digital data for processing. This data is used to detect objects, measure their speed, and determine their distance. The ADC's accuracy and speed directly impact RADAR system performance, making the choice of ADC architecture crucial.

Figure 1.2 illustrates the block diagram of a down converter receiver architecture commonly used in RF systems, including RADAR applications. The process begins with the antenna capturing the RF signal and passing it through the first band-pass filter (BF_1) to reject out-of-band signals. The filtered signal is then amplified by a low noise amplifier (LNA), which enhances the signal with minimal noise, thereby improving the signal-to-noise ratio (SNR).

Following amplification, the mixer combines the amplified RF signal with a local oscillator signal to convert the RF signal to an intermediate frequency (IF). The second

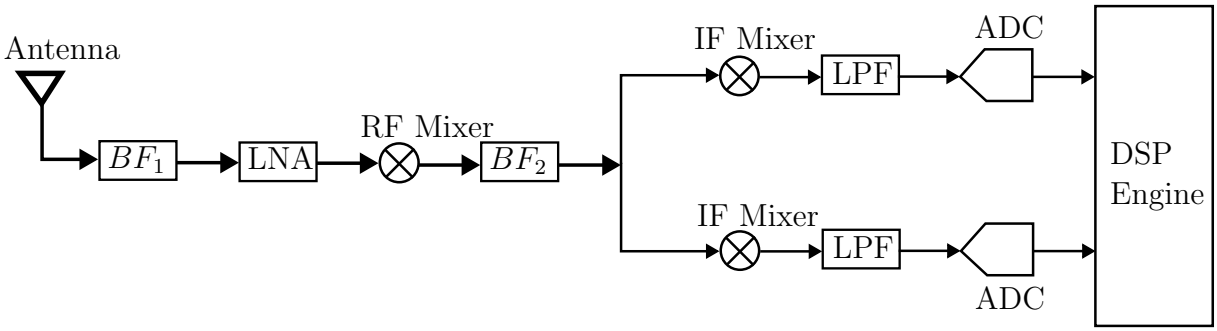


Figure 1.2: Basic dual down-conversion receiver

band-pass filter (BF_2) filters the mixed signal, selecting the desired IF. If further down conversion is required, IF mixers are employed to convert the IF signal to baseband frequency. Subsequently, a low-pass filter removes high-frequency components. The resulting signal is then converted by the ADC, and the output is sent to a digital signal processing engine (DSP) for processing the digitized signal.

In RADAR applications, the RF signal typically has a frequency in the tens of gigahertz range. After IF down conversion, the resulting signal frequency ranges from gigahertz to hundreds of megahertz. Further down conversion can bring the frequency down to the megahertz range. Introducing the ADC earlier in the receiver chain is advantageous, as it allows for the use of digital filters instead of analog ones. This transition eliminates imbalance-related distortions that arise from mismatches between analog components and simplifies the design process. However, this approach requires ADCs that can operate at higher speeds, underscoring the necessity of high-speed ADCs in RADAR applications, and even enabling the possibility of direct RF sampling [6].

RADAR systems are widely used in various applications, including:

- **Automotive RADAR:** Used for advanced driver assistance systems (ADAS) and autonomous driving to detect obstacles [7], monitor traffic, and assist in parking.
- **Aerospace and Defense:** Critical for navigation [8], surveillance, and targeting in military aircraft, missiles, and ground-based defense systems.
- **Weather Monitoring:** Employed in weather RADAR systems [9] to detect precipitation, measure storm intensity, and track weather patterns.

- **Marine Navigation:** Used in shipborne RADAR systems to detect other vessels [10], navigate in poor visibility, and avoid collisions.

- **Air Traffic Control:** Essential for monitoring and managing aircraft movements [11] in and around airports to ensure safety and efficiency.

The wide range of RADAR system applications underscores the ever-present necessity of ADCs operating at high speeds with high resolution. Continuous CMOS technology scaling contributes to these high-speed ADCs by providing higher transistor densities and faster switching rate. This allows for more complex and efficient ADC architectures on a single chip, enhancing performance while reducing power consumption and cost.

Figure 1.3 illustrates CMOS scaling trends, showing how different technologies achieve varying Signal-to-Noise-and-Distortion Ratios (SNDR) at different conversion rates. As conversion rates increase, achieving high SNDR becomes more challenging, particularly in advanced nodes like $\leq$ 20nm. This trade-off highlights the need to optimize speed and accuracy in ADC design, balancing these competing demands carefully.

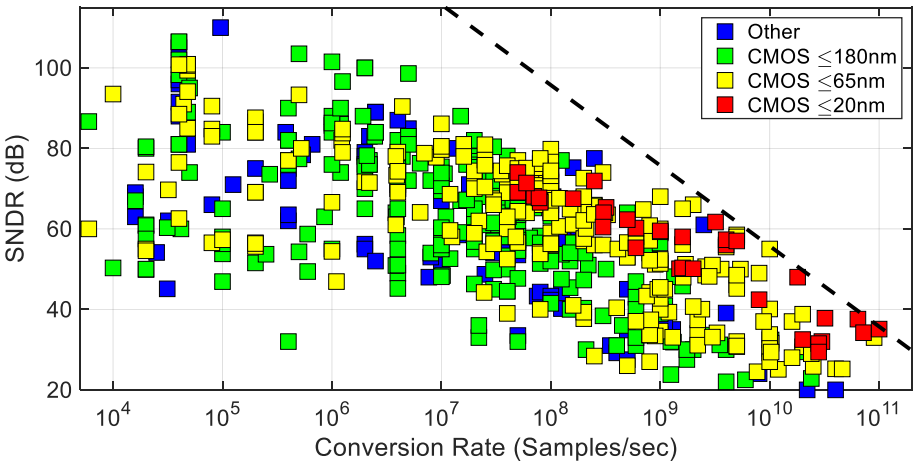


Figure 1.3: Scaling trends in CMOS technology, showing SNDR versus conversion rate for different technology nodes [12].

Of the many ADC architectures, SAR ADCs stand out due to their excellent power efficiency and high resolution, coupled with decent speed. For higher speed applications, SAR ADCs can be implemented in pipeline or subranging configurations, or through time-interleaving techniques.

Pipeline SAR ADCs enhance speed and resolution [13] by segmenting the conversion process into multiple stages (Figure 1.4). Each stage simultaneously performs a part of the conversion, but the determination of the bits is made sequentially. This architecture provides higher throughput and reduced latency. Unlike a simple flash converter, pipeline stages must perform additional tasks within a single clock cycle, including reference value subtraction and signal amplification. The settling time for each pipeline stage is influenced by the operational amplifier's unity-gain bandwidth, the gain factor dictated by the capacitor ratio, and the number of settling time constants required for an exponential function to achieve the desired accuracy. Increased resolution demands more time, thus reducing the sample rate and Nyquist bandwidth. Moreover, gate delays, wiring delays, and worst-case design margins further constrain the maximum speed of pipeline SAR ADCs.

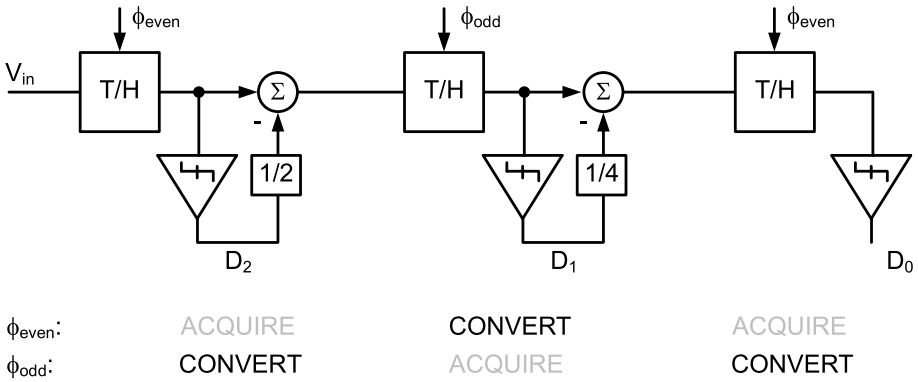


Figure 1.4: Pipeline SAR ADC architecture where the conversion process is divided into three stages [12].

To address these limitations, Time-Interleaved (TI) SAR ADCs have been developed [14]. TI SAR ADCs achieve high-speed conversion by interleaving multiple SAR ADCs in time, each sampling the input signal at different intervals. This technique increases the overall sampling rate by a factor of N, depending on the number of interleaved channels. In a TI SAR ADC, the same device works at a fraction of the full sampling frequency f_s, with a multiplexer interleaving these signals in time (Figure 1.5). While this approach significantly boosts the sampling rate, it introduces sensitivity to channel mismatches such as offset, gain, timing, and bandwidth, which must be carefully managed to ensure accurate conversion.

While pipeline and TI SAR ADCs both aim to enhance speed and resolution, they do so

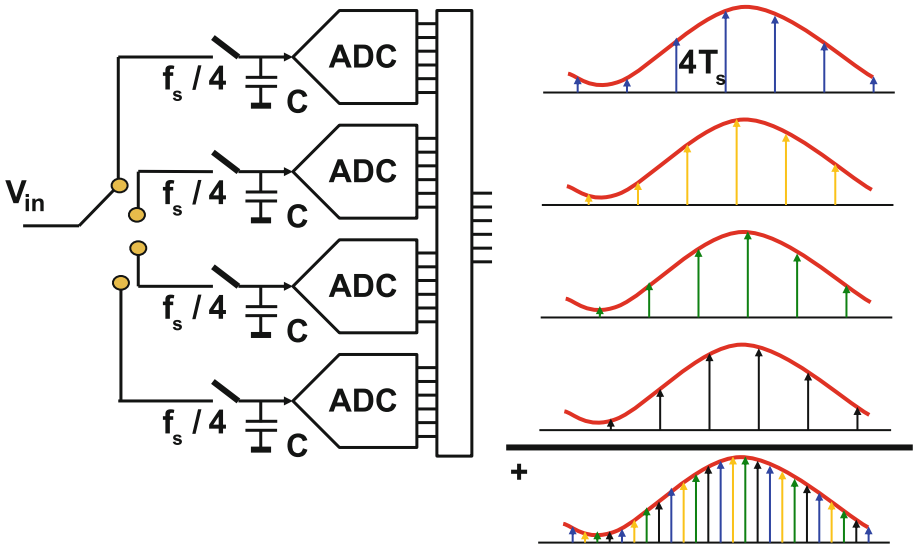


Figure 1.5: Time-Interleaved (TI) SAR ADC architecture where multiple SAR ADCs are interleaved in time, each sampling the input signal at different intervals to effectively increase the overall sampling rate [15].

through different means. Pipeline SAR ADCs leverage concurrent stages and sequential bit determination, whereas TI SAR ADCs interleave multiple ADCs to achieve higher overall sampling rates at the cost of lower general resolution. Each approach has its advantages and limitations, which must be carefully considered in high-speed ADC system design.

As CMOS technology scales down, analog circuits' gain degrades due to reduced intrinsic gain of transistors. Calibration techniques counteract this by improving gain and correcting offsets [16]. Calibration can adjust the ADC's gain, offset, and linearity, enhancing performance. However, these techniques introduce trade-offs in area, power, and speed. Calibration circuits require additional chip area and power, and the calibration process can add latency to the conversion. Despite these trade-offs, calibration is necessary for most modern high-speed ADC designs to improve accuracy and performance.

1.1 Objectives and Scope of the Thesis

This thesis focuses on designing and optimizing Successive Approximation Register (SAR) Analog-to-Digital Converters (ADCs) for high-frequency RADAR applications. The primary goals are to achieve a balance between high speed and high resolution in ADC

design.

A critical aspect of this process is selecting the appropriate SAR architecture for RADAR applications. The performance limits of SAR ADCs are significantly influenced by the comparator. Therefore, a thorough analysis and selection of promising comparator architectures, particularly those based on latches and clocked designs, are essential for achieving high-speed performance.

Latch-based comparators are suited for high-speed applications due to their fast response times and robust performance. Additionally, the use of clocked circuit is crucial for low power consumption. By focusing on these architectures, we aim to enhance SAR ADCs' efficiency and accuracy in RADAR systems.

A critical aspect of this thesis involves the development of a MATLAB model to simulate the probability of error arising from metastability and noise as a function of comparator speed. This allows for better leverage of the speed and the resolution of the converter. While this approach is not entirely novel, recreating the work of Boirs Murmann et al. [17] allows for future modifications that can enhance the model's applicability to specific electrical circuits, thereby leading to future innovations in the analog design process.

Following the development of the MATLAB model, the thesis studies two empirical methods for obtaining the input-referred noise of the comparator through simulation using Cadence Virtuoso. Once the model is developed, this detailed analysis aids in fine-tuning the procedure, ensuring accurate simulations and more effective adjustments to the design of SAR ADCs for high-frequency RADAR applications.

Such analysis is crucial for understanding and mitigating potential error sources, improving SAR ADCs' reliability and performance. By enhancing these findings for practical circuit design, this thesis aims to advance SAR ADCs' understanding and application in high-speed RADAR systems.

The objectives of this thesis are summarized as follows:

- Investigate and select suitable comparator architectures based on latches and clocked designs for high-speed SAR ADCs in RADAR applications.

- Provide a simulation model to estimate the probability of error, considering noise and metastability.
- Evaluate different procedures for characterizing converter noise contributions.
- Offer design guidelines and methodologies for future development of high-performance SAR ADCs in various high-speed applications.

This work aims to make a significant contribution to microelectronics by gathering effective solutions and methodologies for designing and implementing high-performance SAR ADCs in RADAR systems. The advancements detailed in this thesis will help address the growing demand for efficient, accurate, and reliable data conversion systems in modern electronic applications.

1.2 Structure of the Thesis

The structure of this thesis is organized as follows:

- **Chapter 1: Introduction**
  This chapter provides an overview of the importance of ADCs in high-frequency applications, particularly in RADAR systems. It outlines the thesis objectives and scope, highlighting advancements in SAR ADC architectures.
- **Chapter 2: Fundamentals of Sampling Theory and Noise**
  This chapter explains the fundamental concepts of analog-to-digital conversion, covering ADC operation basics and noise theory, essential for understanding subsequent chapters.
- **Chapter 3: Analog-to-Digital Conversion**
  This chapter details the characteristics of different types of ADCs, with a particular emphasis on the behavior and outcomes of Successive Approximation Register (SAR) ADCs and hybrid SAR architectures. Additionally, it explores the key distinctions between synchronous and asynchronous SARs.

- **Chapter 4: Latch-Based Comparators**

  This chapter explores various latch-based comparator architectures, providing a comparative analysis of their advantages and limitations in high-speed, high-resolution SAR ADCs.

- **Chapter 5: Case Study and Probability Model**

  This chapter presents a detailed case study of a specific comparator architecture, simulated using Cadence Virtuoso. It describes methods for obtaining input-referred noise and creating a simulation environment to estimate the probability of error due to metastability and noise.

- **Chapter 6: Conclusion and Future Work**

  The final chapter summarizes the key findings, discusses the implications of the research, and suggests future directions based on the results of this study.

- **Appendix: MATLAB Codes**

  The appendix includes MATLAB codes used in the simulations and analyses presented in the thesis, supporting reproducibility and facilitating further research.

This structured approach aims to provide a comprehensive examination of SAR ADCs, focusing on their design, optimization, and application in high-frequency RADAR systems. Each chapter builds on the previous one, developing a deep understanding of the subject progressively.

CHAPTER 2

Fundamentals of Sampling Theory and Noise

2.1 Sampling Aspects

In microelectronics, the most common sampling circuit includes a switch and a capacitor. Sampling involves capturing the value of a signal at specific time intervals, defined by the sampling frequency f_s. These intervals are determined as follows:

$$t = \frac{n}{f_s} = nT_s, \quad n = -\infty, \ldots, -1, 0, 1, \ldots \infty \tag{2.1}$$

Here, T_s represents the time interval between consecutive samples. Between these sampling instances, the signal's value is not mathematically defined, although internal operations are performed on the sampled data during this interval for time-discrete signal processing. If T_s remains constant, the continuous-time signal is uniformly sampled.

The Dirac delta function plays a crucial role in the sampling process. This function is essential in defining the sampling frame, which can be represented as the following sequence of pulses:

$$\sum_{n=-\infty}^{\infty} \delta(t - nT_s) \tag{2.2}$$

Figure 2.1 shows a signal $A(t)$ sampled at f_s. In the frequency domain, this signal has a bandwidth from 0 Hz to BW. Sampling involves multiplying $A(t)$ by the above pulse sequence, resulting in a discrete-time signal:

$$A_s(t) = \sum_{n=-\infty}^{\infty} A(t-\tau)\delta(\tau - nT_s) \tag{2.3}$$

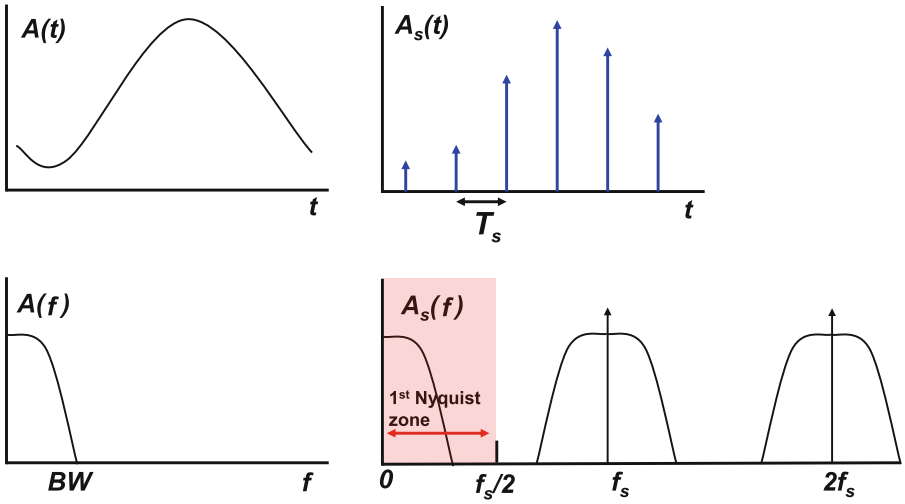


Figure 2.1: Illustration of the sampling process [15].

The Fourier transform of the sampled signal $A(t)$ results in the original signal's spectrum repeating at intervals of the sampling frequency $f_s = 1/T_s$, resulting in:

$$A_s(f) = \sum_{k=-\infty}^{\infty} A(f - kf_s) \tag{2.4}$$

A significant implication of sampling theory is that frequency components equidistant from multiples of f_s will overlap in the sampled data. For instance, $f_1 = 4f_s + 5$ kHz and $f_2 = 7f_s - 5$ kHz both produce a component at 5 kHz after sampling.

This sampling property is useful for down-mixing communication signals but can also lead to aliasing, where unwanted signals overlap with desired ones. Figure 2.2 illustrates how different sine waves (1 MHz, 19 MHz, 39 MHz) sampled at 20 MS/s can result in the same sampled data.

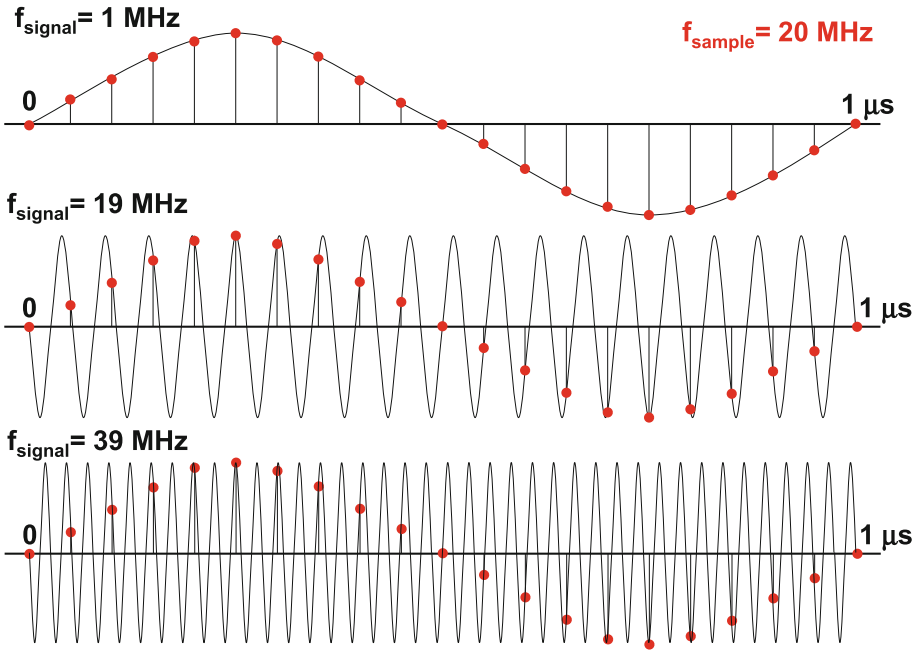


Figure 2.2: Sampling of three time-continuous signals at 20 MS/s [15].

2.1.1 Shannon–Nyquist Theorem

The Nyquist criterion, also known as the Nyquist-Shannon [18] sampling theorem, states that to accurately reconstruct a continuous-time signal from its samples, the sampling frequency f_s must be at least twice the signal's bandwidth. Mathematically, this is expressed as:

$$f_s > 2BW \tag{2.5}$$

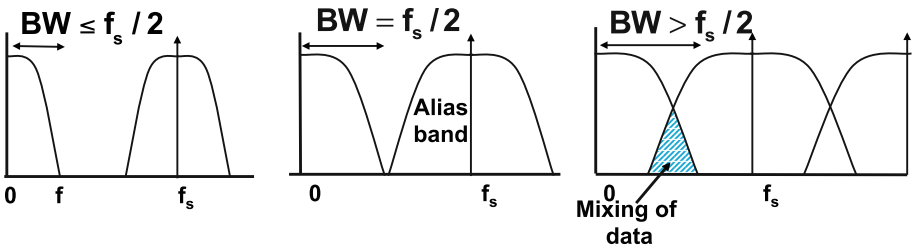


Figure 2.3: Effect of sampling frequency on signal aliasing [15].

Thus, the Nyquist criterion limits usable bandwidth to half the sampling rate. This choice ensures that there is no aliasing, which is the distortion that occurs when higher frequency components are misinterpreted as lower frequencies during the sampling process,

as seen in Figure 2.3.

2.2 Quantization

During quantization, the signal level at each sampling moment is compared with a set of discrete analog reference values, each corresponding to specific digital codes. The amplitude of the input signal is truncated to the closest reference value, thus discretizing the signal. Truncation errors, or quantization errors, arise due to the discretization of both time and amplitude in analog-to-digital conversion. The power of these errors fundamentally limits the precision of the conversion.

Binary coding is widely used in digital systems for amplitude-discrete representations due to its seamless integration with converter designs that employ switches. In such systems, each binary digit, or "bit," can be either 0 or 1. The term word width or resolution denotes the number of bits B needed to convert an analog sample into a digital word. With B bits, a binary signal can represent 2^B possible digital values, which define the quantization levels for a continuous amplitude signal. A common binary representation is the "straight binary representation":

$$B_s = \sum_{i=0}^{B-1} b_i 2^i = b_0 2^0 + b_1 2^1 + b_2 2^2 + \cdots + b_{B-1} 2^{B-1} \tag{2.6}$$

In this context, the coefficient b_{B-1} associated with the highest 2^{B-1} term is referred to as the "Most Significant Bit" (MSB), while the coefficient b_0 linked to the lowest 2^0 term is known as the "Least Significant Bit" (LSB). When the LSB changes from 0 to 1, it selects the next higher quantization level, thereby defining the step size between levels as "one LSB".

Takes as an example the voltage range from 0 to 0.8 V that is divided into 8 intervals by 7 decision levels at 0.1 V, 0.2 V, ..., 0.7 V. This setup yields a resolution of $B = 3$ bits. The difference between the original signal and its quantized representation defines the quantization error. However, there is some ambiguity in both literature and practice regarding the converter's operating range. Some sources assert that the full-scale range

is equal to the reference voltage, expressed as $V_{range} = 2^B \times V_{LSB} = 0.8$ V. Alternatively, others suggest that the range extends from 0.1 V to 0.7 V, ensuring the signal is always within 0.1 V (one LSB) of a decision level, leading to $V_{range} = (2^B - 1) \times V_{LSB}$. In the latter case, signals below 0.1 V are encoded as "000," whereas signals above 0.7 V are encoded as "111". Nonetheless, this difference becomes insignificant at higher resolutions.

2.3 Digital-to-Analog Conversion

In the context of the data conversion chain, digital-to-analog converters (DACs) [19] fulfill two essential functions. Firstly, a DAC is always required to convert the digital signal back into the analog domain. On top of that, every ADC relies on some form of digital-to-analog conversion to perform its operation effectively.

Digital-to-analog conversion involves using a reference quantity that is divided into fractions equal to the LSB or its multiples. The two main approaches for combining these basic units are unary and binary representations.

In the unary representation, 2^B identical elements (such as resistors, capacitors, or currents) are used in the conversion process. However, the area required for a unary coded converter increases exponentially with B since each element needs a switch, making it impractical for resolutions much higher than 10–12 bits.

The unary numerical value is created as follows:

$$B_u = \sum_{i=0}^{i=2^B-1} b_i = b_0 + b_1 + b_2 \ldots + b_{2^B-1} \tag{2.7}$$

where each coefficient b_i equals either "0" or "1". The analog equivalent is formed by summing copies of the physical equivalent A_{LSB} of an LSB:

$$A_u = \sum_{i=0}^{i=2^B-1} b_i A_{LSB,i} = b_0 A_{LSB,0} + b_1 A_{LSB,1} + b_2 A_{LSB,2} \ldots + b_{2^B-1} A_{LSB,2^B-1} \tag{2.8}$$

In this representation, a value that is 1 LSB higher can be easily generated by adding

a new unitary element to the previously selected ones, as illustrated in the upper part of Figure 2.4.

In contrast to unary representation, binary representation employs elements arranged to generate voltages or currents in an exponential sequence. Given that a basic switch has two positions, 2 is a practical choice for the base. The binary numerical value is formed as follows:

$$B_b = \sum_{i=0}^{i=B-1} b_i 2^i = b_0 + b_1 2^1 + b_2 2^2 \ldots + b_{B-1} 2^{B-1} \tag{2.9}$$

or in the physical domain:

$$A = \sum_{i=0}^{i=B-1} b_i A_{LSB+i} = b_0 A_{LSB} + b_1 A_{LSB+1} + b_2 A_{LSB+2} \ldots + b_{B-1} A_{MSB} \tag{2.10}$$

A binary coded converter can represent all integers from 0 to $2^B - 1$ by switching weighted elements on and off. For instance, transitioning from 15 to 16 requires turning off all previously selected elements and switching on the element with the value 16, as depicted in the lower part of Figure 2.4.

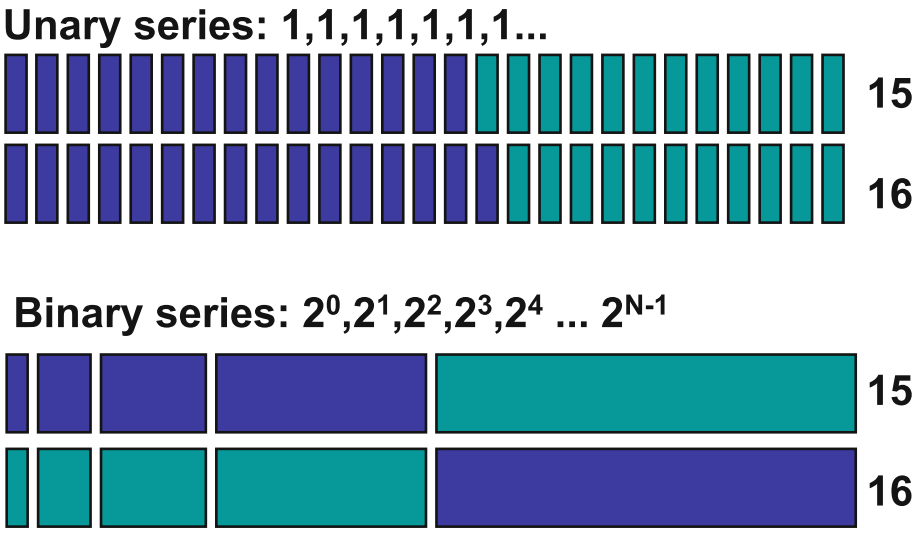


Figure 2.4: Comparison of unary and binary representation methods in digital-to-analog conversion. The unary series uses identical elements, while the binary series uses an exponential series with base 2 [15].

While the unary method guarantees monotonicity, the binary method is vulnerable to non-monotonicity if a higher exponent element is smaller than the sum of the lower exponential elements (see Figure 2.4). This non-monotonicity can lead to instability and potential oscillations in control loops.

2.4 Noise

The elemental sampling circuit introduced in the first section, consisting of a switch and a capacitor, presents several challenges in physical implementation. The sampling process modulates the entire continuous-time spectrum into the $0 - f_s/2$ band and all Nyquist zones. If the input signal is influenced by $1/f$ noise and/or thermal noise, these components will manifest in the samples. Effective anti-alias filtering can eliminate noise and other unwanted signals. Generally, the basic sampling circuit contributes minimally to $1/f$ noise, with follower or opamp circuits being the main sources. However, it introduces non-negligible thermal noise to any processed signal.

2.4.1 Thermal Noise

Thermal noise, resulting from the random motion of electrons in conductors, is introduced by a noise source with resistance R. This phenomenon is explained by the Equipartition Theorem in physics, which states that thermodynamic energy is equivalent to the electrical noise energy on a capacitor. Thermal noise is fundamentally characterized by its dependence on temperature and is described by the formula:

$$\frac{kT}{2} = \frac{Cv_{C,\text{noise}}^2}{2} \implies v_{C,\text{noise}} = \sqrt{\frac{kT}{C}} \tag{2.11}$$

In this context, the resistor value does not directly affect this first-order expression. While a higher resistor value increases the noise power density, it also decreases the relevant bandwidth. Consequently, the noise voltage is only determined by the capacitor value. For instance, a 1 pF capacitor produces a noise voltage of 65 μV_{rms}.

2.4.2 Quantization Noise

Quantization and sampling are distinct processes. Sampling converts a continuous-time signal into discrete-time samples, while quantization maps the amplitude of these samples to a finite set of discrete levels. Quantization error arises from approximating a continuous signal to discrete levels and is inherently non-linear. This error becomes significant, particularly for analog-to-digital converters (ADCs) with resolutions exceeding 6 bits. Analyzing the power contained in the error signal is crucial for evaluating ADC performance.

In low-resolution ADCs, quantization error is highly dependent on the input signal, resulting in distortion. However, with a sufficiently large number of quantization levels (higher than 7 bits), these errors can be statistically approximated, leading to a more accurate representation of the error signal.

Despite the constraints imposed by the assumptions of a uniform distribution and equally spaced quantization levels, the following formula remains sufficiently accurate for many applications in order to approximate quantization error as white noise:

$$\text{Quantization power} = \frac{A_{LSB}^2}{12} = \left(\frac{\text{full-scale}}{2^B\sqrt{12}}\right)^2 \tag{2.12}$$

Here, A_{LSB} denotes the physical magnitude of the least significant bit, "full-scale" represents the analog range of the conversion, and B refers to the resolution.

The formula above illustrates that quantization power is inversely proportional to the resolution. As the resolution increases, the quantization error diminishes, resulting in a more accurate representation of the analog signal. Understanding and minimizing quantization error is essential for optimizing the performance of high-resolution ADCs employed in applications requiring precise signal conversion.

2.4.3 Jitter Noise

In previous discussions, we have assumed that sampling moments are defined with infinite precision. However, in practical applications, two unwanted mechanisms can disrupt this

ideal scenario: clock skew and jitter. Each has distinct root causes and requires different mitigation strategies.

Clock skew refers to a deterministic, fixed delay between multiple clock paths. For instance, variations in the propagation path lengths of clocks can result in skew. This issue is rarely problematic in single-path sampling systems. Nonetheless, in multi-path systems, clock trees must be meticulously designed to address this concern.

Table 2.1: Jitter specifications of some commercially available parts [15].

Part	**Description**	**Jitter**
2011	Quartz 50–170 MHz	3 ps_{rms}
8002	Programmable oscillator	25 ps_{rms}
1028	MEMS+PLL combi 100 MHz	95 ps_{rms}
6909	RC oscillator 20 MHz	0.2%
555	RC oscillator/timer	>50 ns_{rms}

Jitter, on the other hand, arises because time-defining signals have limited bandwidths, resulting in non-ideal rising edges. Devices that generate noise, such as oscillators and PLLs, add additional noise to these edges. As a result, the sample moments deviate from their optimal positions, causing the sampling circuit to capture the signal at varying times.

Jitter can also be approximated by a flat spectrum, where the variance of the amplitude σ_A due to jitter is given by:

$$\sigma_A^2 = \omega^2 A_{rms}^2 \tau_{jitter}^2 \tag{2.13}$$

where ω is the angular frequency of operation, A_{rms} is the RMS amplitude of the signal whose SNR we are computing, and τ_{jitter} is the RMS jitter value. Table 2.1 presents typical jitter values for several commercially available components.

2.4.4 Signal to Quantization Noise Ratio (SQNR)

When the quantization error can be approximated as “white noise”, its power often becomes the dominant noise source up to a 14-bit resolution.

Many system definitions employ sine wave-related specifications. Consequently, sinusoidal signals are the standard for characterizing the performance of analog-to-digital converters. The signal-to-quantization-noise ratio (SQNR) is thus expressed as the ratio between the power of the input sinusoidal signal and the power of the quantization noise:

$$\text{SQNR} = 10\log_{10}\left(\frac{\text{Signal power}}{\text{Quantization error power}}\right) = 20\log_{10}\left(\frac{V_{signal,rms}}{V_{Q,rms}}\right) \tag{2.14}$$

A signal $\hat{A}\sin(\omega t)$ swings over the entire range $2^B A_{LSB}$, so $\hat{A} = 2^{B-1}A_{LSB}$. Substituting the signal power and quantization error power gives the signal-to-quantization-noise ratio:

$$\text{Signal power} = \frac{1}{T}\int_{t=0}^{t=T}\hat{A}^2\sin^2(\omega t)dt = \frac{\hat{A}^2}{2} = \frac{2^{2B}A_{LSB}^2}{8} \tag{2.15}$$

$$\text{Quantization power} = \frac{A_{LSB}^2}{12} \tag{2.16}$$

$$\text{SQNR} = \frac{\text{Signal power}}{\text{Quantization power}} = \frac{3}{2}2^{2B} \tag{2.17}$$

$$\text{SQNR in dB} = 10\log_{10}\left(\frac{3}{2}2^{2B}\right) = 1.76 + B \times 6.02\,\text{dB} \tag{2.18}$$

This formula is a commonly used approximation for designing ADCs.

Since the quantization power is uniformly distributed over the Nyquist bandwidth $0 - f_s/2$ and the bandwidth of interest (BW) may be significantly smaller, a higher SQNR can be achieved within that bandwidth:

$$\text{SQNR} = 1.76 + B \times 6.02 + 10\log_{10}\left(\frac{f_s/2}{BW}\right)\,\text{dB} \tag{2.19}$$

This idea is used under the term “oversampling.”

2.4.5 Signal-to-Noise-and-Distortion Ratio (SNDR)

The Signal-to-Noise-and-Distortion Ratio (SNDR), also referred to as SINAD, represents the ratio of the signal power to all unwanted components, including quantization errors, thermal noise, and distortion. SNDR is measured when both the signal and noise are present within the measurement time interval. Signal-related noise contributions, such as those from jitter, are included in the total noise. The Dynamic Range (DR) defines the usable range, while the Signal-to-Noise Ratio (SNR) specifies the quality within that range.

The SNDR is given by:

$$\text{SNDR} = 10\log_{10}\left(\frac{\text{Power of 1st harmonic}}{\text{Power of all unwanted components}}\right) \tag{2.20}$$

With further specification and remembering that $A_{rms} = \hat{A}/\sqrt{2}$, where $A_{rms} = A_{peak-to-peak}/2\sqrt{2}$:

$$\text{SNDR} = 10\log_{10}\left(\frac{\hat{A}^2/2}{10^{HD2/10} + \frac{kT}{C} + \frac{A_{LSB}^2}{12} + \omega^2 A_{rms}^2 \tau_{jitter}^2 + \sigma_{int}^2}\right) \tag{2.21}$$

Here, the input-referred noise due to comparator thermal noise is denoted as σ_{int}. Additionally, the terms related to the quantization error, thermal noise due to sampling, and jitter are added. The equation can be extended to account for higher-order harmonic, spurious signals, and other noise contributions if needed.

The Effective Number of Bits (ENOB), is a measure of the performance of an analog-to-digital converter (ADC). It quantifies the resolution and accuracy of the ADC by comparing its actual performance to that of an ideal ADC with a given number of bits. ENOB is calculated by using the SNDR in the Formula 2.18:

$$\text{ENOB} = \frac{\text{SNDR} - 1.76}{6.02} \tag{2.22}$$

2.4.6 Differential Noise

In many electronic systems, signals are preferably handled by differential circuits (Figure 2.5). This approach offers greater flexibility in common-mode signal selection, cancels even-order harmonics, and suppresses power supply and substrate noise. The distinctions between single-ended and differential sampling in the case of thermal noise are outlined as follows:

In a single-ended configuration, the signal power for a sine wave with amplitude V_{in} is $\frac{V_{in}^2}{2}$, and the noise power is $\frac{kT}{C}$. The signal-to-noise ratio (SNR) is given by:

$$\mathrm{SNR}_{single} = \frac{V_{in}^2}{2kT/C} \tag{2.23}$$

In a differential sampling circuit, the signal is sampled twice, resulting in the signal power being quadrupled to $\frac{(2V_{in})^2}{2}$, while the noise power is doubled to $\frac{2kT}{C}$. This configuration provides a 2x improvement in SNR, equivalent to a 3 dB gain. The signal-to-noise ratio for the differential configuration is:

$$\mathrm{SNR}_{diff} = \frac{V_{in}^2}{kT/C} \tag{2.24}$$

When evaluating the Signal-to-Noise-and-Distortion Ratio (SNDR) in differential applications, it is crucial to note that noises introduced by the Track and Hold (T&H) circuit, such as thermal noise, quantization noise, or jitter, are doubled because two T&H circuits are employed. However, the input-referred noise, which is associated with the operation of the comparator, is considered only once since the comparator is activated a single time.

Thus, for differential applications, the SNDR can be expressed as:

$$\mathrm{SNDR} = 10\log_{10}\left(\frac{\hat{A}^2}{10^{HD3/10} + \frac{kT}{C} + \frac{A_{LSB}^2}{12} + \omega^2 A_{rms}^2 \tau_{jitter}^2 + \sigma_{int}^2/2}\right) \tag{2.25}$$

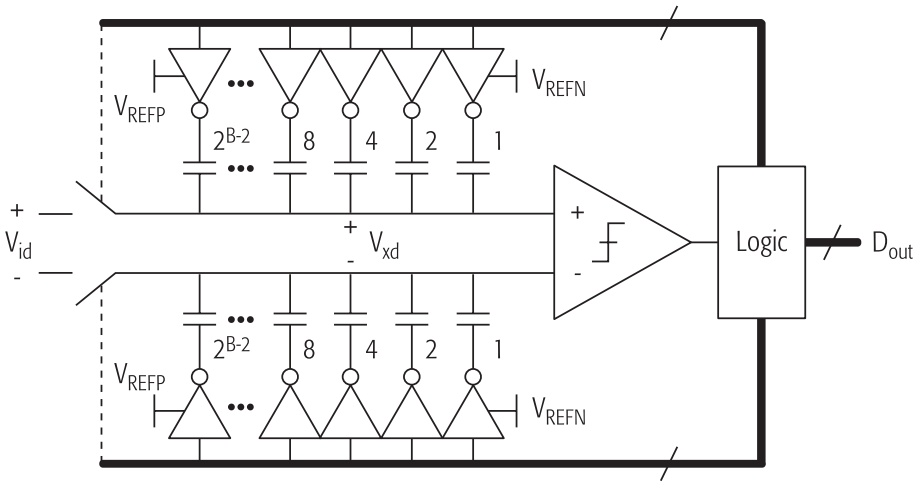


Figure 2.5: Differential sampling for a SAR ADC [20]. The figure shows how the differential input V_{id} is sampled by two Track and Hold (T&H) circuits, one for each side of the differential signal. The differential voltage V_{xd} is then compared by the comparator, and the digital output D_{out} is produced by the logic circuitry. This differential configuration enhances noise immunity and linearity while reducing even-order harmonics.

CHAPTER 3

Analog-to-Digital Conversion

Analog-to-digital converters (ADCs) are fundamental components in modern electronic systems, enabling the conversion of analog signals into digital data for further processing. This chapter covers the basic theory of ADC classification, focusing on Successive Approximation Register (SAR) ADCs.

ADCs can be categorized into four main types based on their conversion timing. These types, known as "Nyquist-rate converters," produce a single digital output value for each input level and convert a bandwidth close to $f_s/2$. They tend to operate at the speed limit of their architecture and technology. Nyquist-rate analog-to-digital converters include:

- **Flash Converters (Parallel Search Converters)**: Flash converters [21] perform the conversion instantaneously, requiring the input signal and all necessary reference levels to be present simultaneously. They do not require an explicit track-and-hold (T&H) or sample-and-hold (S&H) circuit, as they incorporate latches in each comparator. Flash converters are known for their exceptional speed, making them suitable for applications requiring high speed and modest accuracy. However, providing 2^B reference levels and comparators results in exponential increases in area and power consumption with increasing resolution B. In practical cases, flash architectures are limited to about 6 bits of resolution. This way, trying to achieve resolutions of $B \geq 6$ leads to large and power-hungry circuits.

- **Sequential Search Converters**: These converters approximate the signal value by

selecting new reference levels at each clock transition based on previously processed information. The transition between steps is usually based on binary representation, as discussed in the previous chapter. A key design choice is whether to use the same stage for processing all steps, or separate stages for each subsequent resolution step. The latter approach, known as pipelining, allows multiple samples to be processed simultaneously, although the total processing time for one sample remains the same.

Examples in this category include successive approximation conversion [22], pipeline conversion [23], and multi-step conversion [24]. The combination of high accuracy and relatively high speed makes these converters suitable for many industrial and communication applications.

- **Linear Search Converters**: Linear search converters generate all potential conversion levels in increasing or decreasing order and compare them to the input signal, resulting in extremely slow conversion speeds. However, these converters require minimal hardware and are tolerant of various forms of component variation. An example is the dual-slope converter [25]. The robustness of these converters makes them popular for low-speed and harsh conditions, such as image sensors.

Oversampled converters, on the other hand, differ in their sample rate and implementation. These converters sample at a rate much higher than required by the Nyquist theorem, using only a few reference levels. Accuracy is achieved through precise timing pulses. Sigma-Delta modulators ($\Delta\Sigma$ modulators) are a common type of oversampled converter. These modulators generate a bitstream from which digital filters can extract the Nyquist samples, providing accuracy over a larger number of samples at the cost of lower output speed.

Table 3.1: Comparison of ADC Types

Type	**Cycles per Conversion**	**Complexity**	**Opamps**
Flash	~ 1	High $\sim 2^B$	No
Pipeline	~ 1	Moderate $\sim B$	Yes
SAR	$\sim B$	Low ~ 1	No

Table 3.1 compares three common types of ADCs: Flash, Pipeline, and Successive Approximation Register (SAR) ADCs. Each type has distinct characteristics in terms of

cycles per conversion, complexity, and the use of operational amplifiers (opamps).

As said before, flash ADCs are the fastest, requiring only one cycle per conversion. However, many communications applications require an effective number of bits (ENOB) greater than 6, eliminating this topology as a potential solution. In fact, the flash ADCs plateaued more than 10 years ago (at ~ 7.5 GS/s).

Pipelined ADCs are in essence an unrolled version of a SAR ADC, trading latency for speed. Each stage performs a coarse conversion, amplifying the reside to the following stage. Since all stages run in parallel, the pipeline also complete conversions in approximately one cycle. On the other hand, their complexity is moderate, growing linearly with the number of bits (B), and requiring nearly an opamp per each stage. They are suitable for applications requiring a balance of speed and accuracy, such as high-speed communications. With a (single-channel) pipelined ADC, it is possible to deliver ~ 11 ENOB at 1 GS/s [6].

Unlike the previous options, SAR ADCs require a number of cycles equal to the resolution (B) for each conversion. But their complexity is low and does not increase significantly with resolution, making them very efficient in terms of power consumption and area. SAR ADCs do not use operational amplifiers, further simplifying their design. In fact, modern SAR ADCs rely heavily on MOS switches and latches, which strongly benefit from aggressively CMOS scaling. This efficiency and simplicity make SAR ADCs ideal for a wide variety of applications [20] (Figure 3.3), since low-speed instrumentation and measurement, to high-speed communication or RADAR systems, where a balance of speed, accuracy, and power efficiency is crucial.

The benefits of CMOS scaling are so significant that SAR ADCs are widely used as fundamental components in hybrid architectures to enhance either speed or resolution. A pertinent question arises: why not employ faster stages instead of SAR ADCs, thereby reducing the complexity? The answer lies in the inherent limitations of other Nyquist-rate converters, such as flash and pipeline ADCs. These alternatives depend on the use of multiple operational amplifiers (opamps) or similar circuits, which are relatively slow and do not gain as much advantage from technology scaling as the components in SAR ADCs. Consequently, even though a pipeline ADC requires only one clock cycle per conversion,

this cycle can take longer than the sub-cycles necessary for SAR ADC conversions.

As a result, we will delve into the architecture and operation of Successive Approximation Register (SAR) ADCs. The following sections will provide an in-depth exploration of SAR ADC architecture, its design considerations, and practical implementations.

3.1 Successive Approximation: The Algorithm

The successive approximation register (SAR) ADC architecture dates back to the 1950s and saw it debut in CMOS in the 1970s. Being the main application drivers back then, the telephony and relatively low-speed instrumentation. The Successive Approximation (SA) algorithm is a method used in ADCs to approximate the input signal's value efficiently. Unlike flash converters, which require a single clock edge, or linear converters, which require 2^B clock cycles for a linear approximation, the SAR converter completes the conversion in B cycles. Each cycle refining the signal approximation.

Figure 3.1 shows an abstract flow diagram of the SA algorithm. At the start, a sample $V_{S/H}$ is taken, and the B output bits b_{B-1} to b_0 are reset to 0. In the first comparison cycle, the coefficient corresponding to the most significant bit (MSB) b_{B-1} is set to 1, and the DAC output is set to a value $V_{DAC} = 0.5V_{REF}$, so that the sampled input level is compared to this value. If the result is positive $V_{S/H} > V_{DAC}$, the bit $b_{B-1} = 1$ is kept. If not, the value is set back to $b_{B-1} = 0$. This cycle is repeated for all required bits, from the MSB to the least significant bit (LSB).

Figure 3.2b visually represents this process. The voltage difference between the input and the DAC value narrows down step by step until the voltage is nulled to within a small residual quantization error, leading to the final digital output code that best approximates the input voltage.

Figure 3.2a illustrates a circuit implementation of the SAR algorithm. SAR designers have optimized the design by implementing the DAC and T&H using a single binary-weighted capacitor array (Figure 2.5). Modern arrays utilize fringe capacitors, reducing the unit capacitance to as small as 100 attofarads [26].

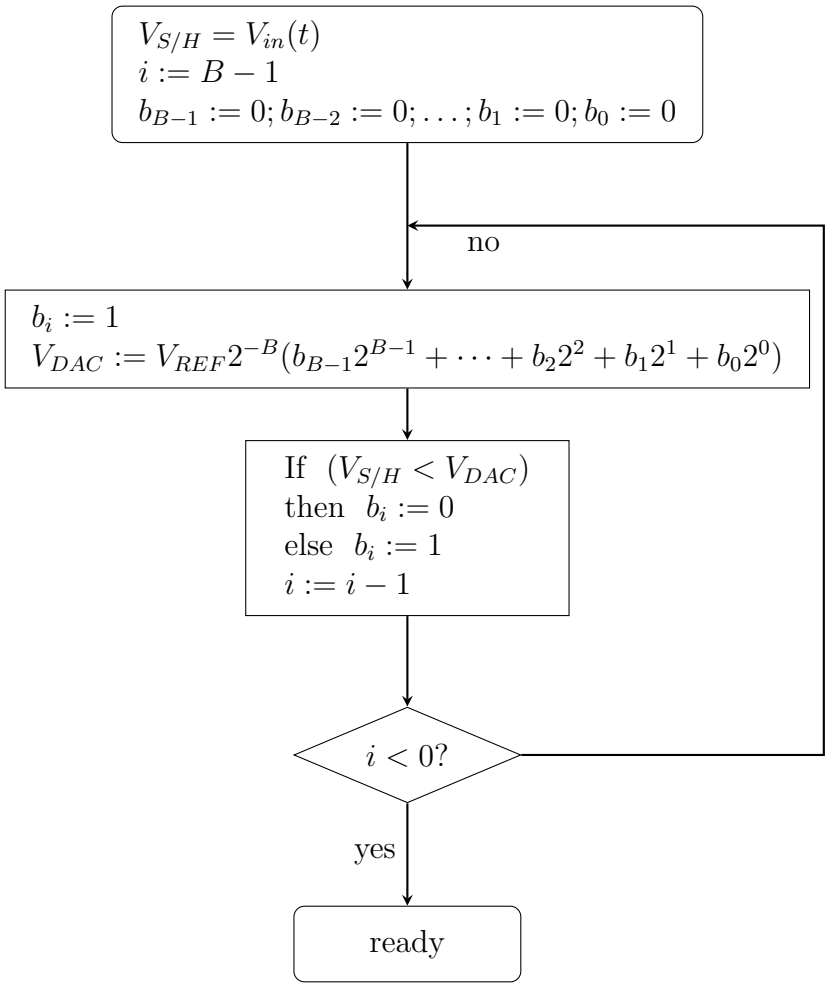
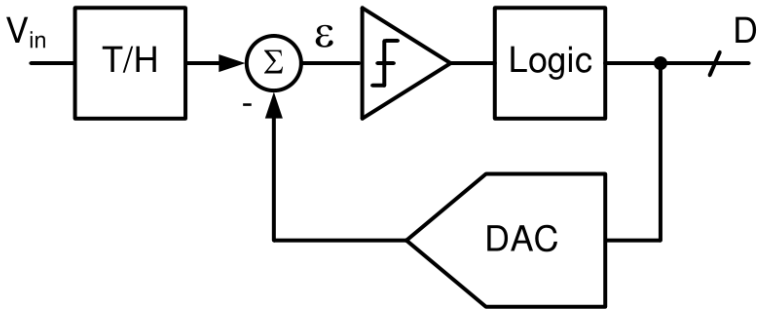


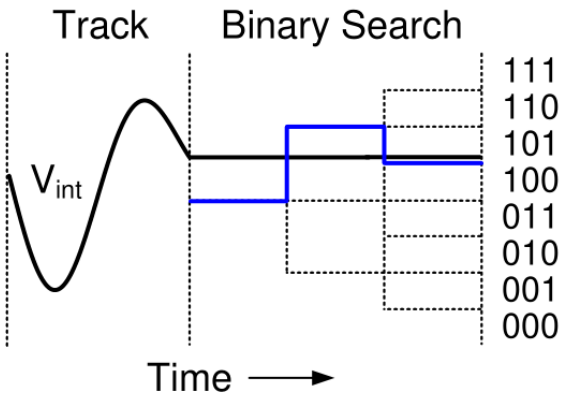
Figure 3.1: Flow diagram of the SAR algorithm.

Due to the conversions being made in B cycles, the internal clock f_{clock} must run at least $B\times$ faster than the sample clock f_s. Furthermore, for each sample we must accommodate a reset for each cycle and a sample-and-hold action, making more than B clock cycles needed.

With clock speeds in modern technology going as high as 5-10 GHz, it is difficult to go much beyond 1 GS/s even at moderate resolutions of B = 6-8. The answer to this problem is the use of hybrid structures such as TI or pipelined.



(a) Circuit implementation of the SAR algorithm.



(b) Timing diagram of the SAR algorithm showing the track and binary search phases.

Figure 3.2: Circuit implementation and timing diagram of the SAR algorithm.

3.2 Hybrid SAR Architectures

In modern analog-to-digital converter design, various architectures are often combined, resulting in "hybrid converters" which incorporate a secondary converter architecture as a sub-block.

Figure 3.3 presents an aperture plot displaying the Signal-to-Noise and Distortion Ratio (SNDR) against the input frequency for different ADC architectures, including Flash, Pipeline, Sigma Delta, and various SAR configurations. The plot illustrates how different hybrid SAR architectures achieve varying balances of resolution and speed:

- **Standard SAR (blue):** Generally operates with an SNDR range of 50–80 dB and an input frequency range up to several megahertz.

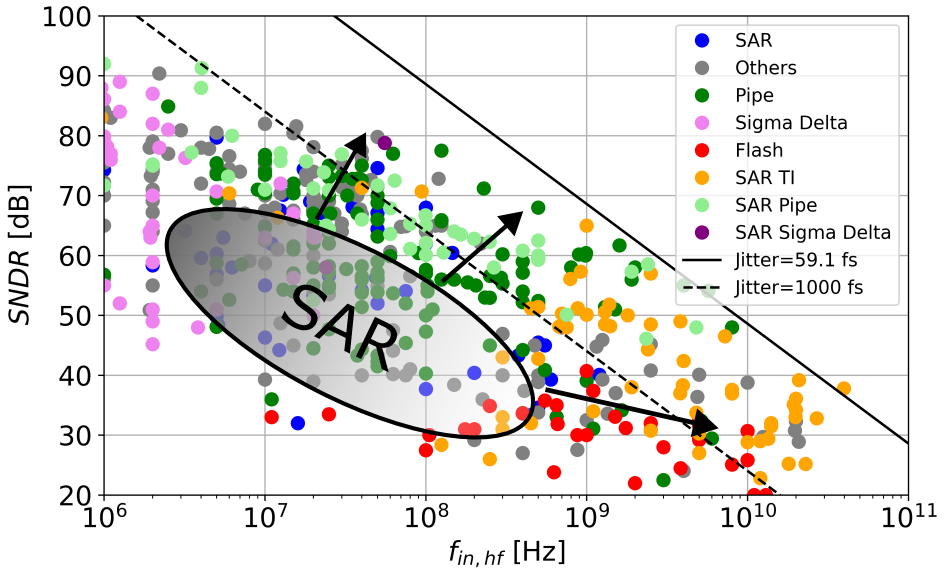


Figure 3.3: Aperture plot showing SNDR versus input frequency for various ADC architectures.

- **SAR TI (orange):** Time-Interleaved SAR ADCs aim to increase speed by interleaving multiple SAR ADCs. This configuration typically achieves higher input frequencies, often in the GHz range. TI is effective for aggregating speed, but the gained speed tends to force us to sacrifice resolution. This is caused by the hard bound that clock jitter imposes on the achievable speed-resolution product.

- **SAR Pipe (light green):** Pipelined SAR ADCs enhance the balance between speed and resolution, achieving moderate input frequencies (up to several hundred MHz) with improved SNDR, typically ranging from 60-90 dB.

- **SAR Sigma Delta (purple):** Sigma Delta SAR ADCs target higher resolution through oversampling and noise shaping, resulting in higher SNDR values (up to 100 dB) but operating at lower input frequencies, typically below 100 MHz.

This analysis highlights the significance of SAR ADCs across various fields. For instance, at the lower end of sampling rates, SAR designs are utilized in biomedical instrumentation [27]. In the moderate speed range, SAR-assisted pipeline ADCs operate at tens of megahertz with very low power consumption (approximately 1 mW) [28]. Lastly, in the ultra-high speed regime, designs achieving 90 GS/s are made possible by time interleaving techniques [29].

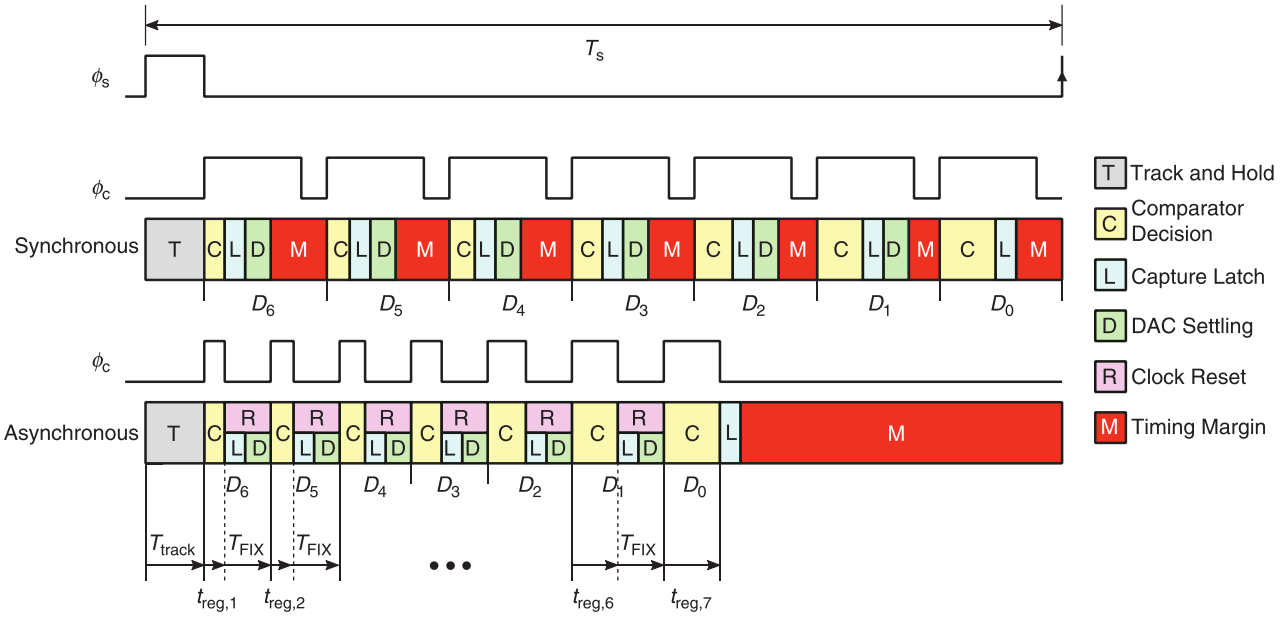


Figure 3.4: Timing diagrams comparing the synchronous and asynchronous topologies [17]

Overall, the aperture plot effectively illustrates the trade-offs between speed, resolution, and jitter performance across various ADC architectures, highlighting the versatility and adaptability of SAR ADCs and their hybrid architectures for different application requirements, including their suitability for RADAR systems due to their balance of speed and accuracy, especially in the case of pipelined SARs.

3.3 Synchronous and Asynchronous

Managing metastable events is a crucial challenge in this architecture, especially for high-speed applications. Designers are often compelled to run each converter at the highest possible speed, typically around 1 GS/s. This assumes that the comparator must make B decisions in less than 1 ns. Under these circumstances, if some decisions take longer than usual, the comparator may not resolve in time, leading to metastable events.

In the synchronous topology, one clock period T_{clk} is allocated to each bit resolution in the SA algorithm. The clock period must be long enough to accommodate the comparator decision, the latch capture, and the DAC settling time (Figure 3.4, upper). If the total time used by these processes exceeds the time allocated in T_{clk}, a metastability error

occurs. The number of comparator time constants (τ) that fit into the time available for regeneration $(T_{clk} - T_{fix})/\tau$ is sufficient to fully characterize the metastability behavior in this case. Here, T_{fix} is defined as the sum of the times dedicated to latch capture and DAC settling, as these times are constant across all bit cycles.

To understand the problem with this architecture, we must first explore what causes the comparator decision time to vary. As we will see in the following chapter, the regeneration time of a latch-based comparator can be expressed as:

$$T_{reg} = \tau \ln\left(\frac{V_{DD}}{|V_{S/H} - V_{DAC}|}\right) \tag{3.1}$$

where $V_{S/H}$ is the voltage stored in the S&H and V_{DAC} is the voltage supplied by the DAC. Studying this equation, we can observe that when $V_{S/H}$ and V_{DAC} are very close, the regeneration time can increase indefinitely.

However, due to the nature of the binary search algorithm, only one bit decision can reside within less than ± 0.5 LSB in each sample resolution. This decision is often called the "hard" decision, while the rest are "easy" decisions.

In designing synchronous SAR ADCs, we must allocate a number of τs dedicated to regeneration, assuming that any decision could be a "hard" one. This approach is quite wasteful, as we know that only one decision will take longer, while the rest will be faster. It would be more efficient to proceed from one bit cycle to the next immediately after regeneration, allocating a total time for the sample resolution that better suits the SA behavior. A first-order solution to this problem is the use of asynchronous timing [30]. This approach also allocates more time for the "hard" decision, mitigating the risk of metastability.

In Figure 3.4, bottom, we can see the timing diagram of the proposed architecture. The main difference with synchronous architecture is that bit cycles are resolved asynchronously without relying on an external clock. The only external clock needed is the one controlling the sample of the S&H. This fact also relaxes the specifications of the external clock.

Furthermore, in the synchronous architecture, the same timing margin (colored in red

in the figure) is allocated for each bit. If a bit cycle does not use this time, it is wasted. In contrast, in the asynchronous architecture, we allocate a single block of time margin that can be consumed as needed. This approach is a very reliable way of improving both operation speed and probability of metastability.

We must note that we still have a T_{fix} for each bit cycle. In this case, it is designated for the capture latch and DAC settling, which now run in parallel with the comparator's internal reset. If, instead of considering the total metastability probability P_{meta}, we take a more realistic approach such as the probability of occurrence of an error greater than a certain magnitude X, another benefit of this architecture becomes apparent. In this scenario, the MSB bits can borrow the time allocated for T_{fix} and regeneration of the LSBs, thereby decreasing the probability of this code error occurring.

To compare the probability of metastability (P_{meta}) of synchronous and asynchronous architectures, we first define the number of comparator time constants available for regeneration per bit cycle, N:

$$N = \frac{\left(T_s - T_{track} - T_{Fix,total}\right)/\tau}{B} \tag{3.2}$$

In this context, $T_{Fix,total}$ represents the total fixed time for capture latch and DAC settling. For synchronous architectures, $T_{Fix,total}$ is equal to $B \times T_{Fix}$, whereas for asynchronous architectures, it is equal to $(B-1) \times T_{Fix}$.

Taking into account these definitions, we can define P_{meta} for both synchronous and asynchronous architectures [17] as:

$$P_{meta} \approx \begin{cases} 2^{B+1} e^{-\left(BN - \frac{T_{easy}}{\tau}\right)} & \text{for asynchronous} \\ 2^{B+1} e^{-N} & \text{for synchronous} \end{cases} \tag{3.3}$$

where $\frac{T_{easy}}{\tau}$ is the average of total time constants due to "easy" regeneration bit cycles for each input voltage and can be computed theoretically from [17]:

$$\frac{T_{easy}}{\tau} = \frac{1}{\tau}\sum_{i=1}^{B} t_{reg,easy,i} \approx [1+\ln(2)](2^{-B}-1)+B\left[1+\left(2^{-B}-\frac{1}{2}\right)\ln(2)\right]+\frac{B^2\ln(2)}{2} \tag{3.4}$$

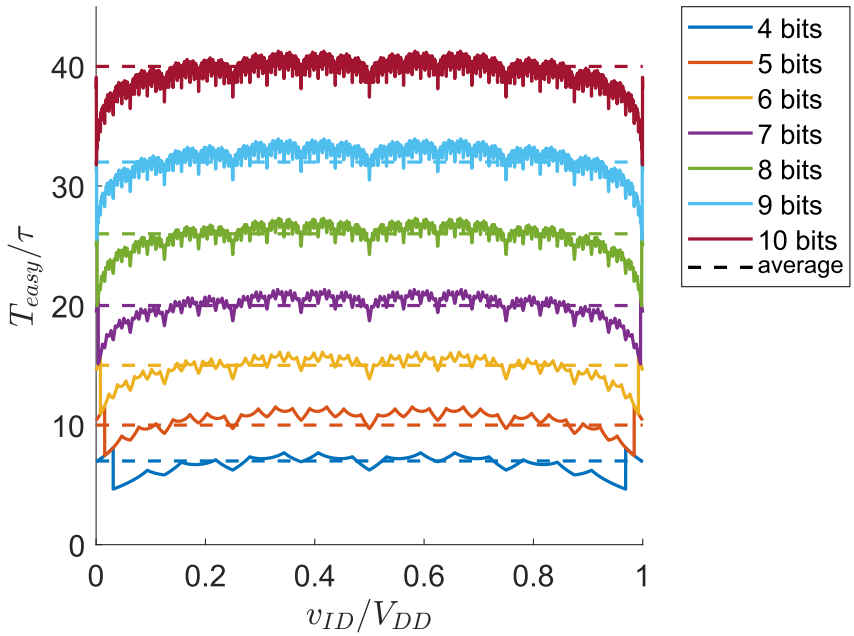


Figure 3.5: Total regeneration time and average of easy decisions as a function of ADC input voltage

Another way of computing T_{easy} is by simulating the binary search algorithm and computing the regeneration time due to each bit cycle from Equation 3.1. By repeating this process for all bits and for each possible input value of the ADC, we obtain Figure 3.5. The average is computed by taking into account all the total regeneration times calculated for each input value. Notice that by computing only T_{easy} we have skipped the regeneration time due to decisions with $\left|V_{S/H} - V_{DAC}\right| \leq 0.5$ LSB.

In [31], we see another more robust but computationally demanding method in which P_{meta} is computed from the ribbons formed in the plot of regeneration time. These ribbons result from different regeneration times due to the nature of the binary search algorithm. When we introduce an input voltage very close to any of the voltage values produced by the DAC, we observe a spike in regeneration time. On the other hand, if all decisions happen to occur far from the DAC voltage values, there will be a decrease in the regeneration time. These combined processes explain the existence of these ribbons.

From now on, we will consider the total number of comparator time constants available during the converter duty cycle (N_{DC}) instead of N, which provides a more representative measure:

$$N_{DC} = \frac{T_s - T_{track}}{\tau} \tag{3.5}$$

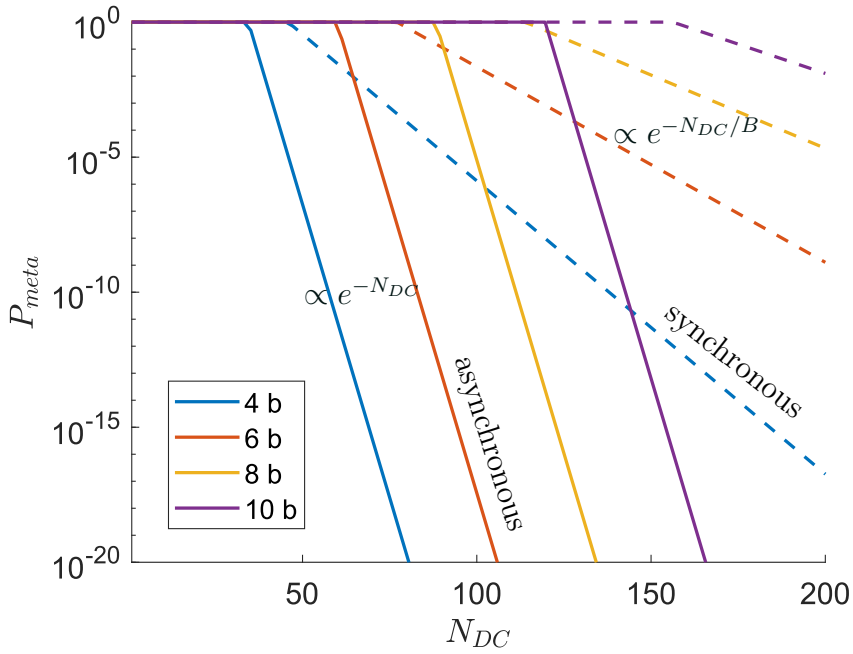


Figure 3.6: Comparison of P_{meta} for asynchronous and synchronous architectures versus N_{DC} with $N_{Fix} = 8$

In Figure 3.6, we observe the comparison between the probabilities of metastability for several bit resolutions. In this simulation, we have established $N_{Fix} = T_{Fix}/\tau = 8$. The results clearly show a faster decrease in metastability for asynchronous architectures compared to synchronous ones for the same value of N_{DC}. This also implies that, for the same P_{meta}, we can aggressively reduce N_{DC} of the comparator, thereby speeding it up.

From this analysis, we can conclude that asynchronous SAR architectures offer a significant advantage over synchronous ones in terms of reducing metastability probability or increasing speed. By allowing bit cycles to be resolved asynchronously, these architectures can more efficiently manage the timing margins and reduce the appearance of metastable states. This results in better performance, especially in high-speed applications where minimizing metastability is critical.

We will explore this topic in more depth in the following chapters when discussing the computation of the probability of error due to noise and metastability.

CHAPTER 4

Latch-Based Comparators

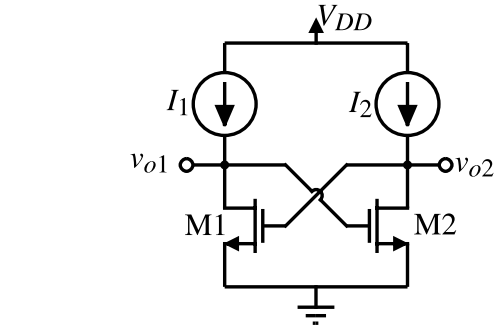
A comparator is typically an opamp-based circuit used for comparing an analog signal to a reference voltage. The output of the comparator is binary and depends on the relationship between the two signals being compared, making it a crucial component in ADCs for analog-to-digital conversion.

Comparators can generally be divided into two major groups: open-loop and regenerative comparators. Open-loop comparators are essentially opamps without compensation, causing their output to switch between the two power supply rails. On the other hand, regenerative comparators rely on positive feedback, generally using latches, to compare the two signals. There is also a third type which combines pre-amplification using an opamp with subsequent regeneration employing latches for extremely fast operation.

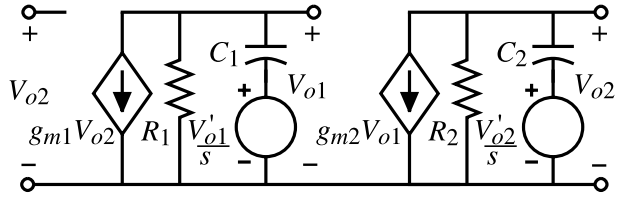
Given our focus on RADAR applications, this chapter will concentrate on the latter types of comparators, specifically those using regenerative techniques for enhanced speed and performance.

4.1 Latches

In order to accomplish the regeneration of the signal, designers make use of latches or bistables. The simplest form of a latch consists of two cross-coupled MOSFETs, as shown in Figure 4.1a. Normally, a clock is used to disable the latch and establish an initial



(a) NMOS latch



(b) NMOS latch small signal model

Figure 4.1: Equivalent model of NMOS latch [32]

voltage at the terminals v_{o1} and v_{o2}. Afterward, the clock enables the regeneration of the signal from the initial condition. Depending on the relationship between the two terminals, one of the outputs will go high while the other will go low.

In Figure 4.1b we can see the small model of the NMOS latch circuit. Let us assume that the initial values of v_{o1} and v_{o2} have been established and find the time it takes for the latch to operate. The equivalent voltage values of v'_{o1} and v'_{o2} in the Laplace domain represent the initial values of v_{o1} and v_{o2} and are step functions.

We may write the nodal equations as [32]:

$$g_{m1}V_{o2} + G_1V_{o1} + sC_1\left(V_{o1} - \frac{V'_{o1}}{s}\right) = g_{m1}V_{o2} + G_1V_{o1} + sC_1V_{o1} - C_1V'_{o1} = 0 \qquad (4.1)$$

and

$$g_{m2}V_{o1} + G_2V_{o2} + sC_2\left(V_{o2} - \frac{V'_{o2}}{s}\right) = g_{m2}V_{o1} + G_2V_{o2} + sC_2V_{o2} - C_2V'_{o2} = 0 \qquad (4.2)$$

where G_1 and G_2 are the conductances seen from the drains of M1 and M2 to ground and C_1 and C_2 are the capacitances seen from the drains of M1 and M2 to ground. Solving Eqs. 4.1 and 4.2 for V_{o1} and V_{o2} gives:

$$V_{o1} = \frac{R_1 C_1}{sR_1C_1 + 1} V'_{o1} - \frac{g_{m1}R_1}{sR_1C_1 + 1} V_{o2} = \frac{\tau_1}{s\tau_1 + 1} V'_{o1} - \frac{g_{m1}R_1}{s\tau_1 + 1} V_{o2} \tag{4.3}$$

and

$$V_{o2} = \frac{R_2 C_2}{sR_2C_2 + 1} V'_{o2} - \frac{g_{m2}R_2}{sR_2C_2 + 1} V_{o1} = \frac{\tau_2}{s\tau_2 + 1} V'_{o2} - \frac{g_{m2}R_2}{s\tau_2 + 1} V_{o1} \tag{4.4}$$

where τ_i is the time constant R_iC_i. Assume that both transistors are identical so that $g_{m1} = g_{m2} = g_m$, $R_1 = R_2 = R$, and $C = C_1 = C_2$, which gives $\tau_1 = \tau_2 = \tau$. Let us define the difference between V_{o2} and V_{o1} as ΔV_o and the difference between V'_{o2} and V'_{o1} as ΔV_i. Therefore,

$$\Delta V_o = V_{o2} - V_{o1} = \frac{\tau}{s\tau + 1} \Delta V_i + \frac{g_m R}{s\tau + 1} \Delta V_o \tag{4.5}$$

Solving for ΔV_o gives

$$\Delta V_o = \frac{\tau \Delta V_i}{s\tau + (1 - g_m R)} = \frac{\tau \Delta V_i}{s\tau + 1 - g_m R} = \frac{\tau' \Delta V_i}{s\tau' + 1} \tag{4.6}$$

where

$$\tau' = \frac{\tau}{1 - g_m R} \tag{4.7}$$

Taking the inverse Laplace transform of Eq. 4.6 gives:

$$\Delta v_o(t) = \Delta v_i e^{-t/\tau'} \tag{4.8}$$

if $g_m R$ is greater than one. The time constant of the latch, as expressed in Eq. 4.7, is given as

$$\tau' \approx -\frac{\tau}{g_m R} = -\frac{C}{g_m} \equiv -\tau_L \tag{4.9}$$

If C is mostly the gate-source capacitance, then the latch time constant can be expressed as

$$\tau_L = \frac{0.67WLC_{ox}}{\sqrt{2K'(W/L)I}} = 0.67C_{ox}\sqrt{\frac{WL}{2K'I}} \tag{4.10}$$

Equation 4.10 shows the strong dependence of the latch time constant on the channel length. Therefore, removing the notation used to differentiate the small signal, the latch time response can be expressed as:

$$\Delta V_{out}(t) = e^{t/\tau_L}\Delta V_i \tag{4.11}$$

Thus, the propagation time delay of a latch can be found by setting Eq. 4.11 equal to V_{DD}. The result is:

$$t_p = \tau_L \ln\left(\frac{V_{DD}}{\Delta V_i}\right) \tag{4.12}$$

This model can be further refined by defining the upper and lower voltage limits of the comparator as V_{OH} and V_{OL} and defining the minimum value from which the correct binary output is ensured, $V_{logic} = \alpha\,(V_{OH} - V_{OL})$, where α is defined such that $0 < \alpha \leq 1$.

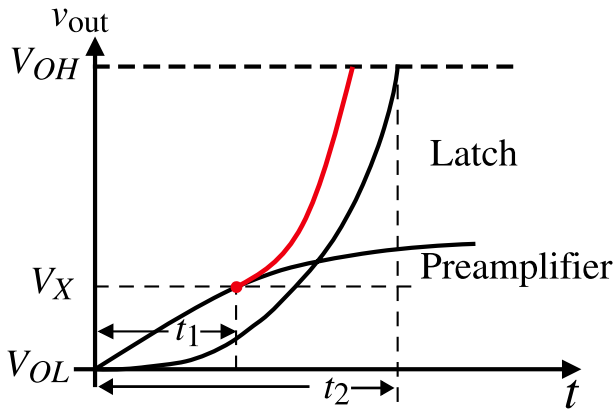


Figure 4.2: Comparison of time taken for latch and regenerative comparator with latch to reach to logic level

In numerous applications, signals are processed in discrete time rather than in a continuous form. Bistable devices can serve as comparators in these scenarios. A key characteristic of regenerative circuits is their transient response, which exhibits a positive exponential growth. This implies that if the input signal is small, a significant amount of time is required for the signal to reach the region where the exponential response becomes steep.

To address this issue, a preamplifier can be cascaded with the latch. The characteristic of the amplifier, represented in Figure 4.2, can be expressed as $\Delta V_{out} = A_v \left(1 - e^{-t/\tau_c}\right) \Delta V_i$.

The role of the preamplifier is to quickly amplify the input signal before it reaches the latch, thereby circumventing the slow initial rise of the exponential response. This method has led to the development of high-speed latch-based comparators, such as the Strong-ARM. The following derivation in the subsequent sections is based on the work presented in [33].

4.2 Strong-ARM

In memory design and advanced digital circuits, the search for comparators without static power consumption led to the development of circuits where a clock allows the precharge of the nodes to a certain voltage, enabling the circuit to discharge them based on the processed information. A comparator meeting these requirements was introduced in [34], and it was named "Strong-ARM" to commemorate its use in Digital Equipment Corporation's StrongARM microprocessor.

Since then, the Strong-ARM circuit has become quite popular for three main reasons: it has zero static power consumption, making it ideal for aggressive power-saving applications; it directly produces rail-to-rail outputs in less time than classical open-loop comparators; and its input-referred noise primarily originates from the differential pair.

The Strong-ARM comparator, depicted in Figure 4.3, comprises a clocked differential pair M_{1P}/M_{1N}, a tail current source T_1, a cross-coupled inverter pair M_{2P}/M_{2N}, M_{3P}/M_{3N}, and four reset devices M_{4P}/M_{4N}, M_{5P}/M_{5N}. Its operation can be described in four distinct phases.

In the first phase, the reset phase ($CLK = V_{SS} = 0$ V), T_1 is off and M_{4P}/M_{4N}, M_{5P}/M_{5N} pull nodes $X+/X-$ and $O+/O-$, respectively, to V_{DD}.

The second phase, the primary integration phase, begins with $CLK = V_{DD}$. At this point, T_1 turns on and the input pair devices M_{1P}/M_{1N} enter saturation. Nodes $X+/X-$ discharge proportionally to the differential current, which depends on the initial difference ΔV_I.

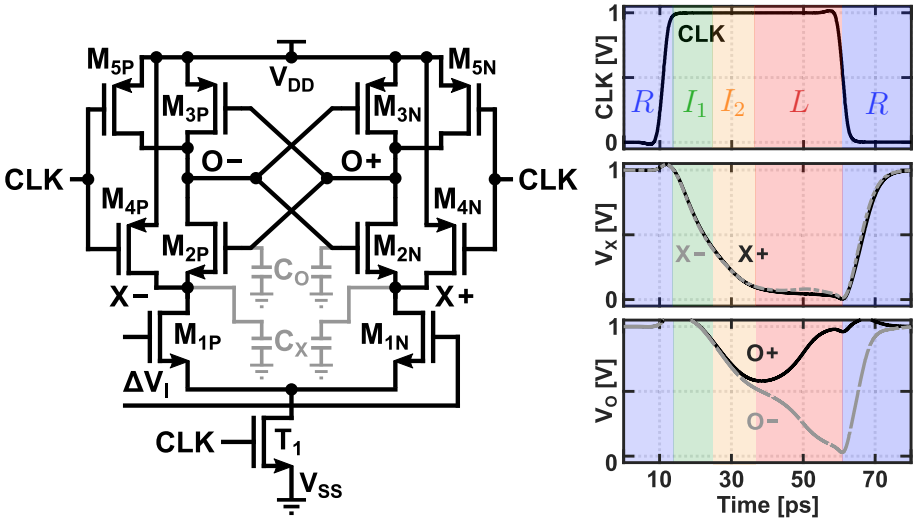


Figure 4.3: Strong-ARM comparator and waveforms in 28 nm technology [33]. The reset (R), first integration (I_1), second integration (I_2) and latch (L) phases are also represented

This phase continues until nodes $X+/X-$ drop one NMOS threshold voltage V_{TH} below V_{DD}, initiating the third phase, the secondary integration phase. Here, M_{2P}/M_{2N} turn on, and the output nodes $O+/O-$ discharge with an initial difference of ΔV_X, providing further amplification.

The third phase continues until the output nodes $O+/O-$ drop one PMOS threshold below V_{DD}, preceding the fourth and final phase of the comparator:

$$\left|\frac{\Delta V_O}{\Delta V_I}\right| = 2g_{m1}\frac{|V_{TH}|}{I_{T1}} \cdot \left(\frac{C_X + C_O}{C_O}\right) \approx 2g_{m1}\frac{|V_{TH}|}{I_{T1}} \tag{4.13}$$

The final expression holds true for $C_O \gg C_X$, which is common since C_O includes both the input capacitance of any subsequent circuit and the comparator's self-load.

The final phase, latching, begins with M_{3P}/M_{3N} turning on. Due to the cross-coupled inverter pair, the drain of the PMOS with the lower gate voltage is pulled to V_{DD}, while the other drain is pulled to V_{SS}. The latching delay, which provides a differential output perceived as a clear logical level (typically $V_{DD}/2$), arises from the exponential input-output latch characteristic:

$$t_{SAC,latch} = \frac{C_O}{g_{m,eff}} \ln\left[\frac{V_{DD}/2}{|\Delta V_O|}\right] = \frac{C_O}{g_{m,eff}} \ln\left[\frac{V_{DD}(I_{T1}/g_{m1})}{4|V_{TH}||\Delta V_I|}\right] \tag{4.14}$$

In this expression, $g_{m,eff}$ is the effective transconductance of the cross-coupled pairs, primarily comprising g_{m3} as M_{3P}/M_{3N} are mainly responsible for initiating the latching.

Lastly, we can estimate the input referred noise by focusing on the differential input pair noise, which typically dominates. This can be expressed as:

$$\overline{v_{SAC,inpre}^2} = 2\frac{kT\gamma\left(I_{T1}/g_{m1}\right)}{C_X V_{TH}} \tag{4.15}$$

There are several known issues with this architecture. The first is the trade-off between latch speed and preamplifier gain and noise. From Eq. 4.10, we can see that one method of reducing the comparator time constant, and thus increasing speed, is by increasing the tail current I_{T1}. However, in this configuration, increasing I_{T1} will also lower the preamplifier gain, as shown in Eq. 4.13. This results in a lower ΔV_O for the latch to regenerate, thereby negating the potential speed increase. Additionally, the increase in current will lead to a corresponding increase in input-referred noise, as indicated in Eq. 4.15.

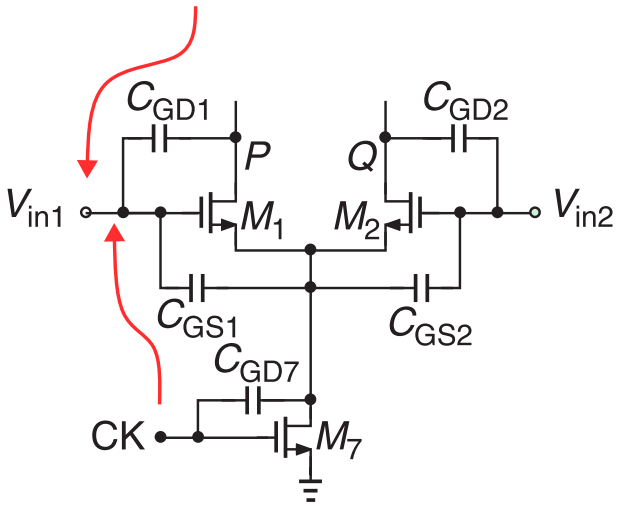


Figure 4.4: Kickback noise paths [35]

The second problem is the occurrence of "kickback" currents drawn from the inputs and the supply. Due to the integration of the latch in parallel with the preamplifier, the high transient currents necessary for regeneration can affect the input nodes through capacitive noise paths (Figure 4.4). This can modify the initial voltage ΔV_I and potentially lead to a malfunction of the comparator, especially for small input signals.

The main takeaway from this analysis is the need to modify this structure by cascading the preamplifier and the latch in separate instances. This approach led to the development of the second most common type of latch-based comparator, the Double-Tail comparator.

4.3 Double-Tail comparator

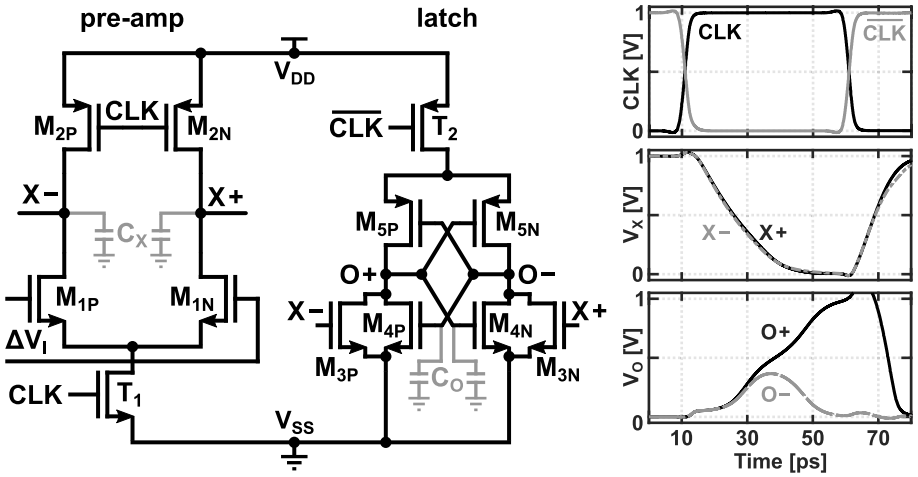


Figure 4.5: Double-Tail comparator and waveforms in 28 nm technology [33]

To overcome the limitations presented in the Strong-ARM analysis, such as the use of a single current for controlling both the integration of the preamplifier and the latching, as well as to reduce device stacking, the split input-latch configuration shown in Figure 4.5 was proposed in [36]. Device stacking is also particularly relevant in the context of lower supply voltages derived from aggressive CMOS scaling.

The pre-amplifier stage includes a clocked differential pair M_{1P}/M_{1N} with reset devices M_{2P}/M_{2N}, and a tail current source T_1 controlling the integration. The latch contains a cross-coupled inverter pair M_{4P}/M_{4N}, M_{5P}/M_{5N}, and a tail current source T_2 controlling the latching. The intermediate devices M_{3P}/M_{3N} amplify ΔV_X while transferring it to the latching stage.

The operation of this comparator can be described in three main phases. During the reset phase ($CLK = V_{SS}$, $\overline{CLK} = V_{DD}$), T_1 and T_2 are off, and M_{2P}/M_{2N} and M_{3P}/M_{3N} pull nodes $X+/X-$ and $O+/O-$ to V_{DD} and V_{SS}, respectively. After the reset, when $CLK = V_{DD}$, $\overline{CLK} = V_{SS}$, nodes $X+/X-$ discharge through the input

pair with slopes proportional to the differential current in the pre-amplifier. Almost simultaneously, the output nodes charge with slopes proportional to the differential latch current, and intermediate M_{3P}/M_{3N} provide further amplification while transferring ΔV_X to the latch. Assuming M_{1P}/M_{1N} and M_{3P}/M_{3N} function as largely overlapping cascaded integrators, the amplification towards the latch is:

$$\left|\frac{\Delta V_O}{\Delta V_I}\right| \approx 4 g_{m1} g_{m3} \frac{C_X}{C_O} \left(\frac{V_{X,DC}}{I_{T1}}\right)^2 \tag{4.16}$$

The term $V_{X,DC}$ represents the voltage drop on $X+/X-$ such that the intermediate devices are no longer able to clamp the outputs to V_{SS}. In both stages, the current on each side is approximated as $I_{T1}/2$ and $I_{T2}/2$, which is fair for small enough ΔV_I and ΔV_X respectively, following the same assumptions that lead to Eq. 4.13.

The time of regeneration of the latch can be computed as in Eq. 4.14:

$$t_{DTC,latch} = \frac{C_O}{g_{m,eff}} \ln\left[\frac{V_{DD}/2}{|\Delta V_O|}\right] = \frac{C_O}{g_{m,eff}} \ln\left[\frac{V_{DD}(I_{T2}^2/g_{m1}g_{m3})}{8V_{TH}^2|\Delta V_I|(C_O/C_X)}\right] \tag{4.17}$$

Finally, the input-referred noise can be estimated as:

$$\overline{v_{DTC,inpre}^2} \approx 2\frac{kT\gamma\,(I_{T1}/g_{m1})}{C_X V_{TH}} \tag{4.18}$$

Here we can observe several improvements from the previous comparator. Firstly, maximizing g_{m1} and g_{m3} results in an increase in gain, as seen in Eq. 4.16. Additionally, a greater ΔV_O leads to a shorter regeneration time for the latch, and there is also a reduction in input-referred noise (Eq. 4.18).

Secondly, the introduction of an additional degree of freedom, namely the tail current I_{T2}, allows for the optimization of I_{T1} for maximum gain and low noise (Eqs. 4.16 and 4.18), while controlling I_{T2} for minimal latch regeneration time (Eq. 4.17).

Splitting the input and latch stages also reduces the required headroom to two threshold voltages and one overdrive voltage (one less than the Strong-ARM comparator), making this topology more suitable for lower supply voltages.

Lastly, the noise paths from the latch to the input voltage signals are reinforced by

two parasitic capacitors instead of just one, thus reducing the possible kickback current from the latch. The kickback current from the power supply is also reduced since I_{T1} is minimized to increase gain and no longer supplies power to the latch.

However, there are still some issues to address. The first issue is the lack of cascode devices in the pre-amplifier stage. These devices provide further amplification to the differential pair by remaining in saturation longer, thereby reducing the output noise and regeneration time[1].

The second issue is the strong dependency of the regeneration time on the amplitude of the input signal ΔV_I. This causes the comparator delays for different input signals to be quite disparate.

4.4 Triple-Tail Feedforward Comparator

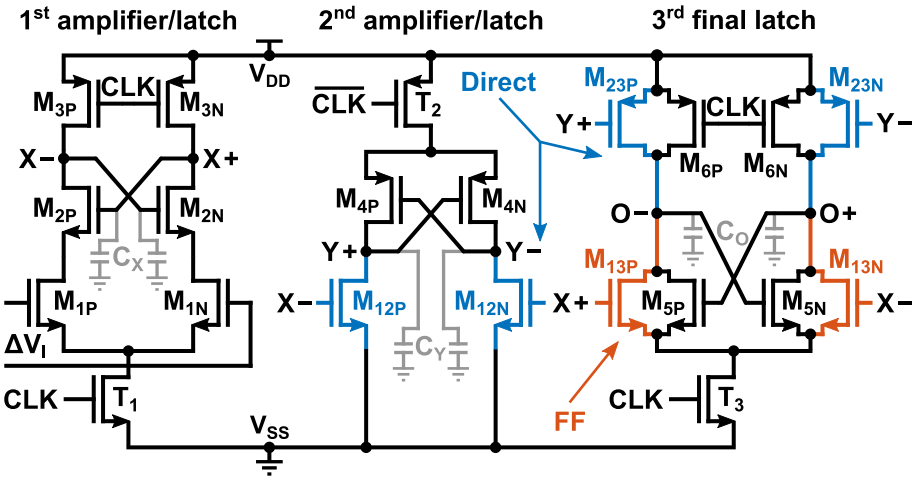


Figure 4.6: Triple-Latch feedforward comparator [33]

The Triple-Tail feedforward comparator, presented in Figure 4.6, was proposed by Ramkaj et al. in [37] to address some of the issues observed in the Double-Tail comparator.

This architecture consists of a stage-1 amplifier M_{1P}/M_{1N} with a cascoded NMOS half-latch M_{2P}/M_{2N}, followed by a stage-2 amplifier M_{12P}/M_{12N} with a PMOS half-latch M_{4P}/M_{4N}. Both stages drive the final stage NMOS latch M_{5P}/M_{5N} through a

[1]This problem could be addressed by modifying the first stage to the configuration seen later in the stage-1 of the Triple-Tail comparator.

parallel direct path M_{23P}/M_{23N} and a feedforward path M_{13P}/M_{13N}. This configuration allows for separate optimization of each stage, improving the total delay, noise/offset, and sensitivity to ΔV_I. Stage-1 is designed to provide a specific noise and a relatively high gain to sufficiently attenuate the noise and kickback of the following stages. Stage-3 is optimized for minimal regeneration time given its total load (following circuit and self-load). Stage-2 serves as an additional gain stage, increasing its differential output exponentially before the final latch. By horizontally cascading the stages, the required headroom of this topology is reduced by one to two threshold voltages compared to the Double-Tail and Strong-ARM, lowering its minimum V_{DD} by the same amount.

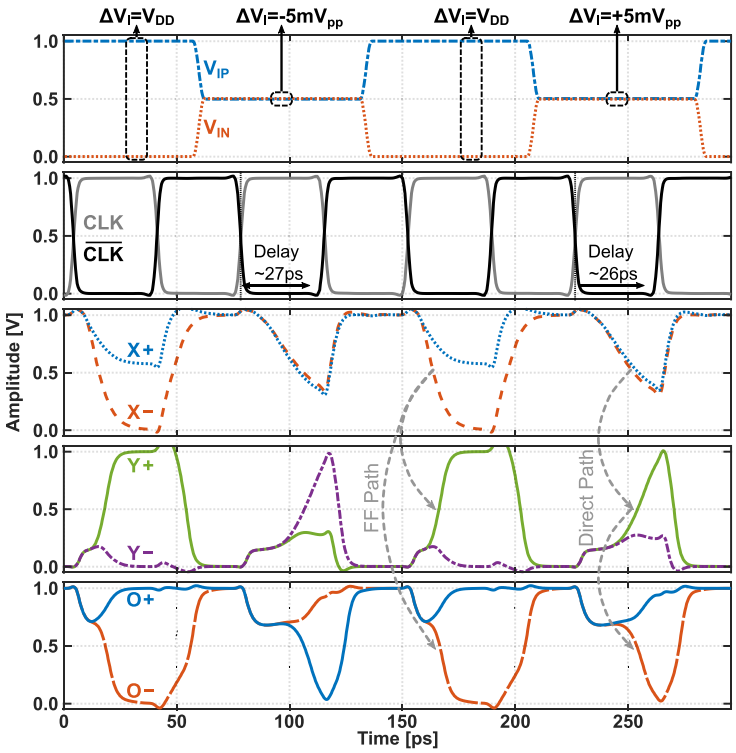


Figure 4.7: Waveforms of the Triple-Tail comparator [33]

The operation of the TLFF comparator, as represented in Figure 4.7, consists of several phases. During the reset phase ($CLK/\overline{CLK} = V_{SS}/V_{DD}$), all stages are off; nodes $X+/X-$ and $O+/O-$ are pulled to V_{DD}, while nodes $Y+/Y-$ are discharged to V_{SS}. The intermediate devices M_{12P}/M_{12N} and M_{23P}/M_{23N} perform a dual role, functioning both as reset devices and transconductors[2]. Due to the very high data rate, the additional

[2]In simulations, we observed better gain and noise reduction using another set of reset transistors to the drains of transistors M_{1P} and M_{1N}, similar to what is used in the Strong-ARM design. This enables maximum initial overdrive voltage of the differential input pair.

devices M_{6P}/M_{6N}, gated by the clock, are included to speed up the reset and minimize any memory effect.

The integration phase begins by simultaneously turning on the three tail devices, T_1, T_2, and T_3, through the symmetrical $CLK/\overline{CLK}$ ($CLK/\overline{CLK} = V_{DD}/V_{SS}$). Concurrently, the cascaded stage-2/stage-3 PMOS/NMOS cross-coupled pairs also turn on together with a large initial overdrive of about $V_{DD} - |V_{TH}|$. This is a key speed-boosting attribute of the proposed TLFF compared to the Strong-ARM and Double-Tail comparators.

ΔV_Y is transferred to the final stage outputs by the direct path M_{23P}/M_{23N}, where it is combined with the transferred ΔV_X by the feedforward path M_{13P}/M_{13N}, producing ΔV_O, prior to the final latching by the NMOS cross-coupled pair M_{5P}/M_{5N}.

The already partially regenerated ΔV_O is obtained as follows:

$$|\Delta V_O| = \left[8\frac{V_{Y,DC}^2 V_{O,DC}}{I_{T_2}^2 I_{T_3}/(g_{m1}g_{m12}g_{m23})}\frac{C_Y}{C_X} \times e^{2g_{m4}\frac{V_{Y,DC}}{I_{T_2}}} + 4\frac{V_{O,DC}^2}{I_{T_3}^2/(g_{m1}g_{m13})}\frac{C_O}{C_X}\right]|\Delta V_I| \quad (4.19)$$

where $V_{Y,DC}$ is the required voltage level at nodes $Y+/Y-$ for devices M_{23P}/M_{23N} to be unable to clamp $O+/O-$ to V_{DD} and $V_{O,DC}$ is the voltage drop built in nodes $O+/O-$.

The regeneration time due to both latches is given by:

$$t_{TLFF,latch} = \frac{C_O}{g_{m13} + g_{m23} + g_{m5}} \ln\left[\frac{V_{DD}/2}{|\Delta V_O|}\right] \quad (4.20)$$

Lastly, the input-referred noise adheres to the same formula as in the Strong-ARM or Double-Tail configurations. This is because the high gain of the first stage compensates for the increased number of transistors, with the differential input pair being the primary contributor.

$$\overline{v_{TLFF,inpre}^2} \approx 2\frac{kT\gamma\,(I_{T1}/g_{m1})}{C_X V_{TH}} \quad (4.21)$$

These expressions highlight the major speed enhancements of the TLFF over its prior art comparators. The larger ΔV_O due to the greater signal gain significantly reduces

$t_{TLFF,latch}$. Additionally, unlike in the Double-Tail, where we could only increase the tail current for speed, here we can obtain further gain without significantly increasing the tail currents due to the extra gain stages and initial maximum device g_m in all stages.

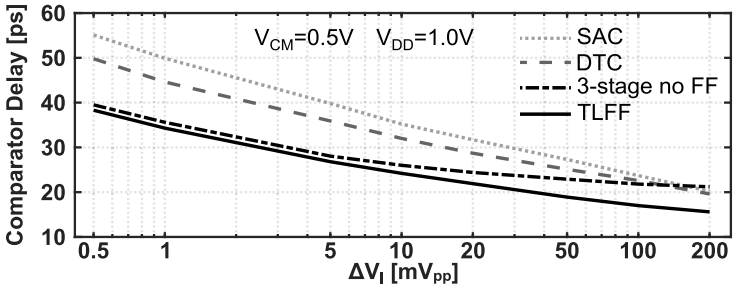


Figure 4.8: Comparator delay comparison between Strong-ARM, Double-Tail, Triple-Tail, and Triple-Tail with feedforward [33]

The feedforward path is also a major advantage compared to other triple-stage comparators because it allows greater input signals to directly affect the output of the third stage without the gate delay that would be introduced by the second stage using the direct path. These advantages result in a significant decrease in comparator delay compared to its alternatives, as seen in Figure 4.8. The increased gain causes faster regeneration time for small input signals, while the feedforward paths enable a reduction for larger inputs.

In conclusion, the Triple-Tail feedforward comparator addresses the limitations of previous architectures by optimizing each stage separately for gain, noise, and speed. This design significantly improves performance, making it suitable for high-speed and low-voltage applications.

CHAPTER 5

Case Study and Probability Model

In the design of an asynchronous SAR ADC, the comparator is the most critical component. To achieve an optimal design, it is essential to derive mathematical expressions that elucidate first-order phenomena and provide initial estimates for transistor sizes. These initial values can subsequently be refined through a process of electrical simulation and verification to meet the desired specifications.

However, obtaining mathematical expressions for more complex physical systems can be challenging. In the case of clocked comparators, there is no constant DC operating point. Consequently, to derive these expressions, we must approximate the system's behavior as a succession of several operating points, assuming these DC points remain constant during the application time.

Furthermore, electrical simulations of more complex systems, such as a complete SAR ADC, can be time-consuming. Consequently, it is impractical to perform numerous simulations for parameter selection, necessitating an initial set of "good" parameters. This is where numerical simulation of models becomes crucial.

For SAR ADCs in RADAR applications, several trade-offs must be considered for proper optimization. The primary factor is speed, dictated by the input frequency of the specific application. For RADAR, this typically corresponds to hundreds of megahertz after down conversion, since direct RF sampling remains outside the scope of this thesis. In the initial model, the parameter that dictates comparator speed is the comparator time constant, τ. As indicated in Equation 4.10, the minimum transistor length, L, specified by

the technology, must be used to maximize speed. Typically, other capacitive contributors exist besides the latch self-load. Therefore, increasing the latch transistor width, W, will reduce τ until the capacitive contribution of the latch itself becomes dominant.

Another key design parameter is the probability of error, which is determined by the combination of metastability and noise. As discussed in previous chapters, there is a trade-off between the number of comparator time constants used for the converter's duty cycle (N_{DC}) and the probability of metastability (i.e., the faster we try to run the comparator, the higher the probability of metastability and the number of code errors per second). Regarding noise, the main contributors are quantization noise and input-referred noise. In this context, input-referred noise is primarily due to thermal noise, as the comparator typically operates in regions where thermal noise dominates over flicker noise.

As observed in [38], the main contributor to noise in these types of comparators is the differential pair. Their contribution can be approximated by:

$$\sigma_{M_1}^2 = \frac{2kT\gamma}{C_X\mathcal{F}} \quad \text{with} \quad \mathcal{F} = \frac{V_{Tn}}{V_{\text{ov}}} \tag{5.1}$$

Given that the overdrive voltage $V_{\text{ov}} = \frac{2I_D}{g_m}$, we arrive at the same expression as Equation 4.15 when considering both differential pair transistors' contributions. From these expressions, we conclude that, in order to reduce noise, we must either increase C_X by increasing the W/L ratio of the differential pair (consequently reducing the overdrive voltage) or decrease the overdrive voltage by reducing the current driven by these transistors. Both actions reduce comparator speed, clearly illustrating the trade-off between noise and speed.

Overall, there is a distinct trade-off between the global probability of error, influenced by metastability and noise, and operational speed.

The objective of this chapter is to develop a simulation model that, given the input resolution of the comparator, the sampling frequency, and the desired probability of error, aids the designer in computing the required τ and input-referred noise (σ_{in}).

5.1 Probability of Error

5.1.1 Metastability Only Model

In this first part, our focus is to develop a simulation model to compute the probability mass function (PMF) for each possible code error due to metastability alone. The combined model with noise will be developed in the following sections.

5.1.1.1 Metastability Margins

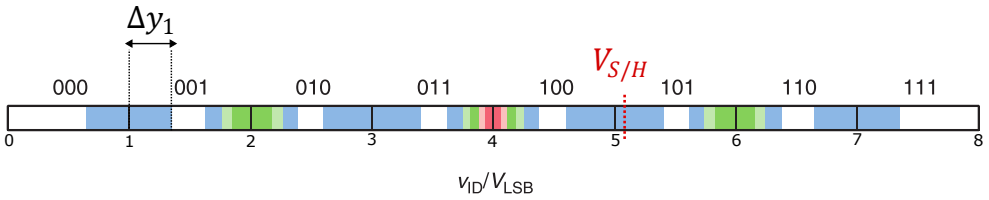


Figure 5.1: Margin of metastability for 3-bit case

In Figure 5.1, we observe the metastability regions around each DAC voltage value. There are two main methods for computing these regions to obtain the total probability of metastability. The first method is brute-force, where we sweep the entire voltage range with multiple points and simulate the behavior of the comparator at each of these points using Equation 3.1. Once completed, the probability of metastability is the number of points where the comparator is metastable divided by the total number of points.

$$T_{reg} = \tau \ln \left(\frac{V_{DD}}{|V_{S/H} - V_{DAC}|} \right) \tag{3.1}$$

This method is straightforward and simple to implement. However, it has a significant drawback when the probability of metastability is very low, which is the case in most practical scenarios. The resolution of the probability is limited by the total number of points used in the simulation. For instance, to compute a probability below 10^{-15}, we would need at least a vector containing 10^{15} points, making this process highly time-consuming and impractical for higher resolutions.

The second method, proposed in [17] and used in this discussion, is built using the graphical information provided by Figure 5.1. Firstly, we assume that metastability only

occurs around each DAC value. Thus, we start our simulations centered on each DAC value and add a very small quantity Δy_i where i represents the number of the DAC value under study. We further assume that Δy_i is the same to the right and left of the DAC values, which holds true for very small values (in fact, if the value of Δy_i is small, the decision time controlled by the binary search will only differ by this quantity, making both sides practically symmetrical).

Next, we apply an iterative optimization algorithm to compute these metastability margins for each DAC value and compute the total probability as:

$$P_{meta} = \sum_{i=0}^{2^B-1} 2 \times \Delta y_i \tag{5.2}$$

To optimize this process when metastability is high, we limit the upper bound of Δy_i to half an LSB, $\max(\Delta y_i) = \frac{1}{2^{B+1}}$, which is valid because if the comparator is metastable at all points, all Δy_i's will be $\frac{1}{2^{B+1}}$ and the probability of metastability will be 100%.

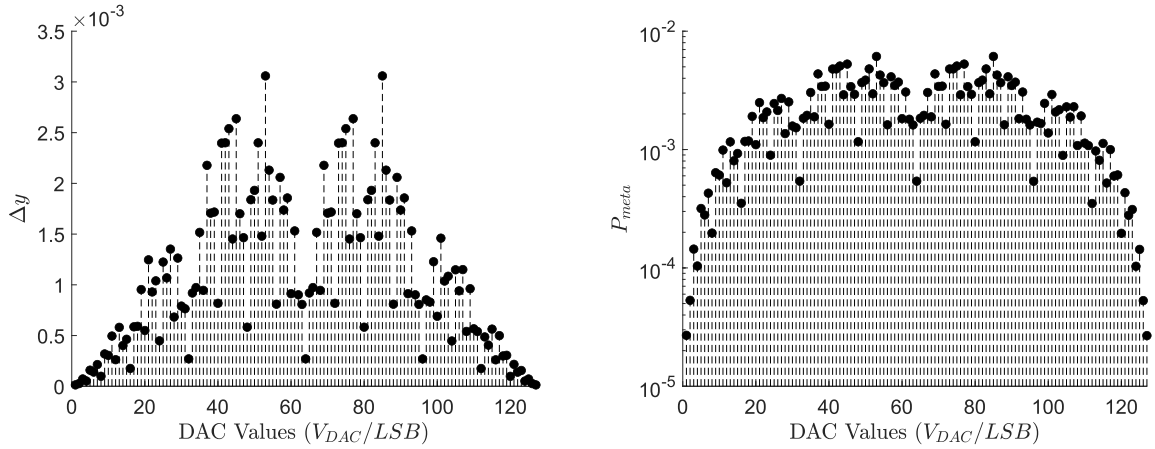


Figure 5.2: Probability of metastability and Δy for each DAC value computed for $B = 7$, $N_{DC} = 75$, and $N_{Fix} = 8$

In Figure 5.2, the computed Δy_i and the probability distribution for each DAC value, represented on a logarithmic scale, are illustrated for a 7-bit resolution, $N_{DC} = 75$, and a comparator time constant per bit resolution related to the latch capture and DAC settling of $N_{Fix} = 8$. This figure clearly highlights the preferential regions where all the bit resolutions made by the binary search algorithm occur faster than in other cases.

We have computed these Δy_i using two methods: the bisection method implemented

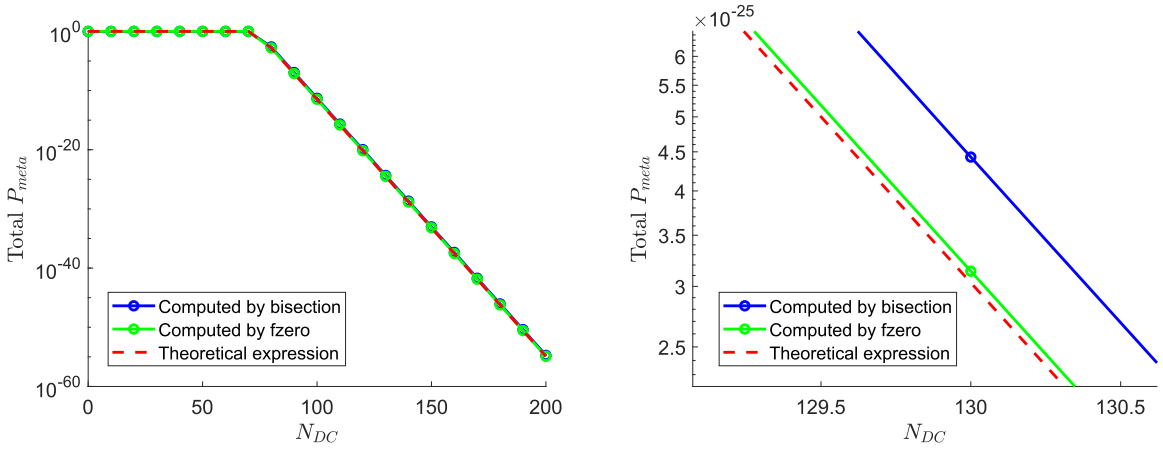


Figure 5.3: Comparison of computed probability of metastability using MATLAB function *fzero* and bisection against the theoretical expression. Computed for $B = 7$ and $N_{Fix} = 8$

manually and using the MATLAB integrated function *fzero*. In Figure 5.3, we observe the difference between these two methods used to compute the total metastability probability versus the theoretical expression from Equation 3.3.

The computation performed using the *fzero* function provides a higher degree of precision and incurs only a marginal increase in computation time. Consequently, the results obtained through this method will be utilized in the subsequent sections.

5.1.1.2 Probability Mass Function

To compute the probability mass function (PMF), we must transform the previously computed probability of metastability into the probability of occurrence of a specific code error. This is achieved through the following approach:

These comparators typically have a bistable element at their output to store the result. We assume that this bistable is designed to store bits sequentially, starting from the reset code of all bits being zero. Consequently, if no valid logic signals reach its input, the output will remain at the reset code from the bit where metastability occurs onward.

Referring back to Figure 5.1, we can observe different colors representing regions of metastability. In the 3-bit case, the blue regions indicate where the least significant bit (LSB) experiences metastability, the dark green areas correspond to metastability in the second most significant bit (MSB), and the dark red areas indicate metastability in the

MSB.

The magnitude of the error varies depending on the bit where metastability occurs. We will illustrate this using the case represented in the figure with $V_{S/H}$ for the 3-bit example.

In this case, the voltage should be represented by the code 101. Table 5.1 shows the binary process with all possible cases of metastability.

Table 5.1: Binary search procedure and all possible code errors for the 3-bit case

Binary Search	X00	1X0	10X	101
↓	↓	↓	↓	↓
Metastability	000	100	100	101
↓	↓	↓	↓	↓
Code Error	-5	-1	-1	0

To compute the new margins for each bit, we modify the previously used script and compute a new set of variables, $\Delta y_{i,b}$, where i represents the DAC value and b the bit under study. The result for this computation in the 7-bit case is shown in Figure 5.4a.

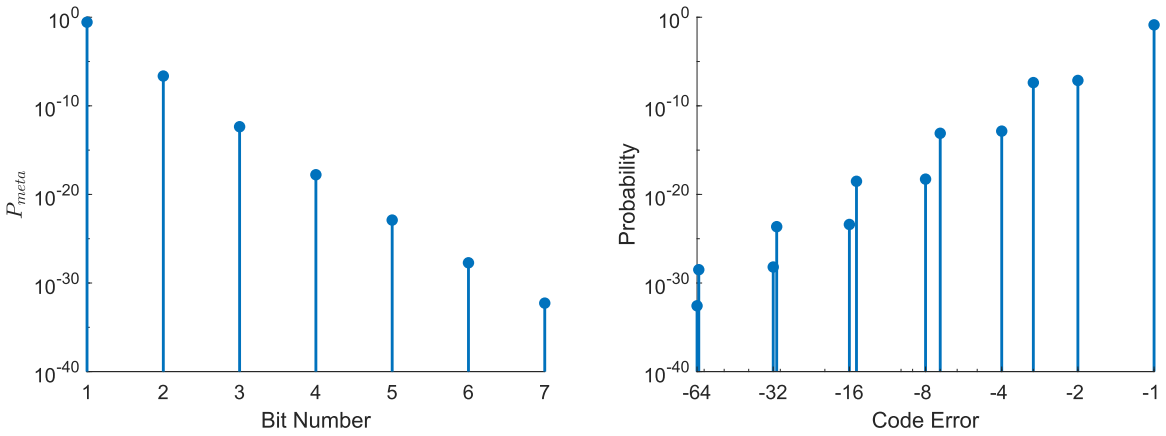


Figure 5.4: Probability of metastability for each bit (a), and probability mass function (PMF) computed for $B = 7$, $N_{DC} = 75$, and $N_{Fix} = 8$ (b)

Finally, to compute the PMF observed in Figure 5.4b, we simulate each $\Delta y_{i,b}$ individually, compute the ideal code it would result in without metastability, and then compute the actual code. The difference between the ideal and actual codes gives the code error. The probability of each code error is then summed and plotted against each code error.

We observe the appearance of code errors in groups of two. This is due to the metastability error occurring to the left or right of the DAC values, leading to a variation of just one LSB between the two or ±1 in code error.

5.1.2 Noise Only Model

To compute the total probability of code error, considering both metastability and noise, we should ideally develop a combined model accounting for their interaction. However, this approach is highly time-consuming. As demonstrated in [17], a first-order approximation using the convolution of the PMF due to metastability alone and the PMF due to noise alone can accurately describe the results obtained from the combined model. Therefore, our focus in this section will be exclusively on computing the PMF due to noise.

Given that computing errors due to noise requires extensive simulation data to obtain precise probability values, we will adopt a more statistical approach based on the work presented in [39].

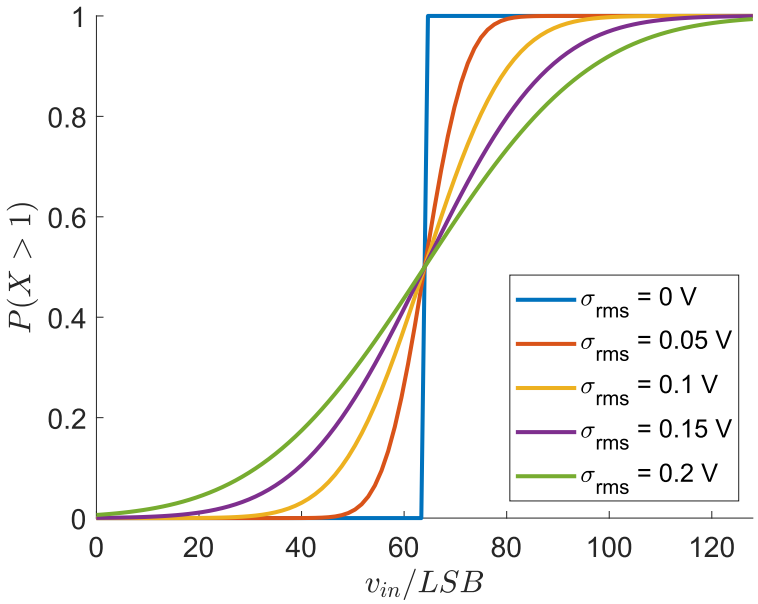


Figure 5.5: Simulated cumulative distribution function due to thermal noise

At each comparison, the input-referred noise, σ_{in}^2, along with the applied voltage, v_{in}, appears at the comparator input. Thus, the probability density function (pdf) of the

noise sum, V_n, is Gaussian with variance σ_{in}^2:

$$f_{V_n}(v_n) = \frac{1}{\sqrt{2\pi}\sigma_{\text{in}}} e^{\frac{-v_n^2}{2\sigma_{\text{in}}^2}} \tag{5.3}$$

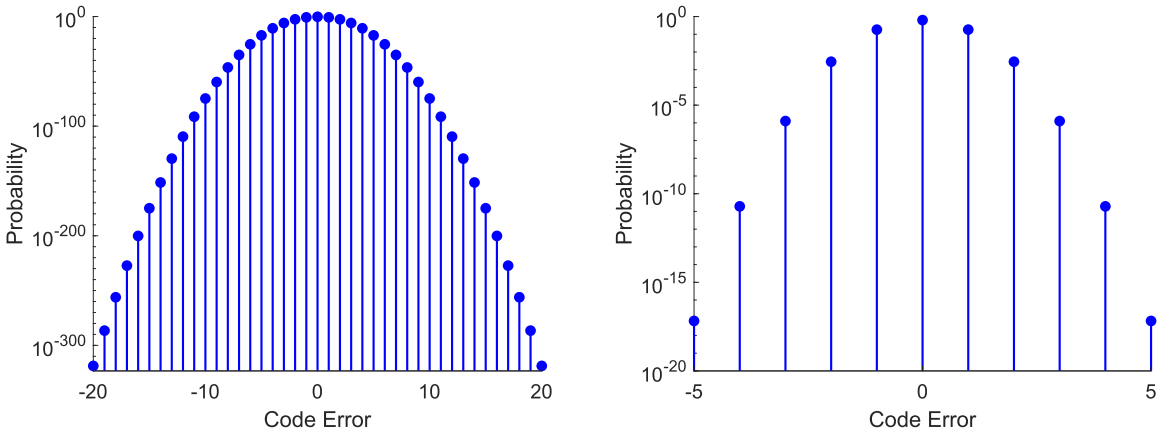
The probability of the MSB being 1 for a given v_{in} can be computed as follows:

$$\begin{aligned} P[D_{B-1} = 1|v_{in}] &= P\left[V_n + v_{in} \geq \frac{1}{2}\right] = P\left[V_n \geq \frac{1}{2} - v_{in}\right] \\ &= \int_{\frac{1}{2}-v_{in}}^{\infty} \frac{1}{\sqrt{2\pi}\sigma_{\text{in}}} e^{\frac{-x^2}{2\sigma_{\text{in}}^2}} dx \\ &= \Phi\left(\frac{v_{in} - \frac{1}{2}}{\sigma_{\text{in}}}\right) \end{aligned} \tag{5.4}$$

where $\Phi(x)$ is the cumulative distribution function (CDF) of the standard normal distribution:

$$\Phi(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{x} e^{-\frac{t^2}{2}} dt \tag{5.5}$$

This can be computed in MATLAB using the *erf(x)* or *erfc(x)* functions for increased precision.



(a) (b)

Figure 5.6: Probability mass function due to noise only (a), and zoomed region of PMF (b) computed for $B = 7$ and ENOB $= 6$

To compute the conjugate probability:

$$P[D_{B-1} = 0|v_{in}] = 1 - \Phi\left(\frac{v_{in} - \frac{1}{2}}{\sigma_{\text{in}}}\right) \stackrel{\text{def}}{=} \Phi_c\left(\frac{v_{in} - \frac{1}{2}}{\sigma_{\text{in}}}\right) \tag{5.6}$$

In Figure 5.5, we observe the probability of obtaining a 1 at the comparator output for the entire range of v_{in} voltages, varying the value of the Gaussian deviation (i.e., the amount of input-referred noise being injected).

To compute the total probability due to the succession of decisions made by the binary search:

$$P[D = d|v_{in}] = \prod_{i=1}^{B} \Phi_{B-i}\left(\frac{v_{in} - d_{c,B-i}}{\sigma_{\text{in}}}\right) \tag{5.7}$$

where $d_{c,B-i}$ can be computed recursively using:

$$d_{c,B-i} = 2^{-i} + \sum_{j=B-i+1}^{N-1} 2^{-(B-j)} d_j \tag{5.8}$$

and d_j represents the logic value of the previously resolved bits.

In Figure 5.6, we observe the PMF due to noise, computed using Equations 5.7 and 5.8 for each possible output code and for each possible input value. ENOB is used here as a proxy for computing σ_{in}, also accounting for quantization noise.

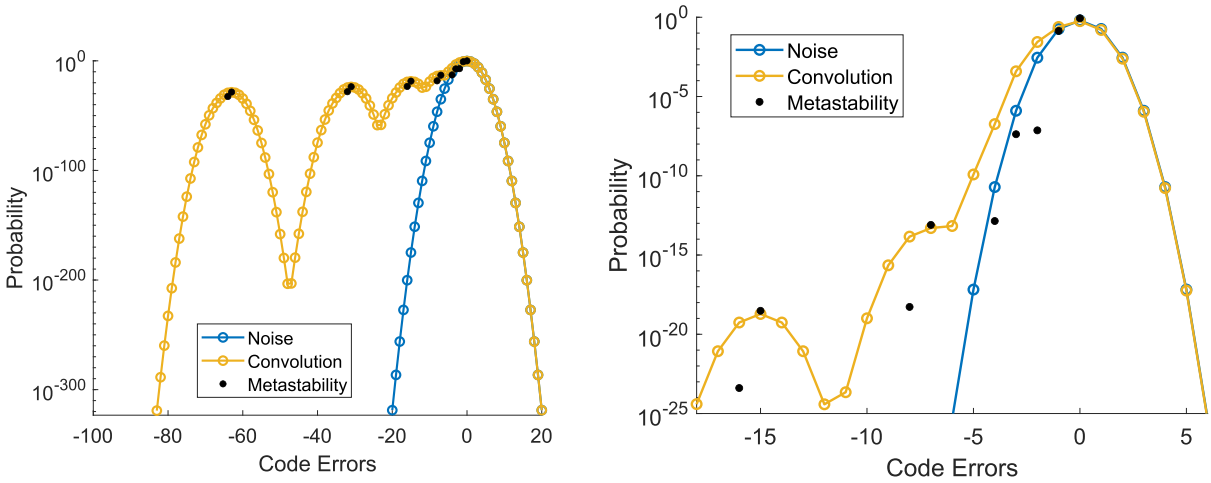


Figure 5.7: Convolved PMF of metastability and noise (a), and zoomed region of PMF computed for $B = 7$, $N_{DC} = 75$, $N_{Fix} = 8$, and ENOB = 6 (b)

5.1.3 Combined model of metastability and noise

In Figure 5.7, we observe the convolved PMF obtained from the metastability-only and noise-only procedures. The feasibility of computing the combined model through the convolution of individual models is explained by supposing that both contribute additively. Thus, at each point where we have metastability, there will also be contributions due to noise, assuming both sources to be independent.

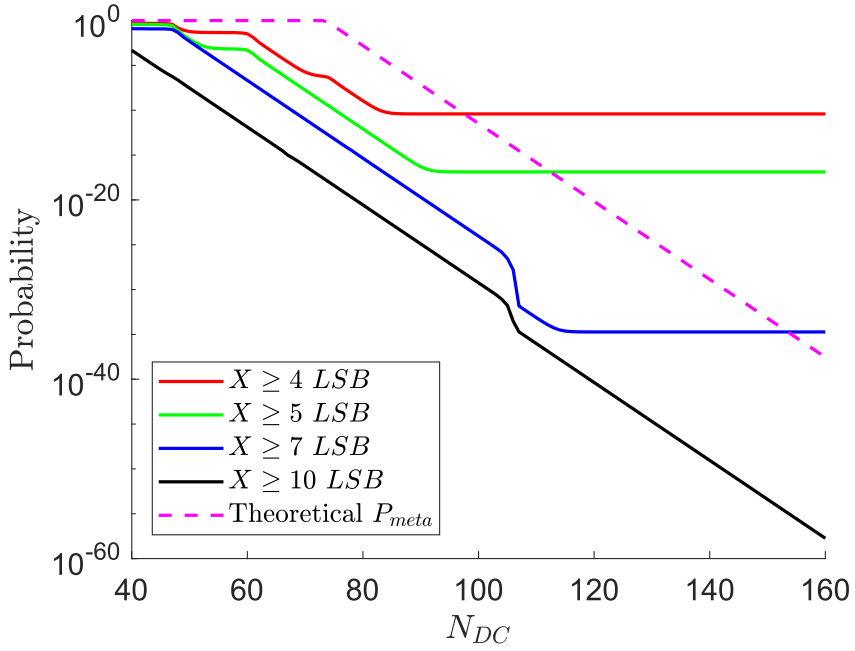


Figure 5.8: Probability of Code Error greater than X computed for $B = 7$, $N_{Fix} = 8$, and ENOB $= 6$

Finally, we can compute the probability of occurrence of a certain code error greater than X. The value of X will determine the input precision of the comparator in terms of the number of LSBs. For example, for $X \geq 4$ LSB in the 7-bit case, we would be considering a precision of roughly 31 mV.

In Figure 5.8, we observe these probabilities for several values of X as a function of the number of N_{DC}. The theoretical probability of metastability alone is represented to show that, at high speeds, the dominant contributor to the PMF is metastability (all probabilities have the same slope as the theoretical expression in this region). For lower speeds, we observe asymptotes due to thermal noise. The position of the asymptote varies depending on the input resolution.

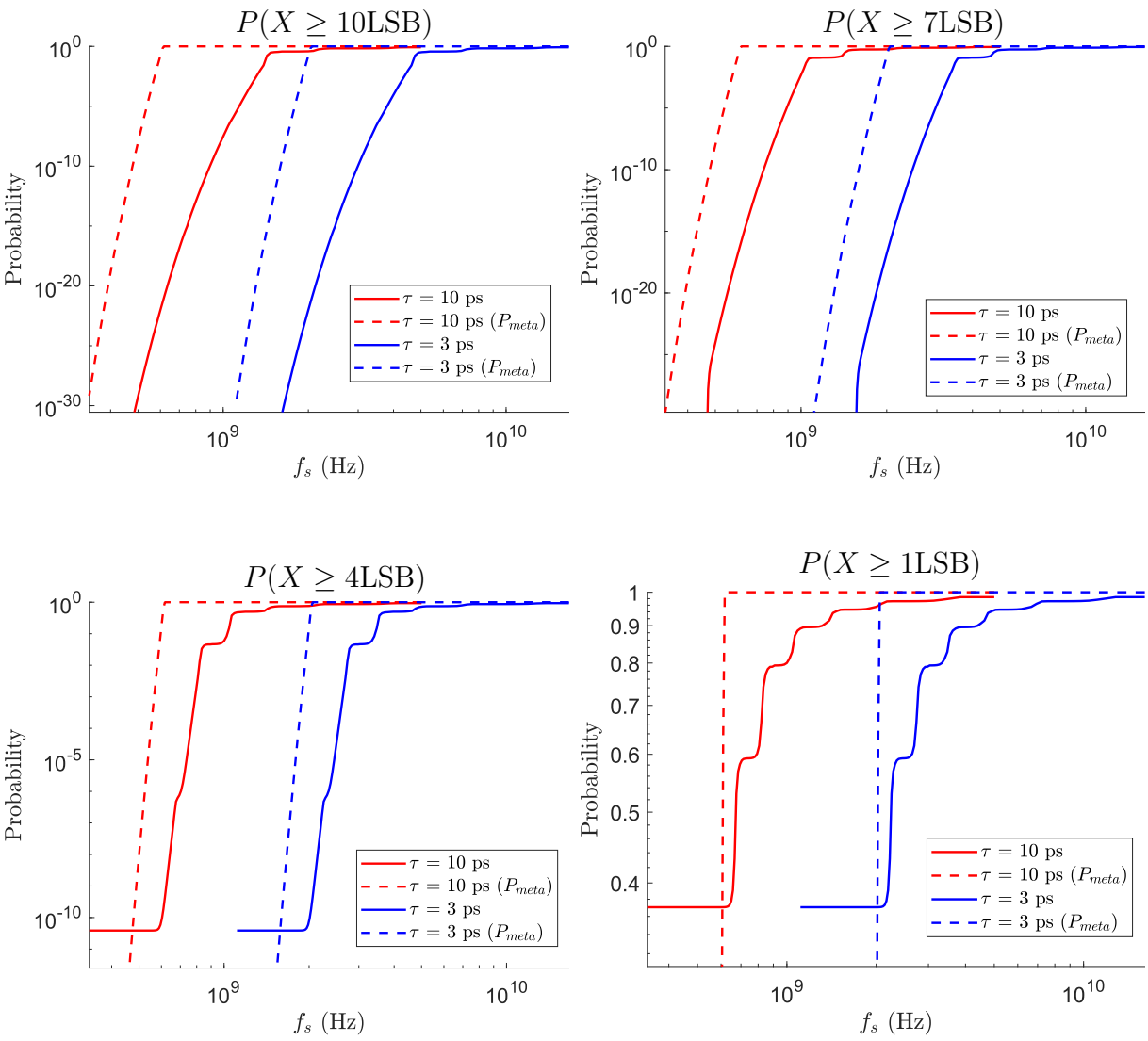


Figure 5.9: Probability of error versus sampling frequency for comparator time constants $\tau = 10$ ps and 3 ps.

In Figure 5.9, we have plotted the probability of error as a function of the sampling frequency, assuming a comparator duty cycle of 50%. Each of these probabilities has been plotted for comparator time constants of $\tau = 10$ ps and 3 ps, which represent a good range for state-of-the-art latch-based comparators. The purpose of these graphs is to aid the designer in selecting the required comparator time constant and input-referred noise for a given probability of error, sampling frequency, and input resolution.

5.2 Obtaining the Input-Referred Noise of the Comparator

To refine CMOS transistor sizing, it is essential to have efficient methods for characterizing the required parameters, specifically the comparator time constant τ and the input-referred noise σ_{in}.

Characterizing τ involves performing a transient simulation to measure the time it takes for the output signal to reach the minimum voltage level where the logical value is valid. This process is computationally inexpensive and can be automated for iterative optimization.

In contrast, characterizing the input-referred noise requires significant computational effort, posing challenges when optimizing transistor sizing for speed or power consumption.

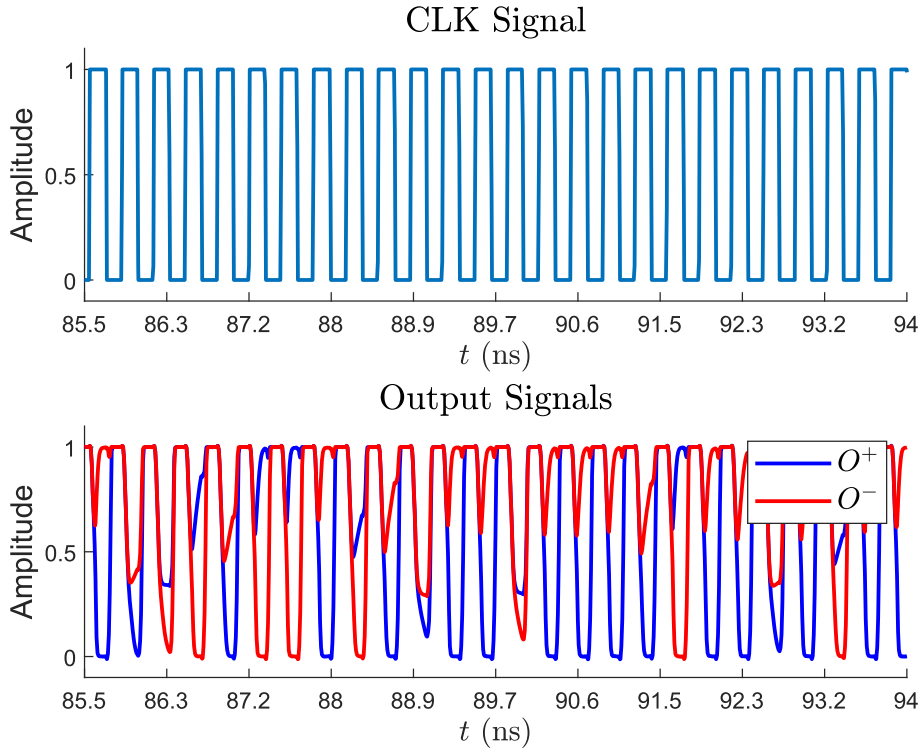


Figure 5.10: Example of transient simulation with noise for $v_{\text{in}} = 0.25$ mV, illustrating the decision errors when the input is reasonably low.

The method illustrated in Figure 5.10 is typically used for this purpose. It involves running extended transient simulations with noise at a constant clock frequency for a specific input voltage value. The comparator outputs are recorded to obtain a statistical

probability of the output being a logic 1 (or alternatively, the probability of being a logic 0).

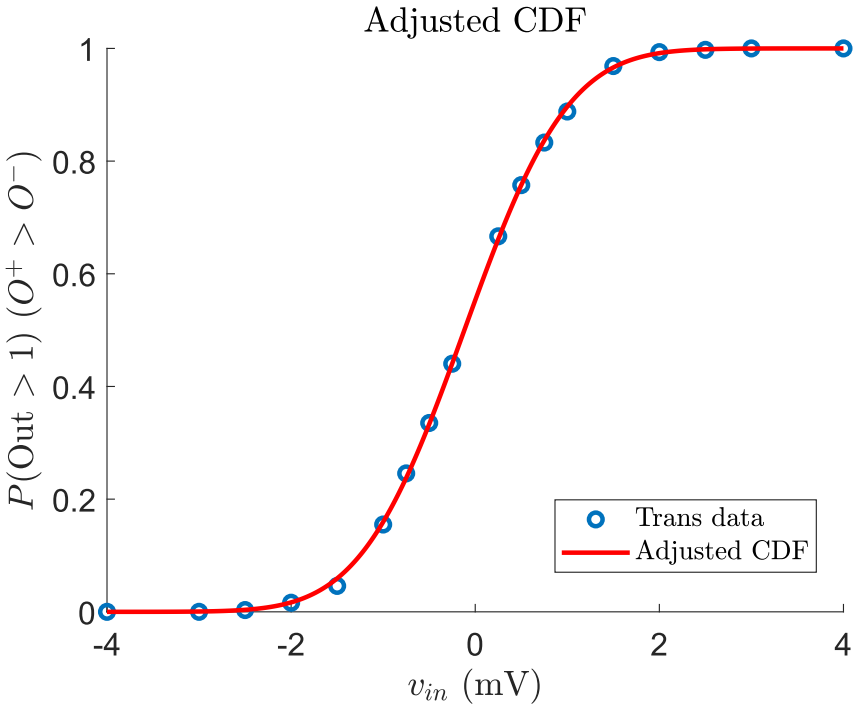


Figure 5.11: Adjusted CDF

The transient data, represented by dots in Figure 5.11, is obtained by performing this procedure over a voltage sweep in the range of -4 mV to 4 mV. Finally, this data is fitted to the curve described by the CDF, as seen in Figure 5.5:

$$\Phi\left(\frac{x-\mu}{\sigma}\right) = \frac{1}{2}\left[1 + \text{erf}\left(\frac{x-\mu}{\sigma\sqrt{2}}\right)\right] \tag{5.9}$$

where σ represents the deviation of the input-referred noise and μ is the intrinsic offset, excluding the contribution of mismatch.

Table 5.2: Transistor dimensions referred to Figure 5.12 with unitary length $L_x = 30$ nm and unitary width $W_x = 100$ nm

M_1	M_2	M_3	M_4	M_5	M_6	M_{12}	M_{13}	M_{23}	T_1	T_2	T_3
80/1	34/1	10/1	20/1	8.6/1	2.5/1	20/1	6/1	10/1	20/1	22/1	20/1

To validate this simulation, we employed the Triple-Tail Latch comparator depicted in Figure 5.12 with the transistor sizing detailed in Table 5.2, the same values given in [33] for 28 nm technology. Using this method, we obtained an input-referred noise

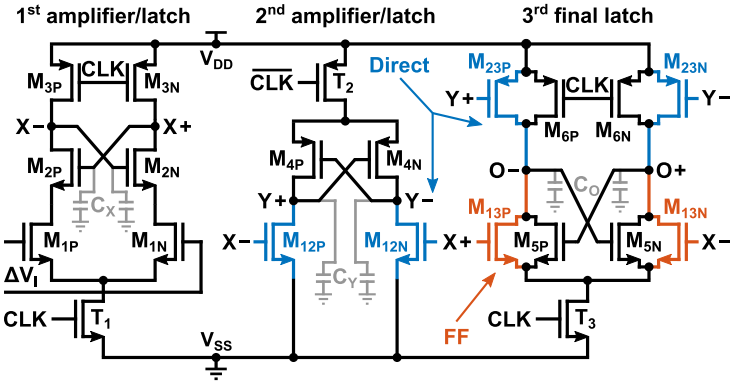


Figure 5.12: Feedforward Triple-Tail schematic

$\sigma_{\text{in,rms}} = 0.89$ mV and an offset of $\mu = -0.12$ mV. The simulations were completed within 15 minutes using parallel simulations, with post-processing in MATLAB necessary for fitting the data.

A more time-efficient method is proposed in [40], utilizing the Periodic Steady State (PSS) simulation available in some simulation software, such as Cadence Virtuoso, for characterizing periodic circuits like oscillators. The key is to use a constant clock frequency (as in the previous simulation, typically the one later imposed by the sampling frequency) so that the system output becomes periodic.

PSS analysis linearizes the circuit around the specified periodic frequency rather than using an operating point, as in DC analysis. Once set up, two additional analyses are required: PAC and PNOISE, analogous to AC analysis and noise analysis in DC domain. The challenge lies in selecting the observation time for the PAC and PNOISE analysis when the output gain and noise are computed in the final analysis.

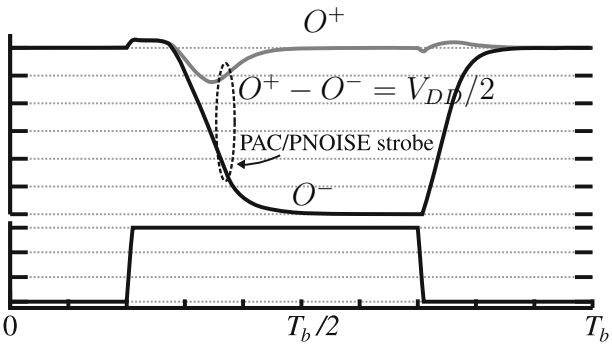


Figure 5.13: Illustrative position of observation time for PAC as indicated in [41]

In Figure 5.13, we observe the rule of thumb provided in [41], the same authors of [40].

They conclude that using the point where both outputs differ by $V_{DD}/2$ is a good starting point for selecting the time instant for PAC, as the comparator is already in regeneration at this time. This is necessary because, at this point, there is no further sampling of the input signal, and the SNR is held almost constant [42].

Using this method, we need to perform a single simulation with the described analyses, which are much more time-efficient than a single transient simulation. Once the simulation is run, since PNOISE presents the output noise as a Power Spectral Density (PSD), we need to integrate it over the simulation time range:

$$\sigma_{n,o}(t_{\text{obs}}) = \sqrt{\int_0^{1/2T_b} \text{PSD}(f, t_{\text{obs}})\, df} \tag{5.10}$$

After obtaining the output noise, we divide by the stage gain to obtain the input-referred noise:

$$\sigma_{n,i}(t_{\text{obs}}) = \frac{\sigma_{n,o}(t_{\text{obs}})}{G} \tag{5.11}$$

Using this method, we obtained $\sigma_{n,i}(t_{\text{obs}}) \approx 0.93$ mV, representing a relative error of only 4.5% compared to the result obtained from the transient simulation.

Table 5.3: Percentage Contribution to Total Input-Referred Noise

Transistor	Contribution (%)
M_{1P}	23.6
M_{1N}	23.6
M_{3N}	6.3
M_{3P}	6.3
M_{4N}	5.7
M_{4P}	5.7

This method also allows us to identify the transistors contributing the most to the input-referred noise. Table 5.3 displays these values, showing that the primary contributors are the transistors of the differential pair, as anticipated.

On the whole, the transient simulation method, while more time-consuming, provides a reliable measure of these parameters. The PSS, PAC, and PNOISE analysis methods offer a more efficient alternative, yielding comparable results with significantly reduced

simulation time. These methodologies facilitate the optimization of comparator design in asynchronous SAR ADCs for RADAR applications, balancing speed, power consumption, and noise performance. Transient simulations can be utilized to refine the results already obtained with the PNOISE method.

5.3 Route Map for Designers: Steps to Design Good Comparators for RADAR Applications

Designing comparators for RADAR applications involves several critical steps to ensure optimal performance. Below is a systematic route map indicating the essential steps designers should follow:

1. **Define Required Specifications:**
    - Determine the required input resolution.
    - Specify the sampling frequency.
    - Establish the desired probability of error.
    - Choose an initial guess for N_{fix} based on preliminary electrical simulations.
2. **Initial Comparator Selection:**
    - Start with a simple comparator topology, such as the Strong-ARM.
    - If your application is highly sensitive to kickback, consider using a double-tail comparator instead.
3. **Simulate Probability of Code Error:**
    - Simulate the probability of code error greater than the specified threshold X against the number of comparator time constants used for the converter's duty cycle (N_{DC}).
4. **Vary Effective Number of Bits (ENOB):**

- Adjust the ENOB until the asymptote due to noise coincides with the required probability of error.
- This step helps in calculating the maximum tolerable input-referred noise.

5. **Calculate Comparator Time Constant (τ):**
   - Obtain the comparator time constant (τ) from the number of comparator time constants used for the converter's duty cycle (N_{DC}).
   - Note that τ values ranging from 10 ps to 3 ps are the lowest possible, considering current technologies.
6. **Compute Noise through PNOISE Analysis:**
   - Use the Periodic Steady State (PSS) simulation to linearize the circuit around the specified periodic frequency.
   - Perform PNOISE analysis to ensure the maximum input-referred noise is satisfied.
   - Integrate the noise Power Spectral Density (PSD) over the simulation time range to calculate the input-referred noise.
7. **Refine Solution with Transient Simulation:**
   - Conduct transient simulations to further refine the solution obtained from PNOISE analysis.
   - Validate the input-referred noise and offset obtained through the PNOISE analysis by comparing with transient simulation results.
8. **Iterative Adjustment:**
   - If the desired τ is not compatible with the required input-referred noise, either modify the initial guess for N_{fix} and repeat the process or consider using an alternative comparator topology from those presented in this thesis.
   - Continue iterating until a feasible solution is found, or reevaluate the initial requirements if necessary.

Following this route map ensures a systematic approach to designing comparators that meet the stringent requirements of RADAR applications, balancing the trade-offs between speed, noise performance, and error probability.

CHAPTER 6

Conclusion and Future Lines

This thesis has synthesized and condensed various published works to create a comprehensive framework for the design and optimization of Successive Approximation Register (SAR) Analog-to-Digital Converters (ADCs), particularly for RADAR applications. By replicating the individual outputs of each work and combining them under a single workflow, we have established useful design guidelines and a full modeling process for SAR ADCs. The effort to integrate these methodologies into a coherent structure provides significant benefits, allowing for easy modifications and adaptations through clear MATLAB code.

6.1 Conclusions

Throughout this work, we have achieved the following key milestones, presented in chronological order:

- **Analysis of Latch-Based Comparators**: We examined various latch-based comparator architectures, including the Strong-ARM and Double-Tail comparators, identifying their strengths and limitations, and proposed the Triple-Tail Feedforward comparator to address these issues.
- **Discussion of Trade-offs**: We analyzed the trade-offs between comparator speed, probability of error, and noise performance, providing guidelines for optimizing these

parameters to meet the specific requirements of SAR ADCs in RADAR applications.

- **Modeling of Metastability for Synchronous and Asynchronous Architectures**: We developed models to estimate the probability of metastability for both synchronous and asynchronous architectures, highlighting the benefits of asynchronous designs in reducing metastability.

- **Noise Modeling and PMF Due to Noise**: We derived mathematical expressions to estimate noise in comparators and developed a probability mass function (PMF) due to noise, validated through extensive simulations.

- **Combined Model of Metastability and Noise**: We integrated the individual models of metastability and noise to create a comprehensive model. This model can detect when metastability or noise is the dominant factor depending on comparator speed.

- **Method for Determining Input-Referred Noise**: We presented methods to efficiently characterize the comparator time constant τ and input-referred noise σ_{in}, including both transient simulations and advanced techniques using Periodic Steady State (PSS) and PNOISE analyses.

This integration of multiple methodologies into a single, clear MATLAB framework provides an adaptable and powerful tool for SAR ADC design, facilitating modifications and optimizations as needed. Overall, this work has established a solid foundation for the design and optimization of SAR ADCs, emphasizing the critical role of the comparator and providing practical tools for achieving high performance.

6.2 Future Lines

Building upon the foundations laid in this thesis, several avenues for future research and development can be pursued:

- **Incorporation of Second-Order Phenomena**: While this work combined existing frameworks, future efforts could focus on refining the comparator model used in

simulations by incorporating second-order phenomena. This could include specifying the minimum voltage for logic detection, the maximum, and minimum comparator voltage levels (V_{OH} and V_{OL}) or including the dependency of the speed resolution with common mode voltage. This enhancement would improve the accuracy of physical design predictions and aid designers in achieving optimal performance.

- **Exploration of Hybrid Architectures**: Extending this work to hybrid architectures, such as pipe-SAR ADCs, which are particularly relevant for RADAR systems, could provide further insights and design guidelines.

- **Development of Power Consumption Models**: Exploring and developing power consumption models to include in the design guidelines will be crucial for optimizing SAR ADCs for low-power applications.

- **Study of CDAC Design**: Investigating the design of Capacitive Digital-to-Analog Converters (CDACs) to include their contribution in the modeling process will help in creating more accurate and comprehensive design frameworks.

- **Modeling of Reference Architectures**: For pipeline-SAR architectures, modeling residual amplifiers (RAs) will be essential to understand their impact on overall performance and to develop effective design strategies.

In summary, this thesis has successfully integrated various published works to create a comprehensive modeling framework for SAR ADC design. The significant effort to replicate and combine individual contributions under a single workflow has resulted in a robust tool for designers. Future research can build upon this foundation, increasing model complexity and incorporating more sophisticated design elements to further aid designers in developing high-performance ADCs.

CHAPTER 7

MATLAB Codes

This appendix contains the MATLAB codes used for the simulations and analyses presented in the thesis. The following listings include scripts and functions crucial for the development and validation of the models discussed.

7.1 Metastability margins

Listing 7.1: MATLAB function for simulating the margin of metastability Δy of each DAC value.

```
function Dx = compute_meta_margin_fzero(B, NDC, Nfix, tolerance, Bmeta,
    upper_range)

    % Set default values if not provided
    if nargin < 4 || isempty(tolerance)
        tolerance = realmin;
    end
    if nargin < 5 || isempty(Bmeta)
        Bmeta = 1;
    end
    if nargin < 6 || isempty(upper_range)
        upper_range = 1/(2^(B+1));
```

```
12  end
13
14  % Parameters
15  lsb = 1 / (2^B);
16
17  % DAC values
18  kix = 1:2^(B) - 1;
19  DACx = kix * lsb;
20
21  % Initialize Dx
22  Dx = zeros(1, length(DACx));
23
24  % Options for fzero with user-defined tolerance
25  options = optimset('TolX', tolerance);
26
27  % Calculate Dx for each DAC value
28  for i = 1:length(DACx)
29      % Define the function whose root we are trying to find
30      fun = @(Dx) NDC - compute_Nreg(DACx(i), Dx, B, Nfix, Bmeta);
31
32      % Use fzero to find the root within the specified range
33      try
34          Dx(i) = fzero(fun, [realmin, upper_range], options);
35      catch
36          % Update Dx based on the sign of fun
37          if fun(realmin) > 0
38              Dx(i) = 0; % not metastable
39          else
40              Dx(i) = upper_range; % metastable in all domain
41          end
42      end
43  end
44
```

```
    % Helper function to compute Nreg
    function Nreg = compute_Nreg(DACx, Dx, B, Nfix, Bmeta)
        XSH = DACx + Dx;
        Xin = zeros(1, B+1);
        Xin(B+1) = XSH;
        for b = B:-1:1
            Xin(b) = Xin(b+1) - 1/2^(B-b+1) * sign(Xin(b+1));
            % Correct any Xin(b) that is 0 to be Dx
            if Xin(b) == 0
                Xin(b) = Dx;
            end
        end
        Nreg = sum(log(1./abs(Xin(Bmeta:B)))) + (B-Bmeta) * Nfix; % Total
            regenerative time (all bits summed) relative to regenerative tau
    end
end
```

7.2 Probability of metastability

Listing 7.2: MATLAB function for simulating the probability of metastability as function of N_{DC}

```
function Pmeta_total = compute_Pmeta_total_fzero(B, NDC, Nfix, tolerance)
    % Parameters
    lsb = 1 / (2^B);

    % DAC values
    kix = 1:2^(B) - 1;
    DACx = kix * lsb;

    % Initialize Dx
    Dx = zeros(1,length(DACx));
```

```
11
12  % Options for fzero with TolX set to realmin
13  options = optimset(@fzero);
14  options.('TolX') = realmin;
15
16  for i = 1:length(DACx)
17      % Define the function whose root we are trying to find
18      fun = @(Dx) NDC - compute_Nreg(DACx(i), Dx, B, Nfix);
19
20      % Use fzero to find the root within the specified range
21      try
22          Dx(i) = fzero(fun, [realmin, 1/(2^(B+1))], options);
23      catch
24          % Update Dx based on the sign of fun
25          if fun(realmin) > 0
26              Dx(i) = 0;
27          else
28              Dx(i) = 1 / (2^(B+1));
29          end
30      end
31  end
32
33  % Helper function to compute Nreg
34  function Nreg = compute_Nreg(DACx, Dx, B, Nfix)
35      XSH = DACx + Dx;
36      Xin = zeros(1, B+1);
37      Xin(B+1) = XSH;
38      for b = B:-1:1
39          Xin(b) = Xin(b+1) - 1/2^(B-b+1) * sign(Xin(b+1));
40          % Correct any Xin(b) that is 0 to be Dx
41          if Xin(b) == 0
42              Xin(b) = Dx;
43          end
```

```
        end
        Nreg = sum(log(1./abs(Xin(1:B)))) + (B-1) * Nfix; % Total regenerative
            time (all bits summed) relative to regenerative tau
    end

    Pmeta_total = sum(Dx) * 2;
end
```

7.3 Probability mass function due to metastability

Listing 7.3: MATLAB function for simulating the PMF due to metastability only

```
function [filtered_Pcode_error, filtered_code_errors] =
    compute_meta_probability_of_code_errors(B, NDC, Nfix)
    % Function to compute the probability of each possible code error for B
        bits

    % ADC parameters
    lsb = 1 / (2^B);

    % DAC values
    kix = 1:2^B - 1;
    DACx = kix * lsb;

    %% Fzero method
    % Dy values for each bit in each DAC value
    DxB_values = cell(1, B);
    for i = 1:B
        DxB_values{i} = zeros(1, length(DACx));
    end

    for m = 1:B
```

```
19      if m == 1 % uses previous bound as upper bound
20          DxB_values{m} = compute_meta_margin_fzero(B, NDC, Nfix);
21      else
22          DxB_values{m} = compute_meta_margin_fzero(B, NDC, Nfix, [], m,
                max(DxB_values{m-1}));
23      end
24  end
25
26  %% Calculate the probability of each possible code error for B bits
27  code_errors = [0, -kix]; % Include 0 in code errors
28  Pcode_error = zeros(1, length(code_errors));
29
30  for j = 1:length(DACx) % positive extension (+Dy)
31      Xin = DACx(j) + DxB_values{1}(j);
32
33      % Compute probabilities and update code error probabilities
34      ki = 0;
35      for i = B:-1:1
36          kib = 2^(i-1);
37          ki = ki + kib;
38          Ydac = ki * lsb;
39
40          if DxB_values{i}(j) ~= 0 % if metastability
41              if i == B
42                  prob = DxB_values{i}(j);
43              else
44                  prob = DxB_values{i}(j) - DxB_values{i+1}(j);
45              end
46
47              code_error = (ki - kib) - kix(j);
48              idx = find(code_errors == code_error, 1);
49              if ~isempty(idx)
50                  Pcode_error(idx) = Pcode_error(idx) + prob;
```

```
            end
        end

        % Comparator
        if Ydac > Xin
            ki = ki - kib;
        end
    end
end

for j = 1:length(DACx) % negative extension (-Dy)
    Xin = DACx(j) - DxB_values{1}(j);

    % Compute probabilities and update code error probabilities
    ki = 0;
    for i = B:-1:1
        kib = 2^(i-1);
        ki = ki + kib;
        Ydac = ki * lsb;

        if DxB_values{i}(j) ~= 0 % if metastability
            if i == B
                prob = DxB_values{i}(j);
            else
                prob = DxB_values{i}(j) - DxB_values{i+1}(j);
            end

            if kix(j) == 1
                code_error = (ki - kib);
            else
                code_error = (ki - kib) - kix(j-1);
            end
            idx = find(code_errors == code_error, 1);
```

```
                if ~isempty(idx)
                    Pcode_error(idx) = Pcode_error(idx) + prob;
                end
            end

            % Comparator
            if Ydac > Xin
                ki = ki - kib;
            end
        end
    end

    Pcode_error(1) = Pcode_error(1) + (1 - 2 * sum(DxB_values{1})); % zero
        correction

    % Filter out zero probabilities
    non_zero_idx = Pcode_error > 0;
    filtered_code_errors = code_errors(non_zero_idx);
    filtered_Pcode_error = Pcode_error(non_zero_idx);
end
```

7.4 Probability mass function due to noise

Listing 7.4: MATLAB function for simulating the PMF due to noise only

```
function Pcode_error_normalized = compute_noise_normalized_probability(B,
    ENOB, sigma_factor)
    % Function to compute normalized probability of code errors

    % ADC parameters
    lsb = 1 / (2^B);

```

```
7   % DAC values
8   kix = 1:2^B - 1;
9
10  % Noise computation for diff input (only quantization and input referred
        noise)
11  SNR = 6.02 * ENOB + 1.76;
12  % sigma2 = 1 / (2 * 10^(SNR / 10)) - lsb^2 / 6; % Differential
13  sigma2 = 1 / (8 * 10^(SNR / 10)) - lsb^2 / 12; % one-sided
14  sigma = sqrt(sigma2);
15
16  % Functions
17  CDF = @(vin,sigma,d) 0.5 * ( erfc( -(vin-d)./sigma/sqrt(2) ) ); % using
        erfc for higher precision when erf(x) -> 1
18  CDF_c = @(vin,sigma,d) 1 - CDF(vin,sigma,d);
19
20  code_errors = 0:kix(end); % Compute only for non-negative code errors
21  npts = 1 / (sigma * sigma_factor); % step smaller than sigma
22  XSH_values = linspace(0, 1, npts); % Vector of XSH values
23  Pcode_error = zeros(1, length(code_errors)); % Initialize array to store
        accumulated probabilities
24
25  % Creations of all possible codes
26  Bcodes = dec2bin([0, kix]);
27
28  % Initialize waitbar
29  h = waitbar(0, 'Computing noise probability...');
30
31  % Iterate over each XSH value
32  for xi = 1:length(XSH_values)
33      XSH = XSH_values(xi); % Current XSH value
34
35      % Iterate over each binary code (each row in Bcodes)
36      for i = 1:2^B
```

```
        Bcode = Bcodes(i, :); % Get the i-th binary code as a string
        Pcode_error_current = 1;
        code_error = bin2dec(Bcode) - floor(XSH / lsb); % DAC-Ideal

        % Skip negative code errors
        if code_error < 0
            continue;
        end

        % Iterate over each bit in the current binary code
        d = zeros(1, B);
        dc = zeros(1, B);
        for j = 1:B
            bit = Bcode(j); % Access each bit as a character
            d(B-j+1) = str2double(bit);
            S = 0;
            for s = B-j+1:B-1
                S = S + 2^(s-B) * d(s+1);
            end
            dc(B-j+1) = 2^(-j) + S;

            % CDF computation
            if d(B-j+1) == 0
                prob = CDF_c(XSH, sigma, dc(B-j+1));
            else
                prob = CDF(XSH, sigma, dc(B-j+1));
            end
            Pcode_error_current = prob * Pcode_error_current;
        end

        % Accumulate the probability of the current code error
        code_error_index = find(code_errors == code_error);
        Pcode_error(code_error_index) = Pcode_error(code_error_index) +
```

```
                Pcode_error_current;
        end

        % Update waitbar
        waitbar(xi / length(XSH_values), h);
    end

    % Close waitbar
    close(h);

    % Normalize the probabilities by the total number of points
    Pcode_error_normalized = Pcode_error / npts;

    % Mirror the results for negative code errors
    Pcode_error_normalized = [fliplr(Pcode_error_normalized(2:end)),
        Pcode_error_normalized];
end
```

Bibliography

[1] Shahriar Mirabbasi, Laura C. Fujino, and Kenneth C. Smith. "Through the Looking Glass—The 2022 Edition: Trends in Solid-State Circuits from ISSCC". In: *IEEE Solid-State Circuits Magazine* 14.1 (2022), pp. 54–72. DOI: 10.1109/MSSC.2021.3128243.

[2] Sandeep Santhosh Kumar et al. "A 750mW 24GS/s 12b Time-Interleaved ADC for Direct RF Sampling in Modern Wireless Systems". In: *2023 IEEE International Solid-State Circuits Conference (ISSCC)* (2023), pp. 1–3. DOI: 10.1109/ISSCC42615.2023.10067793.

[3] Lei Qiu et al. "A 10-bit 300 MS/s 5.8 mW SAR ADC With Two-Stage Interpolation for PET Imaging". In: *IEEE Sensors Journal* 18 (2018), pp. 2006–2014. DOI: 10.1109/JSEN.2018.2790581.

[4] A. Konikov and T. Fedoseeva. "Analog-to-digital data processing tools in the construction industry and in the transportation sector". In: *IOP Conference Series: Materials Science and Engineering* 918 (2020). DOI: 10.1088/1757-899X/918/1/012069.

[5] K. Yoshioka et al. "A 20-ch TDC/ADC Hybrid Architecture LiDAR SoC for 240 × 96 Pixel 200-m Range Imaging With Smart Accumulation Technique and Residue Quantizing SAR ADC". In: *IEEE Journal of Solid-State Circuits* 53 (2018), pp. 3026–3038. DOI: 10.1109/JSSC.2018.2868315.

[6] Ahmed M. A. Ali et al. "A 14 Bit 1 GS/s RF Sampling Pipelined ADC With Background Calibration". In: *IEEE Journal of Solid-State Circuits* 49.12 (2014), pp. 2857–2867. DOI: 10.1109/JSSC.2014.2361339.

[7] S. Patole et al. “Automotive Radars: A Review of Signal Processing Techniques”. In: *IEEE Signal Processing Magazine* 34 (2017), pp. 22–35. DOI: 10.1109/MSP.2016.2628914.

[8] G. Tsandoulas. “Space-Based Radar”. In: *Science* 237 (1987), pp. 257–262. DOI: 10.1126/science.237.4812.257.

[9] I. Holleman et al. “Operational Monitoring of Weather Radar Receiving Chain Using the Sun”. In: *Journal of Atmospheric and Oceanic Technology* 27 (2010), pp. 159–166. DOI: 10.1175/2009JTECHA1213.1.

[10] Deqing Mao et al. “An Efficient Anti-Interference Imaging Technology for Marine Radar”. In: *IEEE Transactions on Geoscience and Remote Sensing* 60 (2022), pp. 1–13. DOI: 10.1109/TGRS.2021.3068787.

[11] W. Shrader. “Radar Technology Applied to Air Traffic Control”. In: *IEEE Transactions on Communications* 21 (1973), pp. 591–605. DOI: 10.1109/TCOM.1973.1091691.

[12] Boris Murmann. *ADC Performance Survey 1997-2024*. [Online]. Available: https://github.com/bmurmann/ADC-survey.

[13] Chun C. Lee and M. Flynn. “A SAR-Assisted Two-Stage Pipeline ADC”. In: *IEEE Journal of Solid-State Circuits* 46 (2011), pp. 859–869. DOI: 10.1109/JSSC.2011.2108133.

[14] Jie Fang et al. “A 5-GS/s 10-b 76-mW Time-Interleaved SAR ADC in 28 nm CMOS”. In: *IEEE Transactions on Circuits and Systems I: Regular Papers* 64 (2017), pp. 1673–1683. DOI: 10.1109/TCSI.2017.2661481.

[15] M.J.M. Pelgrom. *Analog-to-Digital Conversion*. 4th. Cham: Springer International Publishing, 2022. ISBN: 978-3-030-90807-2. DOI: 10.1007/978-3-030-90808-9.

[16] Jingpeng Zhou et al. “All-Digital Background Calibration of a Pipelined-SAR ADC Using the “Split ADC” Architecture”. In: *2023 IEEE International Symposium on Circuits and Systems (ISCAS)*. 2023, pp. 1–5. DOI: 10.1109/ISCAS46773.2023.10195888.

[17] Andrew Yu et al. “Understanding Metastability in SAR ADCs: Part II: Asynchronous”. In: *IEEE Solid-State Circuits Magazine* 11 (2019), pp. 16–32. DOI: 10.1109/MSSC.2019.2922890.

[18] C.E. Shannon. “Communication in the Presence of Noise”. In: *Proc. IRE* 37 (1949). Reprinted in Proc. IEEE, vol. 86, pp. 447-457, 1998, pp. 10–21.

[19] C. Laperle. “Advances in high-speed ADCs, DACs, and DSP for optical transceivers”. In: *2013 Optical Fiber Communication Conference and Exposition and the National Fiber Optic Engineers Conference (OFC/NFOEC)*. 2013, pp. 1–30. DOI: 10.1364/OFC.2013.OTH1F.5.

[20] Boris Murmann. “The Successive Approximation Register ADC: A Versatile Building Block for Ultra-Low-Power to Ultra-High-Speed Applications”. In: *IEEE Communications Magazine* 54.4 (2016), pp. 78–83. DOI: 10.1109/MCOM.2016.7452270.

[21] K. Balasubramanian. “A Flash ADC with Reduced Complexity”. In: *IEEE Transactions on Industrial Electronics* 42 (1995), pp. 106–108. DOI: 10.1109/41.345853.

[22] L. Kull et al. “A 3.1 mW 8b 1.2 GS/s Single-Channel Asynchronous SAR ADC With Alternate Comparators for Enhanced Speed in 32 nm Digital SOI CMOS”. In: *IEEE Journal of Solid-State Circuits* 48 (2013), pp. 3049–3058. DOI: 10.1109/JSSC.2013.2279571.

[23] J. Álvarez et al. “Low-power pipeline ADC for wireless LANs”. In: *IEEE Journal of Solid-State Circuits* 39 (2004), pp. 1338–1340. DOI: 10.1109/JSSC.2004.831477.

[24] K. Ohhata. “A 2.3-mW, 1-GHz, 8-Bit Fully Time-Based Two-Step ADC Using a High-Linearity Dynamic VTC”. In: *IEEE Journal of Solid-State Circuits* 54 (2019), pp. 2038–2048. DOI: 10.1109/JSSC.2019.2907401.

[25] Gopal Singh et al. “A Novel Dual-Slope Resistance to Digital Converter With Lead Resistance Compensation”. In: *IEEE Transactions on Instrumentation and Measurement* 72 (2023), pp. 1–10. DOI: 10.1109/TIM.2023.3238021.

[26] Pieter Harpe. “A Compact 10-b SAR ADC with Unit-Length Capacitors and a Passive FIR Filter”. English. In: *IEEE Journal of Solid-State Circuits* 54.3 (Mar. 2019), pp. 636–645. ISSN: 0018-9200. DOI: 10.1109/JSSC.2018.2878830.

[27] Pieter Harpe, Eugenio Cantatore, and Arthur van Roermund. "A 10b/12b 40 kS/s SAR ADC With Data-Driven Noise Reduction Achieving up to 10.1b ENOB at 2.2 fJ/Conversion-Step". In: *IEEE Journal of Solid-State Circuits* 48.12 (2013), pp. 3011–3018. DOI: 10.1109/JSSC.2013.2278471.

[28] Yong Lim and Michael P. 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.

[29] Lukas Kull et al. "22.1 A 90GS/s 8b 667mW 64× interleaved SAR ADC in 32nm digital SOI CMOS". In: *2014 IEEE International Solid-State Circuits Conference Digest of Technical Papers (ISSCC)*. 2014, pp. 378–379. DOI: 10.1109/ISSCC.2014.6757477.

[30] Shuo-Wei Michael Chen and Robert W. Brodersen. "A 6-bit 600-MS/s 5.3-mW Asynchronous ADC in 0.13-μm CMOS". In: *IEEE Journal of Solid-State Circuits* 41.12 (2006), pp. 2669–2680. DOI: 10.1109/JSSC.2006.884231.

[31] Yida Duan. "Design Techniques for Ultra-High-Speed Time-Interleaved Analog-to-Digital Converters (ADCs)". PhD thesis. EECS Department, University of California, Berkeley, May 2017. URL: http://www2.eecs.berkeley.edu/Pubs/TechRpts/2017/EECS-2017-10.html.

[32] P.E. Allen and D.R. Holberg. *CMOS Analog Circuit Design*. New Delhi, India: Oxford University Press, 2016.

[33] Athanasios T. Ramkaj et al. "A 28 nm CMOS Triple-Latch Feed-Forward Dynamic Comparator With <27 ps / 1 V and <70 ps / 0.6 V Delay at 5 mV-Sensitivity". In: *IEEE Transactions on Circuits and Systems I: Regular Papers* 69.11 (2022), pp. 4404–4414. DOI: 10.1109/TCSI.2022.3199438.

[34] T. Kobayashi et al. "A Current-Mode Latch Sense Amplifier and a Static Power Saving Input Buffer for Low-Power Architecture". In: *1992 Symposium on VLSI Circuits Digest of Technical Papers*. 1992, pp. 28–29. DOI: 10.1109/VLSIC.1992.229252.

[35] Behzad Razavi. "The StrongARM Latch [A Circuit for All Seasons]". In: *IEEE Solid-State Circuits Magazine* 7.2 (2015), pp. 12–17. DOI: 10.1109/MSSC.2015.2418155.

[36] Daniel Schinkel et al. "A Double-Tail Latch-Type Voltage Sense Amplifier with 18ps Setup+Hold Time". In: *2007 IEEE International Solid-State Circuits Conference. Digest of Technical Papers*. 2007, pp. 314–605. DOI: 10.1109/ISSCC.2007.373420.

[37] Athanasios T. Ramkaj, Michiel S. J. Steyaert, and Filip Tavernier. "A 13.5-Gb/s 5-mV-Sensitivity 26.8-ps-CLK–OUT Delay Triple-Latch Feedforward Dynamic Comparator in 28-nm CMOS". In: *IEEE Solid-State Circuits Letters* 2.9 (2019), pp. 167–170. DOI: 10.1109/LSSC.2019.2936152.

[38] Pierluigi Nuzzo et al. "Noise Analysis of Regenerative Comparators for Reconfigurable ADC Architectures". In: *IEEE Transactions on Circuits and Systems I: Regular Papers* 55.6 (2008), pp. 1441–1454. DOI: 10.1109/TCSI.2008.917991.

[39] Wenpian Paul Zhang and Xingyuan Tong. "Noise Modeling and Analysis of SAR ADCs". In: *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 23.12 (2015), pp. 2922–2930. DOI: 10.1109/TVLSI.2014.2379613.

[40] João Silva et al. "Methods for Fast Characterization of Noise and Offset in Dynamic Comparators". In: *2021 19th IEEE International New Circuits and Systems Conference (NEWCAS)*. 2021, pp. 1–4. DOI: 10.1109/NEWCAS50681.2021.9462744.

[41] Taimur Gibran Rabuske and Jorge R. Fernandes. "Noise-aware simulation-based sizing and optimization of clocked comparators". In: *Analog Integrated Circuits and Signal Processing* 81 (2014), pp. 723–728. URL: https://api.semanticscholar.org/CorpusID:62673871.

[42] Jaeha Kim et al. "Simulation and Analysis of Random Decision Errors in Clocked Comparators". In: *IEEE Transactions on Circuits and Systems I: Regular Papers* 56.8 (2009), pp. 1844–1857. DOI: 10.1109/TCSI.2009.2028449.