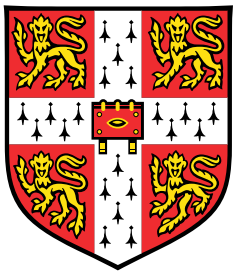
Design Techniques for High-Performance SAR A/D Converters



Mojtaba Bagheri

Department of Engineering
University of Cambridge

This thesis is submitted for the degree of
Doctor of Philosophy

Clare Hall College

July 2019

Declaration

This thesis is the result of my own work and includes nothing which is the outcome of work done in collaboration except as declared in the Preface and specified in the text. It is not substantially the same as any that I have submitted, or, is being concurrently submitted for a degree or diploma or other qualification at the University of Cambridge or any other University or similar institution except as declared in the Preface and specified in the text. I further state that no substantial part of my thesis has already been submitted, or, is being concurrently submitted for any such degree, diploma or other qualification at the University of Cambridge or any other University or similar institution except as declared in the Preface and specified in the text. It does not exceed the prescribed word limit for the relevant Degree Committee

Mojtaba Bagheri
July 2019

Abstract

Design Techniques for High-Performance SAR A/D Converter

Mojtaba Bagheri

The design of electronics needs to account for the non-ideal characteristics of the device technologies used to realize practical circuits. This is particularly important in mixed analog-digital design since the best device technologies are very different for digital compared to analog circuits. One solution for this problem is to use a calibration-correction approach to remove the errors introduced by devices, but this adds complexity and power dissipation, as well as reducing operation speed, and so must be optimised. This thesis addresses such an approach to improve the performance of certain types of analog-to-digital converter (ADC) used in advanced telecommunications, where speed, accuracy and power dissipation currently limit applications. The thesis specifically focuses on the design of compensation circuits for use in successive approximation register (SAR) ADCs.

ADCs are crucial building blocks in communication systems, in general, and for mobile networks, in particular. The recently launched fifth generation of mobile networks (5G) has required new ADC circuit techniques to meet the higher speed and lower power dissipation requirements for 5G technology. The SAR has become one of the most favoured architectures for designing high-performance ADCs, but the successive nature of the circuit operation makes it difficult to reach $\sim$GS/s sampling rates at reasonable power consumption.

Here, two calibration techniques for high-performance SAR ADCs are presented. The first uses an on-chip stochastic-based mismatch calibration technique that is able to accurately compute and compensate for the mismatch of a capacitive DAC in a SAR ADC. The stochastic nature of the proposed calibration method enables determination of the mismatch of the CAP-DAC with a resolution much better than that of the DAC. This allows the unit capacitor to scale down to as low as 280aF for a 9-bit DAC. Since the CAP-DAC causes a large part of the overall dynamic power consumption and directly determines both the sizes of the driving and sampling switches and the size of the input capacitive load of the ADC and the kT/C noise power, a small CAP-DAC helps the power efficiency. To validate the proposed calibration idea, a 10-bit asynchronous SAR ADC was fabricated in 28-nm CMOS. Measurement results

show that the proposed stochastic calibration improves the ADC's SFDR and SNDR by 14.9 dB, 11.5 dB, respectively. After calibration, the fabricated SAR ADC achieves an ENOB of 9.14 bit at a sampling rate of 85 MS/s, resulting in a Walden FoM of 10.9 fJ/c-s.

The second calibration technique is a timing-skew calibration for a time-interleaved (TI) SAR ADC that calibrates/computes the inter-channel timing and offset mismatch simultaneously. Simulation results show the effectiveness of this calibration method.

When used together, the proposed mismatch calibration technique and the timing-skew calibration technique enables a TI SAR ADC to be designed that can achieve a sampling rate of $\sim$GS/s with 10-bit resolution and a power consumption as low as $\sim$10mW; specifications that satisfy the requirements of 5G technology.

To my parents

Acknowledgements

There is an endless list of people who contributed to this thesis without the help of whom this thesis would have not been possible.

First, I'd like to thank my current supervisor, Dr. David Hasko, for his kind support and guidance, and also my former supervisor, Professor Arokia Nathan, who was an inspiring supervisor in academic matters and also an encouraging and very supportive friend.

My special thanks goes to Professor Bogdan Staszewski at University College Dubin who provided me with simulation tools as well as tape-out opportunity and also his team members specially Dr. Filippo Schembari and Naser Pourmousavian who helped me a lot with the design and lay-out of the chip. This thesis would have definitely not been possible without their help.

I would also like to gratefully thank Dr. Hashem Zare-Hoseini for his technical advice. He was very kind to be always available to answer my technical questions. He has had a great contribution to the design of the chip.

I'd like to acknowledge HiSilicon (Huawei) who provided funding for this project as well as technical support. I'd also like to thank TSMC for chip fabrication.

Finally, I would like to thank my parents and my brother for their continued love and support during this 4-year PhD.

Table of contents

List of figures

List of tables

Chapter 1

Introduction

High-speed (∼GSample/s) medium-resolution (∼10 bits) analog-to-digital converters (ADCs) are widely used in high-speed communication systems, such as serial links, Ultra-Wide-Band (UWB), and OFDM-based receivers. In designs for digital TV and satellite receivers, the trend is towards software-defined radio, where the embedded A/D converter is moved closer to the antenna. Such ADCs require 8–10 bits of resolution, a large bandwidth to enable subsampling/down-conversion and a power consumption limited to a few hundred milliwatts, in order to be combined with the digital baseband processing in a single IC.

5G (5th generation mobile networks or 5th generation wireless systems) are the next major phase of mobile telecommunications standards beyond the current 4G/IMT-Advanced standards, and another domain where high-performance ADCs are required. 5G networks must be built to meet a number of individual end-user and enterprise needs [1]:

- 1 to 10 Gb/s data rates to support ultra-high definition video and virtual reality application;
- Less than one millisecond latency to support real time mobile control and vehicle-to-vehicle applications and communications;
- Rapid switching time between different radio access technologies to ensure a consistently seamless delivery of services;
- Being able to support tens of millions of applications and hundreds of billions of machines;
- Increased battery life of terminal devices (handsets);
- Increased capacity in dense urban environments;

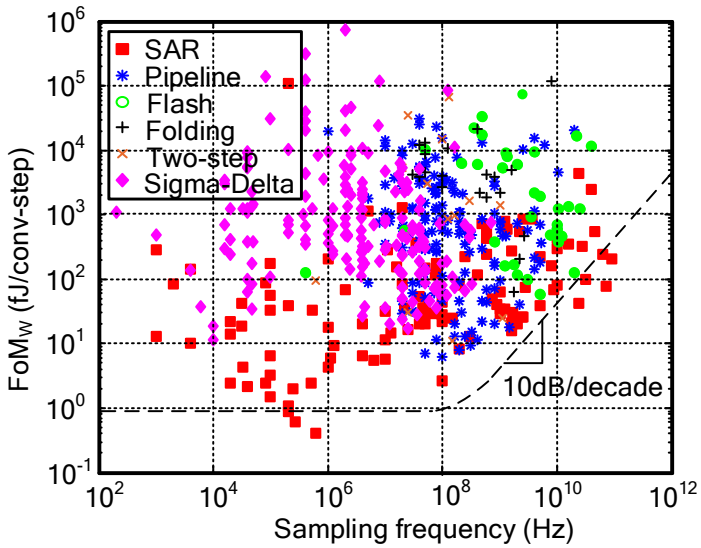


Fig. 1.1 Plot of Walden FoM vs Nyquist sampling frequency for all ADC papers presented at ISSCC and VLSI from 1997-2018.

- Coverage in airplanes and remote areas.

To support the required thousand-fold capacity increase, it will be necessary to simultaneously make efficient use of all of the available continuous and non-continuous electromagnetic spectrum. Furthermore, freeing up additional spectrum will be required. How this new spectrum will be used to achieve 10Gb/s for end users is a major challenge in designing future 5G systems. Specifically for A/D conversion, these 5G requirements call for an ADC with a conversion rate of at least 1GS/s and a resolution of at least 10 bits, together with a low power consumption.

The Successive-Approximation Register (SAR) ADC is a very promising candidate for applications where high-speed medium-resolution is desired. This ADC architecture relies on the high switching speed of certain device technologies, and so is known for its superior energy efficiency, small chip area, and good digital compatibility that can be easily scaled to a new CMOS technology. The attractiveness of the SAR ADC to achieve state-of-the-art performance can be seen in papers presented at important conferences. For instance, Fig. 1.1 illustrates the plot of Walden figure-of-merit (FOM$_W$) [2] versus the maximum sampling frequency for all ADCs presented at the International Solid State Circuits Conference (ISSCC) and at the Symposia on VLSI Technology and Circuits (VLSI) from 1997 until 2018 [3]. The filled red square indicates SAR ADCs. As is clear from the graph, the majority of the ADCs that occupy the outer envelope use the SAR architecture.

When properly implemented, a SAR ADC also benefits from a rail-to-rail input swing and 100% capacitance utilization during input sampling (thus low kT/C noise). In principle, SAR

ADCs can be designed with no static power consumption. While the low-power characteristic of SAR ADCs is very attractive, the large number of decision cycles for a single conversion is a fundamental drawback for high-speed operations. Recent advances in process technology and smart circuit techniques have accelerated conversion speeds, but a single-channel SAR ADC is still unable to satisfy the requirements of 5G applications with low power dissipation. Interleaving multiple SAR ADCs in the time domain is a popular technique to increase the sampling rate. The overall performance of time-interleaved ADCs is very much bound to its constituent sub-ADCs. As such, design of a power-optimized sub-ADC is crucial in designing an efficient time-interleaved (TI) A/D converter.

Among all building blocks of a single-channel SAR ADC, the capacitive digital-to-analog converter (CAP-DAC) deserves the most attention. There are two reasons: first, the linearity of the SAR ADC is mainly determined by the linearity of the CAP-DAC; second, the capacitive DAC not only causes a great part of the overall ADC's dynamic power consumption, but also directly determines the size of the driving and sampling switches, the input capacitive load of the ADC and the kT/C noise power.

Although a small DAC unit capacitance is beneficial, it comes at the cost of a larger random mismatch between capacitors. If an accurate mismatch calibration circuit can guarantee to take care of the mismatch, the size of the capacitive DAC can be kept as small as dictated by the kT/C noise limit[1]. This underlines the importance of an effective mismatch calibration circuit for the SAR ADC.

In this thesis, a stochastic-based mismatch calibration technique is presented, which can compute the mismatch of the capacitors of the DAC with high precision. The calibration process occurs in the foreground. The detection of the mismatch error happens in the analog domain, and the correction occurs in the digital domain. The precision of the proposed calibration technique is not limited by the analog circuit and can be easily determined by a parameter, when designing the digital calibration circuit. A prototype SAR ADC using the proposed calibration technique was implemented in 28nm CMOS technology and measured in the lab. The improvement in the linearity of the ADC after the calibration was applied shows the effectiveness of this approach.

Another factor that needs to be carefully addressed in designing a TI ADC is the mismatch between different channels, e.g. gain mismatch, offset mismatch, bandwidth mismatch and sampling-time mismatch. Amongst these, timing mismatch, also known as timing-skew, is the most challenging to detect and correct. Moreover, the requirement on the accuracy of the timing-skew calibration circuit is tough to meet: for a Giga-sample-per-second TI ADC,

[1] Another limit on the capacitive DAC is the dynamic range reduction due to the top-plate parasitic capacitors.

even a few hundreds of fs timing mismatch can easily degrade the output signal-to-noise-and-distortion-ratio (SNDR) of the ADC by a few dBs.

In this thesis, we propose a timing-skew calibration technique that is not only able to calibrate the timing-skew errors of sub-channels of a time-interleaved ADC, but also to simultaneously correct for the offset mismatches. This calibration works in the foreground by detecting the mismatch errors in the digital domain and correcting them in the analog domain. The effectiveness of the proposed calibration method is validated through simulation.

The remainder of this thesis is organized as follows. Chapter 2 reviews the fundamentals of analog-to-digital conversion and the most common architectures of implementing an A/D converter are briefly covered. Chapter 3 discusses the detailed design and implementation considerations for the building blocks of a conventional SAR ADC, along with some of the state-of-the-art design techniques that are used to boost performance. Chapter 4 reviews some of the most important mismatch calibration techniques for SAR ADCs proposed in the literature. Chapter 5 presents details of the proposed mismatch calibration technique and chapter 6 explains the design details of the prototype ADC in which this calibration method is employed along with the measurement results. Chapter 7 also presents the proposed timing-skew calibration technique for TI ADCs with the simulated results. And finally, chapter 8 draws the conclusions and discusses future work.

Chapter 2

Fundamentals of A/D Conversion

In this chapter, we cover the fundamentals of analog-to-digital conversion. We begin with an ideal converter before discussing the practical limitations and terminology associated with non-idealities. Next, we briefly cover the most commonly employed A/D architectures, comparing their relative advantages and disadvantages.

2.1 Ideal A/D Converter

An *analog-to-digital converter* (ADC) is a block that inputs an analog current or voltage signal and outputs (converts) the magnitude of this signal into an N-bit digital format (code), where N denotes the *resolution* of the ADC. The block digram of an ideal A/D converter is shown in Fig. 2.1.(a) and the input/output transfer curve is shown in Fig. 2.1.(b). Here, V_{in}, V_{REF} and B_{out} are the analog input signal, the reference voltage and the digital output, respectively. The time that is needed by the ADC to complete one conversion is called the *conversion time*. *Sampling rate* is defined as the frequency at which the ADC performers the conversion, which is the inverse of the conversion time. The quantization step is usually referred to as V_{LSB} or simply LSB, which is the change in the input voltage that corresponds to a single bit change at the output code. The fundamental equation that describes the ideal A/D converter of Fig. 2.1 is

$$\frac{V_{\text{REF}}}{2^N}\left(b_0 + b_1 2^1 + b_2 2^2 + ... + b_N 2^{N-1}\right) = V_{\text{in}} + V_X, \tag{2.1}$$

where $b_0 \sim b_{N-1}$ are the binary bits of the output digital code and V_X is the residue error due to the finite resolution of the A/D conversion, also known as the *quantization error*. The

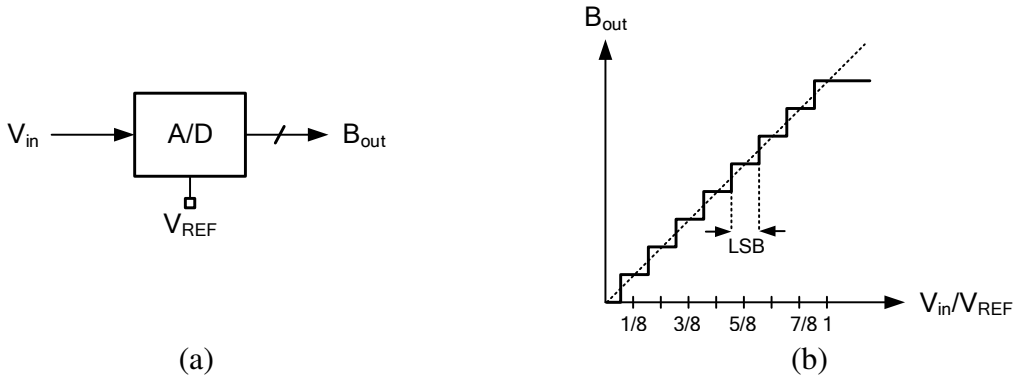


Fig. 2.1 Ideal A/D converter: (a) Block digram, (b) input/output transfer curve for a 3-bit ADC.

quantization error of an ideal A/D is within one LSB voltage:

$$-\frac{V_{\mathrm{LSB}}}{2} < V_X < +\frac{V_{\mathrm{LSB}}}{2} \tag{2.2}$$

Denoting the quantization error by V_Q, the ADC can be modelled as in Fig. 2.2. Assuming that V_Q has a uniform distribution function between $-V_{\mathrm{LSB}}/2$ and $+V_{\mathrm{LSB}}/2$, it can be proven [4] that its RMS voltage is equal to

$$V_{\mathrm{Q,RMS}} = \frac{V_{\mathrm{LSB}}}{\sqrt{12}}. \tag{2.3}$$

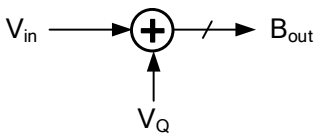


Fig. 2.2 The ADC model with the quantization error.

For a random input voltage, the quantization error can be viewed as being "random", and is often referred to as "noise". In this case, we can define the signal-to-quantization-noise ratio (SQNR) of an ideal ADC as

$$\mathrm{SQNR} = 20\log\left(\frac{V_{\mathrm{in,RMS}}}{V_{\mathrm{Q,RMS}}}\right). \tag{2.4}$$

For a sinusoidal input between 0 and V_{REF}, $V_{\text{in,RMS}} = V_{\text{REF}}/2\sqrt{2}$. $V_{\text{Q,RMS}}$ is also given by (2.3). Therefore,

$$\text{SQNR} = 20\log\frac{V_{\text{REF}}/2\sqrt{2}}{V_{\text{LSB}}/\sqrt{12}} \tag{2.5}$$

$$= 20\log\left(\sqrt{\frac{3}{2}}2^N\right) \tag{2.6}$$

$$= 6.02N + 1.76\text{dB} \tag{2.7}$$

(2.7) is a very useful equation that relates the SQNR of an ADC and the resolution N.

2.2 Non-Ideal A/D Converter

In practice, there are non-idealities that cause the characteristics of an ADC to deviate from the ideal form. In the following, we review some of the most important limitations of an A/D converter.

2.2.1 Offset Error

The *offset error* for an A/D converter is defined as the deviation of the first transition (bit change) that we denote by $V_{00\ldots01}$ from the ideal position (which is $\frac{1}{2}$LSB), that is

$$V_{\text{offset}} = \frac{V_{00\ldots01}}{V_{\text{LSB}}} - \frac{1}{2}\text{LSB}. \tag{2.8}$$

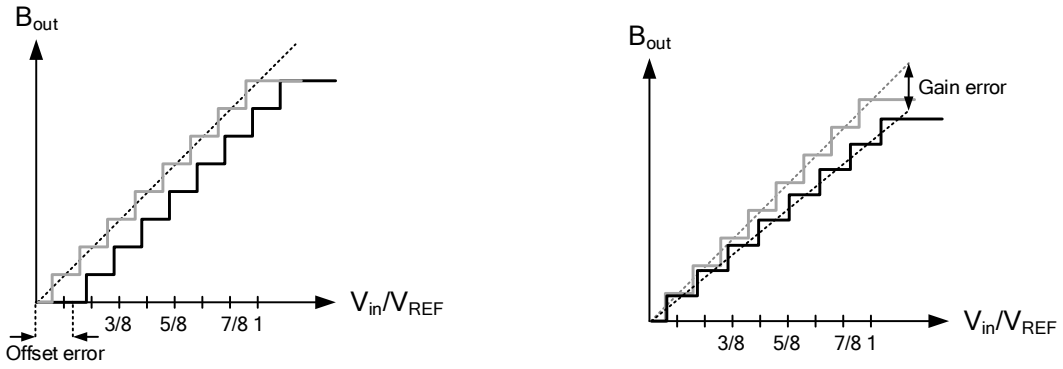
This is illustrated in Fig. 2.3.(a).

2.2.2 Gain Error

The *gain error* is defined as the difference at the full-scale value between the ideal and actual response curves, when the offset error has been removed. For an A/D converter, this is given by

$$V_{\text{gain}} = \left(\frac{V_{11\ldots11}}{V_{\text{LSB}}} - \frac{V_{00\ldots01}}{V_{\text{LSB}}}\right) - (2^N - 2), \tag{2.9}$$

which is shown in Fig. 2.3.(b). Offset and gain errors can be easily compensated by digital pre/post-processing. What is more crucial for the DC performance of an A/D converter is the linearity measures, and most importantly the INl and DNL.



(a) (b)

Fig. 2.3 Illustration of (a) offset error, (b) gain error. Offset error is the deviation of transfer curve from the first transition and gain error is the difference between the ideal (best-fit) and actual response curves.

2.2.3 Differential Nonlinearity Error (DNL)

For an ideal converter, each analog step size is equal to 1 LSB. *Differential nonlinearity* (DNL) is defined as the deviation of every code width from 1 LSB (typically, after gain and offset errors have been removed), or mathematically,

$$\mathrm{DNL}[k] = W[k] - \mathrm{LSB}, \tag{2.10}$$

where $W[k]$ is the width of the k-th code. This makes DNL a vector of size 2^N that characterises the corresponding error associated with all possible output codes. If only one DNL was reported, that would be the maximum DNL value. It can be readily seen that the sum of all the DNLs is equal to zero:

$$\sum_{k=0}^{2^N-1} \mathrm{DNL}[k] = 0. \tag{2.11}$$

An example of an ADC characteristic with non-zero DNL is shown in Fig. 2.4.(a). It is easy to understand that for an ADC, DNL must be larger than -1. In the case where $\mathrm{DNL} = -1$, the code associated with that DNL will be *missing* from the A/D transfer curve and hence will not appear at the output of the ADC.

2.2.4 Integral Nonlinearity Error (INL)

Integral nonlinearity (INL) is defined as the deviation of the code transition from its ideal location, when both the offset and gain errors have been removed, that is

$$\mathrm{INL}[k] = \frac{T[k] - T[\mathrm{k,ideal}]}{\mathrm{LSB}}, \tag{2.12}$$

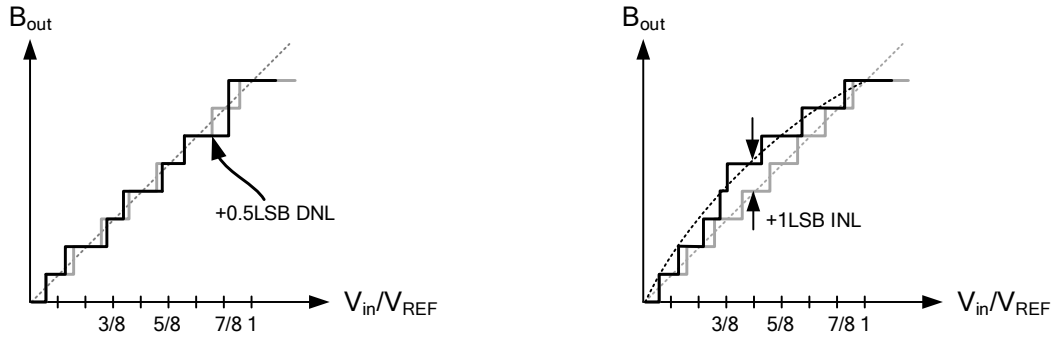
where $T[k]$ and $T[\mathrm{k,ideal}]$ denote the actual and ideal transition points, respectively. A straight line through the endpoints of the A/D transfer curve is usually considered as the reference for determining INL. An alternative reference would be the best-fit line that would represent the characteristic curve. It can be readily shown that

$$\mathrm{INL}[k] = \sum_{i=0}^{k-1} \mathrm{DNL}[i] \tag{2.13}$$

or equivalently

$$\mathrm{DNL}[k] = \mathrm{INL}[k] - \mathrm{INL}[k-1]. \tag{2.14}$$

As for DNL, if only one INL was reported, that would be the maximum individual INL value. An example of an ADC characteristic with non-zero INL is shown in Fig. 2.4.(b).



(a) (b)

Fig. 2.4 Illustration of (a) DNL error and (b) INL error. DNL is the deviation of every code width from 1 LSB and INL is the deviation of the code transition from its ideal location.

An A/D converter is called *monotonic* if the output always increases as the input increases. A converter is guaranteed to be monotonic if $\mathrm{INL}_{\max} < 0.5\mathrm{LSB}$ (or equivalently when $\mathrm{DNL}_{\max} < 1\mathrm{LSB}$). The DNL is *not* defined for the non-monotonic steps.

There are different techniques for measuring the DNL/INL of an ADC. One way, known as *code boundary servo*, is to use an adjustable voltage source to find the exact code trip points. Another very common technique is to apply a signal with known amplitude distribution to the input of the ADC and analyse the digital distribution at the output and is known as the *histogram testing*. Typically a sine wave is used as the input signal for the histogram-based method [5]. Although histogram testing is the most commonly used technique, there are some limitations [6] that should be taken into consideration when following this approach.

2.2.5 Dynamic Range

The *dynamic range* of a converter is defined as the ratio between the maximum and minimum signal magnitudes that can be meaningfully processed. The minimum magnitude is usually determined by the noise level; however, it is still necessary to specify a merit to define the meaningful output. A popular merit is to use the signal-to-noise-and-distortion ratio (SNDR) and to define the maximum magnitude at which the SNDR drops by 3dB for a specific frequency. Similarly, *the effective resolution bandwidth* of an ADC is specified as the bandwidth over which the peak SNDR is within 3dB of its best value for a specific input magnitude . In this context, the *effective number of bits* (ENOB) is computed as

$$\text{ENOB} = \frac{\text{SNDR} - 1.76\text{dB}}{6.02} \text{ bits}, \tag{2.15}$$

which is derived from (2.7) by substituting SNDR for SQNR; ENOB is merely a performance metric relating an actual ADC's performance to that of an ideal ADC.

2.3 ADC Architectures

There exists a number of different architectures for realising an A/D converter. Every architecture comes with its own advantages and disadvantages in terms of the maximum achievable speed, resolution and linearity, power consumption, area, complexity and compatibility with technology scaling.

2.3.1 Serial ADC

This type of ADC consists of a counter that serially counts the number of clock cycles until the input voltage becomes equal to a linearly increasing reference (usually a ramp). There are two ways of realizing a serial ADC, namely *single slope* and *double slope*.

Single-Slope

The single-slope topology is shown in Fig. 2.5. Once the ramp generator starts producing the ramp signal V_{ramp}, the counter starts counting, and the counter stops once $V_{\text{in}} = V_{\text{ramp}}$. Assuming a linear ramp signal, the output of the counter (which is the output of the ADC) is proportional to the input voltage. As can be seen in Fig. 2.5, this type of ADC has a very low complexity and is easy to implement. The linearity of the ADC is mainly determined by the linearity of the ramp signal, and not by component matching. The serial ADC is also inherently monotonic. The downside is the very low conversion speed for they require a digital counter at a highly oversampled clock rate. The generation of a linear ramp can also become challenging for when a high resolution is required. These issues can be alleviated by the dual-slope ADC.

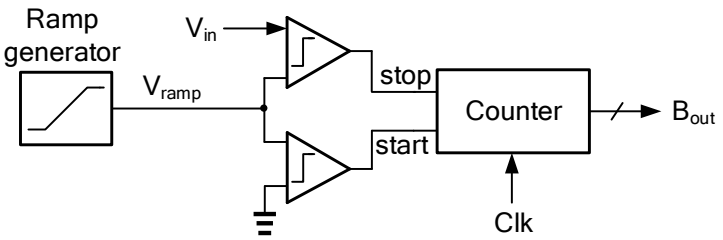


Fig. 2.5 Block diagram of a single-slope serial ADC. The counter stops counting once V_{ramp} becomes equal to V_{in}. This way, the binary output of the counter would be proportional to the input voltage.

Dual-Slope

The dual-slope block diagram is shown in Fig. 2.6.(a). The conversion occurs in two phases. During phase (I), V_{in} is integrated over a fixed period of time, which is $2^N T_{\text{clk}}$ where N is the resolution of the ADC and T_{clk} is the clock period. This would generate a ramp at the output of the integrator, V_X, whose slope, and hence final value, is proportional to V_{in}. In phase (II), the input of the integrator is connected to a constant reference voltage V_{REF} and V_X is de-integrated until it reaches zero during which the counter is counting. Due to the constant slope during the second phase, the output of the counter would be proportional to $V_{\text{in}}/V_{\text{REF}}$. The operation of the dual-slope ADC is illustrated in Fig. 2.6.(b).

It can be shown that the accuracy of the dual-slope A/D converter does not depend on the time constant of the integrator. However, the offset of the op-amp used in the integrator results in the ADC offset, a problem that can be solved either by calibration or by utilizing a *quad-slope* ADC. As for the signal-slope ADC, the conversion speed is still very low. The

application of this type of ADC is mostly for laboratory digital voltmeters (DVM) where high speed is not required.

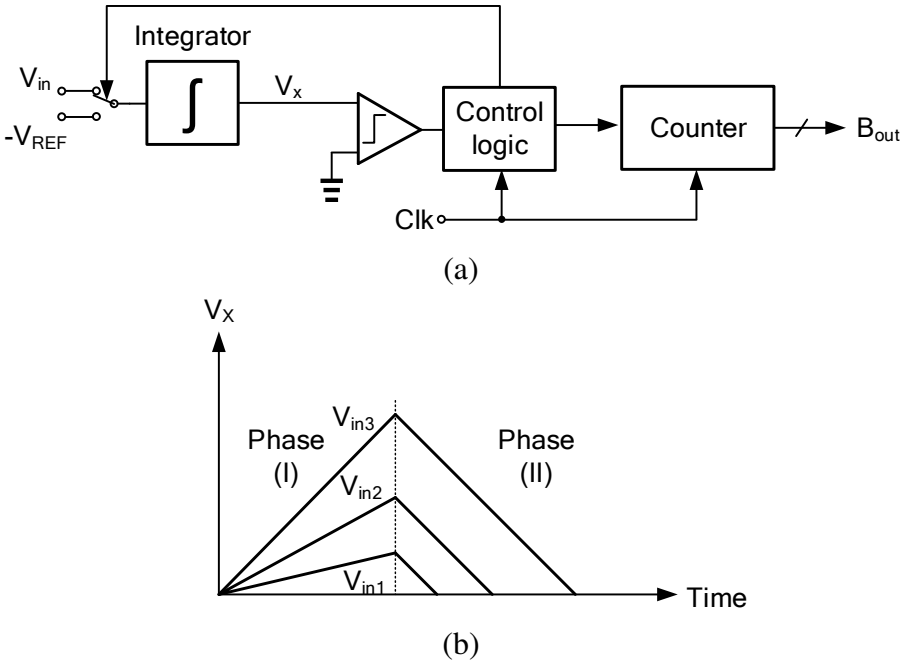


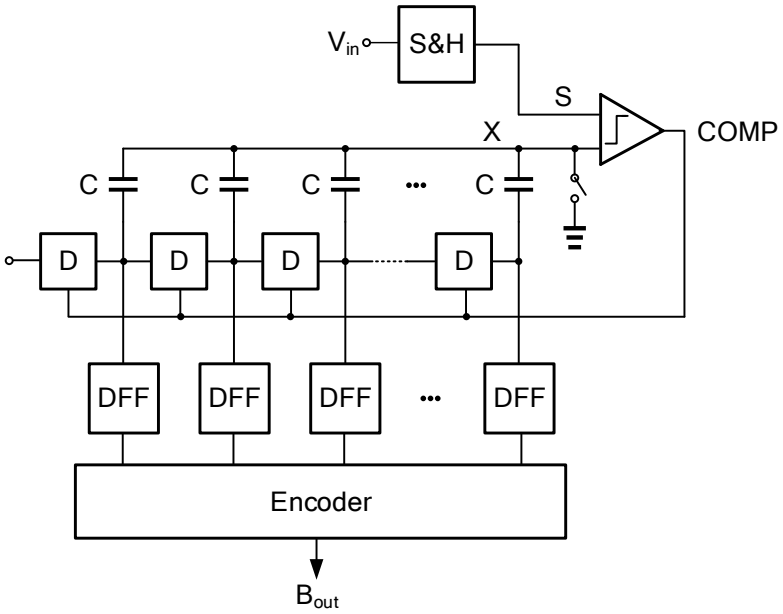
Fig. 2.6 Dual-slope ADC (a) block diagram, (b) timing diagram for three different input voltages. The accuracy of this A/D converter does not depend on the time constant of the integrator.

Asynchronous Digital-Slope

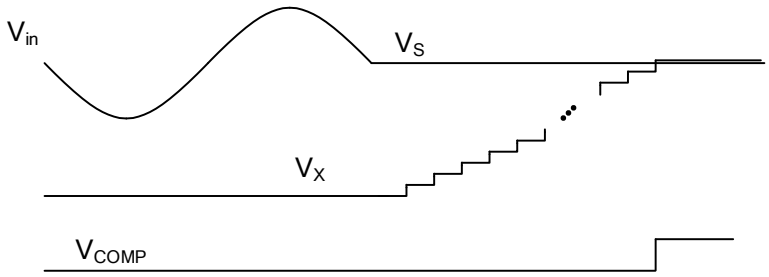
The speed limitation of the synchronous slope-based converter was solved in [7], where an asynchronous digital-slope architecture was introduced. The block diagram of this type of ADC is shown in Fig. 2.7.(a), and its time-domain behaviour is illustrated in Fig. 2.7.(b). As can be seen, the integrator in the dual-slope ADC is replaced by an asynchronous switched capacitor (SC) circuit. The quantization occurs in the time-domain using delay cells, memory cells and an encoder instead of a digital counter. The operation of the ADC is as follows. First, the input voltage is sampled on node *S*. Next, a pulse is applied to the input of the delay cell array and, in a domino manner, it propagates through delay cells one by one. In this way, the delay line acts like an asynchronous thermometer-encoded[1] counter which replaces the original synchronous binary-encoded counter. The array of capacitors, connected to the outputs of the delay line, implements a thermometer-encoded charge-redistribution DAC,

[1]Thermometer encoding is a method of encoding a natural number, n, with n ones followed by a zero.

through which V_X is switched up 1 LSB step by step at a constant interval until it reaches V_S, at which point the comparator toggles and deactivates the delay line. The output thermometer code is then converted to a binary code via the encoder. Since the hardware complexity of the digital-slope ADC grows exponentially with the ADC resolution, this type of ADC does not lend itself to resolutions higher than 8 bits.



(a)



(b)

Fig. 2.7 Asynchronous digital-slope ADC: (a) block diagram, (b) timing diagram. Contrary to a synchronous serial ADC, here no high speed clock is required. The counter is also replaced by delay elements.

2.3.2 Flash ADC

Flash ADC is probably the most straightforward way of realizing an A/D converter. In a flash converter, the input voltage is simply compared with $2^N - 1$ voltages that are equally spaced

between zero and the reference voltage. The reference levels are generated by a reference generator and the comparisons are made by 2^N comparators, as depicted in Fig. 2.8 for the specific case of $N = 3$. Here, generating the reference voltages is realized by a resistor string, but it can be also implemented by capacitors [8]. The output of the comparators form a thermometer code, which is then converted to a binary code via a 2^N-to-1 decoder.

Flash ADCs are inherently very fast [9–11] for they require only one clock cycle for every conversion. However, the complexity of the circuit increases exponentially with the number of bits, that is $2^N - 1$ comparators are needed for an N-bit conversion. This exponential growth also holds for the input capacitive load of the converter, which must be driven by the input driver. In the following, we discuss the most important sources of error in a flash ADC:

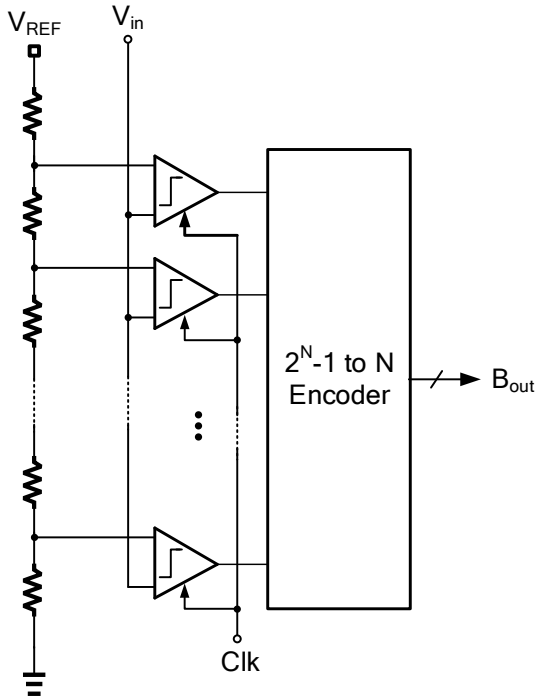


Fig. 2.8 Block diagram of a Flash ADC. Owning to their simple structure, flash ADCs are inherently fast.

Comparator Offset

The comparator offset is probably the most important source of error that can degrade the performance of a flash ADC. There exist several different design and calibration techniques to alleviate the problem of comparator offset. One of the simplest (yet powerful) offset calibration techniques is *resistor averaging* [12–14], which acts as a kind of spatial filtering [15], by simply connecting the output of the pre-amplifiers using resistors. The effectiveness

of this technique depends on the ratio R_2/R_1, which imposes a trade-off with the gain of the amplifiers. Another commonly used technique for offset cancellation is *offset storage*. In this approach, the offset voltage is measured and stored on a capacitor, and can be done in two different ways: *input offset storage* (IOS), shown in Fig. 2.9.(a) and *output offset storage* (OOS), shown in 2.9.(b). It can be shown that this method reduces the offset by a factor given by the pre-amplifier gain, A.

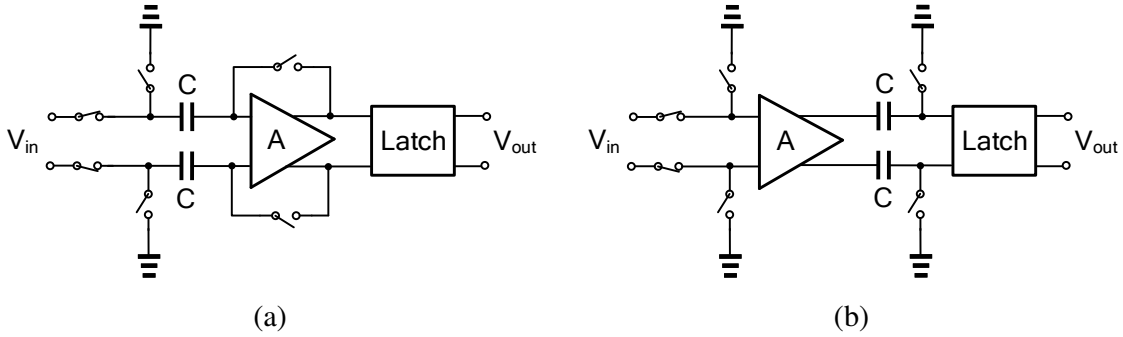


Fig. 2.9 Offset cancellation through offset storage: (a) input offset storage, (b) output offset storage. Offset is reduced by a factor A.

Bubble Error

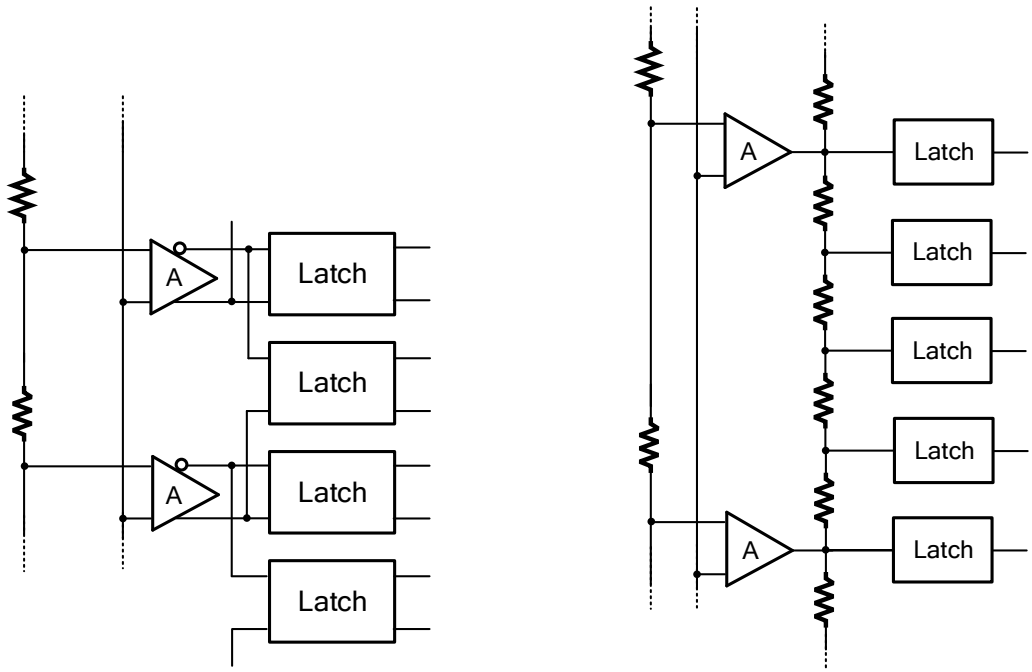
For the encoder to function properly, the outputs of the comparators should form a thermometer code with only one single transition. However, sometimes a lone 1 (the *bubble*) will occur within the string of 0s for various reasons. These bubbles usually occur near the transition point of the thermometer code. A very simple solution is to put a three-input NAND gate at the output of the comparators [16].

Comparator Metastability

Metastability is a problem that occurs in all latching comparators, when the input is near the comparator decision point [17]. This problem occurs when the comparator takes more time to switch to a valid output state than is available in the sampling interval. Different gates interpret the metastable output differently, resulting in an invalid encoder output. Using more stages of latch in the comparator and *Grey coding* are two of a range of methods that can be used to decrease the adverse effect of metastability in flash ADCs.

2.3.3 Interpolating ADC

Interpolation can be used to reduce the number of latches by interpolating between the outputs of the pre-amplifiers [18]. An interpolation of factor M decreases the number of pre-amplifiers by a factor of M. Interpolation by factor 2 in a flash ADC is fairly straightforward; shown in Fig. 2.10.(a), this only requires rearrangement of the connections between the latches and the differential pre-amplifiers. To accomplish higher interpolation factors, extra circuitry is required. This can be achieved using resistors [19], capacitors [20] and current mirrors/current division [21, 22]. Fig. 2.10.(b) illustrates schematically the resistor-based method for a factor 4 interpolation. It is worth mentioning that capacitive interpolation has a side benefit of automatically performing comparator offset cancellation through the offset storage discussed in section 2.3.2.



(a) (b)

Fig. 2.10 Realization of (a) factor 2, (b) factor 4 interpolation. Interpolation of factor M reduces the number of pre-amplifiers by the same factor.

For an interpolation topology to work without error it is necessary that the transfer curves of consecutive pre-amplifiers overlap; however, this limits the pre-amplifier gain. Moreover, the impedance of the interpolation network at the output of the pre-amplifiers typically

reduces the bandwidth and affects the maximum achievable speed. One of the important side-effect of interpolation is the reduction of DNL by the same factor as for the interpolation.

Interpolation reduces the number of pre-amplifiers, thereby reduces the input capacitive load of the ADC. Nevertheless, the number of latches stays the same. To decrease the number of latches, a technique called *folding* is usually adopted in combination with interpolating.

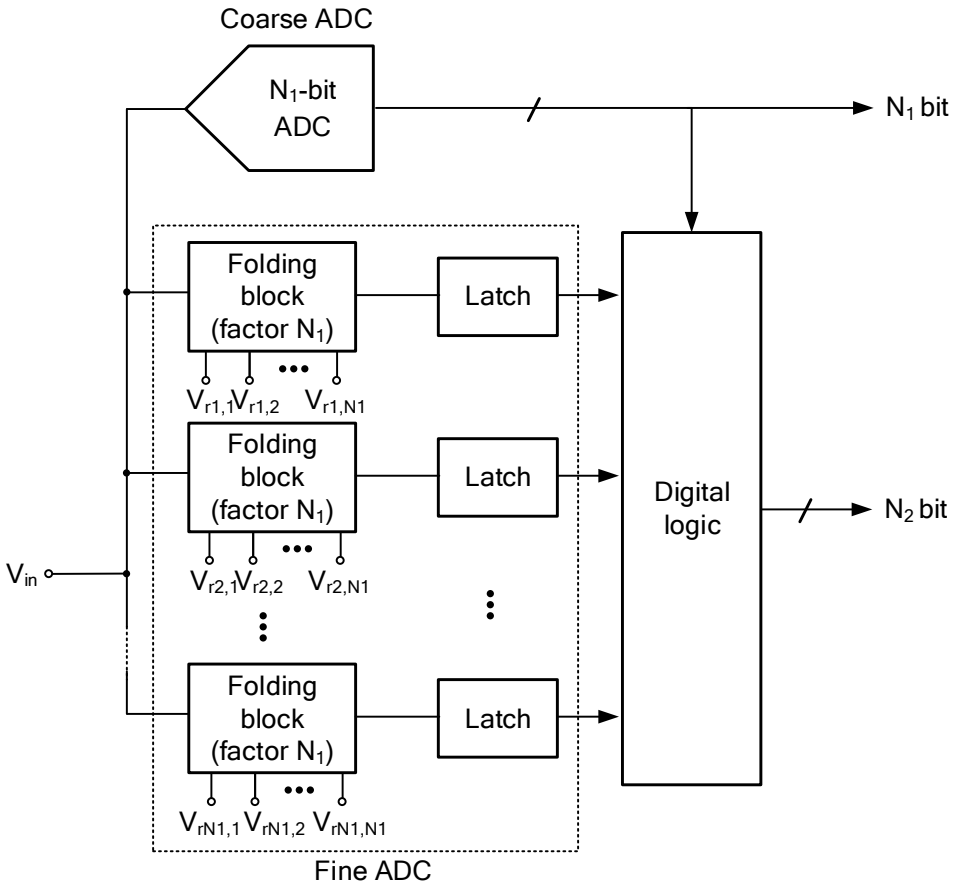
2.3.4 Folding ADC

In a *folding* architecture, two ADCs operate in parallel, namely a coarse ADC that determines the N_1 MSB bits and a fine ADC that resolves the N_2 LSB bits. Defining the *folding factor* (or *folding rate*) to be the number of output transitions for a single folding block as V_{in} is swept over its input range, this is equal to 2^{N_1}. The block diagram of a folding ADC is shown in Fig. 2.11.(a), and the input-output folding transfer curve for a folding factor of 4 is shown in Fig. 2.11.(b). As can be seen, the folding maps the input voltage into a smaller range. The coarse ADC determines which fold the input voltage is in and the fine ADC determines the position of the input within that fold.

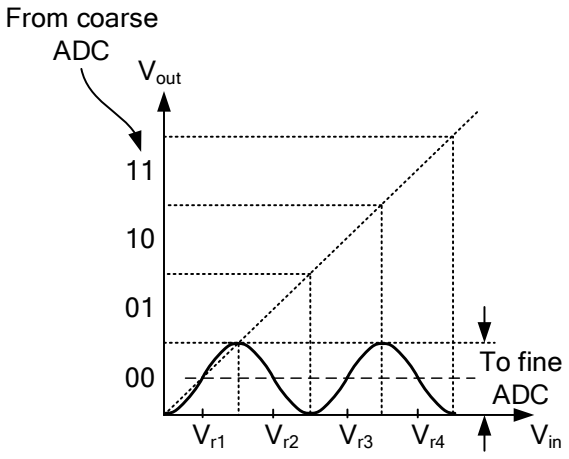
The folding blocks can be realized using cross-coupled differential pairs, as seen in Fig. 2.12 for the case of folding factor of 4. It should be noted that, the rounded edge of the folding characteristic shown in Fig. 2.12.(b) does not cause a non-linearity as long as the *zero-crossings* are accurately positioned. This is only true though if there is no g_m and/or tail current mismatch between the differential pairs forming the folding block [23].

It is worth mentioning that the output signal from a folding block is at a higher frequency than the input signal, with the same factor as the folding factor. This puts a limit on the practical folding rate used in high-speed A/D converters. As mentioned before, the folding topology reduces the number of latches but the input capacitance remains almost unchanged [2]. Therefore, the folding and interpolating architecture are usually used together [24–26]. Folding-interpolating ADCs have the advantageous of both folding and interpolating ADCs together. Fig. 2.13 shows a 4-bit ADC with a folding rate of four and interpolating rate of two.

[2] Assuming that the differential pair used in the folding block has the same size of input transistors that would be otherwise used in the pre-amplifier.



(a)



(b)

Fig. 2.11 (a) Block diagram of a folding ADC with a folding factor of 2^{N_1}, (b) input-output folding transfer curve for a folding factor of 4.

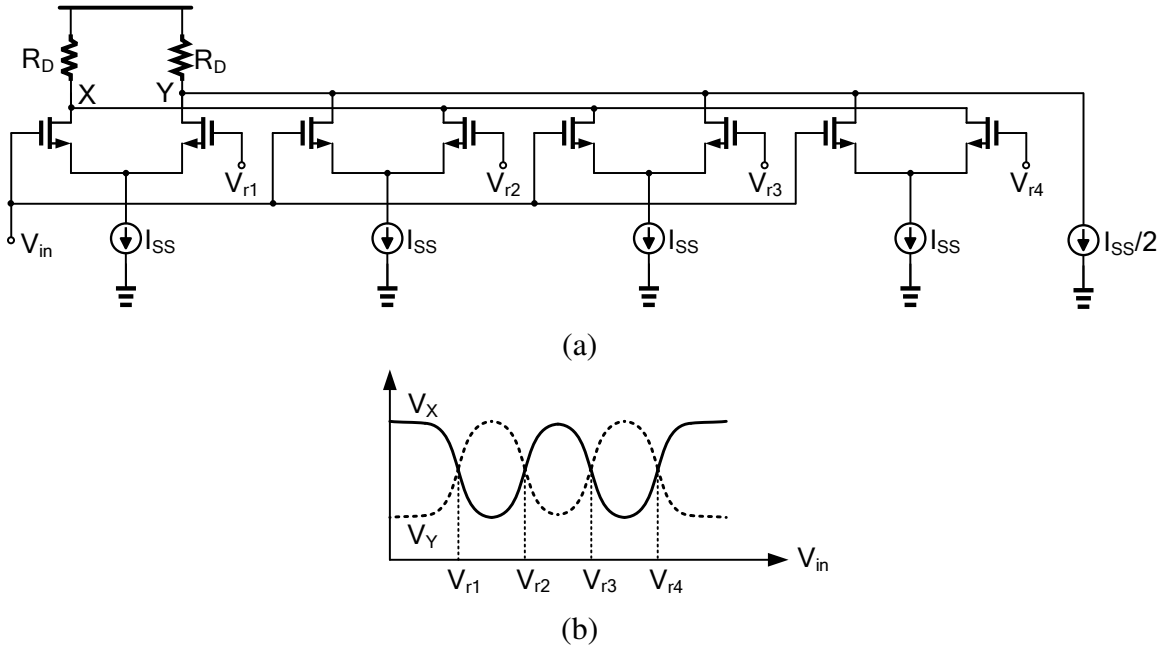


Fig. 2.12 Factor 4 folding: (a) Implementation, (b) voltage of the folding nodes.

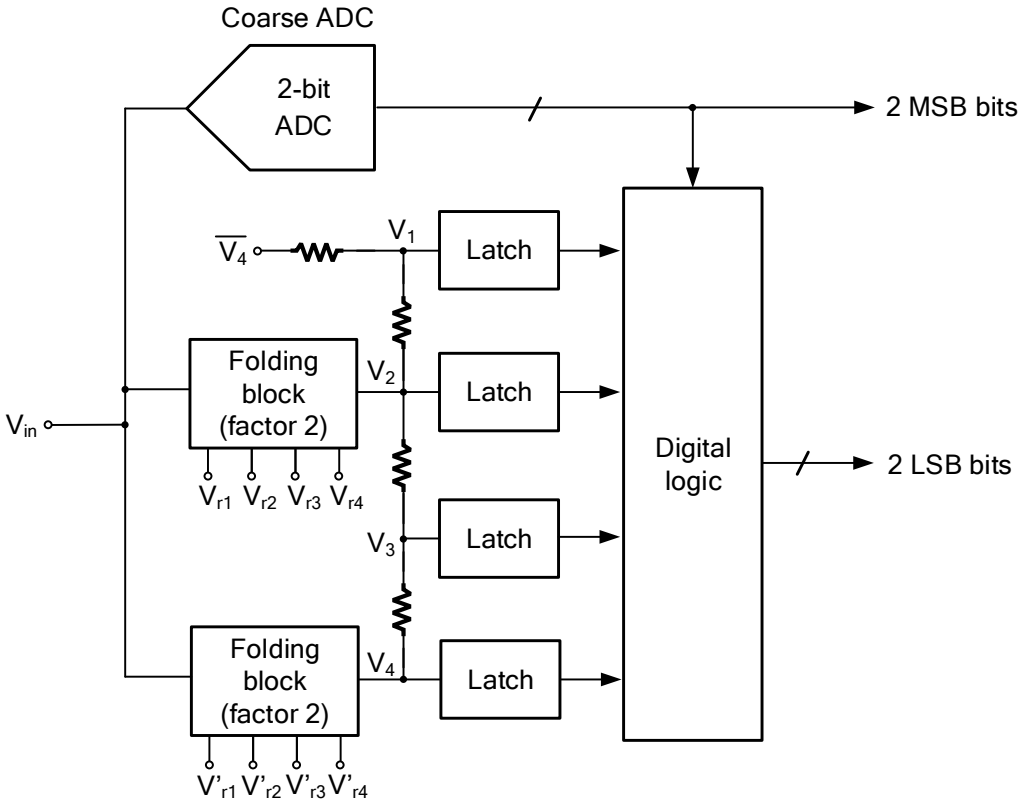


Fig. 2.13 A factor 4 folding-interpolating ADC.

2.3.5 Pipeline ADC

A *pipeline* ADC is a cascade of several low-resolution ADCs to obtain an overall high resolution quantization, as shown in Fig. 2.14.(a). The specific case when there are only two stages is usually known as a *two-step ADC*, whose architecture is shown in Fig. 2.14.(b). The purpose of the interstage gains is twofold: firstly to avoid the LSB of the proceeding stages to become very small and secondly to avoid using multi-references for different stages. In a pipeline ADC, different stages process different samples concurrently. Thus, the throughput rate depends only on the speed of each stage. This concurrency of operation enables pipeline ADCs to reach high speeds, and unlike flash ADC, the number of components (stages) grows only *linearly* with the resolution. Although N input samples are being processed in parallel, N clock cycles are required for each input sample to proceed through the entire pipeline. Hence, there is a *latency* of N clock cycles for every input sample that may be an issue in e.g. control systems. The term pipeline ADC is usually referred to the case of one-bit-per-stage pipeline ADC, whose architecture is shown in Fig. 2.15.

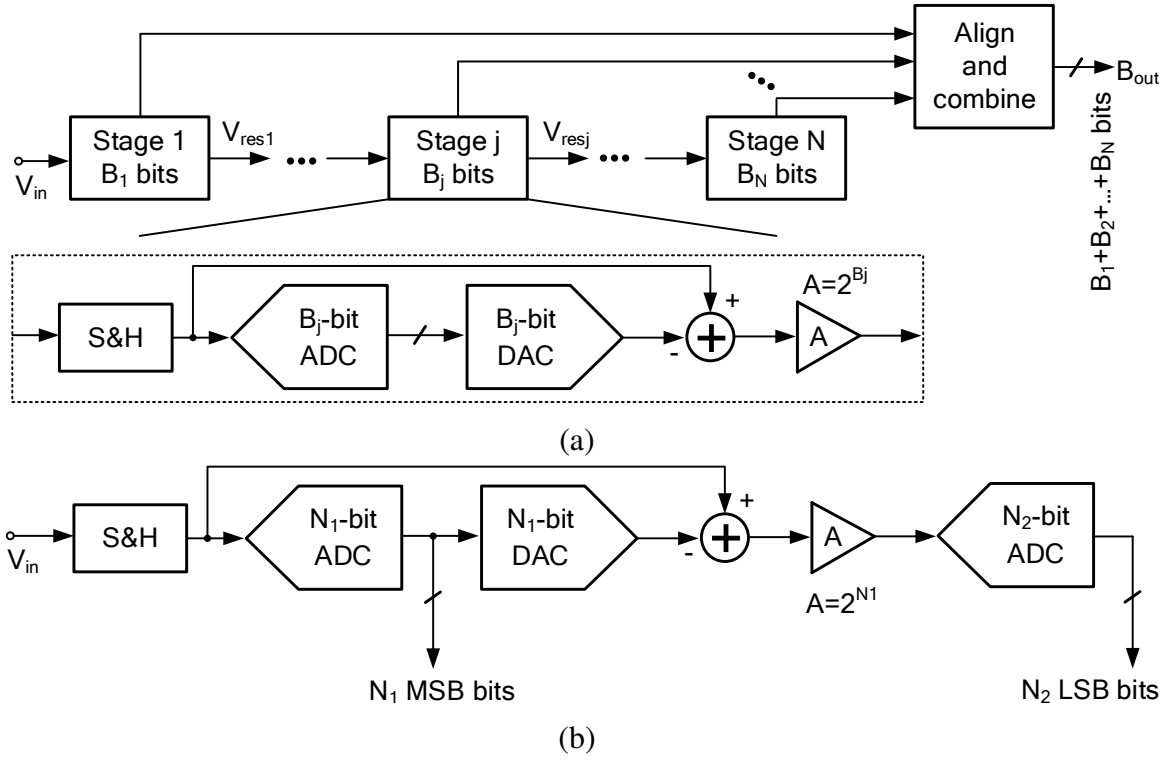


Fig. 2.14 Architecture of a pipeline ADC: (a) generic topology, (b) the specific case of a pipeline ADC with only two stages, also known as two-step ADC. In a pipeline ADC, different stages work in parallel, enabling this ADC to reach high speeds.

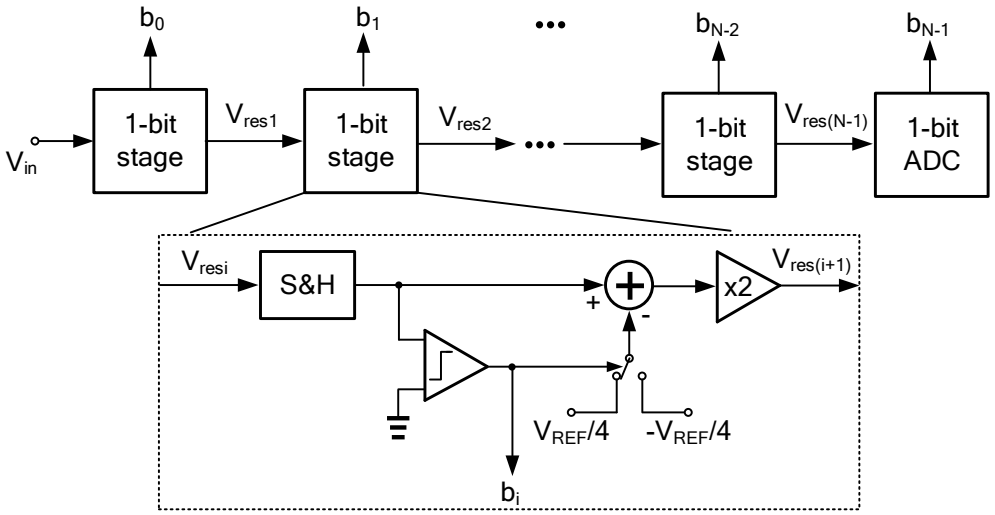


Fig. 2.15 A one-bit-per-stage pipeline ADC.

Any source of error in the constituent blocks of the pipeline ADC may impair the overall performance. These errors include the sub-ADC errors (and most importantly the comparator offset), the gain stage gain and offset errors and the sub-DAC error. The sub-ADC error becomes problematic only if it causes the residue from one stage to grow outside of the allowable input range for the following stage, as shown in Fig. 2.16. As such, the sub-ADC error can be tolerated as long as the residue stays within the next stage's input range or another stage returns the residue to the allowable range before it reaches the last quantizer. This is the core strategy for dealing with the sub-ADC errors in a pipeline ADC for almost all the different techniques that have been discussed in the literature. Example techniques include: using a slightly reduced interstage gain compared to the nominal value [27] or using a higher resolution for sub-ADCs [28, 29].

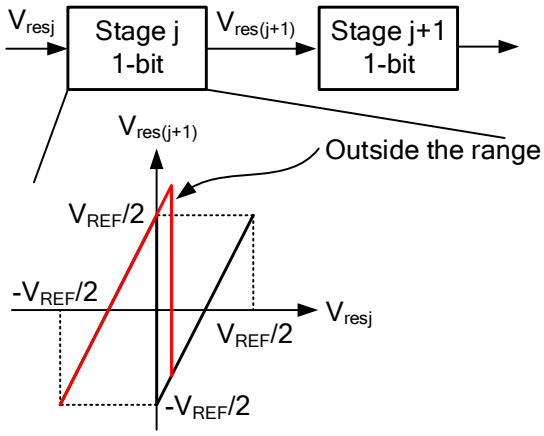


Fig. 2.16 Sub-ADC error for a 1-bit per stage pipeline ADC. Here, the residue voltage has fallen outside the allowable range $\pm V_{\mathrm{REF}}/2$. This causes performance degradation.

A very popular technique to deal with the sub-ADC error is to introduce redundancy by using the so-called *1.5-bit-per-stage* topology [30]. This is shown in Fig. 2.17.(a), where a 1.5-bit stage effectively performs a 3-level quantization of its input, which results in an input-output transfer curve shown in Fig. 2.17.(b). The 1.5-bit stage can tolerate comparator offset as large as $\pm V_{\mathrm{REF}}/4$. The interstage gain error and the sub-DAC error are usually calibrated in the digital domain [31–34].

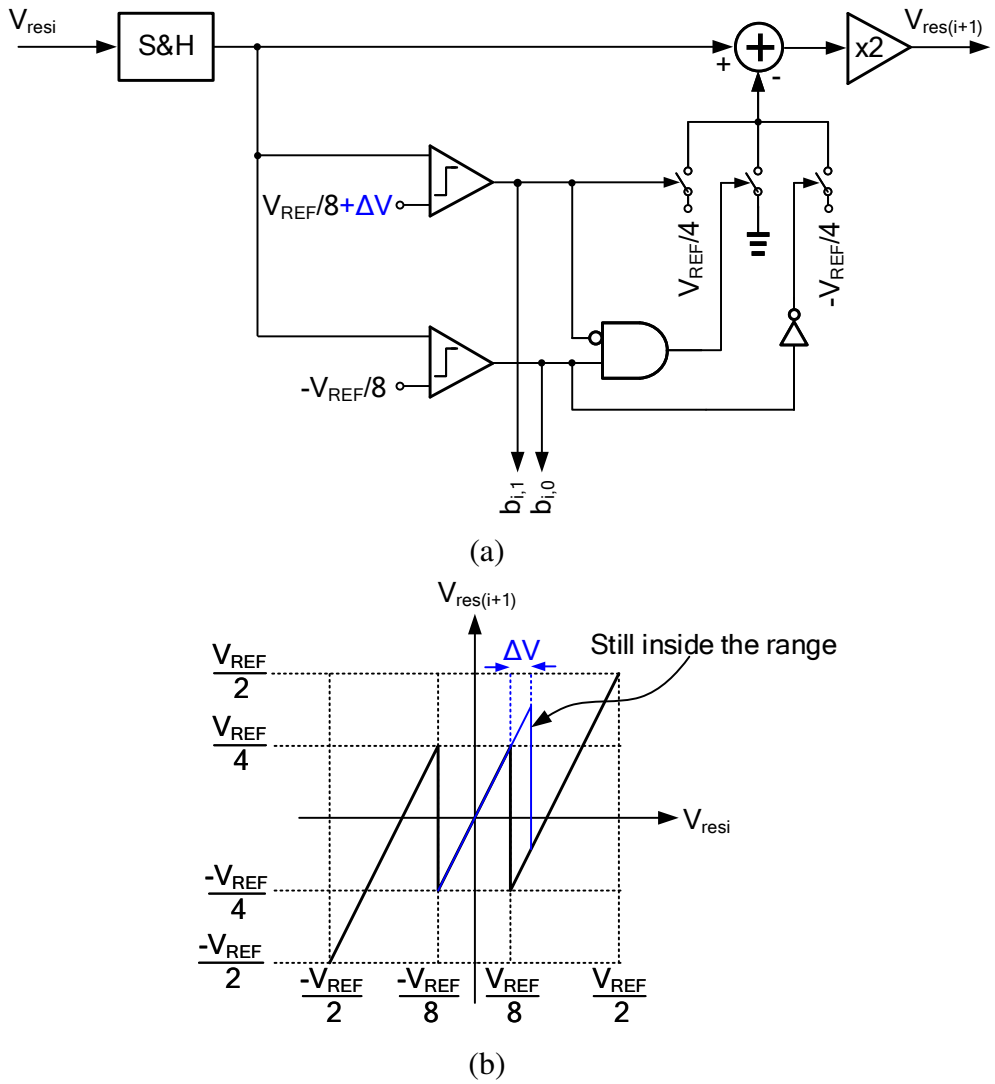


Fig. 2.17 The topology of a 1.5-bit per stage converter: (a) circuit implementation, (b) the input-output transfer curve. A 1.5b/stage converter can tolerate comparator offset as large as $\pm V_{\mathrm{REF}}/4$ without any degradation of the performance.

2.3.6 Oversampling ADC

Nyquist rate A/D converters have a sampling rate of around twice the maximum input signal frequency, whereas *oversampling* ADCs sample at a much higher frequency than that. As we will see later, oversampling brings about a few very important advantages, e.g. this approach relaxes the requirements on matching tolerances, amplifier gains, and the analog anti-aliasing filters and most importantly reduces baseband quantization noise. Basically, and oversampling ADC trades speed with resolution.

Oversampling Without Noise Shaping

As mentioned in section (2.1), the quantization error can be assumed to have a white noise characteristics distributed uniformly across the Nyquist bandwidth, i.e. $\pm f_s/2$, with a power equal to $\Delta^2/12$, where Δ is the quantization step. This results in a spectral density of magnitude

$$S_e(f) = \frac{\Delta}{\sqrt{12}}\sqrt{\frac{1}{f_s}}, \tag{2.16}$$

as depicted in Fig. 2.18.

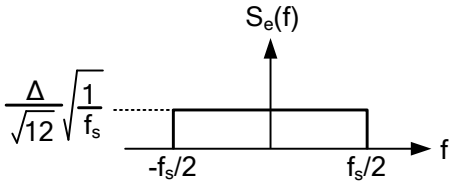


Fig. 2.18 Spectral density of the quantization noise. Δ is the quantization step.

Now, assuming that the sampling frequency f_s is higher than the input signal bandwidth $2f_0$ by a factor of oversampling ratio, OSR, defined as

$$\text{OSR} = \frac{f_s}{2f_0}, \tag{2.17}$$

filtering the quantized signal to get rid of the noise outside of the band $\pm f_0$ would eliminate a portion of the quantization noise. This is illustrated in Fig. 2.19. It can be easily proven that the improved SQNR is given by

$$\text{SQNR} = 6.02N + 1.76 + 10\log(\text{OSR}). \tag{2.18}$$

(2.18) shows 3dB (or 0.5b) improvement of SQNR for every decade increase in the oversampling ratio. The reason for this is that when quantized samples are averaged together, the

signal portion adds linearly, whereas the noise portion adds as the square root of the sum of the squares. Note that this enhancement also takes place with respect to other types of noise, such as thermal noise. Hence, oversampling generally improves the overall signal to noise ratio (SNR) by $10\log(\mathrm{OSR})$.

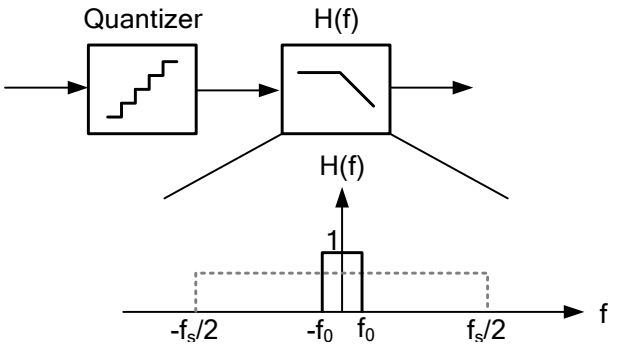


Fig. 2.19 Filtering out the quantization noise using a low-pass filter. This enhances the SQNR by a factor of $f_s/2f_0$.

It should be also noted that while oversampling improves the signal-to-noise ratio, it does *not* improve the linearity. This means for instance, if a 10-bit quantizer is used in an oversampling ADC to achieve 16-bit resolution, the quantizer has to have an accuracy (linearity) of 16-bit. This is why oversampling ADCs typically use a 1-bit quantizer, which is inherently linear.

Oversampling With Noise Shaping

Fig. 2.20 shows the block diagram of an oversampling ADC that includes a delta-sigma ($\Delta\Sigma$) modulator, where noise shaping takes place. The $\Delta\Sigma$ modulator converts the analog signal into a noise-shaped low-resolution digital signal, which is then proceeded with the decimator that converts the oversampled low-resolution digital signal into a high-resolution digital signal at a lower sampling rate. This is the trade-off between accuracy and speed that was mentioned previously.

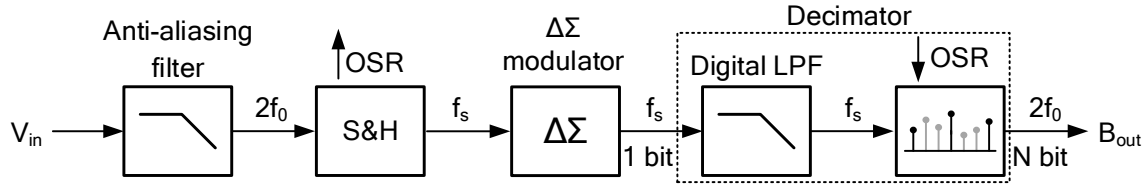


Fig. 2.20 Block diagram of an oversampling ADC with noise shaping. Noise shaping takes place in the $\Delta\Sigma$ modulator.

The block digram and the model of a noise-shaping ΔΣ modulator is shown in Fig. 2.21. The signal transfer function $S_{TF}(z)$ and the noise transfer function $N_{TF}(z)$ can be found to be

$$S_{TF}(z) = \frac{Y(z)}{U(z)} = \frac{H(z)}{1+H(z)} \tag{2.19}$$

$$N_{TF}(z) = \frac{Y(z)}{E(z)} = \frac{1}{1+H(z)}. \tag{2.20}$$

The idea is to choose $H(z)$ in a way that its magnitude is large over the input frequency range, i.e. 0 to f_0, so that the noise transfer function $N_{TF}(z)$ becomes very small in that region, while the signal transfer function $S_{TF}(z)$ remains almost unaffected. In this way, such a loop filter would push the unwanted noise to higher frequencies, where a post filter can then remove this out-of-band noise.

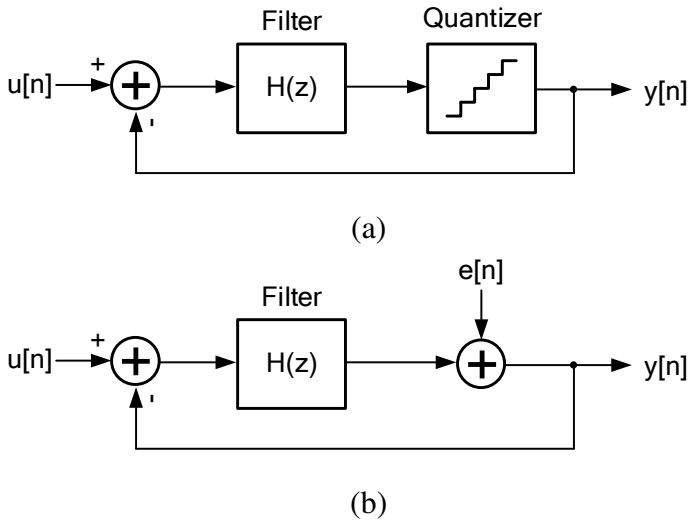


Fig. 2.21 Principle of noise shaping: (a) Block diagram implementation, (b) linear model by replacing the quantizer with an injected noise. Noise shaping is a technique to push the unwanted noise out of the band of interest in order to achieve higher resolutions.

The simplest loop filter $H(z)$ is an integrator that has a transfer function of

$$H(z) = \frac{1}{z-1}. \tag{2.21}$$

This would result in *first-order* noise shaping that can be simply made by a 1-bit quantizer and a 1-bit DAC to form a ΔΣ modulator, as shown in Fig. 2.22. It can be proven that the

SNQR of an oversampling ADC with first-order noise shaping is given by

$$\text{SQNR} = 6.02N + 1.76 + 30\log(\text{OSR}), \tag{2.22}$$

bringing an improvement of 9dB/octave or 1.5-bit/octave. Similarly, higher orders of loop filter would result in an even bigger improvement of SQNR [35]. In general, an L-th order $\Delta\Sigma$ modulator improves the SQNR by $(2L+1)$-bit per octave of OSR. However, high order modulators can give rise to *stability* problems [36]. There are also other practical considerations that one should take into account when implementing $\Delta\Sigma$ modulator. These include the effect of finite op-amp gain and its linearity, kT/C noise, the op-amp noise, the effect of comparator non-idealities, etc. [37] reviews these issues in detail.

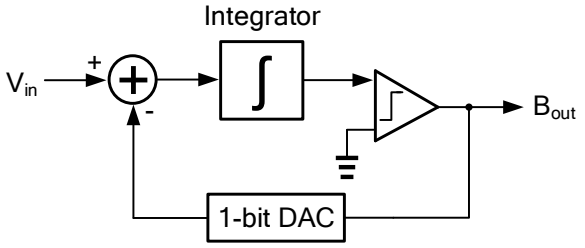


Fig. 2.22 Block digram of a first order modulator. This performs a first-order nosie shaping and improves the SQNR by 9dB/octave.

Another approach to increase the maximum dynamic range of $\Delta\Sigma$ modulators is to use a cascade of lower order modulators to build a high order modulator; such an arrangement has been called MASH (Multi-stAge noise SHaping) [38, 39]. For instance, two 1st order $\Delta\Sigma$ modulators can be cascaded to form a 2nd order modulator. This is shown in Fig. 2.23. Here, the 1st stage quantization error is quantized by the 2nd quantizer and the quantized error is then subtracted from the results in the digital domain. The advantage of a MASH topology is the reduced instability compared to a modulator of the same order, but having only one feedback loop. However, MASH modulators are more sensitive to finite op-amp gain, bandwidth and mismatches, which lead to gain errors.

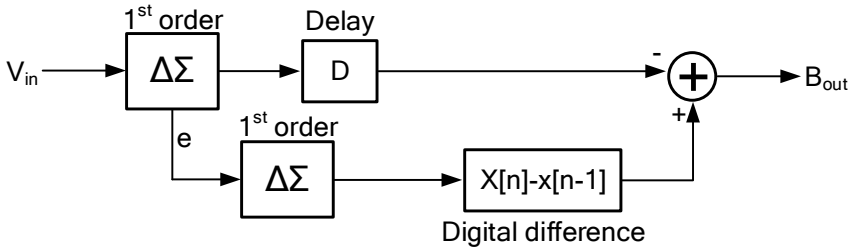


Fig. 2.23 MASH architecture: cascade of 1st order $\Delta\Sigma$ modulators to build a 2nd order one.

2.3.7 SAR ADC

Successive approximation register (SAR) A/D converters are one of the most popular types especially for more advanced technologies. This is mainly due to its low circuit complexity when compared to other topologies. The block diagram of a SAR ADC is shown in Fig. 2.24. As can be seen, it comprises of an input S/H circuit, a comparator, the SAR logic and a D/A converter. The basic operation of a SAR conversion is shown in the signal flow diagram of Fig. 2.25: the reference voltage, which is compared with the input voltage, is halved successively in every cycle and depending on the result of the comparison it is added to or subtracted from the initial D/A voltage. This process occurs for N cycles.

The D/A converter in Fig. 2.24 can be realized in different ways the most versatile of which is to use a capacitive one. A SAR ADC implemented by a capacitive DAC is known as a *charge redistribution* SAR ADC, since the D/A conversion occurs in the charge domain and the comparison (reference) voltages are generated through charge redistribution. One of the advantages of using a capacitive DAC is that the DAC does the sample and hold task, obviating the need for a stand-alone and power-hungry S/H circuit.

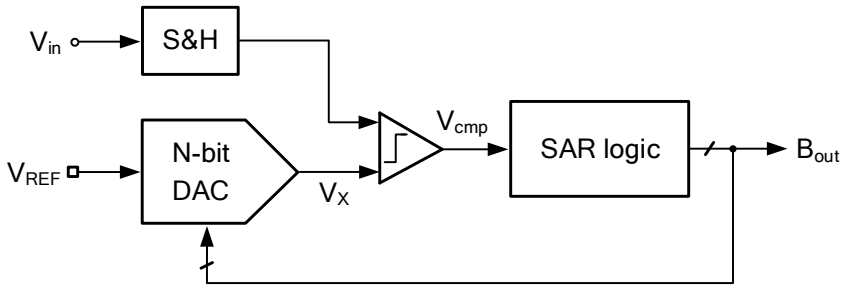


Fig. 2.24 Block diagram of a SAR ADC. SAR ADC is mainly made up of digital blocks. This is an advantage when migrating to a new technology.

The operation of a 5-bit charge redistribution SAR ADC is shown in Fig. 2.26. During the *sampling phase*, V_{in} is sampled on the top plate of the capacitive DAC, X, while all the bottom-plate switches are connected to V_{REF} (Fig. 2.26.(a)). Next, the sampling switch opens, and the *bit cycling* starts by first determining the MSB bit (Fig. 2.26.(b)). This is done by the MSB switch to connect to ground, making V_X equal to $V_{\text{in}} - V_{\text{REF}}/2$. At this point, a comparison operation occurs: if the output of the comparator is a ZERO, it means $V_{\text{in}} < V_{\text{REF}}/2$. In this case, the MSB switch goes back to its initial position, leaving the top plate voltage as before. If the output of the comparator is a ONE, it means $V_{\text{in}} > V_{\text{REF}}/2$ and the MSB switch remains in this state for the rest of the conversion. The MSB bit is the output of the comparator. The same procedure continues until all the bits are determined.

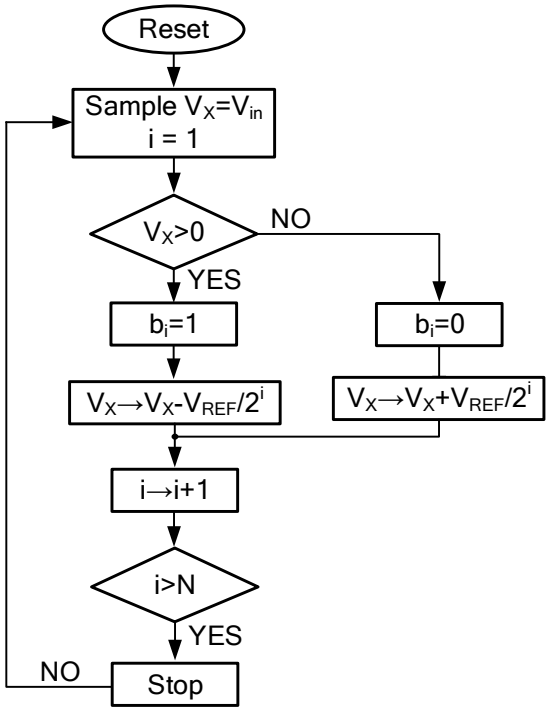


Fig. 2.25 Signal flow diagram of the successive-approximation approach.

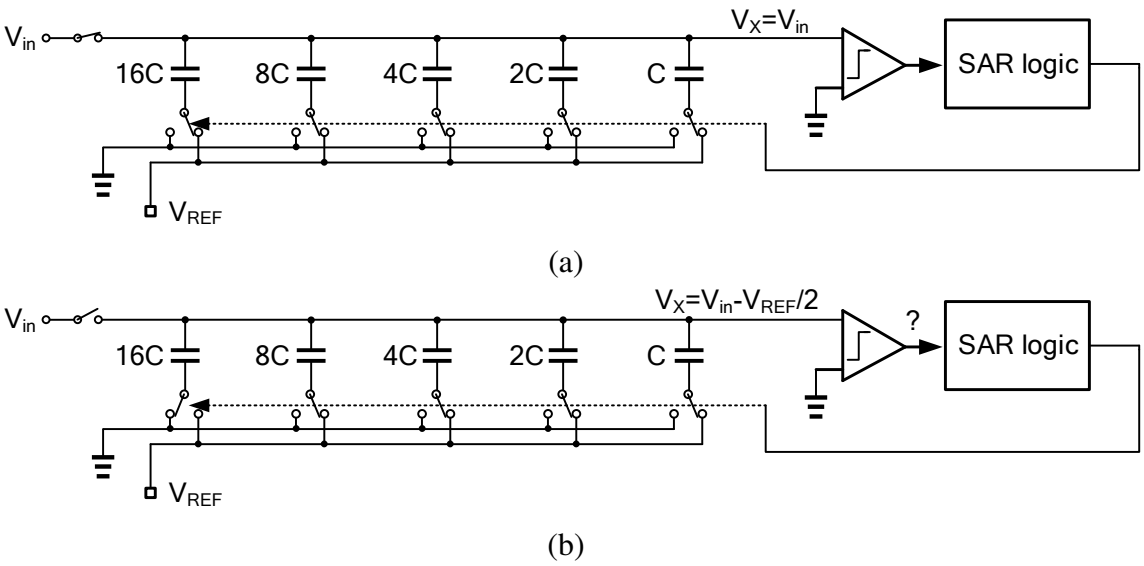


Fig. 2.26 Operation of a 5-bit SAR ADC: (a) sampling phase, when V_{in} is sampled on the top plate of the capacitor array, (b) bit cycling phase to determine the MSB bit. The position of the corresponding bottom plate switch is determined based on the output of the comparator. The bit cycling process continues in the same manner until all the bits are resolved.

It is worth noting that the parasitic capacitor connected to node V_X does not cause any error since the conversion starts by the top plate being connect to V_{in} and ends by it converging to V_{in} again, making the total charge transfer effectively zero. A detailed analysis of the building blocks of a SAR ADC is presented in chapter 3.

In order to increase the performance of a SAR ADC, both in terms of the speed and the accuracy, several architectural methods have been widely discussed in the literature, as follows:

Multi-Bit per Cycle SAR ADC

In a conventional 1-bit per cycle SAR ADC, after every comparison of the comparator, the input range of the ADC is halved, one comparison takes place and one bit is resolved. This process continues until all the N bits of the conversion are determined. In an M-bit per cycle SAR ADC after every cycle, the input range is divided into M equal sectors and $(M-1)$ comparisons occur using $(M-1)$ comparators [40]. Therefore, only N/M cycles are needed for the full conversion, increasing the ADC speed by a factor of M. An example of a 6-bit 2-bit/cycle SAR ADC is shown in Fig. 2.27. In the fist cycle, the SAR logic sets the switching of the DACs in a way to generate the first set of reference voltages, i.e. $\left\{\frac{1}{4}V_{\text{REF}}, \frac{2}{4}V_{\text{REF}}, \frac{3}{4}V_{\text{REF}}\right\}$. The comparators make the first comparison which results in an output of $\{0,1\}$ as the first two bits of the A/D conversion. Next, a new set of reference voltages is generated by the DACs controlled by the SAR logic: $\left\{\frac{5}{16}V_{\text{REF}}, \frac{6}{16}V_{\text{REF}}, \frac{7}{16}V_{\text{REF}}\right\}$. These new V_{REF}s obviously depend on the result of the previous comparison. This comparison yields the next two bits, that is $\{0,0\}$. The conversion proceeds in the same way with the new reference voltages being $\left\{\frac{17}{64}V_{\text{REF}}, \frac{18}{64}V_{\text{REF}}, \frac{19}{64}V_{\text{REF}}\right\}$ and the resultant output bits being $\{1,1\}$. The entire A/D conversion hence gives rise to 6-bit output of $\{0,1,0,0,1,0\}$.

The multi-bit per cycle technique can be viewed as a merger between flash and SAR ADC where the different reference voltages are generated in the same way as in a flash ADC and the conversion happens in a SAR manner. One obvious disadvantage associated with this technique is the larger number of comparators and DACs. Moreover, the input load of the ADC is also increased by a factor of M. The sampling-time skew between the individual DACs and the offset mismatch between the comparators too introduce distortion and degrade the output SNDR of the ADC. To tackle these issues, several methods have been proposed in the literature. In [41–43] only two DACs were used to perform a 2b/cycle SAR conversion, out of which only one DAC was used for decision reference generation (REF DAC), while the other one did the input sampling and residue generation (SIG DAC). [44] intentionally used skewed comparators for generating the reference voltages, hence removing the need for multiple DACs. However, this approach is very process dependant and probably requires

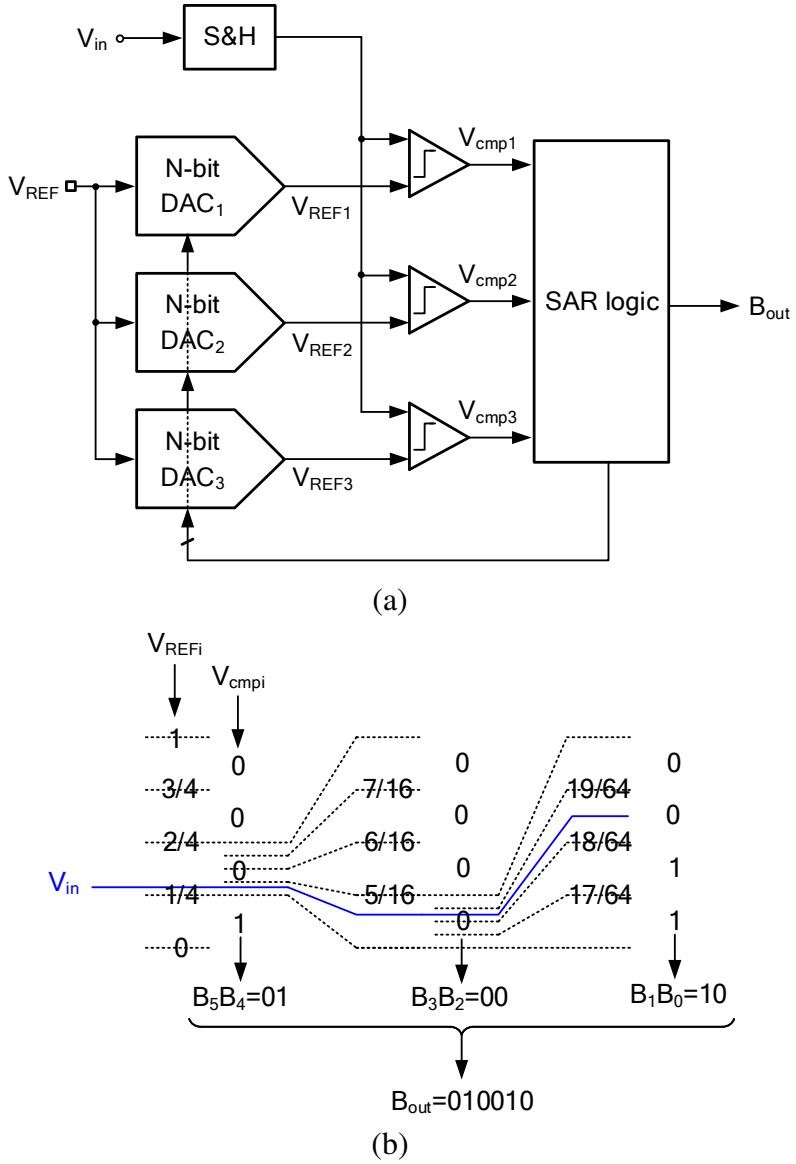


Fig. 2.27 (a) Block diagram of an N-bit 2-bit/cycle SAR ADC, (b) conversion cycle for the case of $N = 6$. Theoretically, a 2-bit/cycle SAR ADC can operate twice as fast as its conventional 1-bit/cycle counterpart.

some sort of calibration to be effective. [45] made use of both the DAC's common-mode and differential-mode voltages to convert 2 bit in every cycle. 3b/cycle and 4b/cycle SAR ADCs have also been reported [46, 47]. Interpolation can also be employed [48–50] to further reduce the area and power overhead of multi-bit/cycle SAR ADCs, as shown in Fig. 2.28. This is a combination of flash, SAR and interpolating ADCs.

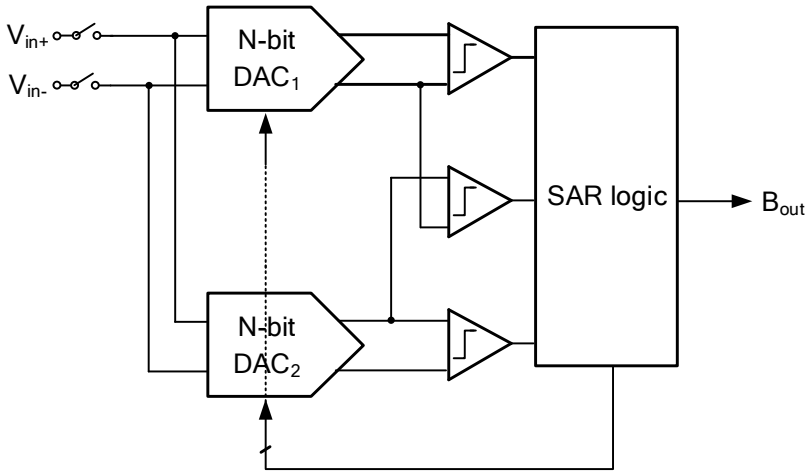


Fig. 2.28 A 2b/cycle SAR ADC with interpolation. Here, interpolation helps to reduce the number of the DACs by one.

Pipelined SAR ADC

The pipeline ADC architecture can achieve both high resolution and high speed [51–53]. Nevertheless, as mentioned in section (2.3.5), pipeline ADCs rely on good component matching and require high-performance op-amps (high-gain and high-bandwidth) which can be hard to achieve in modern CMOS technology, due mainly to the use of low supply voltages. SAR converters have surpassed pipeline converters in terms of energy efficiency and minimal analog complexity, the latter making it well suited to nanometer scale CMOS technology. The disadvantage of SAR ADCs nonetheless is their limited speed due to their serial decision making process. Furthermore, the effective resolution of SAR ADCs is limited by the comparator noise and limited capacitor matching. In order to have good capacitor matching, a large unit capacitor is required, which leads to a large input capacitive load for the ADC, increases the DAC dynamic power consumption and makes the design of a linear input tracking switch challenging. The high energy efficiency of the SAR ADC can be combined with the high conversion rate of the pipelined ADC to bring about the so called *pipelined SAR* ADC [54, 55], whose core architecture is shown in Fig. 2.29. The first N_1 bits are resolved by the coarse SAR ADC, after which the residue voltage, that is freely available on the capacitive DAC[3], is amplified by the residue amplifier to arrive at the second stage of the ADC, the fine SAR ADC, where the rest of the bits are resolved. Usually, the coarse SAR quantizes a larger number of bits with a low interstage gain, resulting in a comparatively small residue for the amplification, thus, relaxing opamp's linearity requirement and power consumption.

[3]Therefore, no separate DAC is required when compared to Fig. 2.14.(b).

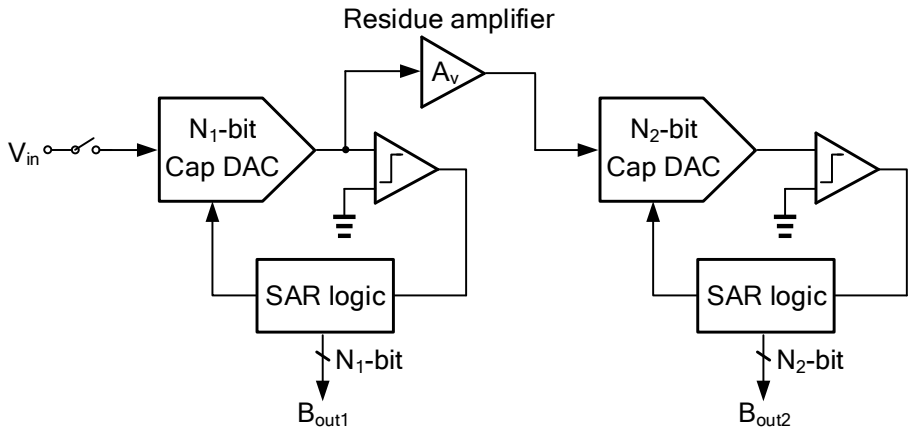


Fig. 2.29 Block digram of a pipeline SAR ADC. It is basically a two-stage pipeline ADC with each stage being a SAR ADC.

The combination of the pipelined SAR ADC with other types of A/D converter to achieve even higher speeds has also been reported. For instance, in [56] a 2b/cycle SAR ADC was used for the first stage. In [57, 58], time-interleaving was also used with a pipelined SAR and in [59, 60], these various types of ADCs were used together.

A major drawback of the pipelined SAR ADC architecture is the presence of an active block, the residue amplifier. Creating an accurate gain, which is usually achieved through a closed-loop topology, demands high power consumption especially in low-voltage technologies. To alleviate this problem, [56, 61] used a dynamic amplifier, where the gain is determined by the open-loop integration time. Offset and output CM calibration of the residue amplifier is also needed. Another issue associated with pipelined SAR ADC is the reference voltage mismatch between the two stages, which calls for circuit adjustment and calibration. The offset mismatch between the two stages' comparators as well as the sampling mismatch also result in comparison errors, that can be partly corrected by redundancy [61].

Noise-Shaped SAR ADC

The same noise shaping technique used in ΔΣ modulators has been employed to reduce the (thermal and quantization) noise in a SAR ADC [62]. The basic idea is to subtract the residue of the previous conversion $V_{RES}[k-1]$, which contains the comparator noise, from the input of the current voltage $V_{\rm in}[k]$, as shown in Fig. 2.30.(a). The transfer function of this system is given by

$$D_{out}(z) = V_{\rm in}(z) + \frac{1}{1+z^{-1}}[Q(z) + V_{\rm n,comp}(z)] \tag{2.23}$$

where $D_{out}(z)$, $Q(z)$ and $V_{\mathrm{n,comp}}(z)$ are the output of the ADC, the quantization noise and the comparator input-referred noise, respectively. (2.23) indicates an all-pass signal transfer function (STF) and a high-pass noise transfer function (NTF). These shape both the quantization noise and the comparator noise, thereby attenuating both at lower frequencies. The noise shaping can be further improved by placing an integrator in the signal path, as shown in Fig. 2.30.(b). This system behaves exactly like a first order ΔΣ modulator. It can be shown that the transfer function of the improved noise-shaping system is

$$D_{\mathrm{out}}(z) = V_{\mathrm{in}}(z) + (1 - z^{-1})Q(z). \tag{2.24}$$

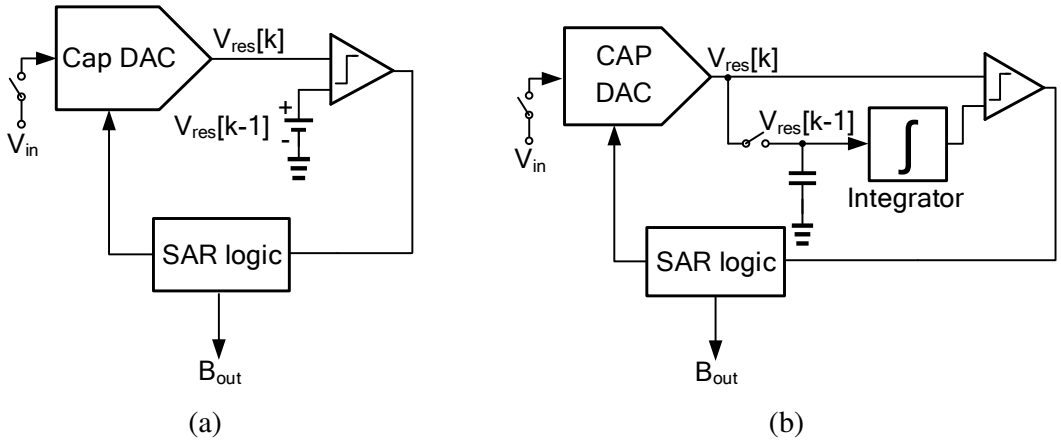


Fig. 2.30 Noise-shaped SAR ADC: (a) block diagram implementation, (b) improvement of the noise shaping by adding an integrator. The noise shaping occurs in the same manner as in a ΔΣ modulator.

In practice, the noise shaping of (2.24) can be implemented by an auxiliary capacitor that shares the charge with the CAP-DAC. [63] uses a 2nd order noise shaping filter with an error-feedback structure. These filters introduce extra noise and increase the ADC area. In addition, opamp-based integrator is hard to scale and its design becomes difficult as the technology and supply voltages scale. To solve these issues, *passive noise shaping* has been employed in [64–66]. [67] managed to achieve a 105dB SFDR using ΔΣ noise shaping and a DAC randomization technique, but had a limitation of maximum input frequency of 1kHz.

Chapter 3

SAR ADC: Circuit Design Considerations and Implementation

In this chapter, we discuss the background theory and the circuit design details of the main building blocks of a SAR ADC. We also provide an overview of the different existing architectures for every block along with the implementation considerations.

3.1 Capacitive DAC

There exist a number of different ways of implementing the DAC for a SAR ADC; the most popular implementation is the capacitive DAC (CAP-DAC) for it also performs the task of the sample-and-hold circuit which obviates the need of having a separate stand-alone S&H. The capacitive DAC also benefits from the fact that it does not consume significant static power.

In subsequent sections, we discuss some of the different architectures for implementing a capacitive DAC for use in a SAR ADC, along with the details of the design. Then, we review various methods for performing the successive approximation approach that is usually referred to as a switching scheme in a SAR ADC. Finally, we introduce the important concept of redundancy in a SAR ADC and the different methods of employing this technique.

3.1.1 Architecture

Active Charge-Redistribution-Based DACs

1. **Thermometer-Code**

Fig. 3.1 shows the most straightforward approach for implementing a capacitive DAC, which is to use 2^N capacitors of the same size to realize an N-bit capacitive DAC. Known as *thermometer-code capacitive DAC*, this architecture is famous for its good DNL performance, that is

$$\sigma_{\text{DNL}} = \frac{\sigma_C}{C}, \tag{3.1}$$

where σ_C is the standard deviation of the unit capacitor C. The monotonicity is also guaranteed for this type of DAC. However, it requires a N-to-2^N decoder. Also, due to the exponential growth of the complexity of the circuit with respect to N, this architecture does not easily lend itself to resolutions greater than 8.

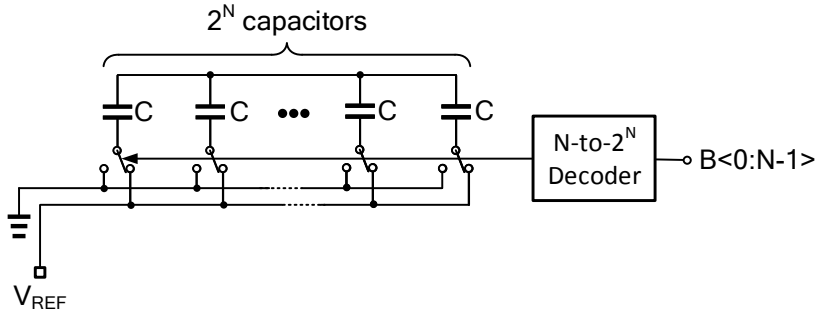


Fig. 3.1 Thermometer-code capacitive DAC. This type of DAC is known for its small DNL error. However, the need for a decoder as well as the large number of capacitors for high resolutions limit its application.

Another drawback of this simple approach is the large value of the total capacitance, that is $2^N C$. Two factors that determine the minimum value of the unit capacitor C are the kT/C noise and the intrinsic capacitor random mismatch, whichever is the most stringent. The requirement imposed by the random mismatch can be worked out from the maximum DNL/INL error. It can be proved that the maximum INL for a thermometer-code DAC is given by

$$\sigma_{\text{INL,max}} \approx 2^{(N/2-1)} \frac{\sigma_C}{C}. \tag{3.2}$$

For $\sigma_{\text{INL,max}}$ to be less than 1 LSB and for a 3σ (99.73%) yield, we should have

$$\frac{\sigma_C}{C} < \frac{2^{(N/2)}}{6}\text{LSB}. \tag{3.3}$$

The above equation puts a limit on the maximum allowable capacitor mismatch for a thermometer-code capacitive DAC.

2. **Binary-Weighted**

Fig. 3.2 shows the implementation of a binary-weighted CAP-DAC. Compared to the thermometer-code DAC, the complexity of a binary-weighted DAC grows only linearly with N. It also does not require an input decoder. However, and similar to the thermometer-code DAC, it calls for a total cap of $2^N C$.

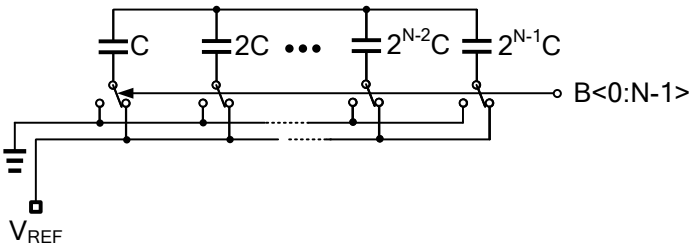


Fig. 3.2 Binary-weighted capacitive DAC. This type of DAC is probably the most popular one own to its simple structure.

It can be proven that the maximum DNL error for the binary-weighted DAC is given by

$$\sigma_{\text{DNL,max}} \approx 2^{(N/2)} \frac{\sigma_C}{C}. \tag{3.4}$$

Again, for $\sigma_{\text{DNL,max}}$ to be less than half a LSB and for a 3σ yield, we should have

$$\frac{\sigma_C}{C} < \frac{2^{(N/2)}}{3} \text{LSB}. \tag{3.5}$$

The above equation puts a limit on the maximum allowable capacitor mismatch for a binary-weighted capacitive DAC.

3. **Bridge-Capacitor**

In order to reduce the total capacitance of the binary-weighted capacitive DAC, one can split the capacitive array into two sections (a LSB and a MSB section) and connect them with a series attenuation bridge capacitor, as shown in Fig. 3.3. It can be shown that with the right value of bridge capacitor, which is given by $C_B = \left(2^{N-M}/(2^{N-M}-1)\right) C$, this configuration can function exactly as an ordinary binary-weighted DAC.

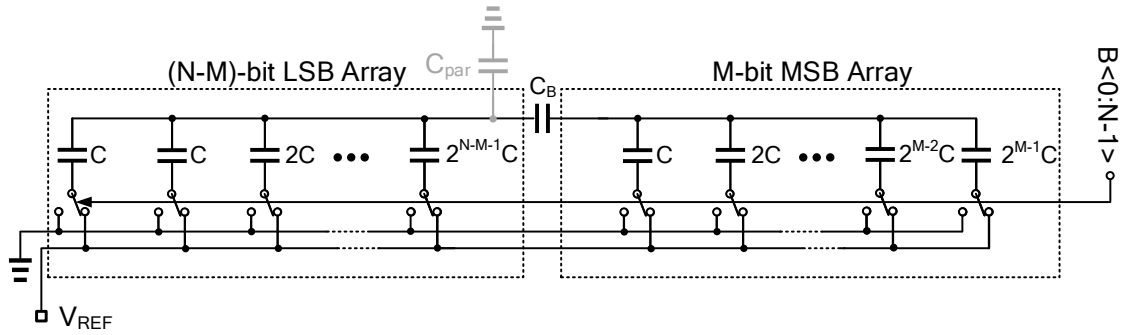


Fig. 3.3 Bridge-capacitor DAC. The parasitic capacitance on the left plate of C_B degrades the linearity.

For the special case where $M = N/2$, the total capacitance of the bridge capacitor DAC is $2^{(N/2-1)}$ times smaller than that of the binary-weighted DAC. This means for the same value of total capacitance, the unit capacitor of the bridge-capacitor DAC can be $2^{(N/2-1)}$ larger than that of the binary-weighted DAC. This relaxes the mismatch requirement of the unit capacitor by a factor of $\sqrt{2^{N/2-1}}$. However, this is not necessarily an advantage since the mismatch requirement for the bridge capacitor is larger by about the same factor.

The main issue associated with this topology is the non-linearity caused by the parasitic capacitor on the LSB plate of C_B, as shown in Fig. 3.3. This non-linearity manifests itself as a discontinuity in the characteristic curve of the DAC, when the transition between the LSB and MSB parts occurs. Different methods of calibration have been proposed in the literature to overcome this issue [68], but these solutions significantly increase the complexity of the circuit.

4. **C-2C**

The "splitting" of the capacitive array into pairs of sub-arrays can be executed repetitively for N times to eventually end up with a configuration shown in Fig. 3.4.

The N-bit C-2C DAC has two advantages over its binary-weighted counterpart: first it only requires $3N$ unit capacitors (C) compared to 2^N capacitors of the binary-weighted DAC, and second its input capacitive load is only $3C$, and independent of the resolution of the DAC, compared to that of $2^N C$ for the binary-weighted capacitive DAC. Nevertheless, this approach is prone to non-linearity errors imposed by parasitics at all of the N intermediate nodes, as shown in Fig. 3.4. In [69] a 7-bit SAR ADC was implemented based on this C-2C topology with a trimming-based calibration of the unwanted parasitic capacitors.

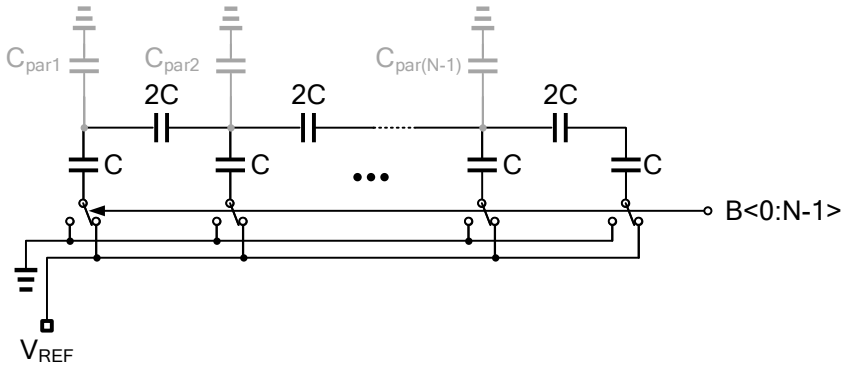


Fig. 3.4 C-2C DAC. This type of DAC requires only $3N$ unit capacitors and introduces an input load of only $3C$. However,the parasitic capacitors limit the linearity.

Passive Charge-Sharing-Based DACs

SAR converters based on charge-redistribution CAP-DAC rely on a precise reference voltage during bit trials, often from a large off-chip/on-chip reference decoupling capacitor or from a power-hungry buffer. During one SAR ADC conversion, the reference voltage is expected to settle at least N times which typically consumes a large portion of the SAR conversion time. As the sampling rate is increased, the reference settling is stressed and often limits the achievable linearity. To address this speed bottleneck, a passive charge-sharing CAP-DAC [70] pre-samples the reference voltage on chip and uses the pre-sampled reference charge to carry out the bit trials. A schematic diagram of a charge-sharing based capacitive DAC is shown in Fig. 3.5.

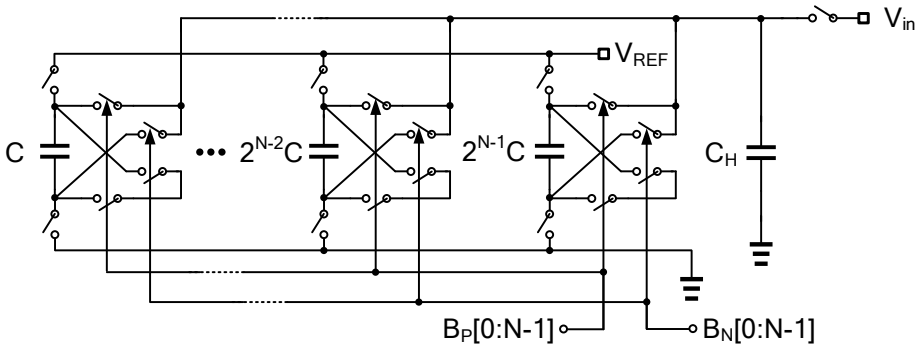


Fig. 3.5 Passive Charge-sharing-based DAC. The reference voltage is sampled on the capacitive array before the conversion starts, and no current is drawn from it during the conversion.

The operation is as follows: initially, the input voltage is sampled onto capacitor C_H, while all other binary-weighted capacitors are charged to V_{REF}. Next, in a process involving N steps, a charge proportional to the value of the binary-weighted capacitors is added or

subtracted from the initial charge stored on C_H until reaches zero. The reference voltage (V_{REF}) is disconnected during the charge-sharing cycle, making this technique immune to reference voltage variations [71]. Also, this type of DAC allows the use of non-linear capacitors. A major drawback, though, is the reduced tolerance to comparator offsets and noise compared to the charge-redistribution method [72]. Moreover, the signal dependent charge injection during the conversion degrades the linearity of the DAC [73].

3.1.2 Switching Schemes

The energy dissipation of the capacitive DAC constitutes a big portion of the energy consumption of the ADC, especially at high frequency. A number of switching techniques have been proposed in the literature to reduce this energy dissipation. Table 3.1 draws a comparison between some of the most famous switching schemes in terms of the energy consumption.

Table 3.1 Various switching schemes and their energy consumption

Switching Scheme	Energy Consumption
Conventional	100%
Split Capacitor [74]	63%
Energy Saving [75]	46%
Charge Averaging [76]	28%
Monotonic [77]	19%
Set-and-Down [78]	19%
Detect-and-Skip [79]	17%
V_{cm}-Based [80]	12%
Merged-Capacitor [81]	7%
V_{cm}-Based Monotonic [82]	3%

Even though some of the switching schemes achieve a significant reduction in energy dissipation, the complexity of the control logic for the switching algorithm make them unsuitable for high-speed applications. Moreover, for a differential ADC, the common-mode (CM) voltage at the output of the DAC changes during the conversion for some of the switching schemes, making it difficult to design a simple and robust comparator. Some of the switching methods also need a third reference voltage.

The *split-capacitor* scheme is a good candidate for high speed applications for it requires a fairly simple and straightforward logic controller. Fig. 3.6 demonstrates the principal behind the split capacitor technique. As can be seen, every specific capacitor in the binary-weighted capacitor array is divided into two halves, where the bottom plate of one capacitor

is initially connected to the positive reference voltage, V_{REFP}, while the other capacitor is connected to the negative reference voltage, V_{REFN}.

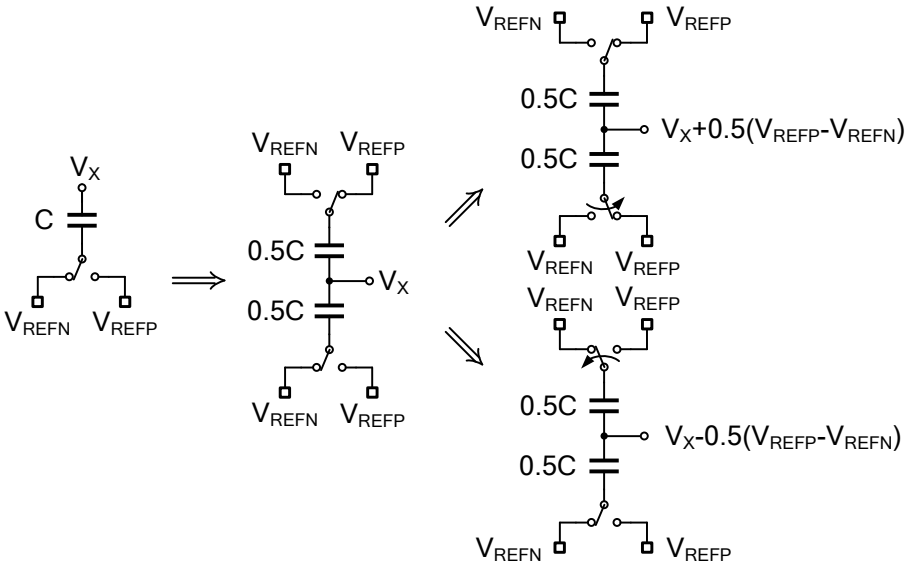


Fig. 3.6 Principle of the split capacitor switching scheme

By applying the principle of the split capacitor of Fig. 3.6 to an N-bit binary-weighted capacitive, the structure of Fig. 3.7 is resulted. In a differential SAR ADC, the polarity bit is determined by only comparing the differential inputs using the comparator before any DAC switching occurs, because of which an $(N-1)$-bit DAC is sufficient for an N-bit A/D conversion. The bottom-plate switch for the positive and negative reference voltages in Fig. 3.7 is just an inverter, as shown in Fig 3.8. The inverters too should be sized in a binary-weighted manner to guarantee a constant RC time constant for all capacitors during bit cycling.

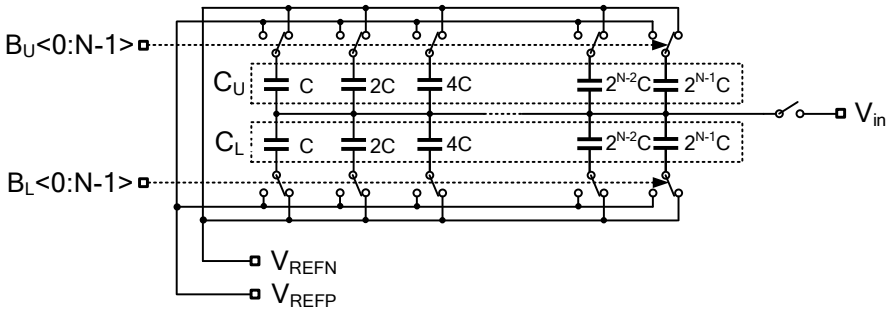


Fig. 3.7 An N-bit binary-weighted split-capacitor array. The split-capacitor architecture guarantees constant CM voltage and requires only two reference voltages. However, it need two independent control signals which would call for a more complex SAR logic circuit.

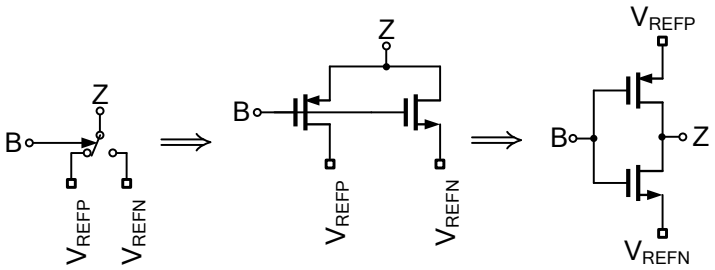


Fig. 3.8 Circuit implementation of the bottom-plate switch

The average energy saving of the split-capacitor array is 37% compared to the conventional approach. Also, because of the full symmetry of the switching, the DAC output CM is constant over the entire conversion, which is particularly important for the design of the comparator. Another advantage of the split-capacitor switching scheme over the conventional scheme is a factor of $\sqrt{2}$ reduction in maximum DNL. As already discussed in 3.1.1 and according to (3.4), for the conventional binary-weighted capacitive array, we have $\sigma_{\text{DNL,max}} \approx 2^{N/2}\sigma_0$. For the split-capacitor structure, this maximum DNL is [81]

$$\sigma_{\text{DNL,max}} \approx \frac{2^{N/2}}{\sqrt{2}}\sigma_0. \tag{3.6}$$

The maximum INL nevertheless is the same for both schemes and is equal to

$$\sigma_{\text{INL,max}} \approx 2^{N/2-1}\sigma_0. \tag{3.7}$$

The split-capacitor architecture only requires two reference voltages, namely V_{REFP} and V_{REFN}. However, it has three separate states, which requires two independent control signals. This means a more complex SAR logic, higher power consumption and larger logic latency.

3.1.3 Redundancy

Redundancy is a simple yet powerful technique that is widely used in all types of ADC including the SAR ADC. Redundancy can be introduced in different ways for the purpose of giving the ADC a tolerance to a certain amount of a specific error. For instance, and as was discussed in section 2.3.5, in a 1.5-bit-per-stage pipeline ADC, a redundant decision level is used at every stage, which makes the ADC tolerant to comparator offset of up to $\pm V_{\text{REF}}/4$. In the SAR ADC redundancy is introduced through one (or more) redundant comparison cycles. The principle is illustrated in Fig. 3.9, where the output voltage, V_X, of a 5-bit single-ended binary-weighted DAC is plotted for a specific input voltage. In Fig. 3.9.(a) no decision error

occurs and V_X converges to within one LSB after 5 cycles. The output code of the ADC is 12 for this case. In Fig. 3.9.(b), V_{REF} experiences a glitch just before the third comparison, making the third decision incorrect. The important point here (that makes the concept of redundancy easier to understand) is that V_X keeps converging towards the LSB region, but due to the wrong decision does not have sufficient time to reside in this interval. If, somehow, the DAC is given more time (say through the introduction of a redundant comparison) then it will eventually converge within 1LSB, as can be seen in Fig. 3.9.(c), and this is the basic concept of redundancy in a SAR ADC. This extra time can be provided in various ways as will be discussed later.

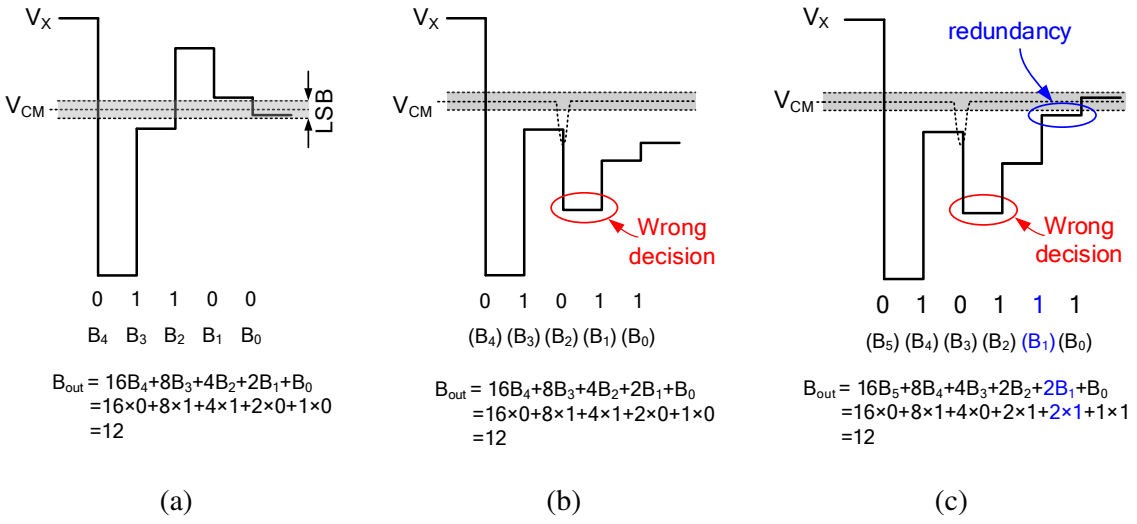


Fig. 3.9 Binary search (a) with no error, (b) with a wrong decision, (c) with correction through redundancy.

The concept of redundancy can also be described in the following way: the introduction of redundancy provides more than one single path (digital output code) for a specific input, making the ADC resilient to a certain range of dynamic errors during the conversion. The redundant search algorithm can compensate for dynamic errors such as settling, reference voltage, or comparator decision errors, but does not correct for, e.g., the DAC nonlinearity or the thermal noise of the comparator [1].

The extent to which the redundancy can correct the occurrence of an error depends on the magnitude of the redundancy introduced. The magnitude of the tolerable decision error in each step is almost equal to the difference between the current bit weight and the sum of all remaining weights. More accurately speaking, if we define the redundancy in the $k-$th

[1] In [83], an extra redundant comparison step was added to mitigate the impact of random noise. This type of redundancy though is not what is usually referred to as redundancy in SAR ADCs.

step, $q(k)$ as

$$q(k) = -p(k+1) + 1 + \sum_{i=k+2}^{N} p(i), \tag{3.8}$$

where $p(i)$ is the weight of the i-th step, it can be proved [84] that if a wrong decision is made in the k-th step, the redundancy can still recover the error if $|V_{in} - V_{\mathrm{REF}}(k)| < q(k)$. Here $V_{\mathrm{REF}}(k)$ is the reference voltage for the k-th step, which is given by

$$V_{\mathrm{REF}}(k) = \sum_{i=1}^{k} d(i-1)p(i), \tag{3.9}$$

where $d(i)$ is the decision of the comparator ("0" or "1") at the i-th step.

Different Types of Redundancy

1. *Radix=2 with Redundant Decision Level*

   The redundancy can be introduced by adding an extra decision level to create multiple trajectories for the same input [85]. This idea is mostly used in pipeline ADCs, as mentioned previously.

2. *Radix=2 with Redundant Comparison*

   As we discussed before, adding an extra comparison could give the ADC enough time to recover from a previous error. This type of redundancy is basically what is shown in Fig. 3.9 and was first introduced in [86]. The implementation comes at little cost by just repeating a capacitor in the binary-weighted capacitor array. The position of the extra capacitor and the comparison associated with it is where the redundancy occurs.

3. *Radix<2 (Sub-Radix)*

   Another way of introducing a redundant decision is by using a ratio of capacitance that is less than 2. This would make the number of all possible binary outputs greater than 2^N, meaning that there would be more than one possible output code for a single input voltage (redundancy). The idea was first introduced in [87] and afterwards became a common practice in designing SAR ADCs [87–101].

   Sub-radix redundancy can be realized in two ways. One way is to use a fixed ratio of $r < 2$ between the capacitors. This, however, calls for non-integer values of the capacitances, which is difficult to implement. Moreover, the radix must be known

precisely in order to construct the proper conversion result. In practice, the radix is typically measured using some form of calibration [92, 102]. In [90, 103] a smart solution for implementing a radix of < 2 was proposed. This is shown in Fig. 3.10, where the radix is equal to $1+\beta/\alpha$ and can be adjusted by choosing α and β.

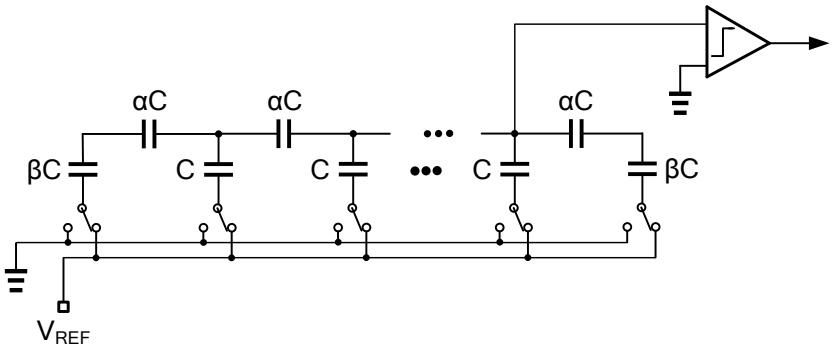


Fig. 3.10 An implementation of non-binary capacitive array.

Another way to realize a sub-radix DAC is to use a variant ratio of less than 2 between capacitors in a way that the resultant values for the capacitors are still an integer multiple of the unit capacitor [97, 98, 100]. For instance, the non-binary weights of $\{128,46,26,20,14,8,6,4,2,1\}$ can be used instead of the binary weights $\{128,64,32,16,8,4,2,1\}$ for an 8-bit DAC. In this case, 10 cycles (instead of 8) is needed for every conversion, which means two redundant cycles. Although, this approach has the advantage of a straightforward layout of individual capacitors compared to the non-integer sub-radix approach, the use of the common-centroid capacitor array arrangement (which improves matching) is impeded. For this reason, redundancy in the digital control is also used [104, 105]. With this approach, the decision weights are stored in a ROM and during every conversion step, an arithmetical unit computes the decision level (an integer number) depending on the comparator result for every specific comparison. A reference voltage is then generated by the DAC through switches that are driven by the digital circuit and a decision level is made [104]. This approach, however, increase the complexity of the digital section.

It is worth mentioning that, regardless of what approach is used for introducing redundancy, the binary output code which includes the redundancy bits should be eventually decoded into an N-bit code. This digital decoder (which is constructed by adders and registers) has to work with the same speed as the ADC itself [86, 106, 107], complicating the ADC design especially for high sampling rates and increasing the ADC power dissipation.

3.2 SAR Control Logic

The successive approximation register, known as the SAR logic, is responsible for the proper switching of the DAC, based on the decision made by the comparator. Even though it is a completely digital block, its power consumption constitutes a big part of the total power budget, when the clock frequency is high. It also plays an important role in the maximum achievable speed of the SAR ADC. Many different techniques have been proposed in the literature to optimize the power and speed of this critical block. The SAR topology is generally realized in two different ways: synchronous and asynchronous, as will be discussed in detail as follows.

3.2.1 Synchronous

The conventional implementation of a successive approximation algorithm in a SAR ADC relies on a synchronous clock to divide time into a signal sampling (tracking) phase and a conversion phase, which progresses from the MSB to the LSB as shown in Fig. 3.11. For an N-bit converter with a conversion rate of f_s, a synchronous approach would require a clock running faster than $(N+1)f_s$.

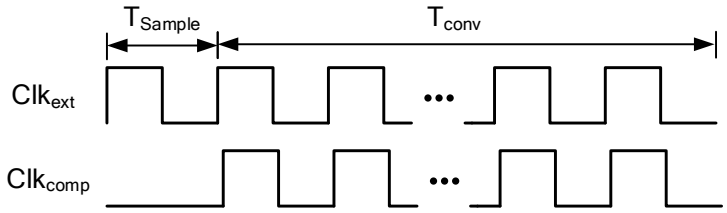


Fig. 3.11 Timing diagram of synchronous SAR scheme. The clock frequency should be at least N times higher than the sampling rate of the ADC. Generation of this clock could be a real challenge especially for high speed SAR ADCs.

A synchronous SAR control logic, in its simplest form, consists of a shift register and an array of Flip-Flops (FF), as shown in Fig. 3.12. The shift register, which itself is also composed of FFs, shifts a pulse which is then used to activate the DFFs one by one to generate a ZERO or a ONE based on the comparator output. In other words, the circuit provides an indication of the working cycle which controls the SA logic to propagate the comparator decisions to the DAC. For an N-bit ADC, an N bit shift register and an array of N DFFs are needed.

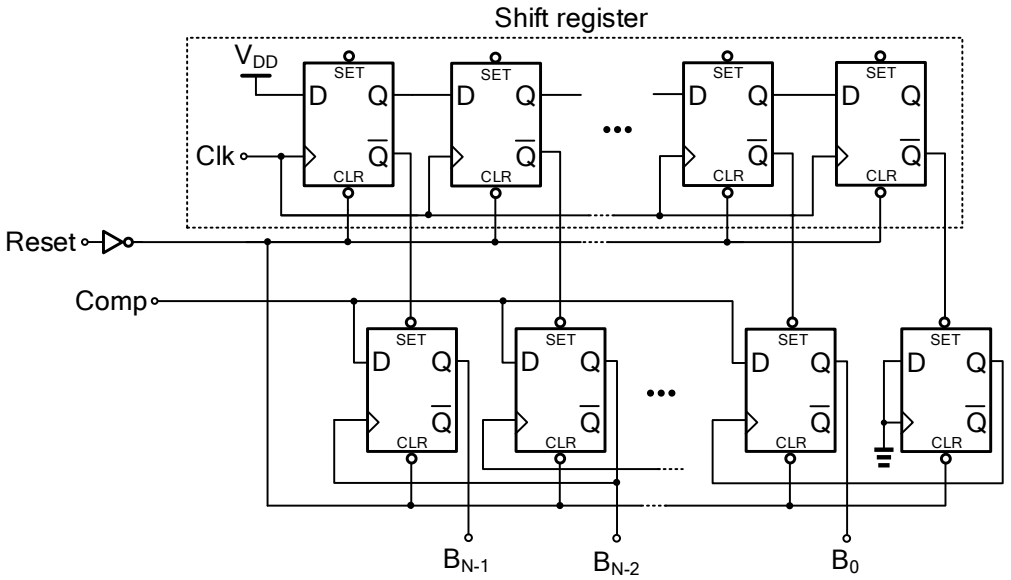


Fig. 3.12 Basic block digram of a synchronous SAR logic controller. The building block is a DFF.

Since DFF is the building block of the SAR control logic, the optimization of the SAR logic comes down to optimizing a single DFF. There are three main architectures for realizing a DFF; it can be realised by using standard logic gates, as shown in Fig. 3.13. Even though this is the most straightforward way of implementing a DFF, it is inferior in terms of both the power consumption and the speed.

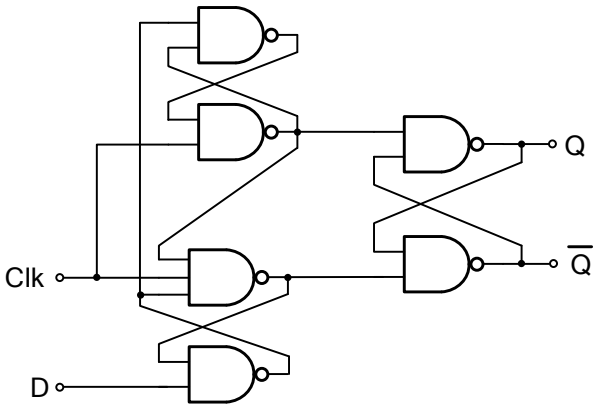


Fig. 3.13 Conventional logic-gate-based DFF. This type of DFF is not suited for high-speed applications.

A switched-based DFF is another way of implementing this block, which works by storing the digital data the same way as done in the analog domain, i.e. using a switch and a capacitor. A schematic diagram of this type of DFF is shown in Fig. 3.14.(a). This is similar

to a conventional Master-Slave DFF and is composed of a Master stage and a Slave stage. When CLK is low (Fig. 3.14.(b)), switch S_1 is ON and the data at D reaches node A. At the same time, switch S_4 is closed, which forces a positive feedback formed by two back-to-back inverters and holds the previous data passed to the Slave stage. Once CLK goes high (Fig. 3.14.(c)), switch S_2 closes and forces positive feedback in the Master stage, which then holds the data on the input of the Master stage waiting for the next clock cycle.

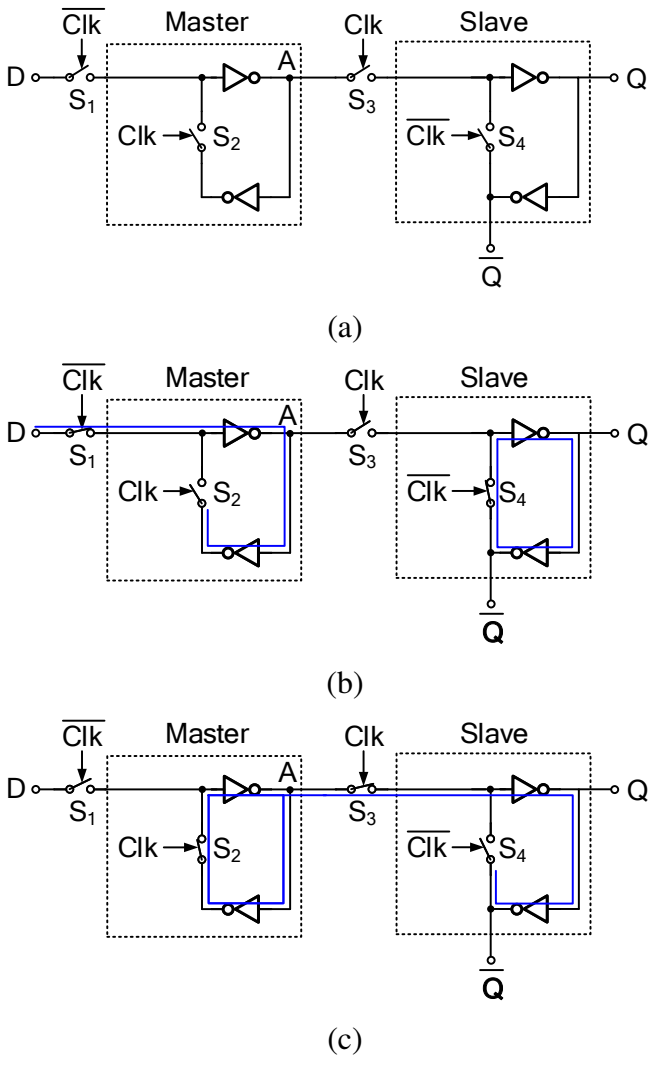


Fig. 3.14 Switch-base DFF: (a) topology, (b) clock is low and the previous data is stored by the back-to-back inverters of the Slave stage, (c) clock is high and the current data is sampled on node A and Q.

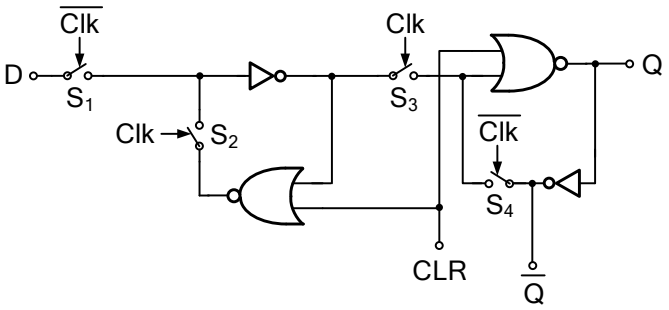


Fig. 3.15 Modification of Fig. 3.14 with the clear signal.

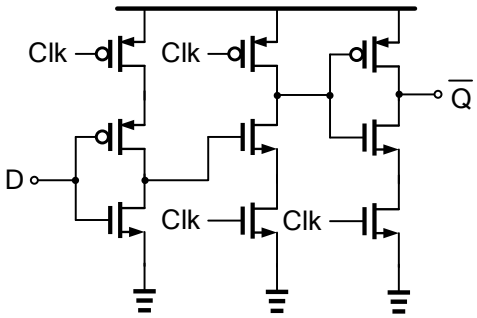


Fig. 3.16 Dynamic DFF. This type of DFF, if designed properly, can achieve high speeds with low power consumption.

The switched-based DFF is superior to its conventional counterpart both in terms of power and speed. Since a reset (clear) signal is also needed for the SAR Logic operation, the circuit of Fig. 3.14 can be easily modified to incorporate a clear signal as well, as illustrated in Fig. 3.15.

The third type of DFF is based on dynamic (C^2MOS) logic. Dynamic logic gates are a whole family of logic circuits that are used to decrease circuit complexity, increase operating speed and reduce power consumption [108, 42]; since the precharging, selection, switching and latch function can be combined together with one dynamic logic gate in this design. A dynamic DFF is shown in Fig. 3.16. The state transition of this FF occurs at the rising edge of the clock signal. While dynamic logic can be faster and less power hungry compared to other types of logic, there are known disadvantages such as glitches, charge sharing, metastability and leakage.

3.2.2 Asynchronous

In synchronous SAR logic, the time allocated for the comparison(s) is equal for all the bit decisions, as shown in Fig. 3.11. Consequently, during an easy decision (where the input

of the comparator is large) the comparator latches quickly and is followed by a period of *inactivity*, hence wasting some of the conversion time. Moreover, for a SAR ADC with a sampling frequency of f_s, a high speed clock of frequency $(N+1)f_s$ is needed. For a high speed SAR ADC, synthesizing this higher-frequency clock, along with implementing the associated clock distribution network, would probably consume more power than the ADC itself. This brings us to the common practice of *asynchronous* SAR Logic for SAR ADCs. In an asynchronous control scheme, the SAR logic starts to function immediately after a comparison is made by the comparator. This is done by a *valid* signal generated by the comparator, which indicates when a comparison is complete. For a differential-output comparator, a simple XOR gate would do the job. The timing diagram of an asynchronous SAR algorithm is shown in Fig. 3.17.

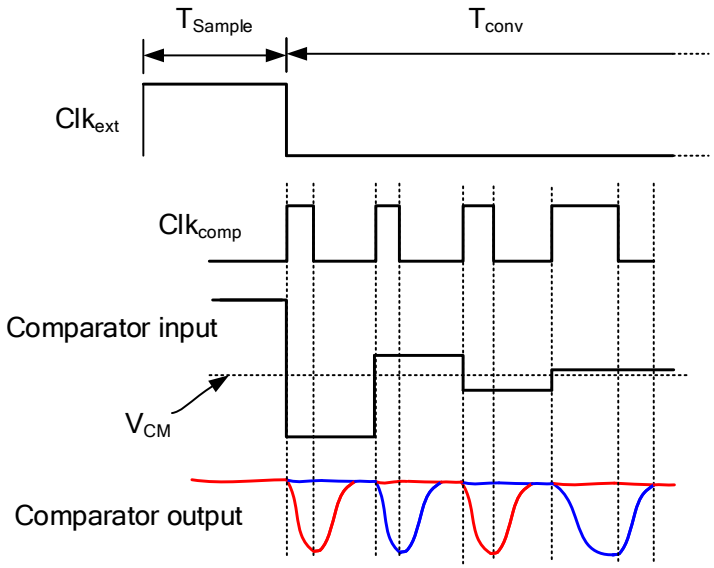


Fig. 3.17 Timing digram of the asynchronous SAR scheme along with the comparator input/output signals. The comparator is reset right after a decision is made (plus a delay for the DAC settling) and the next comparison starts immediately.

The speed improvement of an asynchronous compared to a synchronous scheme depends on many factors (e.g. comparator architecture) and is a multiple-folded problem. In [109], a simple model for the comparator was considered and minimum and maximum limits were found for the ratio $T_{\text{async}}/T_{\text{sync}}$ for an N-bit SAR ADC, where T_{sync} and T_{async} are the total resolving times for the synchronous and the asynchronous SA schemes, respectively. These

limits are given by

$$\frac{T_{\text{async}}}{T_{\text{sync}}}|_{\text{min}} = \frac{1}{2} \tag{3.10}$$

$$\frac{T_{\text{async}}}{T_{\text{sync}}}|_{\text{max}} = \frac{(N-1)\ln 3 + \ln 2 + \frac{N}{2}(N+1)\ln 2}{N(N+1)\ln 2}. \tag{3.11}$$

The ratio $T_{\text{async}}/T_{\text{sync}}$ in (3.11) approaches $1/2$ as N increases. In other words, *an asynchronous SA scheme improves the conversion speed of a SAR ADC by almost a factor of 2*, which is a significant improvement.

3.2.3 Special Techniques

Several innovative techniques have been proposed in the literature to improve the speed and/or power consumption of SAR logic even further.

Even with an asynchronous SAR control scheme, the propagation delay of the comparator can still be excessive, when the input is small. By introducing redundancy, this comparison, if detected, can be skipped. In [110, 111] a comparator time-out scheme was proposed to detect long comparison times and to skip the step if longer than a certain time. The idea is very simple. An extra path is placed in parallel with the comparator to generate a second *Valid* ($Valid_2$) signal along with the Valid signal generated by the comparator ($Valid_1$), as shown in Fig. 3.18. The two *Valid* signals are ORed to make the final *Valid* signal that activates the SAR Logic. Both the comparator and the auxiliary delay path are triggered by the same signal (generally the clock). The signal $Valid_2$ goes high after a fixed certain time T_d once triggered. This means that if the comparator is not able to make a decision fast enough (say within T_d seconds), then the delay path takes over and forces *Valid* to go high.

A different method to reduce the delay of the SAR logic was introduced in [112]. Called *flip-flop bypass* SAR logic, this technique is based on storing the comparator outputs one half clock cycle after applying them to the feedback capacitor DAC. A parallel bypass SAR logic was introduced in [113], where the SAR logic and the comparator work in parallel in order to reduce the delay of the SAR feedback loop, as shown in Fig. 3.19. A logic window is produced to match the result of the comparator, during which the result of the comparator is caught. The delay due to the SAR logic should be slightly shorter than that due to the comparator, so that the logic window is ready before the presence of the valid signal. This technique completely eliminates the delay of the SAR logic from the total loop delay.

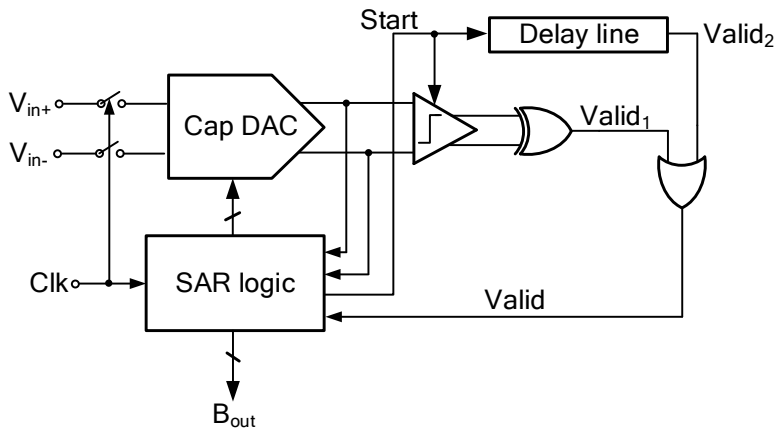
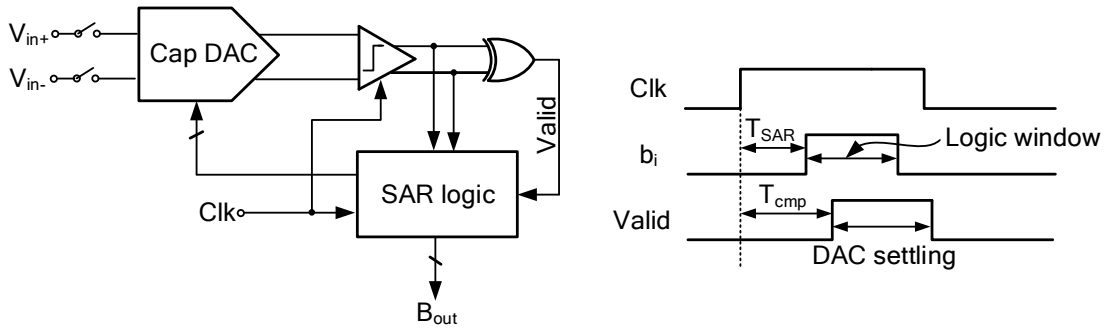


Fig. 3.18 Asynchronous SAR ADC with an auxiliary delay path. In case if the comparator decision process takes longer that the delay of the auxiliary path, the valid signal will generated regardless.



(a) (b)

Fig. 3.19 Parallel bypass SAR logic: (a) circuit implementation (b) timing diagram. This technique completely eliminates the delay of the SAR logic from the total loop delay.

The asynchronous approach can be implemented in a completely different way, as proposed in [114]. Using this approach, N comparators instead of one are used for an N-bit conversion, as shown in Fig. 3.20. Each comparator is responsible for the switching of one capacitor in the DAC, such that the DAC can respond to the results immediately and generate the successively approximated analog signal without being delayed by the digital logic. Once the comparison is finished, a valid signal is generated in the same way as for a conventional asynchronous SAR logic, which triggers the proceeding comparator for the next bit comparison. This procedure propagates in a "domino" fashion until the LSB comparison finishes. Using this method, the conversion time is reduced in three ways compared to the conventional asynchronous SAR ADC. First, no DFF or latch delay is

needed to store the comparator output. Second, comparators are reset simultaneously, and thus, no comparator reset time is required for every comparison cycle. Third, this method allows independent optimization for each comparison cycle. Overall, the optimized critical path for each comparison cycle can be represented as

$$T_d = t_{\text{comp,reg}} + \max\{t_{\text{DAC}}, t_{\text{ready}}\}, \tag{3.12}$$

where $t_{\text{comp,reg}}$, t_{DAC} and t_{ready} are the comparator regeneration time, the DAC settling time and the delay of the *ready* signal generator circuit, respectively. It should be noted that the power consumed by the comparators for this architecture with N comparators is not necessarily larger than that of a conventional SAR ADC where only one comparator is used. This is because here, every comparator makes one single comparison during every conversion, making the total number of comparisons equal to N per conversion. This number is the same as for the conventional SAR ADC where N comparisons are made by one comparator during every conversion. A disadvantage associated with this technique is the offset mismatch between the N comparators, which causes distortion and degrades linearity [115]. Therefore, a part of the power budget for the ADC has to be used for either low-offset comparators or for an offset calibration circuit. Employing redundancy can also compensate for the offset errors before the redundant cycle.

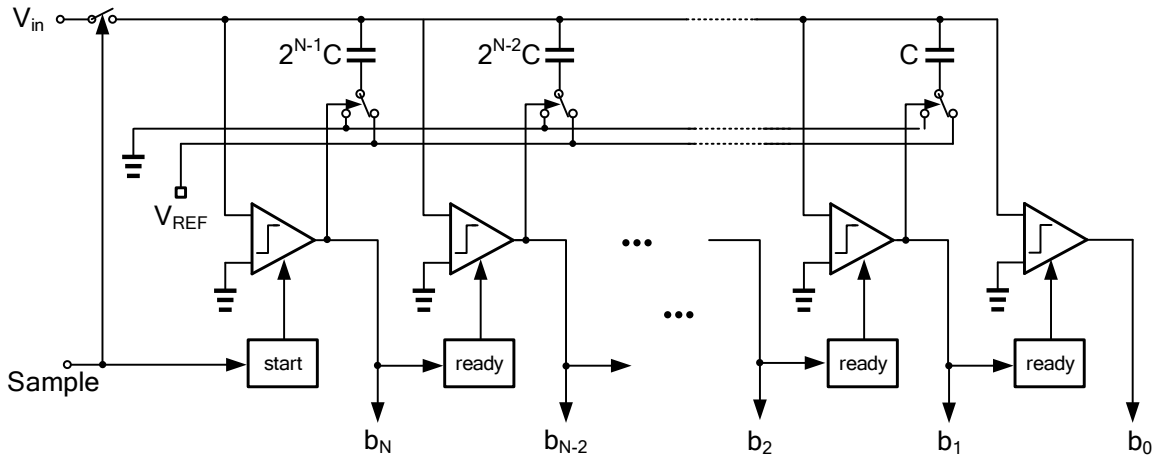
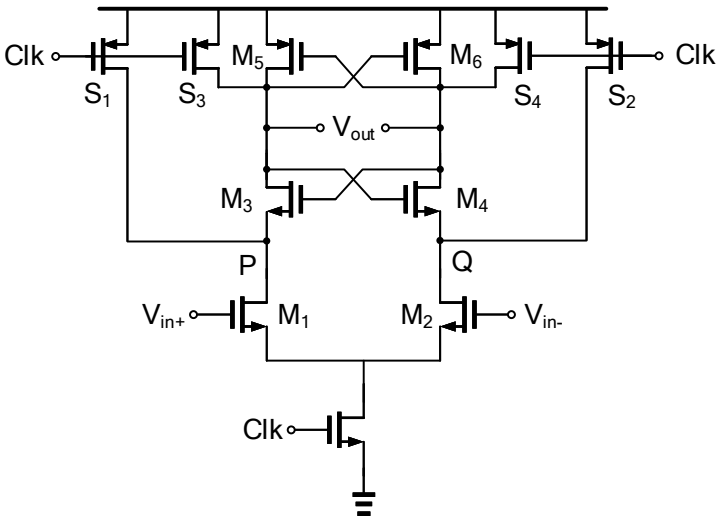


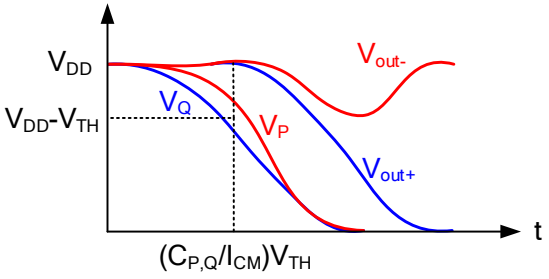
Fig. 3.20 Asynchronous SAR ADC using N comparators. This technique completely eliminates the delay of the DFFs in the SAR logic as well as the reset delay of the comparators.

3.3 Comparator

For low power, a dynamic comparator is better than its static counterpart in that there is no static power consumption. Different architectures for dynamic comparators have been discussed in the literature [95, 116, 117]. For high speed applications, the so called "StrongARM" latch [118, 119] is a famous architecture whose structure is shown in Fig. 3.21.(a). As can be seen in the figure, the StrongARM latch consists of a clocked differential pair, $M_{1,2}$, two cross-coupled pairs, $M_{3,4}$, and four switches $S_1 - S_4$. The operation of the latch is as follows: when *Clock* is low, M_1 and M_2 are off and nodes P, Q, X, and Y are precharged to V_{DD}. When *Clock* goes high, the switches $S_1 - S_4$ turn off, and M_1 and M_2 turn on, drawing a differential current in proportion to $V_{in+} - V_{in-}$ from C_P and C_Q. This makes the voltages at nodes P and Q drop at different rates, causing a differential voltage of



(a)



(b)

Fig. 3.21 StrongARM latch: (a) circuit implementation, (b) voltage of important nodes during one cycle.

$$|V_P - V_Q| \approx (g_{m1,2})|V_{\text{in+}} - V_{\text{in-}}|/C_{P,Q}t, \tag{3.13}$$

where $g_{m1,2}$ is the transconductance of the transistors M_1 and M_2. This voltage drop continues until the NMOS cross-coupled transistors turn on, i.e. when V_P and V_Q fall to $V_{\text{DD}} - V_{\text{TH}}$. Therefore, this phase lasts for approximately $(C_{P,Q}/I_{\text{CM}})V_{\text{TH}}$ seconds, where I_{CM} is the common-mode current drawn from each side. At this point, a positive feedback is created by M_3 and M_4, making $V_{\text{out+}}$ and $V_{\text{out-}}$ split towards the two rail voltages in an exponential manner due to the differential currents caused by the input differential voltage (regeneration). It can be shown that [120] the regeneration time constant is

$$\tau_{reg} = \frac{C_{out}}{g_{m3,4}(1 - C_{out}/C_{P,Q})}, \tag{3.14}$$

where $g_{m3,4}$ is the transconductance of the transistors M_3 and M_4 and C_{out} is the total capacitance at the output of the comparator. Fig. 3.21.(b) shows the behaviour of the different voltages of the StorngARM latch during one clock cycle.

The power consumed by the StrongARM latch is mainly dynamic, arising from the charge and discharge of the capacitances, and is roughly equal to $f_{\text{Clk}}(2C_{P,Q} + C_{\text{out}})V_{\text{DD}}^2$. In [121] a multi-step design methodology was presented for designing a stongARM latch for a given set of specifications.

3.3.1 Double-Tail Comparator

A disadvantage associated with the conventional StrongARM latch of Fig. 3.16 is the stack of four transistors, which requires a large voltage headroom, making this topology unsuitable for low-voltage designs. Moreover, the speed and offset of this circuit depend on the common-mode voltage at the input [122, 123]. An alternative design, known as the *double-Tail comparator*, was proposed in [124] where two separate tails were used for the input stage and the latch stage. The architecture of this type of latch is shown in Fig. 3.22.

Operation

The operation of the double-tail comparator of Fig. 3.22 is as follows: during the reset phase and when *Clock* is low, transistors M_3 and M_4 precharge nodes D_{i+} and D_{i+} to V_{DD}, which causes M_7 and M_8 to discharge the output nodes to ground. When *Clock* goes high, M_1 and M_2 turn on, drawing a differential current in proportion to $V_{\text{in+}} - V_{\text{in-}}$ from C_{Di+} and C_{Di-}.

The operation then proceeds similar to the StrongARM latch with the cross-coupled transistor kicking in and providing a positive feedback, which then takes the outputs to the supply and ground rails. Based on simulation, it was shown in [124] that the double-tail comparator is superior to its StrongARM counterpart both in terms of delay (speed) and input-referred offset.

The double-tail comparator requires high accuracy timing of $\overline{Clk}$ because the latch stage has to detect ΔV_{Di} in a very short time. An alternative topology was proposed in [125] that used the falling edge at nodes D_is for the latch timing. This is shown in Fig. 3.23. Here, the added transistors M_{11} and M_{12} pre-charge both the latch and the inputs of the first stage as well. This helps increasing the comparator sensitivity by increasing the gain of the second stage. Furthermore, this architecture relaxes the clock driving requirements as there is only one clock phase. The task of transistors M_{13} and M_{14} is to reset the nodes X_{i+} and X_{i-} to avoid mismatch voltages between them that would cause comparator offset. It was also shown in [125] that this comparator had better noise performance compared to the one shown in Fig. 3.22.

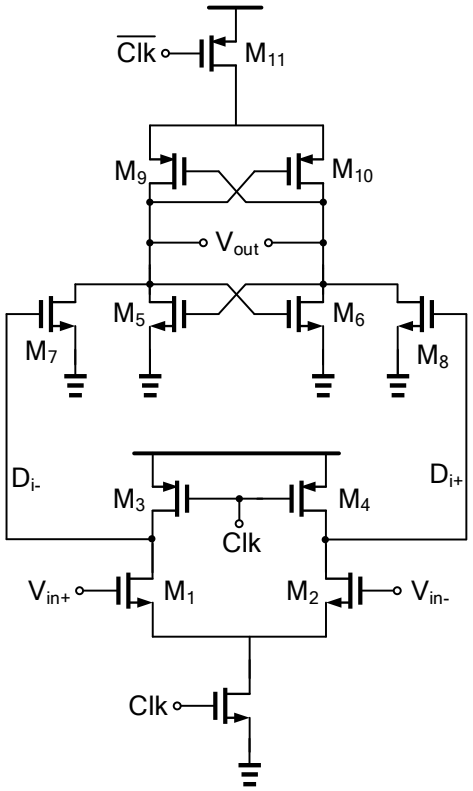


Fig. 3.22 Double-Tail comparator. Two separate tails are used for the input stage and the latch stage.

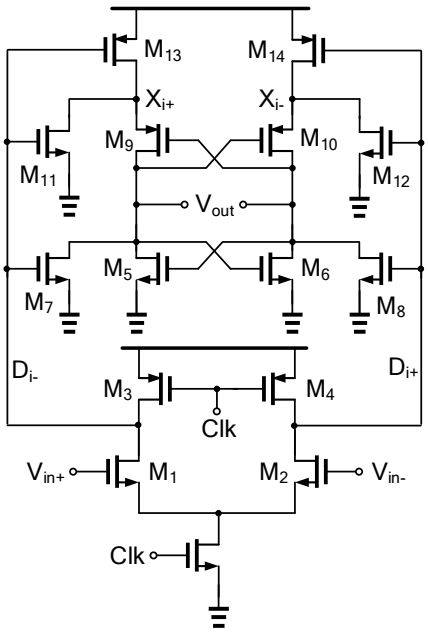


Fig. 3.23 Improved double-tail comparator.

Kick-Back Noise

The voltage variations on the output nodes of a comparator can be coupled, through the parasitic capacitance of the transistors, to its inputs. Since the circuit preceding the comparator does not have zero output impedance, the input voltage is disturbed, which may degrade the accuracy of the comparator. This disturbance is usually called *kickback noise*. Kick-back noise is a common problem for a dynamic comparator. A number of techniques have been proposed to reduce the harmful effect of kick-back noise. The most common solution is to add a pre-amplifier before the comparator [126]. This nevertheless compromises the speed of the comparator and increases its power consumption. Another solution, shown in Fig. 3.24.(a), is to insert switches between the drain of the input pairs and the load (S_1 and S_2), which open when the regeneration starts [127, 128]. This technique also jeopardizes the speed of the latch since the input pair enters the triode region upon the opening of the switches. It also introduces a resistor in the signal path, which slows down the latching operation even further.

Isolating (disconnecting) the inputs of the comparator from the nodes that are sensitive to kick-back noise is another method that was used in [129]. This is done through a series switch, which is kept OFF during the regeneration phase when the kick-back noise is maximum and most disturbing. However, the kick-back noise still corrupts the input signal when the switch is closed. Moreover, the series switch acts as a resistor in the signal path, thereby increasing the time-constant of the sampling network.

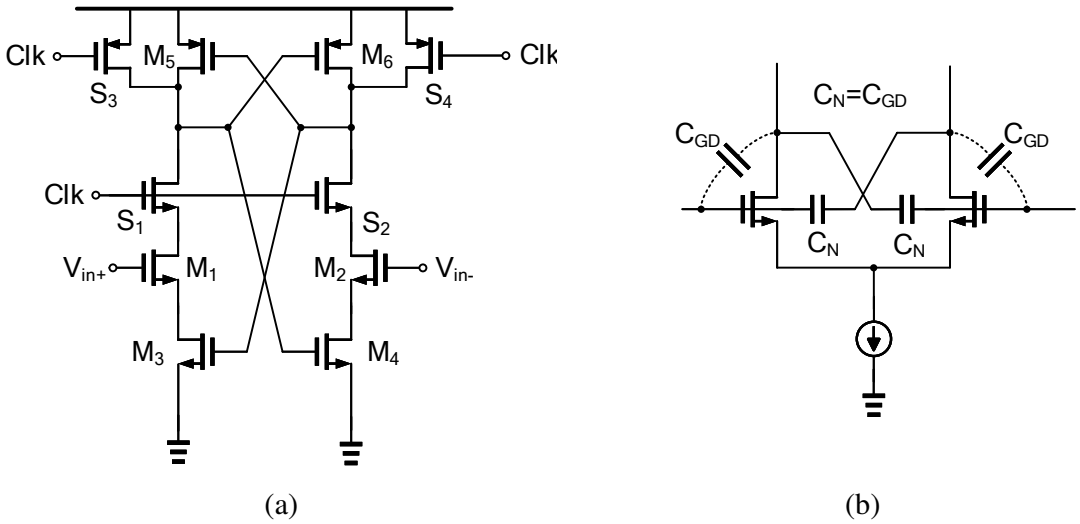


Fig. 3.24 Reducing the effect of kickback noise by (a) putting switches between the input and the output of the latch, (b) capacitive neutralization.

Neutralization is another technique to alleviate the problem of kick-back noise [130, 131], and is done by adding two cross-coupled capacitors with value equal to C_{GD} of the input devices to *neutralize* these capacitors, as shown in Fig. 3.24.(b).

Apart from the aforementioned *differential* kick-back noise, there also exists *common-mode* kick-back noise, which is specific only to dynamic comparators. In a dynamic comparator, once the comparator is clocked, the source and drain of the input transistors will be pulled to ground. This transition draws charge from the gates of the input devices through C_{GS} and C_{GD} and causes common-mode kickback noise on the input. If there is any mismatch between the input transistors, the common-mode kickback noise will be converted into differential one, causing distortion. A dummy comparator was used in [42] to neutralize the common-mode kick-back noise caused by both the C_{GD} and C_{GS} of the input transistors. The idea is shown in Fig. 3.25, where the rising transition of the source and drain of the dummy pairs is in different direction, supplying all the charge drawn by the falling transition of input transistors and largely cancelling the kickback noise if transistors in the replica circuits matches those in the main one. Nevertheless, this only occurs if there is no timing mismatch between the two transitions.

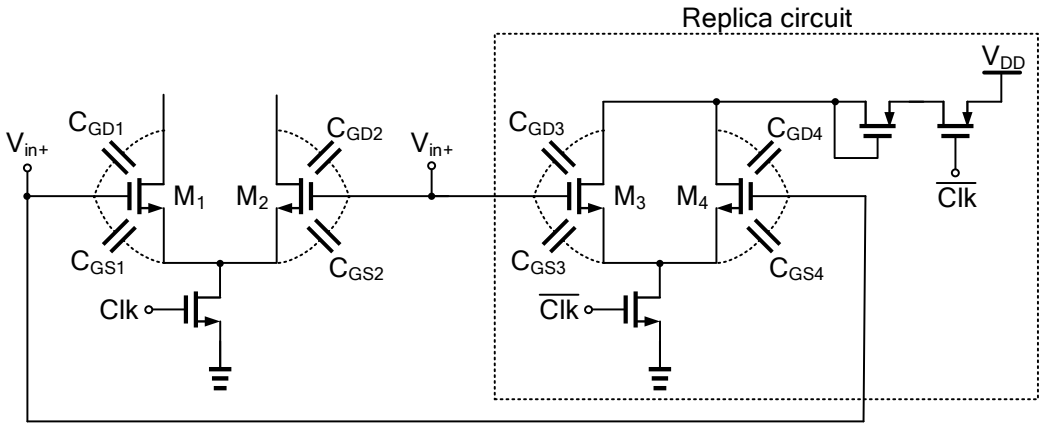


Fig. 3.25 Kickback noise cancellation through a replica (dummy) comparator. This technique is to cancel out both the static and the dynamic kick-back noise.

Thermal Noise

The input-referred noise of the double-tail comparator σ_n can be estimated by [132]

$$\sigma_n \approx \sqrt{8kT\frac{\gamma}{g_{\mathrm{m,in}}}.\mathrm{NBW}}, \tag{3.15}$$

where k is the Boltzmann constant, T is the absolute temperature, γ is a technology-dependent factor (around 2.5), $g_{\mathrm{m,in}}$ is the transconductance of the input devices and NBW is the noise bandwidth. NBW is expressed as

$$\mathrm{NBW} = \frac{1}{2T_{\mathrm{int}}}, \tag{3.16}$$

where T_{int} is the integration time. According to (3.15), to decrease the input noise, we must increase the g_m of the input pairs and/or increase T_{int}. The latter can be achieved by increasing the capacitive load at nodes D_{i+} and D_{i+}, which then lowers the speed. Large input transconductance provides good noise performance, since the preamplifier has a large output voltage difference against noise at the moment of decision. A large input transconductance is achieved by increasing the size of the input pair. Nevertheless, a large differential pair induces more kick-back noise. As such, there is a design trade-off between speed and noise of the double-tail comparator.

3.4 Sampling Network

Sampling of the input voltage on the capacitive DAC can be performed in two different ways: *bottom-plate sampling* and *top-plate sampling*. Using the bottom-plate sampling method, the voltage is sampled on the bottom-plate of the capacitors, the side which is not connected to the comparator. The advantage of the bottom-plate sampling is the charge-injection independent sampling process.

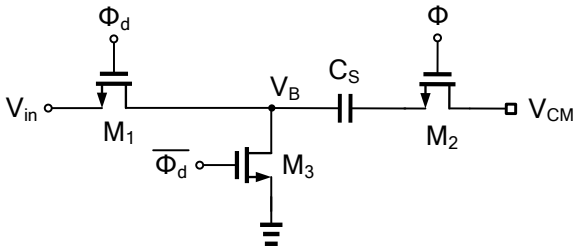


Fig. 3.26 Bottom plate sampling. The advantage of the bottom-plate sampling is the charge-injection independent sampling process.

The configuration is shown Fig. 3.26 and the operation is as follows: In the acquisition mode, M_1 and M_2 are on and M_3 is off. V_{CM} is therefore appears on the top plate of C_s, while the bottom-plate tracks the input voltage V_{in}. In the transition to the hold mode, first Φ goes low, turning M_2 off, and after a small delay Φ_d falls, turning M_1 off and M_3 on. Thus, V_B drops from V_{in} to 0 and hence the charge in V_X is equal to $-V_{\mathrm{in}}$ at the sampling instant. Since M_2 always turns off first , the channel charge of M_1, which is input-dependent, does not introduce any error. Moreover, as the gate-source voltage of M_2 is independent of V_{in}, the channel charge injected by this switch only appears as a constant offset on V_X.

Although bottom-plate sampling yields higher accuracy than top-plate sampling, the complexity of the routing in the layout makes it inappropriate for high-speed applications. However, and regardless of what method of sampling is utilized, using a transistor as the sampling switch exhibits an input-dependent on-resistance, thereby introducing distortion. This issue can be resolved by "bootstrapping," a circuit technique that minimizes the switch on-resistance variation in the presence of large input and output voltage swings [133]. The core idea is shown in Fig. 3.27.(a) where a battery of voltage V_{DD} is inserted between the source and the gate to keep V_{GS} constant (and maximum) during the sampling time. In reality, this can be achieved using a capacitor and a number of switches, as shown in Fig. 3.27.(b). During the tracking phase, S_2 and S_4 go on, putting the precharged C_B on the gate-source of the sampling switch M_s. In the hold mode, S_5 turns on, turning off the sampling switch, while S_1 and S_2 recharge the capacitor to V_{DD} again.

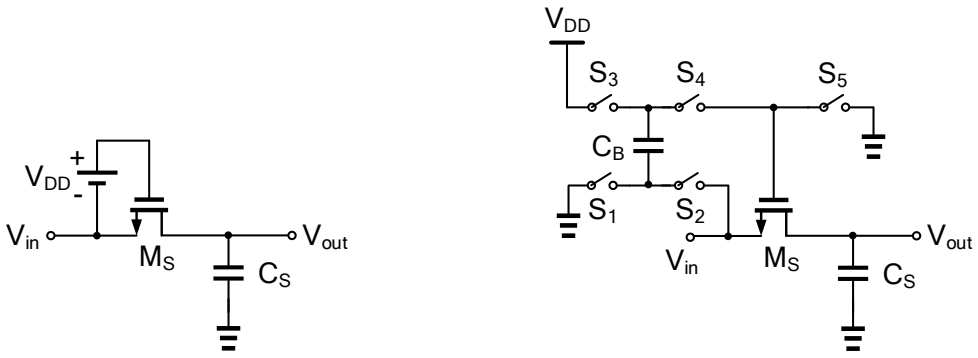


Fig. 3.27 Bootstrapped switch: (a) principal of operation, (b) circuit realization. Bootstrapping minimizes the on-resistance of the sampling switch and makes it input-independent.

The bootstrapped switch guarantees the minimum on-resistance of the sampling switch by forcing its gate-source voltage to be around V_{DD} for all values of the input voltage. The complete bootstrapped switch is shown in Fig. 3.28. As can be seen, there is higher number of transistors than the number of switches in Fig. 3.27.(b). In the following, we explain the function of every transistor:

1. Transistor M_1: this NMOS transistor is simply the replacement of switch S_1.

2. Transistor M_2: this replaces the switch S_2. The gate of this NMOS is bootstrapped to V_G for otherwise it will turn off or at least exhibit a large on-resistance for voltages of V_{in} close to V_{DD}.

3. Transistor M_3: this performs the function of S_3. The gate of this PMOS transistor is also bootstrapped to V_G to avoid it to turning on when V_Y rises above V_{DD}.

4. Transistor M_4 and M_6-M_8: this group of transistors perform the function of switch S_4 all together. M_6 bootstraps the gate of M_4 to node X to avoid a voltage of greater than V_{DD} to appear on its gate-source. However, M_6 would turn off when V_X becomes larger than $V_{\mathrm{DD}} - V_{\mathrm{TH}}$. Therefore, M_7 is added to kick in when this happens and keeps M_4 still ON. The function of transistor M_8 is to turn M_4 off during the tracking mode.

5. Transistors M_5 and M_{5t}: these two transistor together replace the switch S_5. The reason for adding M_{5t} here is to prevent M_5 from experiencing drain-source and drain-gate voltages greater than V_{DD}[2] when V_G rises above V_{DD}.

[2]Any node-to-node voltage of a MOS device greater than the maximum allowable voltage of a specific technology (which is usually $V_{\mathrm{DD}} \pm 10\%$) results in a "stress" on that device, shortening its lifetime.

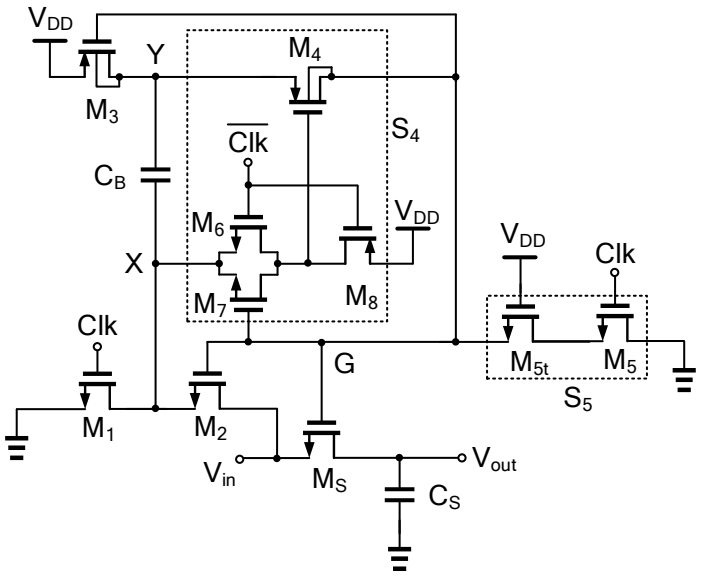


Fig. 3.28 Circuit implementation of the bootstrapped switch.

It should be noted that the since the voltage of node Y can go above V_{DD}, the bulk terminal of transistors M_3 and M_5 should be tied to their sources to avoid the reversed-biased bulk-source diodes to turn on.

The sampling operation via the bootstrapped switch occurs in two phases: the precharge (holding) phase and the tracking phase.

Precharge Phase

During this phase, and when *Clk* is high, the capacitor C_B is precharge to V_{DD} through transistors M_1 and M_3, while transistors M_5 and M_{5t} force the gate of the sampling transistor M_S to ground and push it into OFF state. M_8 also keeps M_4 in OFF region. All other transistors are OFF, as can be seen in Fig. 3.29. The precharge phase starts by the gate of M_S, G, begin discharged to ground with a time constant given by

$$\tau_{1,1} = \left(R_{\mathrm{ON},5} + R_{\mathrm{ON},5\mathrm{t}}\right) C_G, \tag{3.17}$$

where $R_{\mathrm{ON},5}$, $R_{\mathrm{ON},5\mathrm{t}}$ and C_G are the ON resistance of transistors M_5 and the ON resistance of M_{5t} and the total capacitance at node G, respectively. Once G reaches ground, M_3 turns ON, charging C_B to V_{DD} with a time constant of

$$\tau_{1,2} = \left(R_{\mathrm{ON},1} + R_{\mathrm{ON},3}\right) C_B, \tag{3.18}$$

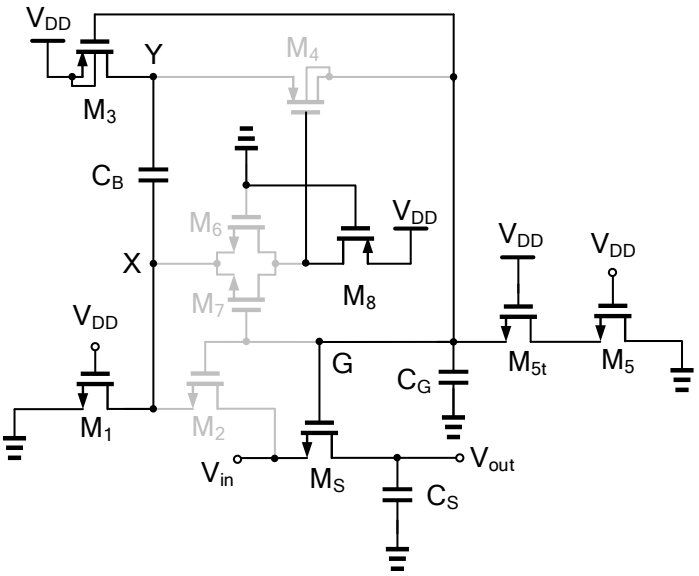


Fig. 3.29 Bootstrapped switch during the precharge phase.

where $R_{\mathrm{ON},1}$ and $R_{\mathrm{ON},3}$ are the ON resistance of transistors M_1 and M_3, respectively. The precharging time constant thus can be viewed as the sum of $\tau_{1,1}$ and $\tau_{1,2}$. Typically, $\tau_{1,1}$ is much smaller than $\tau_{1,2}$, making the precharge time constant $\tau_{\mathrm{precharge}}$ effectively equal to $\tau_{1,2}$.

Tracking Phase

When *Clk* goes low, M_6 is quickly switched on and forces the gate of M_4 to ground (Fig. 3.30.(a)). This happens with a time constant of

$$\tau_{2,1} = R_{\mathrm{ON},6} C_Z, \tag{3.19}$$

where where $R_{\mathrm{ON},6}$ is the ON resistance of transistors M_6 and C_Z is the total capacitance at node Z. Next, and when M_4 is already ON, the top plate of C_B, which was precharged to V_{DD}, appears on the gate of M_S through the ON resistance of M_4. V_G starts to rise and once it reaches the threshold voltage of M_2, this transistor turns ON and connects the bottom plate of C_B to the input voltage V_{in} (Fig. 3.30.(b)). Following this, V_X and V_Y increase by V_{in} with a time constant of

$$\tau_{2,2} \approx \left(R_{\mathrm{ON},2} + R_{\mathrm{ON},4}\right) C_G, \tag{3.20}$$

where $R_{\mathrm{ON},2}$ and $R_{\mathrm{ON},4}$ are the ON resistance of transistors M_2 and M_4, respectively.

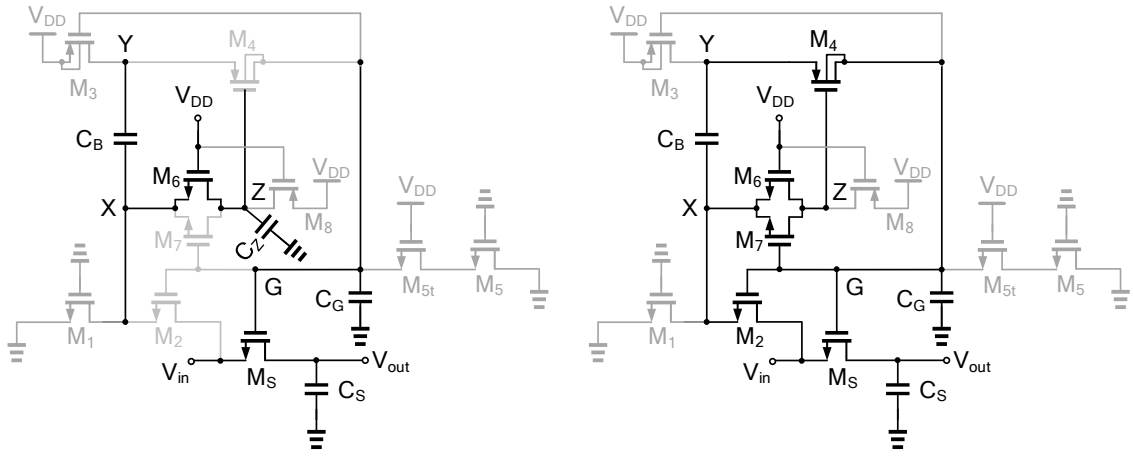


Fig. 3.30 Bootstrapped switch during the tracking phase: (a) node Z discharges to ground, (b) C_B appears on the gate-source of M_S.

A different version of 3.28 is sometimes used where gate of M_3 is driven by a dedicated *charge-pump* circuit taht produces $2V_{\mathrm{DD}}$ [94, 132]. The overhead though is the extra capacitors necessary to build the charge-pump circuit which requires two large capacitors. As for a transmission gate switch, a complementary bootstrapped switch was also proposed, especially for low voltage applications when V_{DD} is close to the threshold voltage of the transistors [134, 135].

Another advantage of bootstrapping is minimizing the harmful effect of input-dependant charge injection, especially for top-plate sampling. This is explained in the following. The charge injection is caused by the charge in the inversion layer of the switch and can be roughly expressed as

$$Q = WLC_{\mathrm{ox}}(V_{\mathrm{GS}} - V_{\mathrm{TH}}), \tag{3.21}$$

where V_{TH} is the threshold voltage of the transistor. When the transistor goes off, this charge is depleted towards both the source and the drain terminals. The distribution mostly depends on the impedance of either side of the transistor. Hence, the voltage jump on the sampling capacitor C_S due to the charge injection once the switch turns off is given by

$$\Delta V = \alpha \frac{WLC_{\mathrm{ox}}(V_{\mathrm{GS}} - V_{\mathrm{TH}})}{C_S}, \tag{3.22}$$

where α is the portion of the charge that is depleted onto the C_S side. As already mentioned, the bootstrapped switch makes V_{GS} constant and independent of the input. To a lesser extent, V_{TH} is also input dependent due to the body effect, especially for high voltage applications. *Bulk-biasing* is a technique to minimize this non-linear effect [136]. The idea is to connect the bulk of the sampling switch to its source during the sampling mode and to the ground during the hold mode. This can be done through some extra switches. A variation of this technique was presented in [137].

3.4.1 Sampling Jitter

Variation in the time at which the sampling switch is opened is known as aperture uncertainty, or jitter, and will result in an error voltage that is directly proportional to the magnitude of the jitter and the input signal slew rate. The sampling jitter is a combination of the external jitter, i.e. the oscillator that generates the sampling clock along with all the proceeding buffers, and the internal ADC aperture jitter itself. The ADC SNR degradation due to jitter can be calculated as

$$\mathrm{SNR}_j[\mathrm{dBFS}] = -20 \cdot \log(2\pi \mathrm{f_{in}} \sigma_{\mathrm{n}}), \tag{3.23}$$

where f_{in} and σ_n are the input frequency and the RMS value of the jitter, respectively. The ADC noise floor (SNR) is then set by the following contributors

1. ADC quantization noise, SNR_q
2. ADC thermal noise (which includes the kT/C sampling noise and the thermal noise of the ADC's constituent blocks), SNR_n
3. Sampling jitter, SNR_j

and can be calculated as

$$\mathrm{SNR_{ADC}} = -20 \cdot \sqrt{\left(10^{-\mathrm{SNR}_q/20}\right)^2 + \left(10^{-\mathrm{SNR}_n/20}\right)^2 + \left(10^{-\mathrm{SNR}_j/20}\right)^2}, \tag{3.24}$$

where SNR_q and SNR_j are given by (2.5) and (3.23), respectively.

Chapter 4

Mismatch Calibration Techniques

In this chapter, we review some of the most important mismatch calibration techniques proposed in the literature. In particular, we focus in on the *mismatch* resulting from the inherent random variation in capacitors that builds up in the capacitive DAC in a SAR ADC. All the mismatch calibration methods discussed here can be used to compensate for this effect, although some methods can be used for other types of mismatch in the A/D converter too.

A mismatch calibration process typically consists of two phases: a *detection* phase, where the error (i.e. capacitive mismatch) is determined in the form of an electrical quantity (e.g. voltage) in the analog domain or as a binary number in the digital domain, and a *correction* phase, where this detected error is used to correct the erroneous output.

4.1 Detection

The mismatch detection process can occur in the background without interrupting the normal conversion of the ADC, or in the foreground upon the start-up of the chip and before the ADC conversion kicks off. Before we proceed and for the sake of clarification for the rest of this article, we define the term bit weight as the numerical weight of a particular bit of the binary output of the ADC, which is basically the (normalized) value of the capacitor(s) in the DAC that would contribute to that specific bit during the conversion process. Therefore, the output of an N-bit ADC can be represented by

$$y = \sum_{i=0}^{N-1} b_i w_i, \tag{4.1}$$

where $B = \{b_{N-1}, b_{N-2}, ..., b_1, b_0\}$ is the binary output and $W = \{w_{N-1}, w_{N-2}, ..., w_1, w_0\}$ is the bit weight vector.

Background

Perturbation-based calibration (also known as *dithering*) is a common method for error detection in an ADC. This technique is widely used to calibrate non-idealities of pipeline ADCs and can also be used with the SAR architecture to calibrate the mismatch of the CAP-DAC. This approach is based on injecting discrete-time single-bit zero-mean *pseudorandom noise* (PN) samples, Δ_{PN}, into the ADC along with the analog input signal V_{in}, and using this as a way to estimate the actual bit weights of the capacitive DAC. In the ideal case where the ADC is perfectly *linear*, i.e. the bit weights that are used to reconstruct the output precisely match the actual bit weights in the DAC, superposition can be used to express the ADC output as

$$D_{\text{out}} = Q(V_{\text{in}} + \Delta_{\text{PN}}) = Q(V_{\text{in}}) + Q(\Delta_{\text{PN}}) = D_{\text{in}} + D_{\Delta_{\text{PN}}}, \tag{4.2}$$

where $D_x = Q(V_x)$ denotes the quantized representation of voltage V_x. The second equal sign in Eq. (4.2) holds if D_{in} is completely independent from (uncorrelated with) $D_{\Delta_{\text{PN}}}$. On the other hand, when the bit weights of the DAC are mismatched with respect to their ideal values, the system is not linear and (4.2) no longer holds. In this case, the *correlation* between D_{out} and $D_{\Delta_{\text{PN}}}$ can be exploited to extract the mismatched bit weights, as shown in Fig. 4.1. The PN signal Δ_{PN} is usually injected via an extra capacitor added to the capacitive DAC [95] or by using the capacitors in the DAC itself [138, 139].

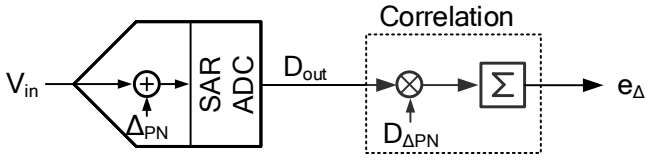


Fig. 4.1 Core idea of correlation-based calibration. The correlation of D_{out} and $D_{\Delta_{\text{PN}}}$ is zero only if the ADC is linear.

Reference [95] proposed a *dual-conversion* approach where each sample is converted twice, once with a positive sign of the PN signal and once with the negative sign, as shown in Fig. 4.2. In this figure, $D_{\text{out}+}$ and $D_{\text{out}-}$ are the ADC outputs corresponding to PN signals $\Delta_{\text{PN}+}$ and $\Delta_{\text{PN}-}$, respectively. The difference between the outputs of the two conversions is then used for the bit weight extraction. This approach has an advantage in that the unknown input signal is subtracted from the calibration path, thus making it unnecessary to accumulate

a large number of data. However, each conversion needs to be performed twice, hence halving the maximum achievable conversion rate of the ADC.

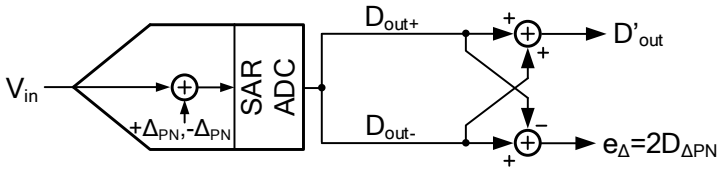


Fig. 4.2 Block diagram of the perturbation-based calibration with dual conversion.

The bit weight extraction can be performed in various ways. One way is to use an adaptive loop. As shown in Fig. 4.3, a least-mean-square (LMS) weight updater can be used to adjust the reconstruction bit weights W, until the error term $e_\Delta = E[D_{\text{out}} \cdot D_{\Delta PN}]$, where $E\{\cdot\}$ is the averaging operator, becomes zero, indicating that W matches the actual bit weights of the DAC. In order to work out all the N bit weights $W = \{w_{N-1}, w_{N-2}, ..., w_0\}$, where N is the resolution of the ADC, the LMS can update all of the bit weights at the same time [97] as

$$w_i(k+1) = w_i(k) - \mu e_\Delta(k), \quad (i = 0, 1, ..N-1). \tag{4.3}$$

In order to reduce the conversion time of the LMS procedure, [140] used N different PN signals to determine the bit weights separately yet simultaneously, as shown in Fig. 4.4.

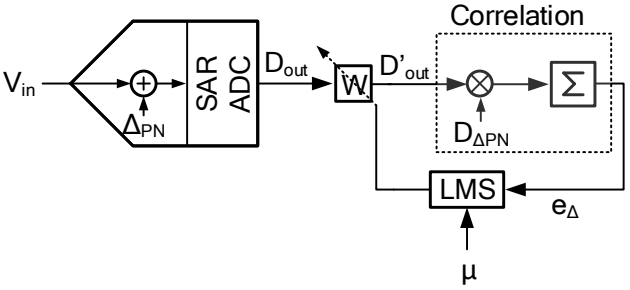


Fig. 4.3 Bit-weight extraction of the perturbation-based calibration through an LMS algorithm.

Bit weight extraction can also be done directly without using an LMS approach [138, 139]. To elaborate, let us denote $C_0 \sim C_{N-1}$ as the capacitors of the DAC in the SAR ADC and $w_0 \sim w_{N-1}$ as their (normalized) corresponding weights. Let us also assume that capacitor C_j is used to add the PN signal through a $\{1, -1\}$ pseudorandom sequence K along with the

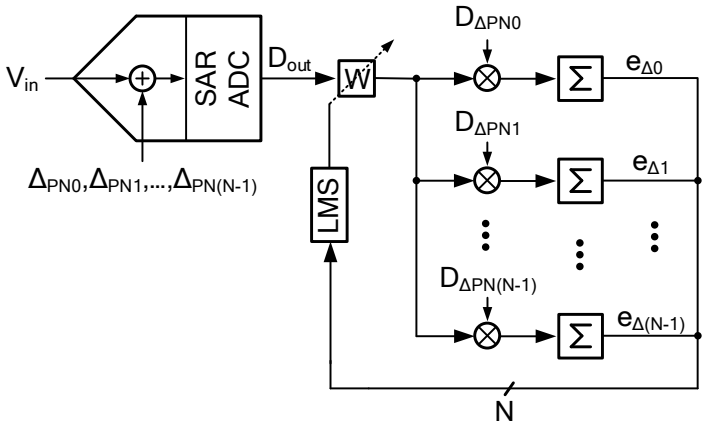


Fig. 4.4 Another implementation of Fig. 4.3. Here N different dithering signals are used.

input signal V_{in}. The operation of the SAR ADC is then described by

$$V_{\text{in}} + Kw_j = \sum_{i=0, i\neq j}^{N-1} b_i w_i + V_q, \tag{4.4}$$

where V_q is the quantization error. The digital representation of (4.4) is

$$D_{\text{in}} + Kw_{Dj} = \left(\sum_{i=0, i\neq j}^{N-1} b_i w_{Di} \right) + D_q. \tag{4.5}$$

By correlating (4.5) with K, we have

$$\sum_{i=0, i\neq j}^{N-1} \overline{Kb_i} w_{Di} = w_{Dj} + \overline{KD_{\text{in}}} - \overline{KD_q} = w_{Dj} + e_j \tag{4.6}$$

where $e_j = \overline{KD_{\text{in}}} - \overline{KD_q}$ is an error term. By running this procedure for the m MSB capacitors, we can get m equations similar to (4.6), which can then be written in a matrix form as

$$A \times \begin{bmatrix} w_{D(N-1)} \\ w_{D(N-2)} \\ \vdots \\ w_{D(N-m+1)} \end{bmatrix} = C - \begin{bmatrix} e_{N-1} \\ e_{N-2} \\ \vdots \\ e_{N-m+1} \end{bmatrix}, \tag{4.7}$$

where A and C are matrices which depend on the correlation between K's, b_is and the bit weights of the LSB capacitors. With a sufficiently long PN sequence, the error terms e_j's

converge to zero. Therefore, (4.7) can be solved to determine the mismatched bit weights $\left\{w_{D(N-1)}, w_{D(N-2)}, \cdots, w_{D(N-m+1)}\right\}$.

Ref. [141] proposed a *split ADC* approach where the capacitor array of the SAR ADC was split into two identical banks and performed two SAR conversions at the same time, as shown in Fig. 4.5. The average of the outputs of the two conversions is taken as the final output, while the difference could be used to calibrate the bit weights, e.g. via an adaptive loop, as shown in the figure. It is worth noting that the difference between the outputs of the two conversions in Fig. 4.5 could be zero even when the two paths still exhibit a mismatch. To circumvent this, [141] also employed a "shuffling" technique for the capacitive array where different unit capacitors were used for different bits in every conversion in a randomized manner. This guarantees the independence of the two conversions and can force e_Δ to zero only if the mismatch is zero. Nonetheless, this approach leads to a very complex connection web between the capacitors in the layout, thereby jeopardizing the conversion speed of the ADC.

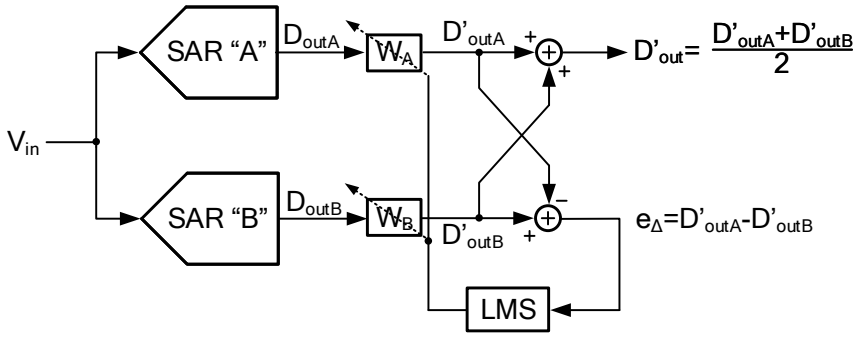


Fig. 4.5 Block diagram of the split ADC calibration. e_Δ is used to determine the ADC bit weights.

In [142], DNL error detection was used to trim the capacitors of the capacitor array with the aid of redundancy. To understand the approach that was used in [142], let us assume that the input-output transfer curve of a SAR ADC with redundancy is as shown in Fig. 4.6. It can be seen that the ADC has a DNL error Δ at the MSB bit which occurs at the transition from 1000...00 to 0111...11 and vice versa. Owing to the redundancy, there is more than one way of representing a single input voltage with binary codes, only if the DAC is mismatch-free. For the specific case of Fig. 4.6, if the MSB capacitor has mismatch, codes A and B would result in different DAC voltages V_A and V_B. This discrepancy in the output voltage can be used to tune Δ back to zero and calibrate the mismatch of the MSB capacitor as follows (Fig. 4.7): if code A (B) is observed during the normal conversion time of the ADC, an extra cycle is added, the switching of the DAC is updated to one associated with code B (A) and

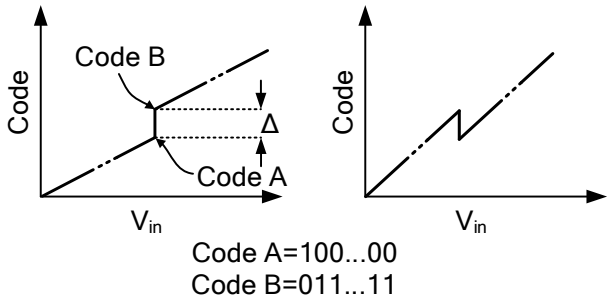


Fig. 4.6 Transfer characteristic of a SAR ADC with mismatch at MSB capacitor.

an additional comparison is performed. If the result of the comparison before and after the switching between code A and B is not the same, it means there is mismatch in the MSB bit that is then eliminated by trimming. The same procedure can be followed to calibrate the mismatch of the other capacitors. We make this clear by considering an 8-bit SAR ADC as an example, whose (normalized) bit weights are $W = \{128, 64, 32, 32, 16, 8, 4, 2, 1\}$. Note that the 4^{th} bit is *redundant*, and its weight is the same as the 3^{rd} bit. Therefore, the two 9-bit codes $B_1 = \{1,0,0,0,x,x,x,x,x\}$ and $B_2 = \{0,1,1,1,x,x,x,x,x\}$ both represent the same digital output if the ADC is linear, i.e. there is no mismatch in the MSB capacitor. This allows the mismatch of the MSB capacitors before the redundant capacitor to be detected as follows. Whenever code B_1 appears at the output of the ADC during a normal conversion, the ADC undergoes an extra 10-th cycle, where the switching of the CAP-DAC is updated to one corresponding to code B_2, followed by an extra comparison. Similarly, when the code B_2 is observed, the CAP-DAC switching is updated to one corresponding to B_1 during the extra comparison cycle. The result of the extra comparison can thus be used to detect the sign of the MSB capacitor mismatch for an appropriate correction method. Similar procedure can be followed to detect the mismatch of the MSB-1 and MSB-2 capacitors. For instance, to detect the mismatch of the MSB-2 capacitor, the extra calibration cycle is activated when the codes $B_3 = \{0,1,0,1,x,x,x,x,x\}$ or $B_4 = \{0,1,1,0,x,x,x,x,x\}$ are observed at the ADC output.

Foreground

Foreground mismatch calibration can be performed by estimating the DNL error of all possible output codes using a known input, e.g. a sinusoidal signal. The DNL estimation can be done in the same way as it is done for the DNL measurement of an ADC, e.g. using a histogram-based method in the time domain or using frequency-based methods [143–146]. Ref. [147] used an averaging method to estimate the DNL as follows. Let us denote $x[k]$ as

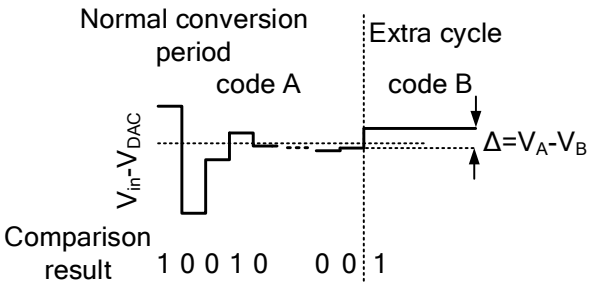


Fig. 4.7 Mismatch calibration based on Fig. 4.6.

the ideal analog sampled signal at the quantizer (ADC) input, with k being the sample index. We denote the quantized number associated with $x[k]$ by $B[k] = Q_N\{x[k]\}$, where the operator $Q_N\{\cdot\}$ is the N-bit quantization performed by the ADC. The output of the ADC, $y[k]$, can then be worked out by using (4.1). Thus, the error associated with $B[k]$ is

$$e[k] = y[k] - x[k]. \tag{4.8}$$

This error includes both the ADC quantization error and nonlinearity errors. The goal is to minimize $e[k]$ through updated bit weights $\widetilde{W}$ that would result in a corrected output $\widetilde{y}[k]$, as shown in Fig. 4.8. It was proved in [148] that this error is minimized when for a specific output code B, $\widetilde{y}[k]$ is set to the average of all sampled inputs $x[k]$ that are mapped to code $\widetilde{B}$. The reconstruction filter in Fig. 4.8 estimates the sampled input voltage and a sinusoidal waveform is employed as input signal during calibration.

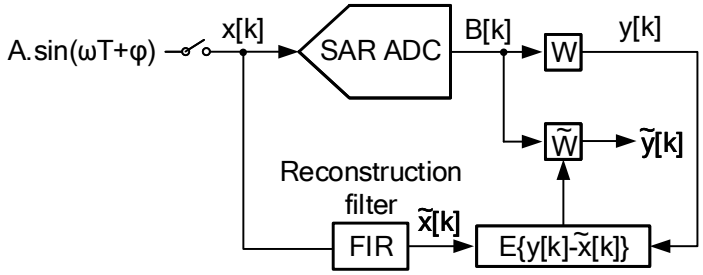


Fig. 4.8 A foreground mismatch calibration based on estimation of the DNL error.

One problem associated with this approach is that the computed DNL includes all types of static and potentially frequency-dependent non-linearities of the ADC, with the CAP-DAC mismatch being only one of them. As a consequence, this error detection mechanism only proves to be accurate for a particular frequency. *Dynamic detection* has been proposed to overcome this limitation [147] where the estimated DNL is not only a function of the

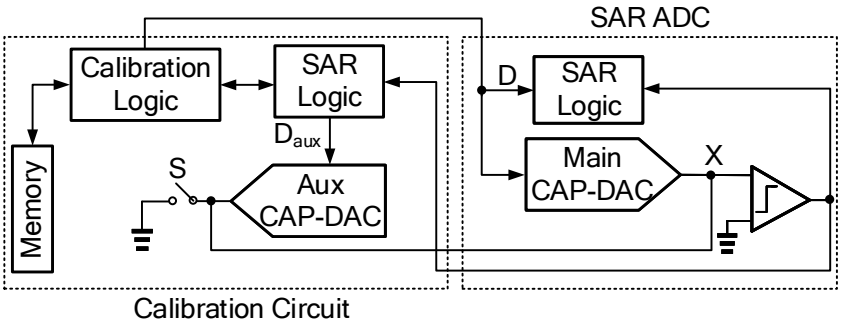


Fig. 4.9 Block diagram of self-calibration using an auxiliary CAP-DAC.

current output code, but also depends on the previous output code(s) or some other additional information, such as the slope of the input signal, as proposed in [149]. In [150], apart from the current output of the ADC, k previous outputs were also used to estimate the DNL.

Self-calibration is another method of foreground mismatch calibration of SAR ADCs, and is based on the DNL error estimation of the individual capacitors using a DAC and can only be applied to binary-weighted capacitive DACs. This technique can be realized in two ways: either by utilizing an *auxiliary* CAP-DAC, as shown in Fig. 4.9, or by using the same DAC of the ADC itself (*main CAP-DAC*). The former can be exploited following a *bottom-up* or a *top-down* approach. In a bottom-up approach [151], the direction of the calibration is from the LSB towards the MSB. Specifically, in the case of the N-bit capacitor array shown in Fig. 4.10, binary-scaling implies

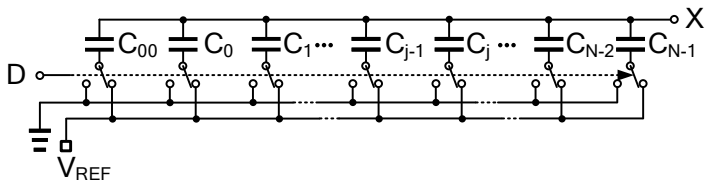


Fig. 4.10 Circuit schematic of an N-bit CAP-DAC.

$$C_j = 2 \cdot C_{j-1} \quad (j = 1, 2, ..N-1). \tag{4.9}$$

Capacitor $C_{00} = C_0$ is added to make the total capacitance a power of two multiples of the LSB capacitor (unit capacitor) C_0, i.e.

$$C_{00} + \sum_{i=0}^{N-1} C_i = 2^N C_0. \tag{4.10}$$

The self-calibration method makes use of the fact that for an ideal binary-weighted capacitor array with no mismatch, the value of any individual capacitor in the array is equal to the sum of all of the lower significant bit capacitors, i.e.

$$C_j = C_{00} + \sum_{i=0}^{j-1} C_i \quad (j = 1,2,..N-1) \tag{4.11}$$

Equivalently, if capacitor C_j in the array of Fig. 4.10 is mismatched with its ideal value and assuming that the LSB capacitors are ideal, then (4.11) is not exact. In other words, if ΔC_j is the deviation of capacitor C_j from its ideal value, then the term

$$C_j - \left(C_{00} + \sum_{i=0}^{j-1} C_i \right) = \Delta C_j \tag{4.12}$$

is its corresponding mismatch. Now assume it is desired to estimate the mismatch of capacitors C_j to $C_{(N-1)}$, assuming that capacitors C_0 to $C_{(j-1)}$ are all mismatch-free[1]. The self-calibration procedure occurs in a multi-phase process as follows:

Phase I: The reset switch S in Fig. 4.9 closes to bring node X to ground (Fig. 4.11(a)). At the same time, the calibration logic sets D to

$$\begin{aligned} D &= \left\{ d_{00}, d_0, d_1, ..., d_{(j-1)}, d_j, ..., d_{(N-2)}, d_{(N-1)} \right\} \\ &= \{1,1,1,...,1,0,...,0,0\} \end{aligned} \tag{4.13}$$

It also sets the binary input signal driving the auxiliary CAP-DAC, D_{aux}, to the mid-code $\{0,0,...,0,1\}$.

Phase II: The reset switch opens and the calibration logic changes D to (Fig. 4.11(b))

$$\begin{aligned} D &= \left\{ d_{00}, d_0, d_1, ..., d_{(j-1)}, d_j, ..., d_{(N-2)}, d_{(N-1)} \right\} \\ &= \{0,0,0,...,0,1,...,0,0\}. \end{aligned} \tag{4.14}$$

Q_X, the charge on node X, thus becomes

$$Q_X = \alpha V_{\text{REF}} \cdot \left[C_j - \left(C_{00} + \sum_{i=0}^{j-1} C_i \right) \right] = \alpha V_{\text{REF}} \cdot \Delta C_j, \tag{4.15}$$

where α is a constant factor that represents the charge sharing between the total capacitance of the main CAP-DAC and that of the auxiliary CAP-DAC, that is $\alpha = C_{\text{DAC,main}}/(C_{\text{DAC,main}} +$

[1]This assumption does not pose any restrictions as the calibration can start from the LSB capacitor.

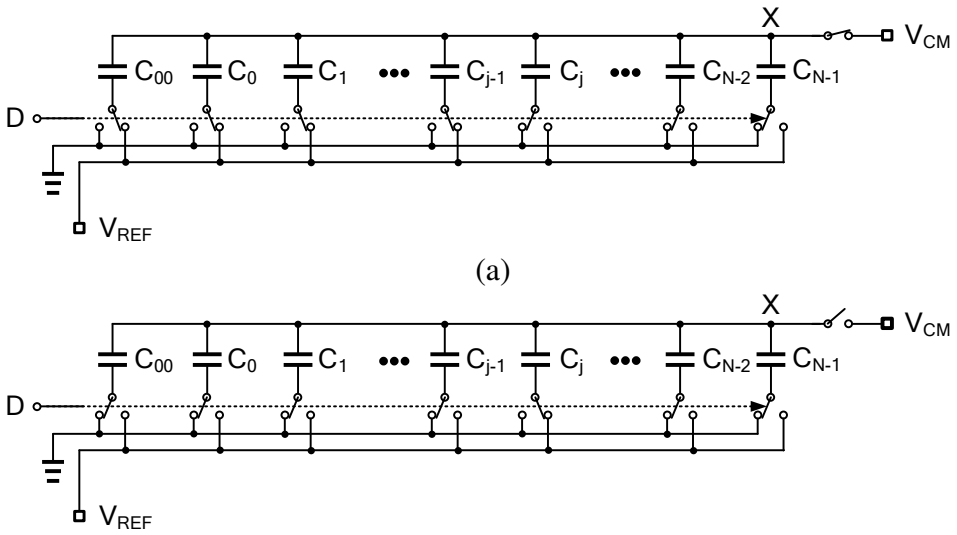


Fig. 4.11 Bottom-up self-calibration: (a) reset phase, (b) mismatch formation phase.

$C_{\mathrm{DAC,aux}}$), where $C_{\mathrm{DAC,main}}$ and $C_{\mathrm{DAC,aux}}$ are the total capacitance of the main DAC and the auxiliary DAC, respectively. This gives rise to a voltage on node X equal to

$$V_X = \frac{\alpha V_{\mathrm{REF}} \cdot \Delta C_j}{C_X}, \tag{4.16}$$

where C_X is the total capacitance at node X.

Phase III: The calibration logic processes the output of the comparator. A negative output implies that $\Delta C_j < 0$. The SAR logic of the calibration circuit then progressively increments D_{aux} in a successive-approximation manner, making the auxiliary CAP-DAC to inject a charge into node X that is proportional to D_{aux}. This sequence continues until the output of the comparator becomes positive. The binary number D_{aux} is then registered as the mismatch of C_j (normalized to the unit capacitor C_0) and is stored in the memory. If the comparator output is positive at the start of this sequence, then $\Delta C_j > 0$. The calibration logic then decrements D_{aux} until the comparator output becomes negative, and again D_{aux} is stored as the mismatch of C_j. It is worth noting that, in order to mitigate the detrimental impact of the thermal noise (e.g. kT/C noise), averaging should be performed by repeating the procedure from phase I to III a sufficient number of times, and averaging the obtained results. The next capacitor to be calibrated is $C_{(j+1)}$. Phases I to III are repeated in the same

way. At the end of phase II, V_X will be equal to

$$V_X = \alpha V_{\text{REF}} \cdot \frac{C_{(j+1)} - \left(C_{00} + \Sigma_{i=0}^{j} C_i\right)}{C_X} = \alpha V_{\text{REF}} \cdot \frac{\Delta C_{(j+1)} + \Delta C_j}{C_X}. \tag{4.17}$$

As can be seen, this time V_X contains the mismatch error for both $C_{(j+1)}$ and C_j. Therefore, at the end of phase III, D_{aux} represents the sum of the mismatches of C_j and $C_{(j+1)}$. By denoting by e_j the estimated mismatch error of capacitor C_j, the mismatch error of $C_{(j+1)}$, which is stored in the memory, can be expressed as

$$e_{(j+1)} = D_{\text{aux}} - e_j. \tag{4.18}$$

Similarly, for all of the subsequent capacitors, the mismatch error would be equal to the value of D_{aux} at the end of phase III minus the sum of the mismatch errors of all of the previously calibrated capacitors. In other words, the binary number representing the mismatch of capacitor C_L is

$$e_L = D_{\text{aux}} - \sum_{i=j}^{L-1} e_i. \tag{4.19}$$

One problem associated with the bottom-up self-calibration approach is the residual voltage at node X at the end of phase III, due to the limited resolution of the calibration DAC (i.e. a quantization error). This quantization error occurs during the mismatch estimation of a particular capacitor and gets doubled for every subsequent capacitor under calibration, resulting in an exponential growth of the error. The accumulated error contributed by all of the capacitors that undergo calibration could then be large, unless a high resolution auxiliary DAC is used. This constraints the number of capacitors whose mismatch is estimated by this approach, using a finite resolution auxiliary DAC.

The *top-down* approach [152–156] is similar to the bottom-up approach. The main difference is the direction of calibration, that is from the MSB capacitor to the LSB one. Moreover, it assumes that the sum of the mismatch errors of all of the capacitors is zero, i.e.

$$\Delta C_{00} + \sum_{i=0}^{N-1} \Delta C_i = 0. \tag{4.20}$$

In general, though, the assumption of (4.20) is not necessarily correct. In other words

$$\Delta C_{00} + \sum_{i=0}^{N-1} \Delta C_i = \Delta S, \tag{4.21}$$

where ΔS is non-zero. This is however not problematic. To explain why, note that the estimated mismatches of the capacitors based on the self-calibration method are all in units of the LSB capacitor C_{00} (i.e. they are normalized to the value of C_{00}). Now, if we define a new value for the LSB capacitor as

$$C'_{00} = C_{00} + \frac{\Delta S}{2^N}, \tag{4.22}$$

then we can write

$$C_j = 2^j \cdot C'_{00} - \frac{\Delta S}{2^{N-j}} + \Delta C_j. \tag{4.23}$$

$$e_L = D_{\text{aux}} - \sum_{i=j}^{L-1} e_i. \tag{4.24}$$

As can be seen in (4.23), this new definition of the LSB capacitance adds an extra term of $-\Delta S/2^{N-j}$ to the capacitor mismatch. We now define a new value for the mismatch of the capacitor based on this redefinition of the unit capacitance as

$$\Delta C'_j = -\frac{\Delta S}{2^{N-j}} + \Delta C_j, \tag{4.25}$$

which satisfies the assumption of (4.20), i.e.

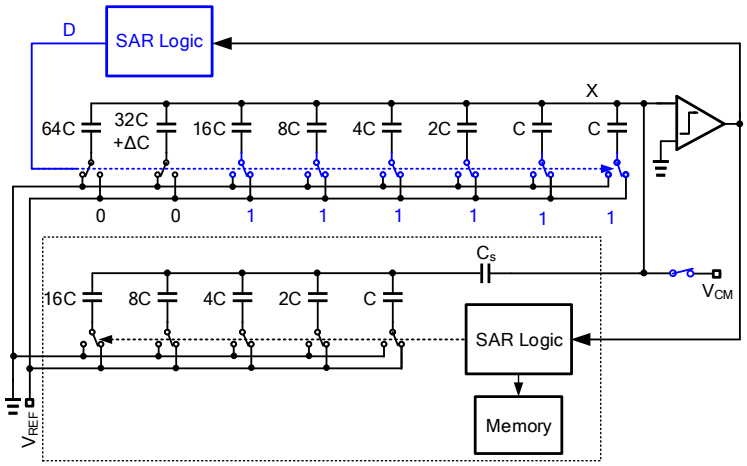
$$\Delta C'_{00} + \sum_{i=0}^{N-1} \Delta C'_i \tag{4.26}$$

$$= \left(\Delta C_{00} + \sum_{i=0}^{N-1} \Delta C_i \right) + \left(\sum_{i=0}^{N-1} -\frac{\Delta S}{2^{N-i}} \right) \tag{4.27}$$

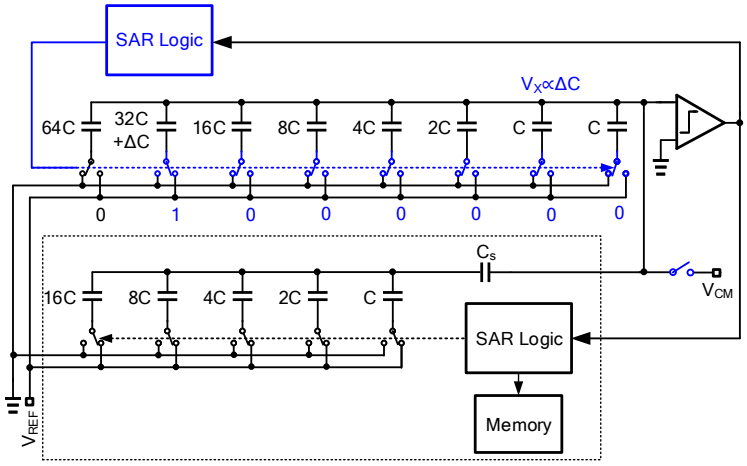
$$= \Delta S - \Delta S = 0. \tag{4.28}$$

In practice, this essentially means that the estimated capacitor mismatches will eventually be expressed in units of the new LSB capacitance C'_{00}.

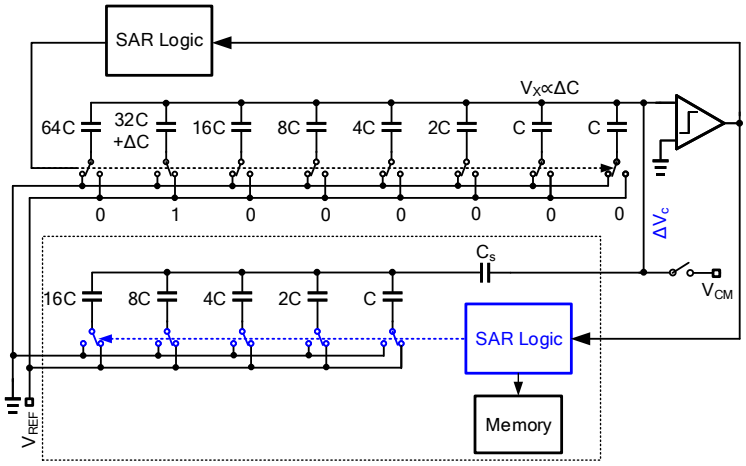
Similarly to the bottom-up approach, the detection process occurs in a three-phase procedure which starts from the MSB capacitor (C_{N-1}) and proceeds as follows:



(a)



(b)



(c)

Fig. 4.12 Bottom-up approach to calibrate the mismatch of capacitor $32C$: (a) reset phase, where X is precharged to V_{CM} and $D = \{00111111\}$, (b) error generation phase, where the reset switch is open and D changes to $\{01000000\}$ (c) error correction phase, where the auxiliary DAC is progressively increased until V_X becomes zero and comparator output changes .

Phase I: With reference to Fig. 4.9, the reset switch S sets node V_X to ground and the calibration logic sets D to

$$\begin{aligned} D &= \left\{ d_{00}, d_0, d_1, ..., d_{(N-2)}, d_{(N-1)} \right\} \\ &= \{1,1,1,...,1,0\}, \end{aligned} \tag{4.29}$$

and D_{aux} to $\{0,0,...,0,1\}$.

Phase II: S opens and the calibration logic sets D to

$$\begin{aligned} D &= \left\{ d_{00}, d_0, d_1, ..., d_{(N-2)}, d_{(N-1)} \right\} \\ &= \{0,0,0,...,0,1\}. \end{aligned} \tag{4.30}$$

Charge redistribution occurs, and charge Q_X becomes

$$Q_X = \alpha V_{\text{REF}} \cdot \left[C_{(N-1)} - \left(C_{00} + \sum_{i=0}^{N-2} C_i \right) \right]. \tag{4.31}$$

From (4.20), it follows that

$$Q_X = \alpha V_{\text{REF}} \cdot 2\Delta C_{(N-1)}, \tag{4.32}$$

which gives rise to

$$V_X = \frac{\alpha V_{\text{REF}} \cdot 2\Delta C_{(N-1)}}{C_X}. \tag{4.33}$$

Phase III: Depending on the output of the comparator, the calibration logic increments or decrements D_{aux} until the comparator output changes. According to (4.33), D_{aux} is therefore equivalent to twice the mismatch error of $C_{(N-1)}$, and it is stored in memory as $e_{(N-1)}$. The mismatch of the next capacitor (the MSB-1 capacitor) is detected in the same way. Following phase I and II and the charge distribution at the end of phase II, Q_X becomes

$$Q_X = \alpha V_{\text{REF}} \cdot \left[C_{(N-2)} - \left(C_{00} + \sum_{i=0}^{N-3} C_i \right) \right] \tag{4.34}$$

$$= \alpha V_{\text{REF}} \cdot \left(2\Delta C_{(N-1)} + \Delta C_{(N-2)} \right). \tag{4.35}$$

This follows

$$V_X = \alpha V_{\text{REF}} \cdot \frac{2\Delta C_{(N-2)} + \Delta C_{(N-1)}}{C_X}, \tag{4.36}$$

resulting in

$$e_{(N-2)} = \frac{1}{2}\left(D_{\text{aux}} - e_{(N-1)}\right). \tag{4.37}$$

Following the same methodology, it can be shown that the mismatch error e_L of the L-th capacitor is equal to

$$e_L = \frac{1}{2}\left(D_{\text{aux}} - \sum_{i=L+1}^{N-1} e_i\right), \tag{4.38}$$

where D_{aux} is the output of the SAR logic of the calibration circuit at the end of phase III. As evident from (4.38), the accumulated error of all previous capacitors *halves* every time when it appears in the estimated mismatch error equation of the capacitor under calibration. Therefore, the *top-down* approach does not present the issue of the exponentially growing quantization error as severe as in the *bottom-up* approach.

As previously mentioned, self-calibration method can also be realized by using the main CAP-DAC of the SAR ADC as the calibration DAC [101, 157]. This is a more hardware-efficient solution. However, the accuracy of the mismatch detection is limited by the resolution of the SAR ADC itself, i.e. one LSB. Therefore, the problem of accumulation of the quantization error is even more severe here. For this reason, this approach is only used to detect the mismatch of a few MSB capacitors, and it is only effective when this mismatch is larger than 1 LSB. This type of deterministic self-calibration, which uses the main CAP-DAC, is indeed what is employed in this work, and it operates jointly with the proposed stochastic quantization.

4.2 Correction

4.2.1 Analog

One of the most straightforward approaches of correcting a capacitor value is trimming [69]. This is performed by progressively adding or removing smaller capacitors to the capacitor which is being calibrated, until the total capacitance matches the desired value. Determining when the desired value has been reached, and thus when trimming should be ended can be accomplished in different ways. In [158], a *reference capacitor* (C_{REF}) is used and the first

calibrated capacitor C_0 is trimmed until its value becomes equal to C_{REF}. Next, C_{REF} plus the calibrated C_0 are used as the new reference capacitor for the second capacitor to be calibrated. This approach continues until all of the capacitors within the binary-weighted capacitor array are calibrated.

When a self-calibration method is used, the calibration DAC itself is normally used to correct the detected mismatches, and this is performed during the normal conversion of the ADC as follows. Let us denote the estimated mismatch error associated to capacitor C_i with e_i ($L \leq i \leq N-1$, with L being the number of LSB capacitors assumed to be ideal, and whose errors are not estimated during the detection process), where e_i is a binary number stored in the memory. If, at some point during the normal operation of the ADC, the binary code driving the main CAP-DAC is the code $D = \left\{ d_0, d_1, ..., d_{(N-2)}, d_{(N-1)} \right\}$, the calibration logic outputs the code

$$D_{\text{aux}} = \sum_{i=L}^{N-1} d_i e_i \tag{4.39}$$

to the auxiliary CAP-DAC. This causes a charge redistribution to occur, correcting the mismatch of the capacitors which have contributed charge to the output of the CAP-DAC up to this point of the conversion. The arithmetic operation of (4.39) can be implemented in various ways [151], one of which is to store the individual capacitor errors (e_L to $e_{(N-1)}$) along with the permutation of the summations of these errors (e.g. $e_L + e_{L+2} + e_{(N-1)}$). This would require a memory of length 2^{N-L}. However, no adder is needed to perform the sum operation in (4.39). Another way is to only store the individual capacitor errors e_L to $e_{(N-1)}$ in the memory, and perform the summation of (4.39) during the conversion. This, however, requires an adder which needs to operate at least N times faster than the sampling rate of the SAR ADC, which appears rather unpractical for high-speed ADCs. The third method is to employ $N-L$ different auxiliary CAP-DACs, one for the error correction (detection) of each capacitor. In such way, neither a memory nor an adder is required. However, the power consumption of the calibration CAP-DAC increases by a factor $N-L$. Moreover, the realization of the detection process should also be modified accordingly.

4.2.2 Digital

When no extra DAC is used to inject a residue charge during the conversion (and in the analog domain), the correction must be performed in the digital domain. This is basically done by using the updated bit weights $\widetilde{W} = \{\widetilde{w}_0, \widetilde{w}_1, ..., \widetilde{w}_{N-1}\}$ containing the mismatch errors of the

capacitors and perform the arithmetic sum of

$$\widetilde{D}_{\text{out}} = \sum_{i=0}^{N-1} D_{\text{out},i}\widetilde{w}_i, \tag{4.40}$$

where D_{out} and $\widetilde{D}_{\text{out}}$ are the output of the ADC before and after correction, respectively. Since the precision of the correction bit weights $\widetilde{w}_0 \sim \widetilde{w}_{N-1}$ can be higher than that of the ADC, the output of (4.40) needs to be *truncated* to the resolution of the ADC. This truncation limits the accuracy of the DNL improvement of the digital correction to 1 LSB. In other words, after the digital correction, the DNL values smaller than 1 LSB will remain unchanged.

Chapter 5

Stochastic Mismatch Calibration

In this chapter, we propose a new mismatch calibration method that we call *stochastic self-calibration*. The detection of the mismatch error occurs in the analog domain, while the correction is carried out in the digital domain. This calibration method is a foreground approach, which should be activated at the start-up of the ADC (chip) and can be switched off thereafter.

This mismatch calibration method is a combination of the self-calibration approach and a stochastic approach that makes use of a noisy comparator to estimate a residue voltage [159]. The idea of stochastic quantization to estimate the mismatch of the CAP-DAC and/or to reduce the quatization noise has been already used in the literature [160, 161]. Our proposed method, however, differs from those in the following ways:

1. The stochastic approach is only employed when the residue voltage is lower than 1 LSB. Otherwise, the calibration starts with a deterministic self-calibration until the residue voltage narrows to below 1 LSB, at which point the stochastic quantization is commenced. The combination of these two calibration approaches substantially decreases the calibration time when the mismatch is (much) larger than the LSB, compared to when only the stochastic approach is adopted.

2. As explained later, in order to estimate the residue voltage using a noisy comparator, the knowledge of the input-referred comparator noise power is required. In the previous publications, this value was either measured off-chip post-fabrication, or assumed equal to the simulated value. Here, we propose a new technique to compute this entirely on-chip.

Thanks to the stochastic nature of the proposed technique, a mismatch estimation with a precision of better than 1 LSB can be achieved. Moreover, it should be mentioned that this method can only be applied to binary-weighted capacitive arrays and will not work with sub-radix DACs.

To explain the stochastic self-calibration approach, we first need to have a short introduction on stochastic processes with a Gaussian distribution, which is related to the proposed method.

5.1 Gaussian Error Function

A random variable X with a (Gaussian) normal distribution has a probability density function (PDF) of the form

$$f_X(x) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left[-\frac{(x-\mu)^2}{2\sigma^2}\right], \tag{5.1}$$

where μ is the mean and σ is the standard deviation of the variable X. Given a real number $x_a \in \mathbb{R}$, the probability that the random variable $X \sim \mathcal{N}(\mu, \sigma^2)$ is less than or equal to x_a is given by

$$P_r\{X \leq x_a\} = \Phi(x_a) = \int_{-\infty}^{x_a} f(x)dx = \int_{-\infty}^{x_a} \frac{1}{\sqrt{2\pi}\sigma} \exp\left[-\frac{(x-\mu)^2}{2\sigma^2}\right] dx, \tag{5.2}$$

$\Phi(x)$ is called the *cumulative distribution function* (CDF).

By defining the *error function*, $\mathrm{erf}(x)$, as

$$\mathrm{erf}(x) = \frac{1}{\sqrt{2\pi}} \int_0^x \exp\left(-y^2/2\right) dy, \tag{5.3}$$

(5.2) can be rewritten as

$$\Phi(x_a) = \frac{1}{2}\left[1 + \mathrm{erf}\left(\frac{x_a+\mu}{\sigma}\right)\right]. \tag{5.4}$$

Now, if $\Phi(x_a)$ is given, x_a can be found by using the *inverse error function*, $\mathrm{erf}^{-1}(x)$ as

$$x_a = \sigma \mathrm{erf}^{-1}(2\Phi(x_a) - 1) + \mu, \tag{5.5}$$

provided that μ and σ are also known.

5.2 Theoretical Background of Stochastic Quantization

What follows is the concept underlying the proposed stochastic calibration, whereas its interaction with the deterministic self-calibration, which operates jointly on it, is discussed later in this chapter.

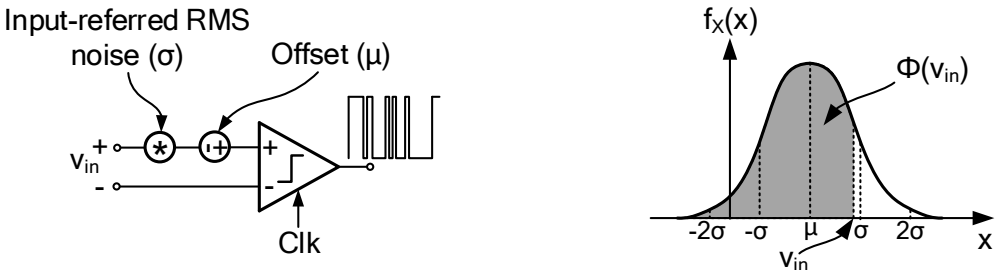


Fig. 5.1 (a) Model of a comparator with its input-referred noise and offset. (b) Illustration of the input distribution function on the Gaussian curve.

Fig. 5.1(a) illustrates a dynamic comparator with an input-referred rms noise and offset σ and μ (expressed in volts), respectively, and an input voltage v_{in}. At the rising edge of the clock (*Clk*), a logic one is asserted at the comparator output if $v_{\text{in}} + \sigma + \mu > 0$, while a logic zero is asserted otherwise. Assuming the noise of the comparator follows a normal distribution, then the input-referred noise and offset can be described by a random variable X_{comp} with (Gaussian) normal distribution of mean μ and standard deviation σ, $X_{\text{comp}} \sim \mathcal{N}(\mu, \sigma^2)$. As such, for the sample space of output ones and zeros, the ratio between the count of ones and the size of the sample space (i.e. the total number of runs, Φ) corresponds to the probability that the comparator input signal is greater than X_{comp}, i.e.

$$\Phi(v_{\text{in}}) = P_r\left(v_{\text{in}} - X_{\text{comp}} \geq 0\right) \tag{5.6}$$

Knowing that the *cumulative distribution function* (CDF) of a random variable X is defined as $F_X(x) = P(X \leq x)$, (5.6) corresponds to the CDF of a normal random variable X_{comp} for an input of v_{in}. Therefore, $\Phi(v_{\text{in}})$ is given by (5.4) as

$$\Phi(v_{\text{in}}) = \frac{1}{2}\left[1 + \text{erf}\left(\frac{v_{\text{in}} + \mu}{\sigma}\right)\right]. \tag{5.7}$$

Therefore, if $\Phi(v_{\text{in}})$ is known, v_{in} can be expressed as

$$\widetilde{v}_{\text{in}} = \sigma \cdot \text{erf}^{-1}[2\Phi(v_{\text{in}}) - 1] - \mu, \tag{5.8}$$

Of course, the accuracy of this estimation depends on the size of the sample space, i.e. the total number of comparisons. It also depends on the magnitude of the input voltage compared to the input-referred noise of the comparator, i.e. v_{in}/σ. Following (5.7), and for a sample space of limited size, the smaller the value of input v_{in}, the higher the accuracy of (5.8). For

instance, for a sample space size of 1000, if $v_{\text{in}} = 3\sigma$ and $\mu = 0$ then $\Phi(v_{\text{in}}) = 0.9999$. This means that, on average, out of 10000 samples, only one of them indicates that the input of the comparator is greater than v_{in}. Therefore, for such an unlikely event, 1000 samples would not be enough to obtain a meaningful statistical output. Since $\text{erf}(1) \simeq 0.85$, as a rule of thumb and for a reasonably large sample space, the outcome of (5.8) can be assumed to be accurate when

$$-1 < \frac{v_{\text{in}} + \mu}{\sigma} < 1, \tag{5.9}$$

which is equivalent to [1]

$$0.25 < \Phi(v_{\text{in}}) < 0.85. \tag{5.10}$$

MATLAB simulations have been performed to assess the estimation error of v_{in} by (5.8), referred to as e_{est}, versus various parameters through three different simulations, as follows.

Simulation 1: The first simulation entails the error of estimation of v_{in} by (5.8) versus the number of comparisons with σ being 1. The error term is defined as

$$e = \max\left\{\widetilde{V}_{\text{in}} - V_{\text{in}}\right\} - \min\left\{\widetilde{V}_{\text{in}} - V_{\text{in}}\right\} \tag{5.11}$$

where $V_{\text{in}} \in [-\sigma, +\sigma]$ and $\widetilde{V}_{\text{in}}$ is the corresponding value of V_{in} estimated by (5.8). Fig. 5.2 shows the result of this simulation. As expected, the error decreases as the number of comparisons increases and as can be seen in the figure, the profile of the curve is a straight line in a logarithmic scale with a slope of $\sim -3.6\sigma$ dB/dec.

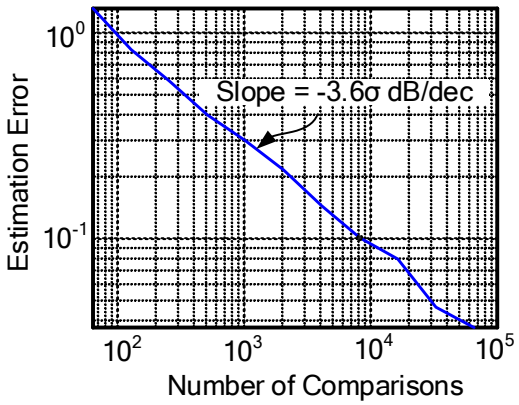


Fig. 5.2 Error of estimation by (5.8) versus the number of comparisons.

[1] This inequality is only used to get a rough initial guess of how close the input voltage is to the input-referred noise of the comparator. It is right that the variation of σ could change the bounds of this inequality, but this would not affect the integrity of the calibration process and would only result in one more/less deterministic step, as will be explained in the next section.

Simulation 2: The next simulation aims to find out for what value of σ, the error of estimation by (5.8) is minimum. For this, we set v_{in} to 1.5 and μ to 0.5 and sweep σ. The number of comparisons is 2^{14} (the value we have chosen for our design). Fig. 5.3 shows the result. As can be seen, the minimum error occurs at around $\sigma = 1$.

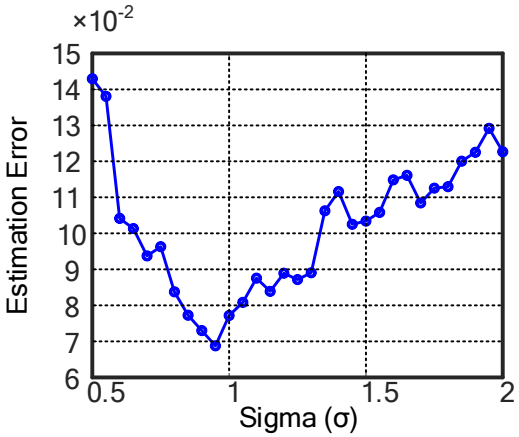


Fig. 5.3 Error of estimation by (5.8) versus σ.

Simulation 3: The next simulation involves running the entire calibration algorithm (modelled in MATLAB) for an ADC with the specification of our design, i.e. 10-bit resolution, and finding out the SNDR degradation versus the number of comparisons (2^c). For each value of c, the calibration algorithm is run 100 times each with a different set of mismatches for the CAP-DAC, and in the end we look at the minimum SNDR of the whole set. Fig. 5.4 shows the result of this simulation. As can be seen, the SNDR starts to degrade for $c < 14$, and this is indeed the value we have chosen for our design.

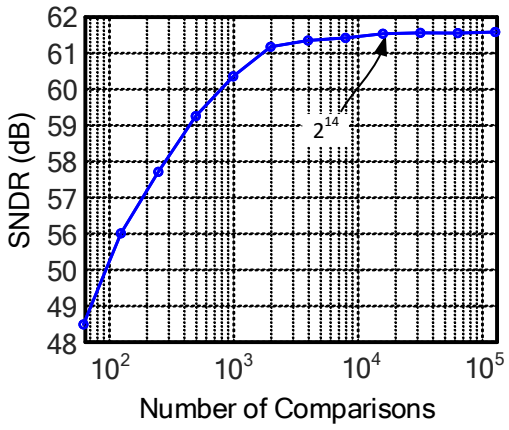


Fig. 5.4 Output SNDR of the ADC versus the number of comparisons.

To sum up, for $-\sigma < v_{\text{in}} < +\sigma$, the relationship between e_{est} and the number of comparisons (N_{comp}) is a straight line in a log-log scale with a slope of $\approx -3.6\sigma$ dB/dec. Also, for (normalized values of) $v_{\text{in}} = 1.5$, $\mu = 0.5$ and $N_{\text{comp}} = 2^{14}$, e_{est} is minimum at $\sigma \approx 1$.

What is also evident from (5.8) is that, in order to estimate v_{in}, the values μ and σ must be known beforehand. Prior works have aimed at accomplishing this through post-fabrication measurements [159]. This is, however, quite inefficient because, firstly, they cannot be fully automated and, secondly, since μ and σ differ among devices, they demand that each IC be separately calibrated, which is both time-consuming and costly from a large-scale production perspective. Here, we propose a novel method to perform this completely on-chip as follows. Eq. (5.8) can be seen as a formula with two unknowns: μ and σ. Therefore, μ and σ can be found by using two different values of v_{in} (v_{in1} and v_{in2}) and solving a system of two equations:

$$\begin{cases} \mu + \sigma \cdot \left[\text{erf}^{-1}(2\Phi(v_{\text{in1}}) - 1)\right] = v_{\text{in1}} \\ \mu + \sigma \cdot \left[\text{erf}^{-1}(2\Phi(v_{\text{in2}}) - 1)\right] = v_{\text{in2}}. \end{cases} \tag{5.12}$$

Solving for μ and σ results in

$$\begin{cases} \sigma = \frac{v_{\text{in1}} - v_{\text{in2}}}{\text{erf}^{-1}(2\Phi(v_{\text{in1}})-1) - \text{erf}^{-1}(2\Phi(v_{\text{in2}})-1)} \\ \mu = \frac{v_{\text{in2}}\text{erf}^{-1}(2\Phi_1(v_{\text{in2}})-1) - v_{\text{in1}}\text{erf}^{-1}(2\Phi_2(v_{\text{in1}})-1)}{\text{erf}^{-1}(2\Phi(v_{\text{in1}})-1) - \text{erf}^{-1}(2\Phi(v_{\text{in2}})-1)}. \end{cases} \tag{5.13}$$

5.3 Calibration Process

According to (5.9), if an input voltage smaller than 1 LSB is to be estimated using a noisy comparator, then σ should be larger than 1 LSB, provided that μ is negligible. Since comparators in SAR ADCs are usually designed so as to exhibit an input-referred noise and offset lower than 1 LSB, they do not satisfy such requirement. However, a possibility is to use a second *noisier* comparator (hereinafter referred to as *auxiliary comparator*), specifically designed for the mismatch calibration process with the proper characteristics. This auxiliary comparator's bandwidth can be more relaxed compared to that of the main comparator as the calibration process can run at a lower speed.

The calibration process is split into two phases: the computation of μ and σ (which is the necessary information for stochastic quantization) and the actual mismatch calibration (inclusive of both the deterministic and stochastic methods).

5.3.1 Computation of μ and σ

In order to satisfy the requirement of (5.9), the auxiliary comparator is designed to have an input-referred noise of $\sim$1.5 LSB and offset of $<$0.5 LSB. The offset is lowered down by means of a dedicated offset calibration circuit which operates before the mismatch calibration process begins. The aim of this initial phase of the complete calibration process is to form the two equations in (5.13), where v_{in1} and v_{in2} are two comparator inputs, and are set to $+$LSB and 0, respectively[2]. These equations are then solved in the digital domain in order to estimate the values of σ and μ and the results are subsequently stored in a memory. The inverse error function $\text{erf}^{-1}(x)$ of (5.13) is also stored as a look-up table. Fig. 5.5 and Fig. 5.6 demonstrate the ASM chart and the data-path circuit of the μ/σ computation process, respectively.

5.3.2 Mismatch Calibration

Let us consider the binary-scaled differential capacitor array shown in Fig. 5.7 as an example so as to explain the flow of the proposed calibration process. Here, for both P and N sides, the value of C_{00} is equal to that of C_0, and $C_j = 2C_{j-1}$, where $1 \leq j \leq N-1$. Let us assume that the mismatch of capacitor C_{Pj} is to be computed, and that the mismatch of all smaller capacitors (also referred to as LSB *capacitors*), which consist of $C_{P0} \sim C_{P(j-1)}$ and $C_{N0} \sim C_{N(j-1)}$, is known (i.e. previously computed). The process evolves in a three-phase procedure that combines a bottom-up deterministic self-calibration by using the main CAP-DAC and a stochastic-based quantization as follows:

Phase I: The reset switches S_{rP} and S_{rN} close, thus precharging nodes X_P and X_N to a common-mode voltage V_{CM}. At the same time, the calibration logic sets D_P to

$$\begin{aligned} D_P &= \left\{ d_{P00}, d_{P0}, d_{P1}, ..., d_{P(j-1)}, d_{Pj}, ..., d_{P(N-1)} \right\} \\ &= \{1,1,1,...,1,0,...,0\} \end{aligned} \tag{5.14}$$

and D_N to all zeros (Fig. 5.8(a)).

Phase II: The reset switches open and the calibration logic changes D_P to

$$\begin{aligned} D_P &= \left\{ d_{P00}, d_{P0}, d_{P1}, ..., d_{P(j-1)}, d_{Pj}, ..., d_{P(N-1)} \right\} \\ &= \{0,0,0,...,0,1,...,0\}. \end{aligned} \tag{5.15}$$

whereas D_N remains unchanged (Fig. 5.8(b)). This causes voltage V_{XP} to become

[2]These voltages can simply be generated by an appropriate control of the CAP-DAC through the SAR logic.

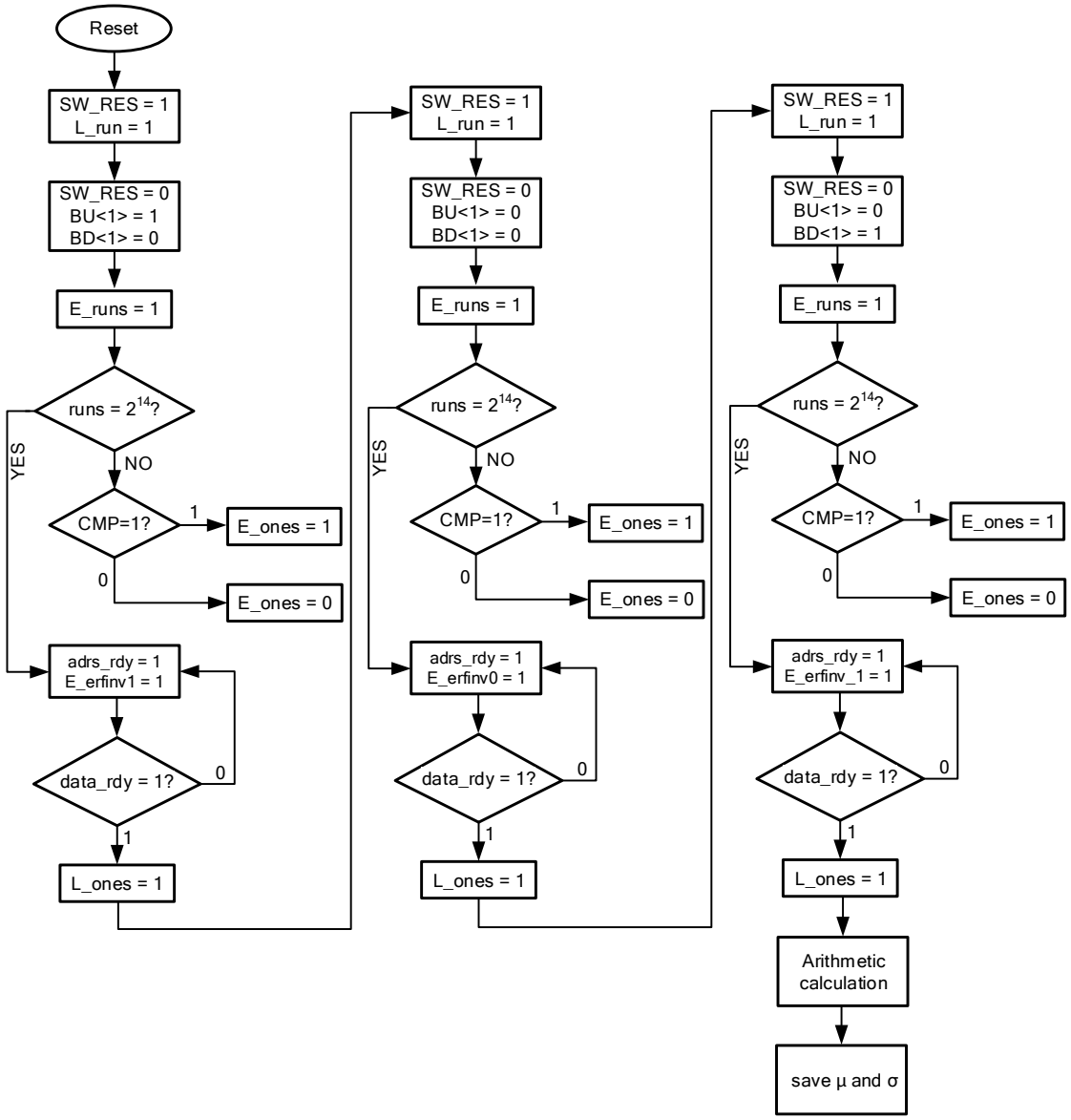


Fig. 5.5 ASM chart for μ/σ estimation.

$$V_{XP} = V_{\mathrm{CM}} + V_{\mathrm{REF}} \cdot \frac{C_{Pj} - \left(C_{P00} + \sum_{i=0}^{j-1} C_{Pi}\right)}{C_{\mathrm{tot}}} \tag{5.16}$$

$$= V_{\mathrm{CM}} + V_{\mathrm{REF}} \cdot \Delta C_{Pj}/C_{\mathrm{tot}}, \tag{5.17}$$

where C_{tot} is the total capacitance of the capacitive array and V_{REF} is equal to $V_{\mathrm{REFP}} - V_{\mathrm{REFN}}$. The voltage at node V_{XN} remains at V_{CM}. Therefore, the differential voltage at the input of

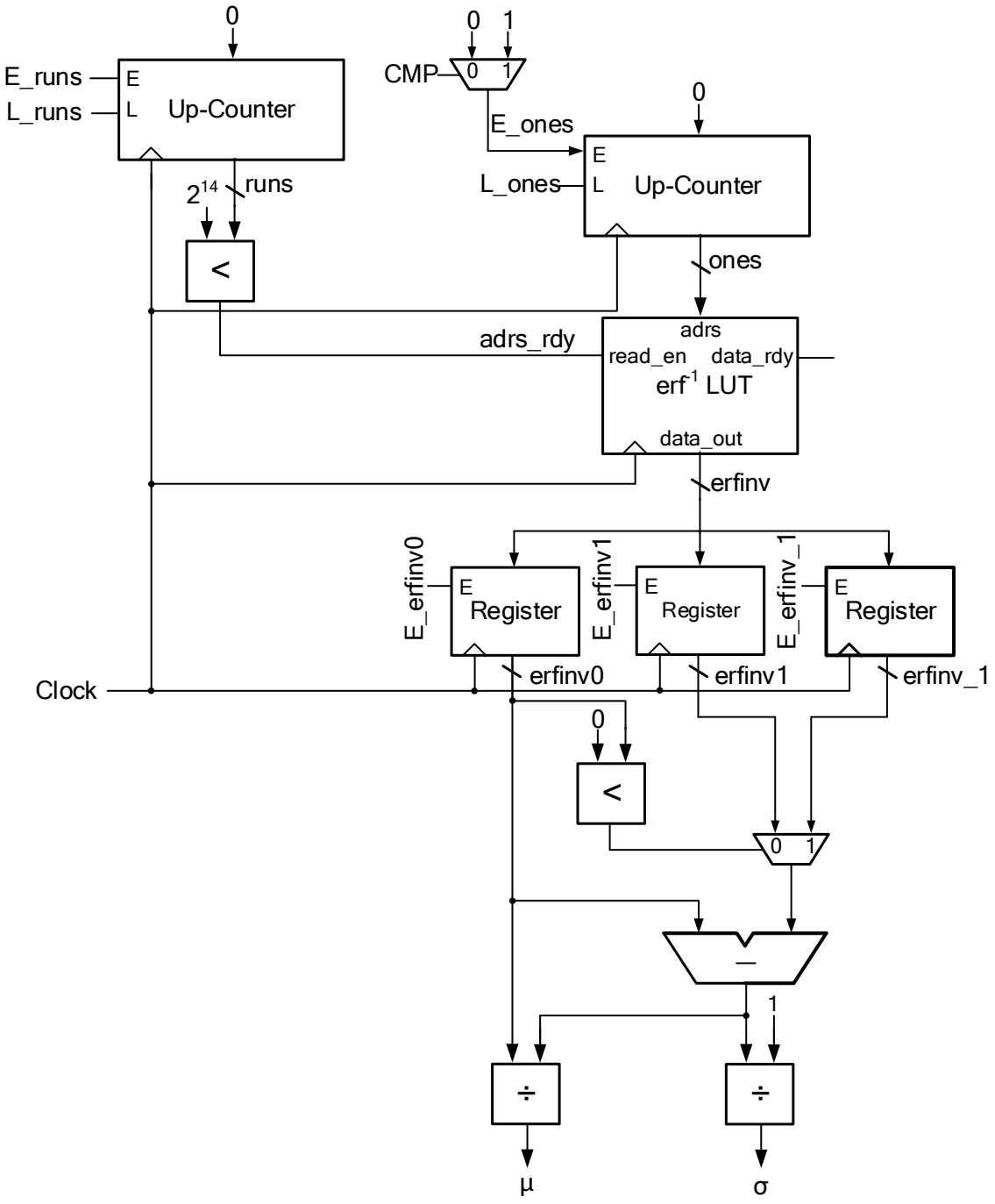


Fig. 5.6 Datapath circuit for μ/σ estimation.

the comparator becomes

$$V_X = V_{\mathrm{XP}} - V_{\mathrm{XN}} = V_{\mathrm{REF}} \cdot \Delta C_{Pj}/C_{\mathrm{tot}}. \tag{5.18}$$

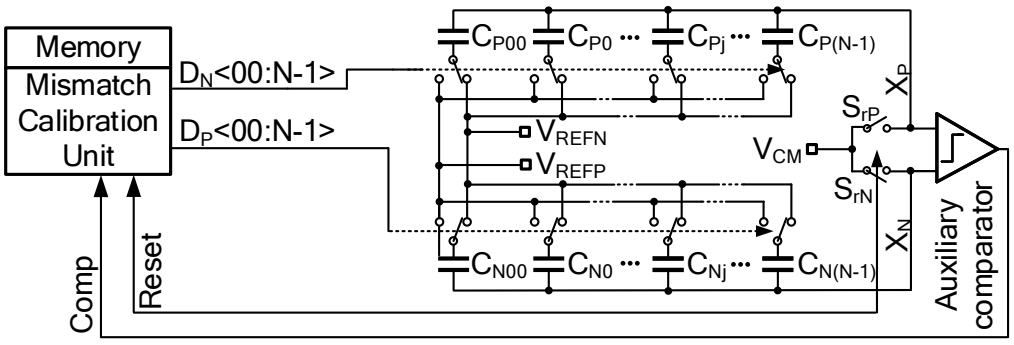


Fig. 5.7 A differential capacitor array to demonstrate the process of stochastic mismatch calibration.

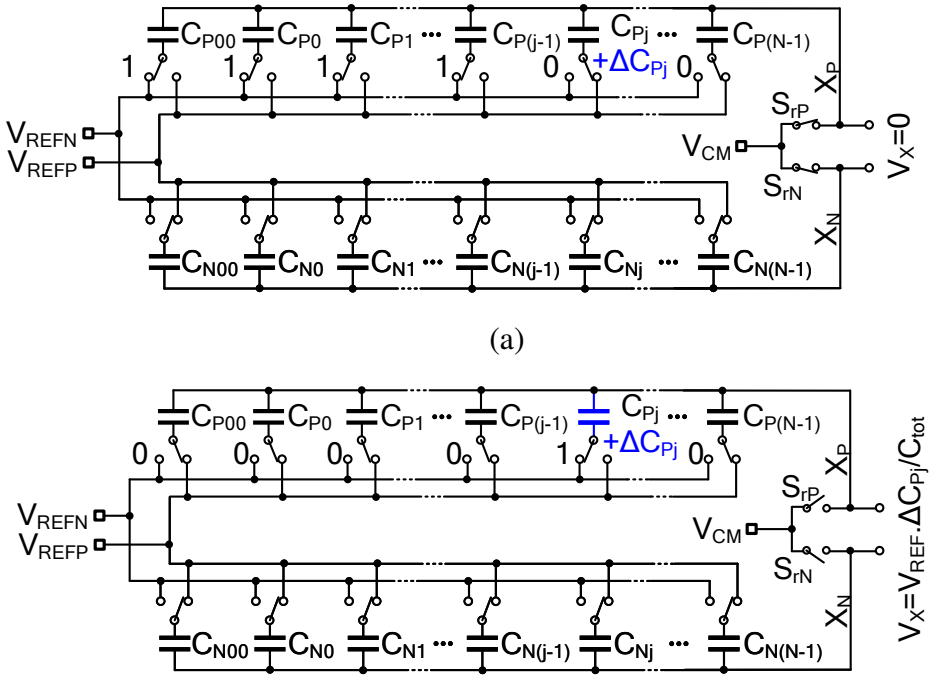


Fig. 5.8 CAP-DAC during the calibration process: (a) phase I, (b) phase II.

Phase III: With (5.18) being the input, the auxiliary comparator is activated to make a large number of comparisons, and the output 1's and 0's are recorded by the calibration unit in order to form the cumulative distribution function $\Phi(V_X)$. According to (5.9) and (5.10), if $0.25 < \Phi(V_X) < 0.85$, then V_X is smaller than 1 LSB, meaning that a *deterministic* self-calibration method would not be able to estimate it. The calibration unit then, by skipping

the deterministic approach and employing the stochastic one, uses the value of $\Phi(V_X)$ and arithmetically solves (5.8) using the values of σ and μ that were previously computed and stored in memory to find V_X.

On the other hand, if $\Phi(V_X) > 0.85$, the input of the comparator is positive, and thus $\Delta C_{Pj} > 0$. The calibration then progressively increments D_N (by skipping the first LSB bit associated to C_{N00}, i.e. D_{N00}) using a successive approximation search along with a counter E (an integer number referred to as the *error counter*) until $\Phi(V_X)$ becomes less than 0.85 (deterministic self-calibration). At this point, the calibration uses the stochastic method previously discussed to estimate V_X. Let us denote the quantified binary number associated with this residual voltage with x_a. The binary number

$$E_{Pj} = E + x_a - \sum_{i=0}^{j-1} e_{Pi} \tag{5.19}$$

would be thus the mismatch of C_{Pj} (normalized to the unit capacitance C_0) and stored in the memory. Here, $\sum_{i=0}^{j-1} e_{Pi}$ is the sum of the mismatch errors of all of the LSB capacitors.

Similarly, if $\Phi(V_X) < 0.25$, then V_X would be negative, implying $\Delta C_{Pj} < 0$. The calibration logic then increments D_P, along with E, until $\Phi(V_X) > 0.25$ (deterministic self-calibration). Once again, the calibration solves (5.8) to compute V_X. This time, the value

$$E_{Pj} = -E + x_a - \sum_{i=0}^{j-1} e_{Pi} \tag{5.20}$$

is stored in the memory as the mismatch of C_{Pj}. It is worth mentioning that the presence of any parasitic capacitance on the top-plate of CAP-DAC only changes the value of C_{tot} in the above equations and does not affect the validity of the calibration algorithm.

The mismatch correction is done as follows. Let us assume that $B_{\text{OUT}} = \{b_0, b_1, ..., b_{N-1}\}$ is a raw digital output of the ADC. Two error terms ΔE_P and ΔE_N are calculated as

$$\Delta E_P = \sum_{i=1}^{N-1} b_i E_{Pi}, \quad \Delta E_N = \sum_{i=1}^{N-1} \overline{b_i} E_{Ni}, \tag{5.21}$$

where $\overline{b_i}$ is the binary inversion of bit b_i, and E_{Pi} and E_{Ni} are the estimated mismatches corresponding to C_{Pi} and C_{Ni}, respectively [see (5.19) and (5.20)]. The final error is then calculated as a sum of ΔE_P and ΔE_N that is added to the raw output of the ADC to produce the calibrated output as

$$\widetilde{B}_{\text{OUT}} = B_{\text{OUT}} + \Delta E_P + \Delta E_N. \tag{5.22}$$

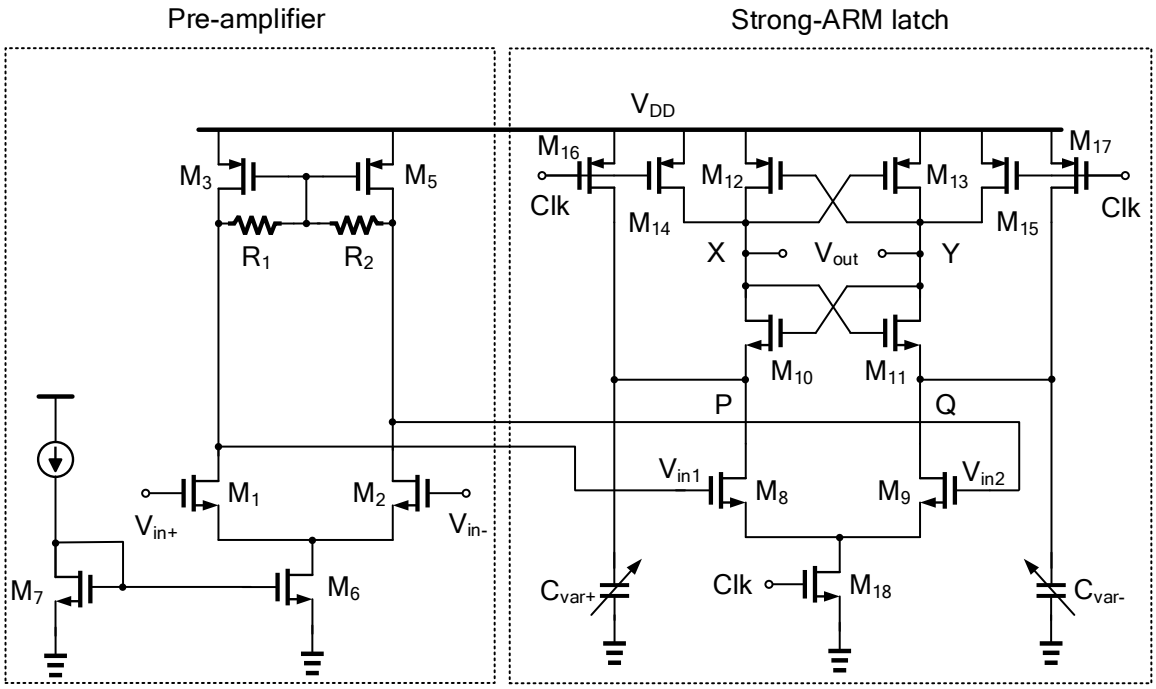


Fig. 5.9 Circuit implementation of the auxiliary comparator used for the calibration.

which is then rounded off to the nearest integer[3].

5.3.3 Auxiliary Comparator

As mentioned before, a dedicated comparator is needed for the proposed calibration. This *auxiliary comparator* needs to have a large input-referred noise in the order of 1LSB but does not need to be high speed, since the calibration can run at lower speeds as well.

Basic Operation

The comparator schematic diagram is shown in Fig. (5.9). As can be seen, it is composed of a static pre-amplifier and a strongARM latch. Two sets of digitally-controlled binary-scaled capacitors are also connected to the output nodes of the latch for offset calibration purposes that will be explained later. The strongARM latch alone could act as the comparator, but with the addition of a pre-amplifier the effects of kick-back noise and input-referred noise are improved.

[3]Note that retaining the fractional bits of (5.22) would result in a marginally better ENOB, but at the expense of more output bits than the nominal resolution of the ADC, which may add unnecessary hardware overhead.

High Noise Implementation

As mentioned previously, the auxiliary comparator needs to have an input referred noise of about 1.5LSB. The total input-referred noise of the comparator of Fig. 5.9 is given by

$$\sigma_{n,in}^2 = \sigma_{n,1}^2 + \sigma_{n,2}^2/A_v^2, \tag{5.23}$$

where $\sigma_{n,1}$ and $\sigma_{n,2}$ are the input-referred noise of the pre-amplifier and the latch, respectively and A_v is the DC gain of the pre-amplifier. According to (5.23), to increase the input-referred noise, we can decrease A_v so as to amplify the effect of the latch noise at the input. The pre-amplifer itself also contributes to the input noise. Having a small transconductance for the input transistors M_1 and M_2 also increases the total input-referred noise. In the final design, the input transistors are sized $4\mu m/30nm$ with a static current of $100\mu A$.

Offset Cancellation

The high noise design of the auxiliary comparator automatically results in a high input offset as well. The input-referred offset of the auxiliary comparator needs to be less than 0.5LSB , as discussed previously. The simulation results show that the standard deviation of the input offset of the designed comparator about 30 times higher than this value. This means an offset calibration circuit is required. There exist various offset cancellation methods [162–166]. Here, we do this by imbalancing the capacitance at nodes P and Q in Fig. 5.9 ($C_{\text{var}+}$ and $C_{\text{var}-}$) through two variable capacitors and establishing various discharge rates at these nodes. It can be shown that [120] during the amplification phase of the latching operation

$$V_P - V_Q = -\frac{g_{m8,9}}{2} \cdot \frac{C_P + C_Q}{C_P C_Q}(V_{in1} - V_{in2})t \tag{5.24}$$

$$+ \frac{C_P - C_Q}{C_P C_Q} I_{CM} t, \tag{5.25}$$

where C_P and C_Q are the total capacitance at nodes P and Q, respectively and I_{CM} is common-mode component of the current passing through M_8 and M_9. As can be seen, the term $(C_P - C_Q)/(C_P C_Q) I_{CM} t$ introduces an offset, which can cancel the comparator's random offset.

To perform offset cancellation, the inputs of the comparator are shorted to ground (common-mode voltage) and the values of $C_{\text{var}+}$ and $C_{\text{var}-}$ are progressively changed until

the comparator output changes sign [4]. The variable capacitor is implemented using an 8-bit binary-scaled capacitor array, as shown in Fig. 5.10, that is controlled by a digital binary input that comes from the calibration unit. The capacitors are implemented by MOS transistors with the bulk, drain and source tied together (MOSCAP). The resolution of the offset calibration is around 0.3LSB, which means that after the calibration the offset of the comparator would be in the range of $\pm$0.3LSB.

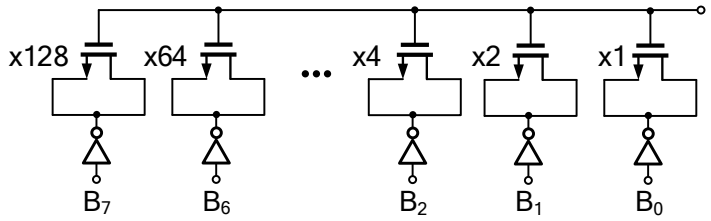


Fig. 5.10 Digitally controlled variable capacitor using MOSFET as a capacitor. This is for the purpose of offset calibration of the auxiliary comparator.

5.4 Deterministic vs. Stochastic Calibration

In this section, we try to more clearly convey the advantage of using a hybrid deterministic-stochastic calibration approach over the previously reported deterministic methods by comparing the performance of the behavioural models of 2 ADCs simulated in MATLAB: a 10-bit ADC and a 12-bit ADC, for the cases of deterministic calibration only and deterministic calibration with stochastic calibration. The model includes a conventional SAR ADC with a differential binary-weighted CAP-DAC with one single redundant capacitor. The bit-weights of the CAP-DAC are $W_1 = \{1,2,4,8,16,32,64,64,128,256\}$ for the 10-bit ADC and $W_2 = \{1,2,4,8,16,32,64,64,128,256,512,1024\}$ for the 12-bit ADC.

A low frequency sinusoidal is considered as the input of the modelled ADCs, and random mismatch (with Gaussian distribution) is introduced to the bit-weights of the CAD-DAC. In order to show the effectiveness of different calibration methods (i.e. the deterministic versus the proposed hybrid deterministic/stochastic), a large mismatch of 20% is considered for the unit capacitor. Figures 5.11 and 5.12 here below show the histogram of the ENOB of a 1000-run simulation for three different scenarios:

Scenario 1: with no mismatch calibration;

Scenario 2: with deterministic mismatch calibration only;

[4] Each time the comparator makes multiple comparisons and the output with the larger number of occurrence would be regarded as the actual output for the offset-calibration algorithm. This *averaging* cancels-out the effect of random noise.

Scenario 3: with the combination of deterministic and stochastic mismatch calibration (which is the proposed work).

As can be observed from the figures, the deterministic mismatch calibration (scenario 2), on average, is rather effective in improving the performance of the ADC compared to the uncalibrated converter (scenario 1). However, the deterministic mismatch calibration still results in an average ENOB of about 1.5-bit lower than the nominal resolution of the ADC. This is due to the following impediments associated with the deterministic approach:

1. The accuracy of the mismatch calibration of each single capacitor of the CAP-DAC is 1 LSB, This means that, for instance, if for a specific scenario where only the MSB capacitor is mismatched with an amount of 0.9 LSB, the calibration would not be able to detect/correct it and the post-calibrated ADC would still have +0.9 LSB INL error.

2. The limited accuracy of the deterministic calibration would also result in the accumulation of quantization error. This accumulation of errors may result in even larger overall error than 1 LSB after the calibration.

Based on the developed numerical models, the average ENOB of the hybrid deterministic-stochastic mismatch calibration is, on the other hand, only a fraction of a bit less than the ideal resolution, as can be seen in the figures.

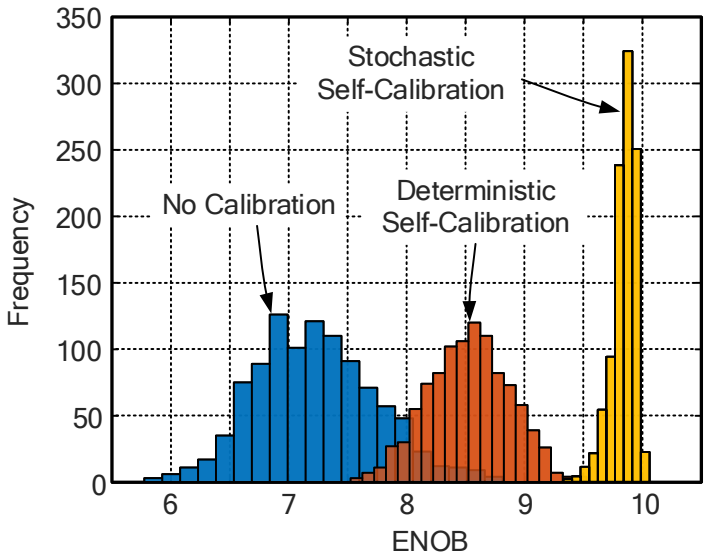


Fig. 5.11 A comparison between deterministic and stochastic approaches for a 10b ADC.

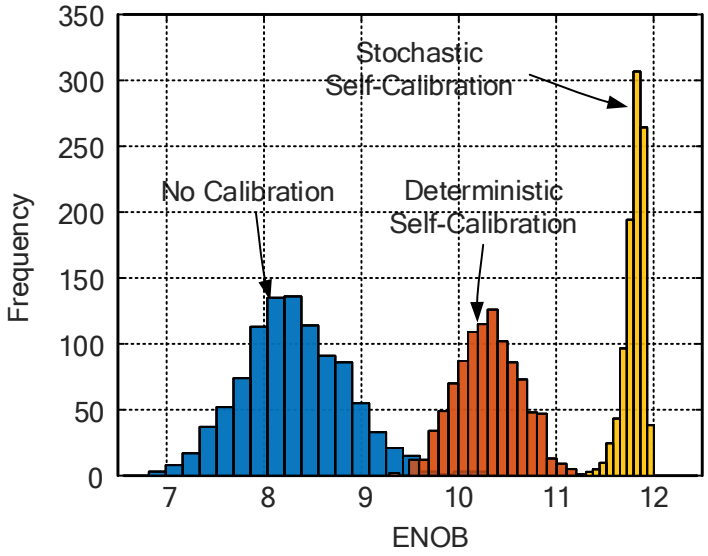


Fig. 5.12 A comparison between deterministic and stochastic approaches for a 12b ADC.

The residual error of the stochastic approach seen in the figure is due to the error imposed by Eq. (5.8) as a result of the limited number of comparisons. As a matter of fact, if the result of (5.8) could be computed precisely, the stochastic based method would compute the *exact* bit-weights and there would be *perfect* reconstruction of the ADC output. However, the inaccuracy in the computation of (5.8) would cause an error that would result in imperfect reconstruction. This residual error of (5.8) is similar to the quantization error of the deterministic approach but with a smaller magnitude (less than 1 LSB). The extend of how small that error is depends on how accurate (5.8) can be computed, which also depends on how many comparisons can be made.

It is also worth noting that the histogram spread of the deterministic calibration follows a Gaussian curve, as can be seen in the figure. This, of course, makes sense as the output SNDR of the ADC is directly affected by the linearity of the CAP-DAC which would also follow a Gaussian curve for large number of runs. The histogram spread of the stochastic calibration, on the other hand, is a bit different, as can be seen in the figures. This is because for this case, the output SNDR is not limited by the linearity of the CAP-DAC but by the accuracy of Eq. (5.8) which is a highly non-linear equation.

Fig. 5.11 shows that the stochastic quantization is able to theoretically improve the average ENOB of the 10-b ADC by almost 1.2b.

Chapter 6

A 10-bit 85MS/s SAR ADC with Stochastic Mismatch Calibration

The ADC architecture is shown in Fig. 6.1. It consists of a 10-bit SAR ADC core, the calibration circuitry, the clock duty-cycle control and the SPI interface. The calibration circuitry comprises the mismatch calibration digital unit and the *auxiliary* comparator along with its offset calibration circuit. The block diagram of the SAR ADC core is shown in Fig. 6.2. The differential input signal $V_{\mathrm{INP/INN}}$ is sampled by two bootstrapped switches on the top plate of the binary-weighted capacitive DAC, which is controlled by an asynchronous SAR logic. The *reset* switches are only used during the calibration phase. Fig. 6.3 also illustrates the timing diagram of the ADC during the three different operational phases. During the offset calibration phase, where the auxiliary comparator's offset is corrected, the Reset signal is asserted, connecting the inputs of the comparator to the common-mode voltage ($V_{\mathrm{CMP}} = V_{\mathrm{CMN}}$, as seen in the figure). The comparator's offset is then cancelled using two arrays of binary-weighted MOS capacitors connected to its output nodes and controlled by signals $B_{\mathrm{OS\_N}}$ and $B_{\mathrm{OS\_N}}$. Afterwards, the mismatch calibration phase starts in a stochastic process which was explained earlier. The ADC then enters the normal binary search conversion phase, where the *Sample* signal is asserted and the input voltage is quantized.

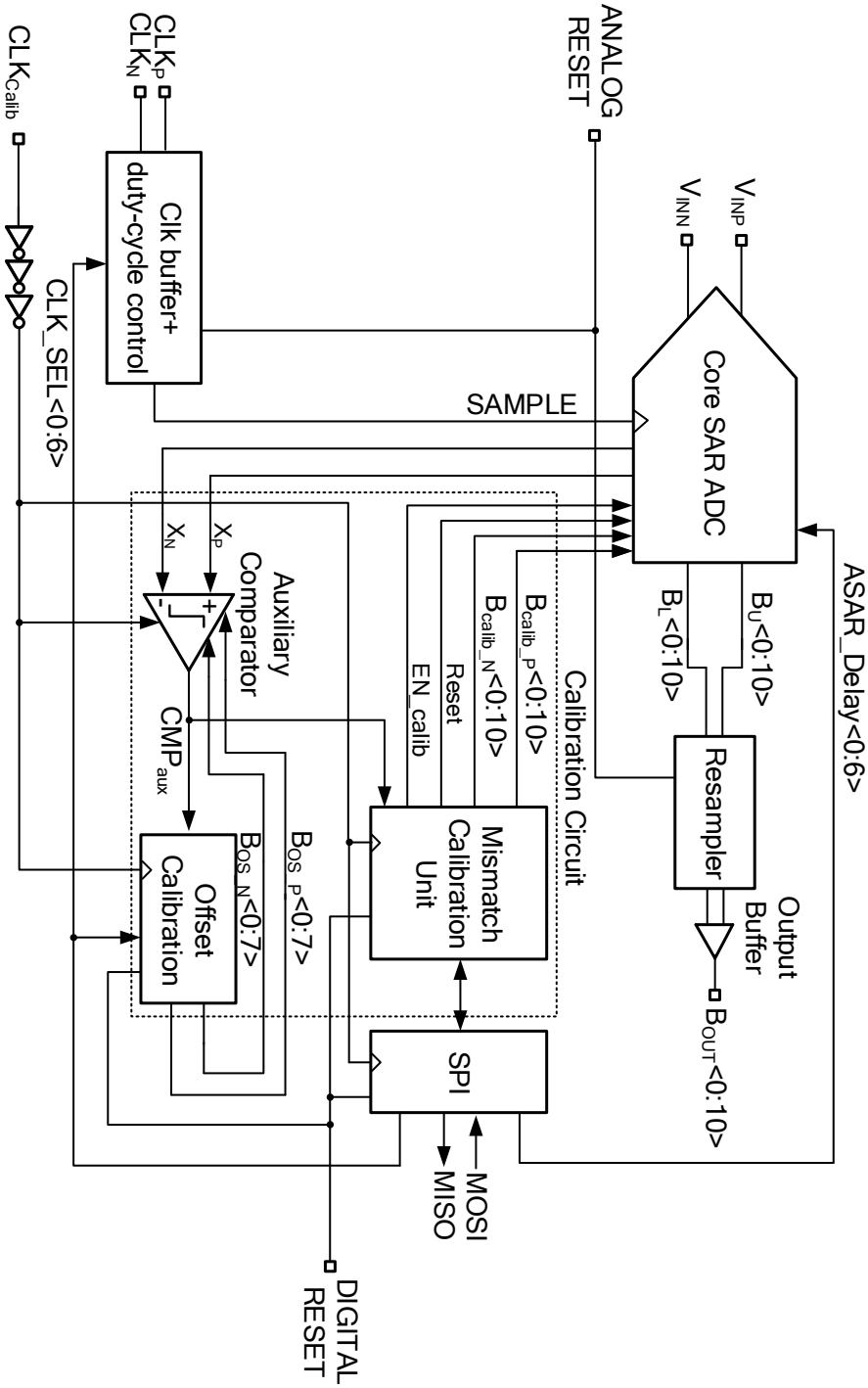


Fig. 6.1 Overall architecture of the ADC with mismatch calibration.

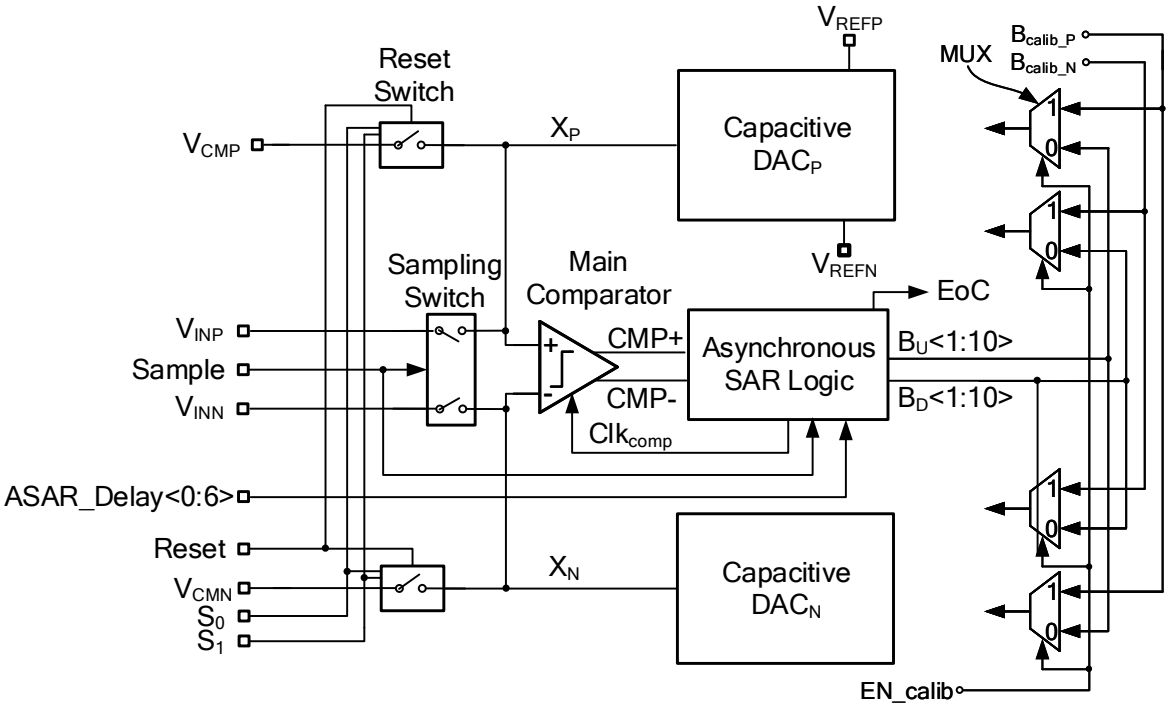


Fig. 6.2 Block digram of the SAR ADC core.

6.1 Capacitive DAC

Thanks to the proposed mismatch calibration, the total size of the capacitive DAC has managed to go as low as 148 fF[1], resulting in a unit capacitance C of only 280 aF. According to Monte-Carlo simulations, the mismatch parameter for the PDK MOM capacitor of the used technology (TSMC 28nm LP CMOS) is $A_C = 0.8 \sim 0.9\% \cdot \sqrt{\text{fF}}$. This did not lead to a considerable mismatch for the value of the unit cap chosen (the Monte-Carlo simulations showed a maximum INL of $\sim \pm 1.1$ LSB). Therefore, eight LSBs of intentional systematic mismatch were also introduced on purpose to the DAC (+4 LSBs to the MSB capacitor and -2 LSBs to the MSB-1 and MSB-2 capacitors) in order to demonstrate the effectiveness of the proposed calibration approach. Split-capacitor topology is used to implement the CAP-DAC, as shown in Fig. 6.4[2].

[1]The kT/C limit is lower than 148 fF for the resolution of the ADC. This value is chosen based on how much dynamic range loss could have been tolerated. This loss is mainly due to the unwanted parasitic capacitance at the top-plate of the CAP-DAC (e.g. metal routing parasitics, input capacitance of the compactor and top-plate parasitic of the DAC itself).

[2]For the sake of simplicity, capacitor C_{00} is not shown in this figure.

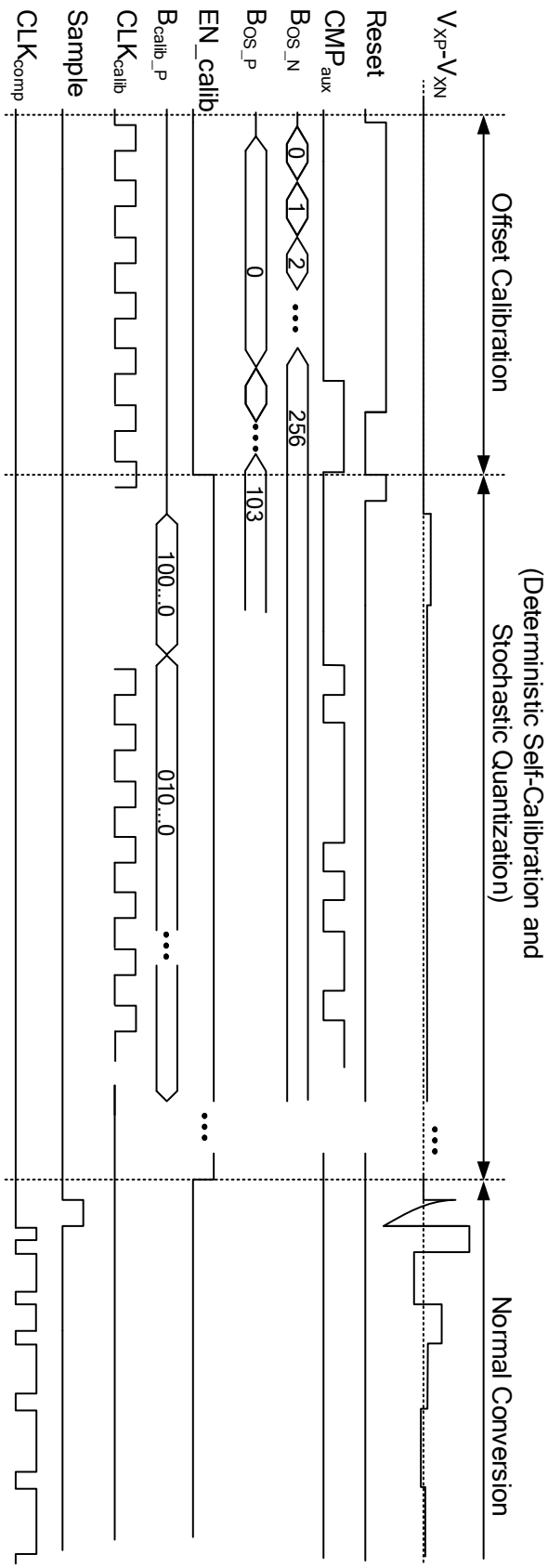


Fig. 6.3 Operational timing diagram of the ADC signals along the three operational phases.

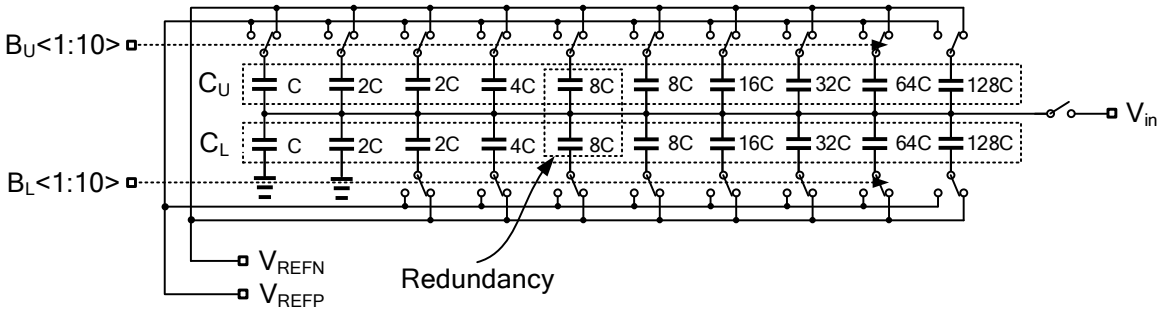


Fig. 6.4 A 9-bit binary-weighted split-capacitor array with redundancy at 8*C*.

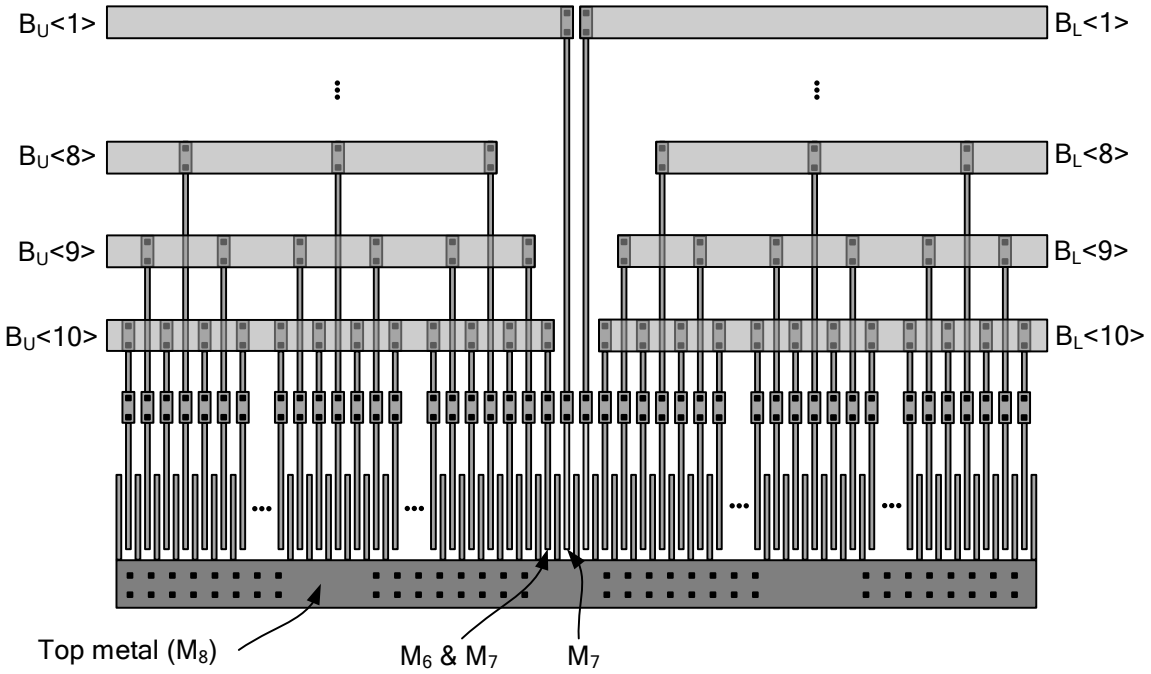


Fig. 6.5 Inter-digitized layout structure of the split-capacitor array.

Redundancy is introduced by adding an additional capacitor of size 8*C*, which can compensate for variation in V_{REF} and/or settling errors with magnitude up to 32 LSB (≈ 60 mV). The CAP-DAC is laid-out using custom fringe capacitors in an inter-digitized manner, as shown in Fig. 6.5. These capacitors are formed between the vertical metal structures. In order to reduce parasitics due to coupling with the substrate, only metal layers 6 and 7 are used. As can be seen in the figure, the unit capacitors are distributed along the structure. This reduces the effect of process gradients in the horizontal direction. To reduce the length of the top-plate connection (metal 8), thereby reducing its parasitic resistance and capacitance, both circuit and layout techniques are adopted. At the circuit level, two LSB capacitors (*C* and

2*C*) are connected to ground on one side of the split-capacitor array [167], as indicated in Fig. 6.4, resulting in an MSB capacitance of 128*C* rather than 256*C*. In the layout, as shown in Fig. 6.5, the LSB capacitor is made by the same number of fingers as the LSB+1 capacitor (1 finger) by using only one layer of metal (M6) rather than two (M6 + M7). This effectively halves the total number of fingers, hence reducing the length of the top metal connection.

6.2 SAR Control Logic

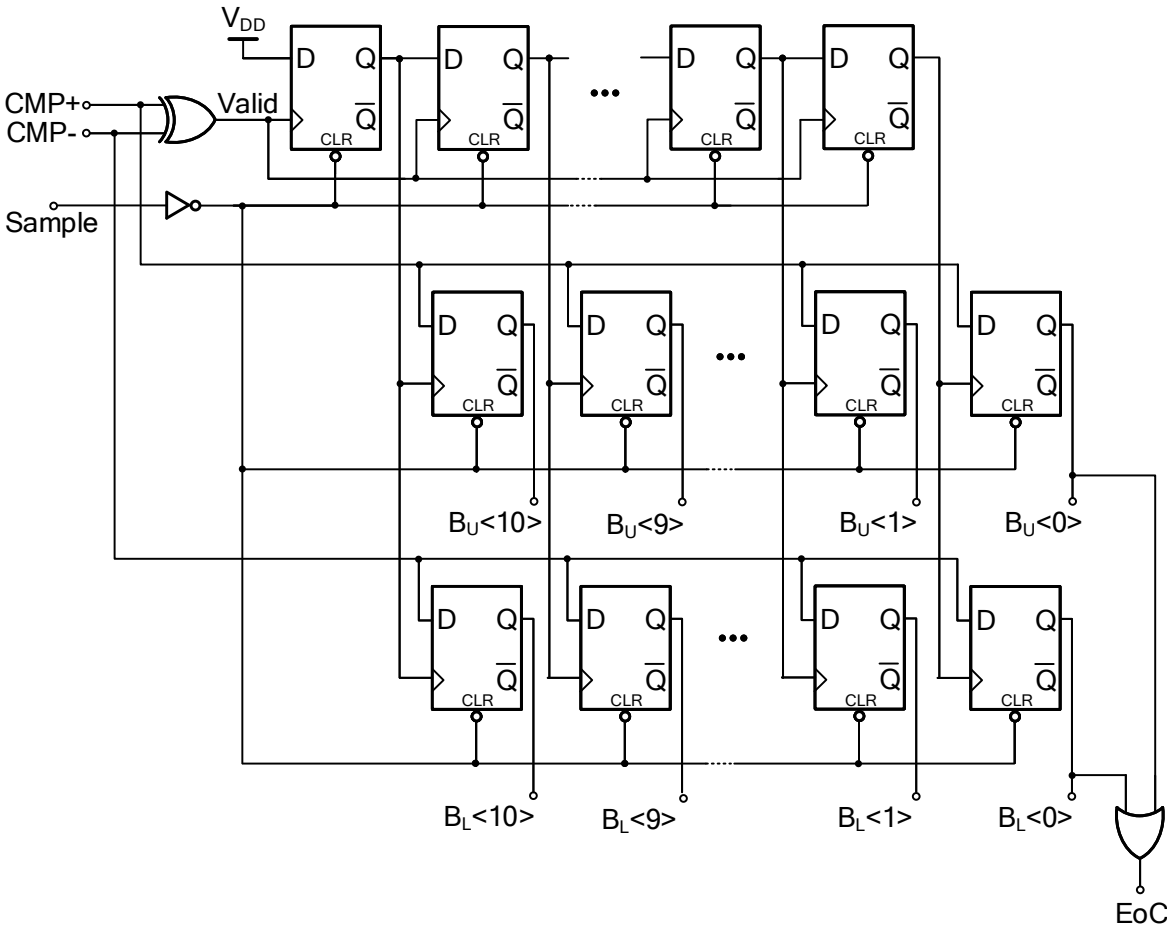


Fig. 6.6 Structure of the SAR logic used for this work.

The topology of the SAR logic that is used for this work is shown in Fig. 6.6. The SAR logic is implemented using transmission-gate-based D-flip-flops (DFFs). A programmable delay block (chain of inverters controlled by SPI) is used in the monostable pulse generator of Fig. 6.7, which introduces a delay, t_{delay}, that needs to be large enough to accommodate the digital logic propagation delays, t_{logic}, and the CAP-DAC capacitors settling time, t_{DAC},

i.e $t_{\text{delay}} > t_{\text{logic}} + t_{\text{DAC}}$. The delay block is implemented as shown in Fig. 6.8 and the delay can be controlled through a one-hot 6-bit signal via the SPI.

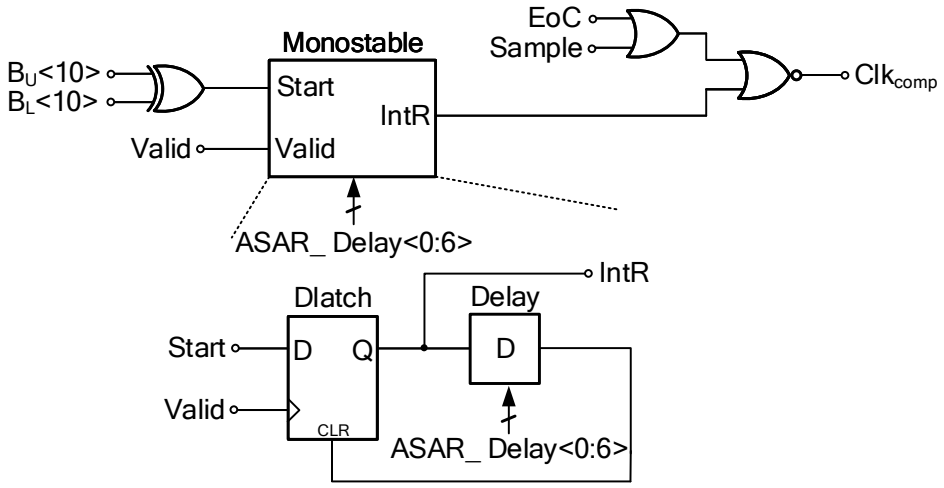
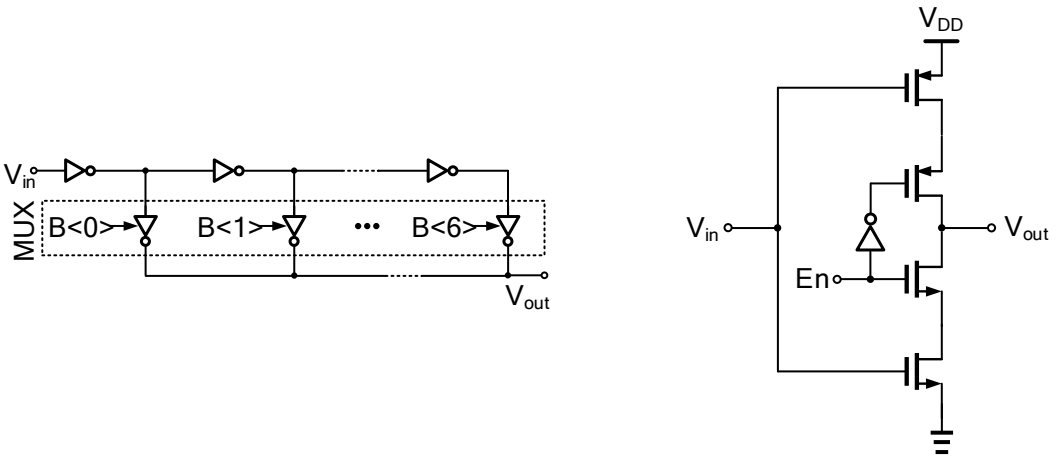


Fig. 6.7 Asynchronous clock generator of the SAR control logic.

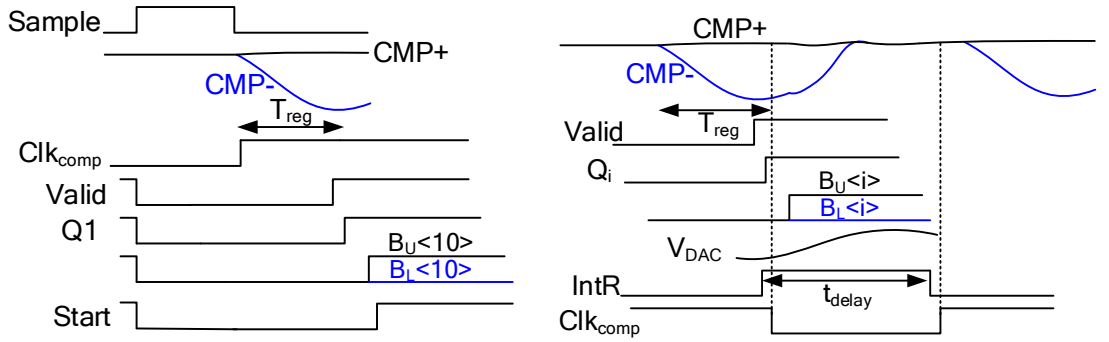


(a) (b)

Fig. 6.8 (a) Implementation of the delay block used in the monostable (b) implementation of the multiplexer.

Timing operation of the asynchronous clock generator is depicted in Fig. 6.9. When *Sample* (sampling signal of the ADC) goes high, the differential input analog signal is sampled on the capacitive DAC, and when it goes low, the comparator makes the first comparison. After a regeneration time, T_{reg}, the outputs of the comparator CMP+ and CMP− diverge. This would generate a *Valid* signal that triggers the top row of DFFs of the SAR logic. The MSB bits $B_U < 10 >$ and $B_L < 10 >$ are determined (depending on the sign of comparator

output) causing *Start* to go high. From then onward and throughout 10 more cycles, the next bits (from MSB-1 to the LSB) are determined as follows (Fig. 6.9(b)). When the comparator makes a successful comparison after its regeneration time, *Valid* goes high. This would clock the first row of DFFs, thus resulting in a bit decision followed by the settling of CAP-DAC top-plate voltage to a new value. *Valid* also triggers the monostable. This leads to generation of a pulse with on-time equal to t_{delay} which resets the comparator for the next bit-decision.



(a) (b)

Fig. 6.9 Waveforms of the SAR logic (a) for the MSB bit, (b) for the bits MSB-1 to LSB.

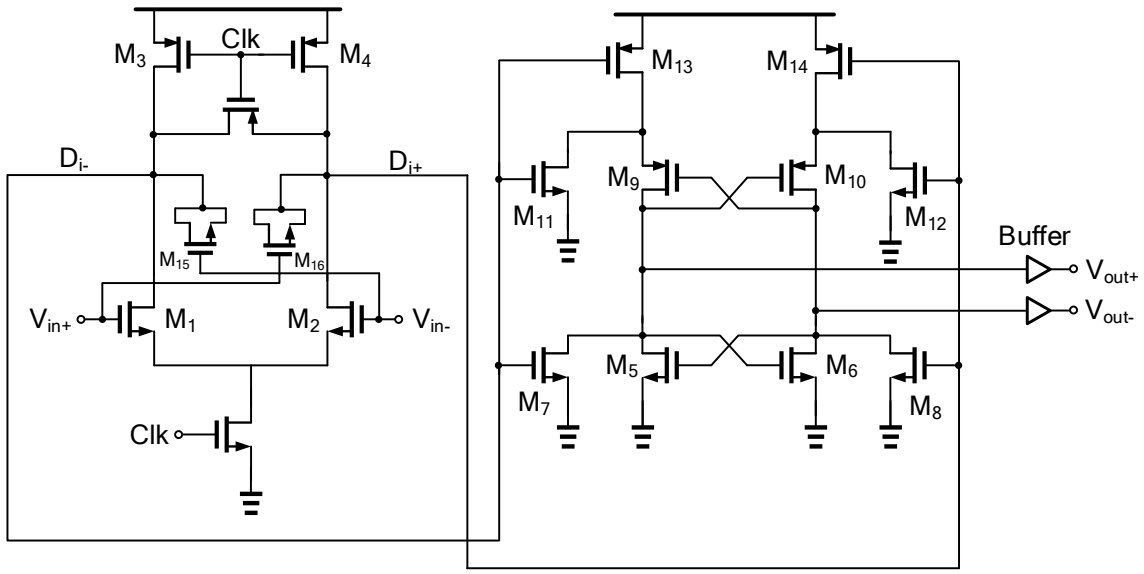


Fig. 6.10 Topology of the comparator used in this work.

6.3 Comparator

Fig. 6.10 shows the schematic of the comparator used here. It is a double-tail comparator based on the topology we discussed in section (3.3.1). Neutralization is used to decrease the effect of differential kick-back noise. Large sizes for the input pairs are used to lower the input-referred noise. The comparator achieves input-referred noise of $\sim$300 μV_{rms} while consuming $\sim$120 μW at the full clock rate of 85 MS/s. The input capacitance of the main comparator is 23 fF. The auxiliary comparator, on the other hand, has an input capacitance of only 3 fF, hence exerting minimal impact on the total top-plate parasitic capacitance of the DAC. The average current consumption of the auxiliary comparator is $\sim$200 μA, which can be neglected for it is turned off during the normal operation. The measurement results also reveal input-referred noise of $\sim$1.75 mV_{rms}, which is consistent with the simulation results.

6.4 Sampling Network

Fig. 6.11 depicts the input sampling network. The sampling switch is bootstrapped, similar to the topology shown in Fig. 3.28. The purpose of the cross-coupled transistors M_{XP} and M_{XN} is to neutralize the signal feed-through through the drain-source capacitance of M_{SP} and M_{SN}, with a size half the size of those. A damping network of series resistor and capacitor is added to damp-out the ringing of the input signal due to the inductive input path (especially the input wirebonds).

6.5 Input Sampling Signal

Fig. 6.12 shows the implementation of the circuit for generating the sample signal for the ADC. A differential clock should be provided to the *clock slicer*, where it is AC coupled by the decoupling capacitors C_ds. The common mode voltage of the clock is added by the self bias circuit, as can be seen in the figure. The output clocks of the clock slicer are then fed to a *duty cycle control* block composed of a chain of DFFs where a 7-bit select signal *CLK SEL<0:6>* adjusts the duty cycle of the sample signal. The select signal comes from the SPI.

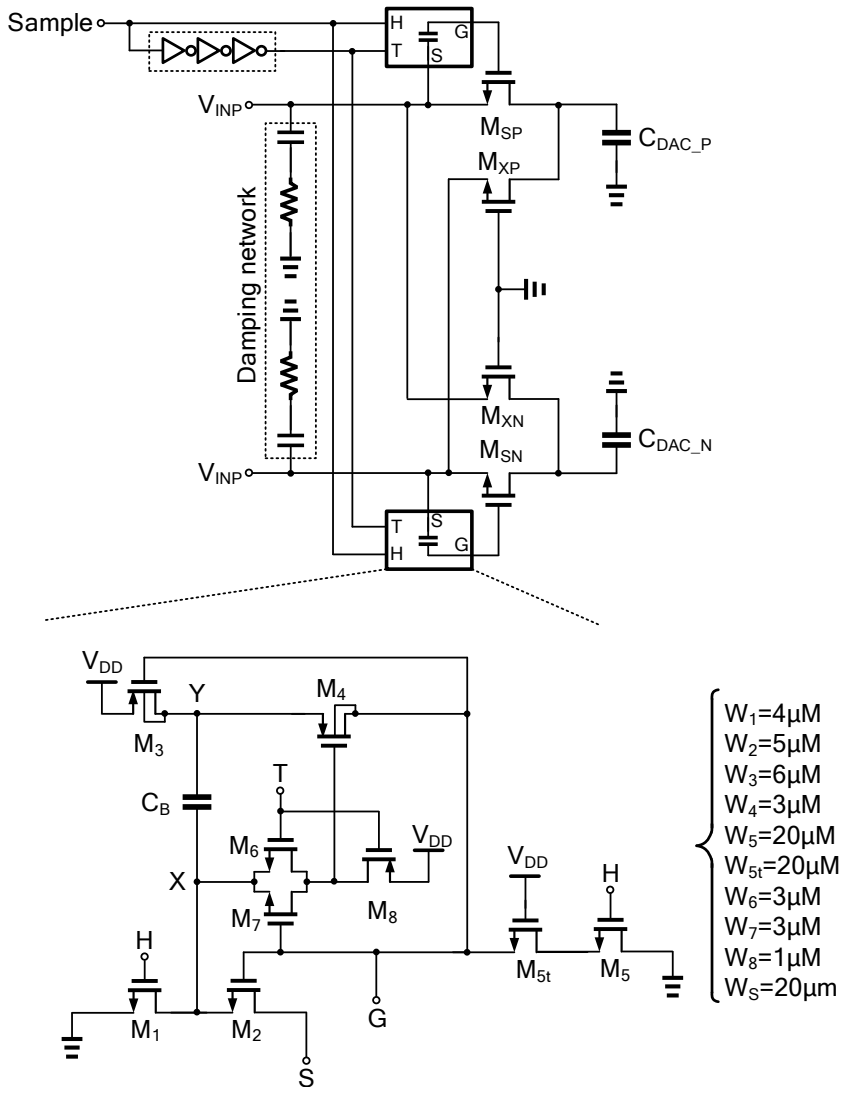


Fig. 6.11 Input Sampling network.

6.6 Mismatch Calibration

The proposed stochastic mismatch calibration algorithm as well as the offset calibration of the auxiliary comparator have been implemented using a standard digital synthesis flow in the targeted 28-nm CMOS. The inverse error function $\text{erf}^{-1}(x)$ is digitally approximated by a 1024×16 LUT, in a way that the approximated values are linearly spread over the curve. That is, more points where the slope of $\text{erf}^{-1}()$ is higher and less points where the slope is lower. The size and precision were determined using MATLAB simulation. For instance, Fig. 6.13 shows the output SNDR versus the precision of the LUT. As can be seen, precision

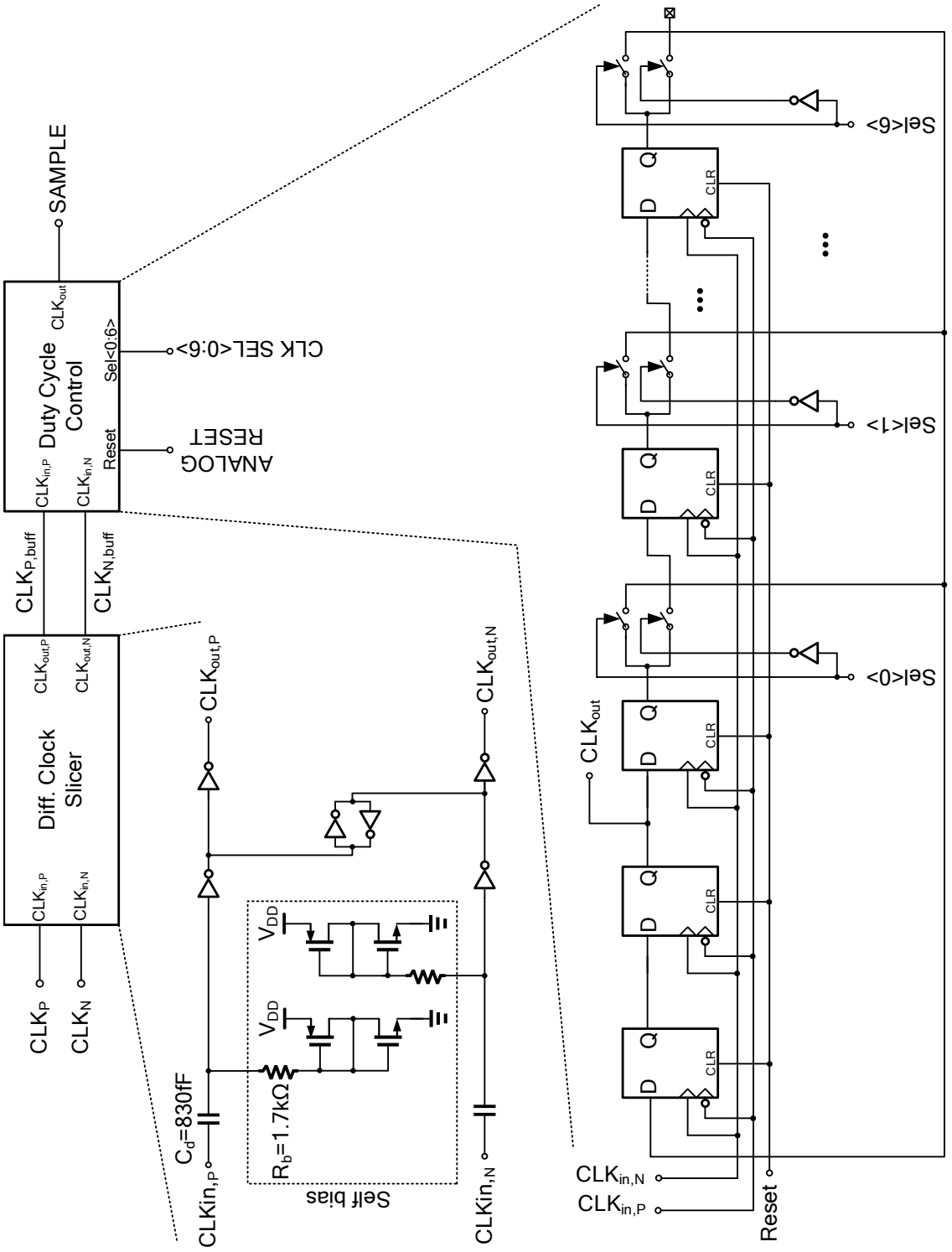


Fig. 6.12 Circuit to generate the sampling signal from the input differential clock.

over 9 bits results in little improvement. We simply chose the next power of two greater than 9 (16) to ease the algorithm calculations. The same simulation was carried out with respect to the number of rows.

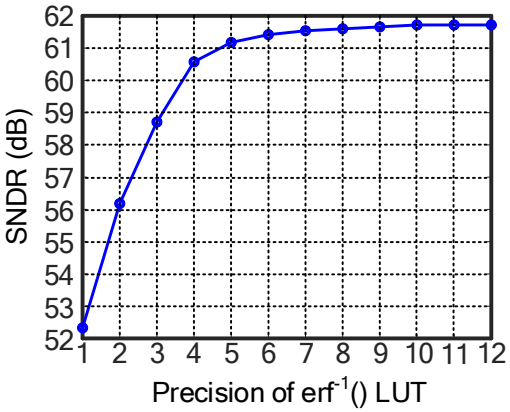


Fig. 6.13 Output SNDR of the ADC versus the precision of the LUT

The entire mismatch calibration circuit is fully implemented on-chip. The mismatch calibration process starts by computing the mismatch from the LSB+1 capacitor all the way up to the MSB capacitor for both the positive and the negative CAP-DACs, resulting in two sets of fractional binary numbers $E_P = \{e_{P,1}, e_{P,2}, ..., e_{P,N-1}\}$ and $E_N = \{e_{N,1}, e_{N,2}, ..., e_{N,N-1}\}$. To estimate the input voltage of the comparator using the discussed CDF method, a total of 2^{14} comparisons are carried out by the comparator.

It is worth mentioning that, since nodes X_P and X_N (i.e. the top-plates of positive and negative CAP-DACs) are left floating during phase III of the calibration process (Section 5.3.2), any leakage current at these nodes may result in an unwanted charge loss. Given the long duration of phase III (2^{14} comparisons), even a small leakage current may result in a considerable discharge of X_P and X_N, thus corrupting the voltage stored on these nodes. A viable solution is to break phase III into multiple shorter sub-phases, each followed by a reset, as explained in the following. Let us assume that the number of times the comparator is clocked during phase III is k ($k = 2^{14}$ here). Hence, the cumulative distribution function $\Phi(V_X)$ would be equal to the total number of logic 1's produced by the comparator divided by k. Similarly, the comparator can be clocked l times, where $l = k/c$ and c is an integer, followed by a reset of nodes X_P and X_N, and this can be repeated c times. This would result in c cumulative distribution functions $\Phi_1(V_X)$, $\Phi_2(V_X)$, ..., $\Phi_c(V_X)$. Due to the *stationary* nature of the comparator random noise, $\Phi(V_X)$ can therefore be expressed as

$$\Phi(V_X) = \frac{\Phi_1(V_X) + ... + \Phi_c(V_X)}{c}. \tag{6.1}$$

This way, the time during which nodes X_P and X_N are left floating is shortened by a factor c. This *averaging* operation is also beneficial in eliminating detrimental effects of other noise sources, such as the kT/C sampled thermal noise of switches S_{rP} and S_{rN} (see Fig. 5.7). Here, $l = 128$ is chosen. This produces 128 CDF functions, which are averaged out according to (6.1) to obtain the final CDF.

The whole calibration process takes about 5 ms using a 50 MHz clock. Once the calibration is completed, the calibration data (E_P and E_N) are sent by SPI for storage. The mismatch is corrected off-chip numerically (as it would be using a downstream DSP) the way it was discussed in Section 5.3.2.

It is worth mentioning that, and as pointed out before, in order to correct the CAP-DAC mismatch of a SAR ADC with split-capacitor architecture, two sets of correction (error) numbers are required; one for the upper row (C_U in Fig. 6.4) and one for the lower row (C_L in Fig. 6.4). This is because these two sets of capacitors are controlled independently by two separate and independent binary numbers $B_U < 1:10 >$ and $B_L < 1:10 >$ in Fig. 6.4. For this reason, it is not possible to correct the mismatch of a split-capacitor-based SAR ADC post-measurement by forming a look-up table based on the DNL/INL curve of the ADC (i.e. the foreground methods we discussed in Chapter 4.)

6.7 Design for Testability

To add testability features to the IC, a number of important node voltages were brought out by the means of an LVDS circuit. These nodes include the sampling signal *SAMPLE*, the clock of the main comparator CLK_{comp}, the valid signal *Valid*, the end of conversion signal *EoC*, the output of the main comparator *CMP+* and the output of the auxiliary comparator CMP_{aux}. The block diagram of the test-point circuit is shown in Fig. 6.14. The LVDS circuit works with a supply voltage of $1.8V$ with an output swing of $\pm 200mV$. Test points can be enabled/disabled locally through SPI in order to observe the point of interest.

6.8 Measurement Protocol

The measurement of the chip was performed in 4 different consecutive modes. The mode selection is done through a two-bit binary number $[S_1, S_0]$ that are set off-chip on the PCB board.

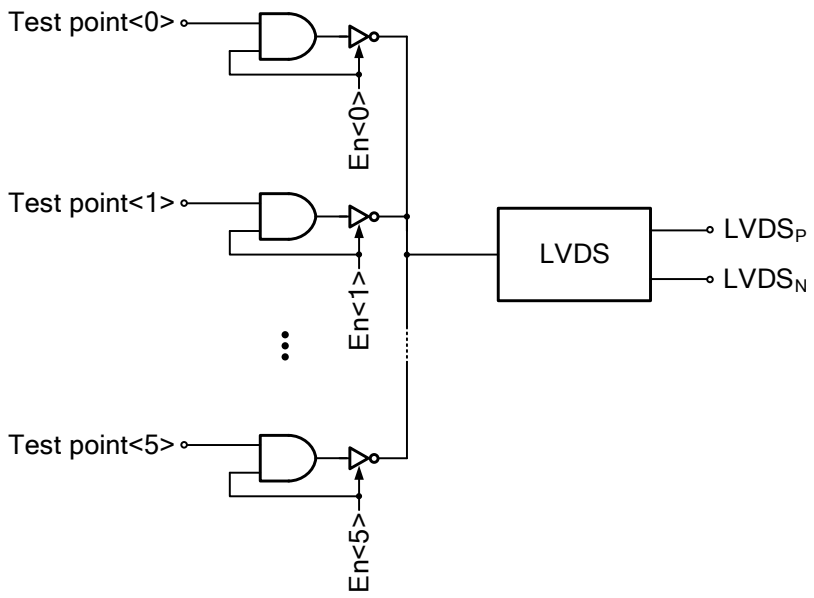


Fig. 6.14 Block digram of the test-point circuit.

- **Mode 0: Characterization of the Auxiliary Comparator**

  The selection binary $[S_1, S_0]$ is set to $[0,0]$ and the test point corresponding to CMP_{aux} is enabled via SPI. At this mode, the positive and negative inputs of the auxiliary comparator are connected to the positive and negative common-mode voltages V_{CMP} and V_{CMN}, respectively. This makes the comparator's input equal to $V_{\text{in,aux}} = V_{\text{CMP}} - V_{\text{CMN}}$. The input-referred offset and noise of the auxiliary comparator are found through the CDF method. This requires sweeping $V_{\text{in,aux}}$ around the common-mode voltage for a number of voltages and for each input voltage, the comparator makes a large number of comparisons to collect the output ZERO's and ONE's. The CDF curve can then be plotted, through which the offset and noise can be computed. The purpose of this mode is especially to find the input-referred noise of the auxiliary comparator to check if it is consistent with the requirement of (5.13).

- **Mode 1: Offset Calibration of the Auxiliary Comparator**

  The selection binary $[S_1, S_0]$ is set to $[0,1]$ and the test point corresponding to CMP_{aux} is enabled via SPI. The positive and negative inputs of the auxiliary comparators are both connected to the common-mode voltage $V_{\text{DD}}/2$. The offset calibration can be done either manually or automatically. The manual option is deliberately provided in case the automatic calibration fails. A dedicated bit *S_OS* that is set by the SPI determines which option is activated. If *S_OS* is zero, the calibration is performed automatically and starts by setting the digital reset and providing the calibration clock

CLK_{calib}. The results are stored in $B_{OSP} < 0:7 >$ and $B_{OSN} < 0:7 >$ and can be checked by SPI. If *S_OS* is one, the calibration is carried out manually by setting $B_{OSP} < 0:7 >$ and $B_{OSN} < 0:7 >$ through SPI.

- **Mode 2: Mismatch Calibration**

  The selection binary $[S_1, S_0]$ is set to $[1, 0]$. This enables the mismatch calibration unit. V_{CMP} and V_{CMN} should be both connected to the common-mode voltage as in mode 1. By applying the calibration clock CLK_{calib}, the calibration process starts and the mismatch of the capacitors, which are presented as binary numbers, are stored in internal registers and can be observed via SPI. A five-bit binary $Y < 0:4 >$, which is the state number of the state machine for the calibration algorithm, is also brought out for testability purposes.

- **Mode 3: Normal Conversion**

  The selection binary $[S_1, S_0]$ is set to $[1, 1]$. This disables all the offset and calibration circuits. The calibration clock CLK_{calib} should be disconnected and a differential clock should be provided through the PCB's SMA connectors *CLKP* and *CLKN*.

6.9 Test Setup

6.9.1 Prototype Chip

The proposed SAR ADC with the stochastic mismatch calibration is fabricated in TSMC 28-nm LP CMOS and occupies a core area of 0.011 mm^2, as shown in Fig. 6.15.

6.9.2 Printed Circuit Board (PCB)

The prototype chip was mounted on a PCB whereby the chip is directly wirebonded to landing pads on the PCB. The PCB is shown in Fig. 6.16. The differential analog inputs and clocks, as well as the single-ended calibration clock, are provided off-chip using SMA connectors. A Teensy microcontroller is responsible for programming the SPI. Analog and digital resets are made by push-bottoms, whereas the selection binary bits S_0 and S_1 are provided by jumpers between the ground and V_{DD}. The bias currents of the auxiliary comparator and the LVDS circuit can be adjusted through a variable resistor.

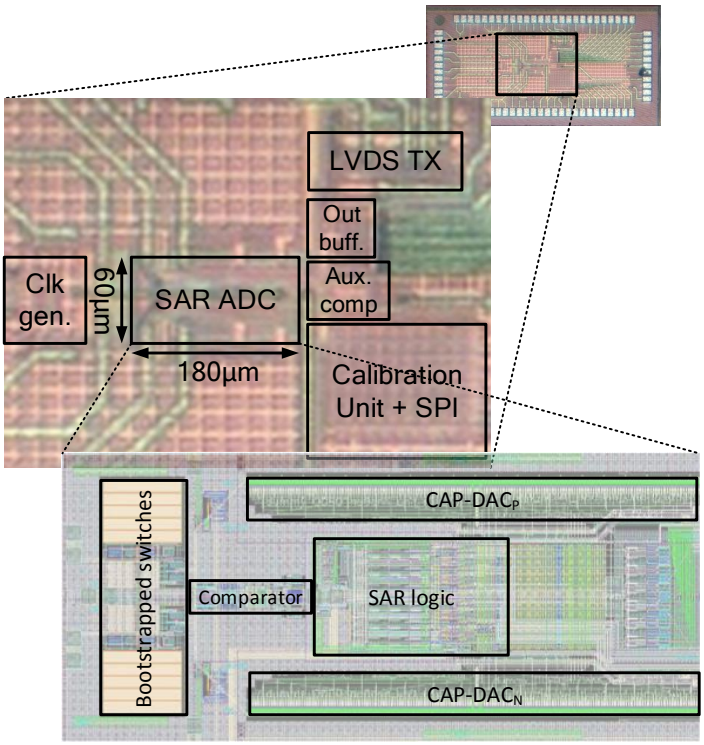


Fig. 6.15 Chip micrograph.

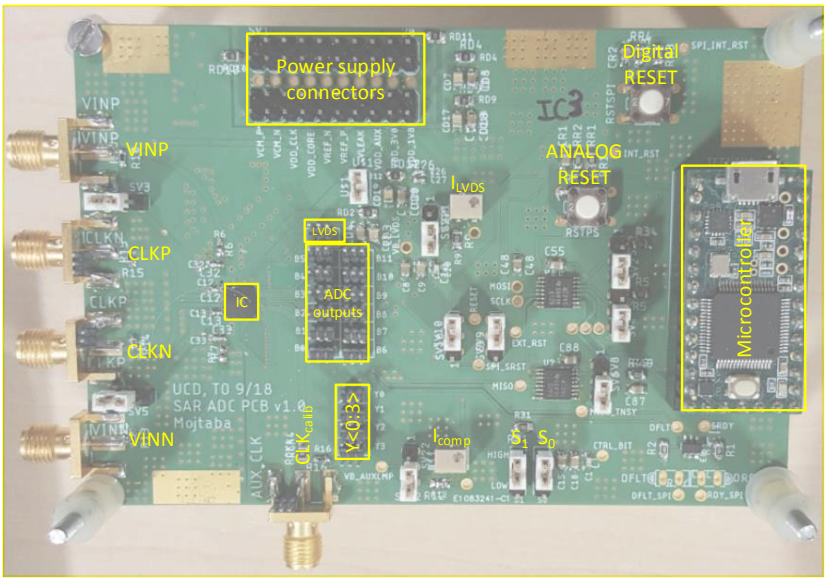


Fig. 6.16 PCB schematic.

6.9.3 Measurement Setup

The block diagram of the test setup used to measure the performance of the prototype ADC is shown in Fig. 6.17. The clock generator is a low-jitter sinewave signal generator. A low-noise signal generator is also used to provide the input signal. Since the harmonic performance of the signal generator is not sufficient for our 10-bit ADC, low-pass filters were used. The 11-bit (10 bits plus one bit of redundancy) output of the ADC is captured by a logic analyzer. The output data of the logic analyzer is then merged and processed with the mismatch data from the SPI in MATLAB on an external computer.

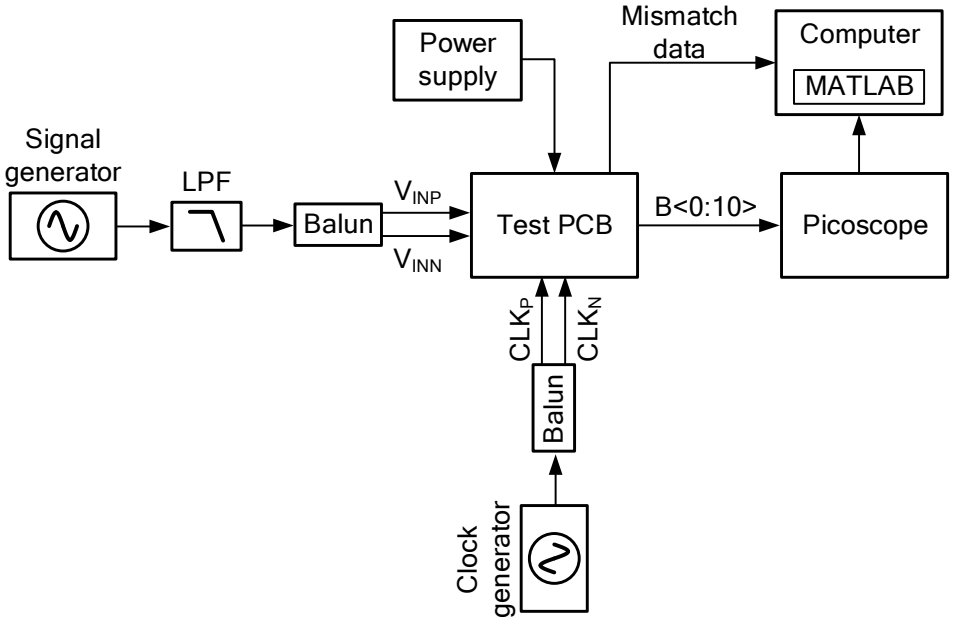


Fig. 6.17 Test setup used to characterized the prototype SAR ADC.

6.10 Measurement Results

To characterise the prototype ADC and measure its static/dynamic performance, the steps mentioned in section 6.8 were followed. Initially, the auxiliary comparator was characterised using the CDF-base method discussed and the setup for mode 0. Fig. 6.18 shows the measured CDF of the auxiliary comparator along with the best-fit theoretical CDF[3]. This shows the auxiliary comparator has an input-referred noise power of 1.75 μV_{RMS}, which is very close to the value captured by simulation.

[3]The discrepancy between the ideal theoretical curve and the measured one is because of the limited size of the sample space (1000 samples for each input voltage).

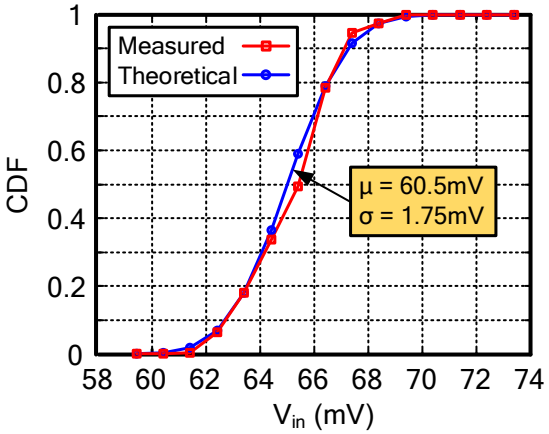


Fig. 6.18 CDF plot of the auxiliary comparator.

Next, following the step for mode 1, the offset of the auxiliary comparator is calibrated. Afterwards, the mismatch calibration is activated and the mismatch data is stored for the correction of the ADC output. The correction is done in MATLAB.

Figure 6.19 shows the DNL and INL plots of the ADC before and after the calibration is applied. Before the calibration, the maximum DNL and INL are equal to +1.1/-1.0 LSB and +4.2/-3.6 LSB, respectively, whereas, after the calibration, their values are improved to +0.7/-0.8 LSB and +0.7/-0.7 LSB, respectively.

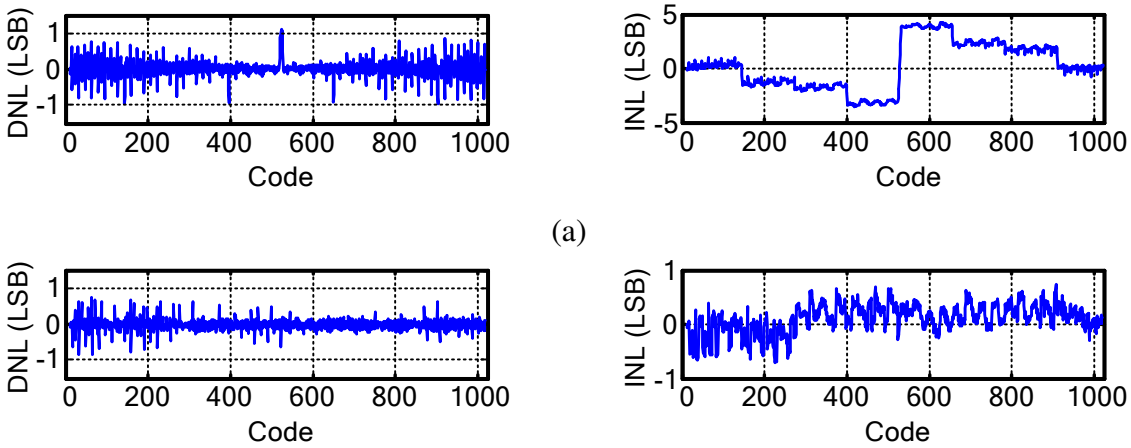
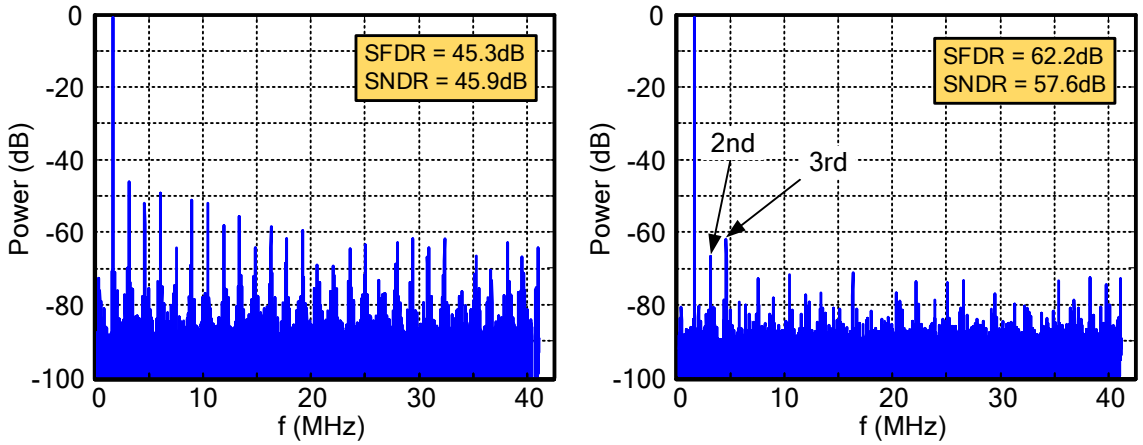


Fig. 6.19 Static performance of the ADC (a) before and (b) after calibration.

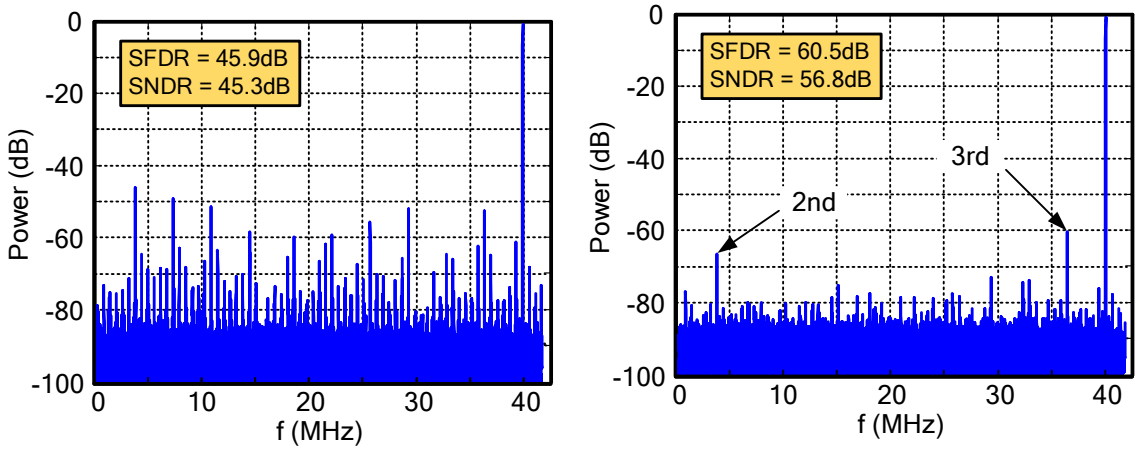
Fig. 6.20 shows the measured FFT spectrum of the ADC with a sampling frequency of 85MS/s and 1.5MHz input, before and after the calibration. As can be seen, without

calibration, the ADC achieves 62.2 SFDR and 57.6 SNDR. After calibration, the ADC's SFDR and SNDR are improved to 16.9dB and 11.7dB, respectively.



(a) (b)

Fig. 6.20 ADC output spectrum at a sampling frequency of 85MS/s and a 1.5MHz input (a) before calibration, (b) after calibration.



(a) (b)

Fig. 6.21 ADC output spectrum at a sampling frequency of 85MS/s and Nyquist-rate input (a) before calibration, (b) after calibration.

Fig. 6.21 shows the ADC's output spectrum with the same sampling frequency for 85MS/s and an input frequency of 39.9MHz. The ADC achieves 60.5dB SFDR and 56.8dB SNDR after calibration, which is an improvement of 14.9dB and 11.5dB for the SFDR and

SNDR, respectively, compared to before calibration. The residual SFDR is mainly limited by the nonlinearity of the input sampling switch while the residual SNDR is limited by the thermal noise of the circuit, including the kT/C noise and the comparator noise.

Fig. 6.22 shows the dynamic performance of the ADC versus the input frequency for with a sampling frequency of 85MS/s. Fig. 6.23 also shows the dynamic performance of the ADC versus the sampling frequency for with a 1.5 MHz sine-wave input.

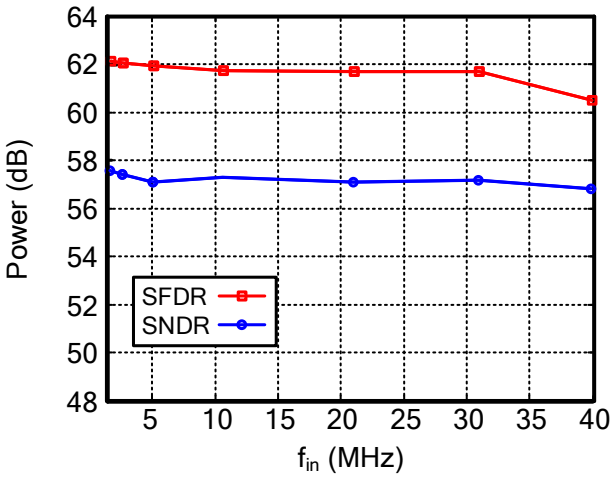


Fig. 6.22 SFDR and SNDR vs. input frequency at a sampling rate of 85MS/s.

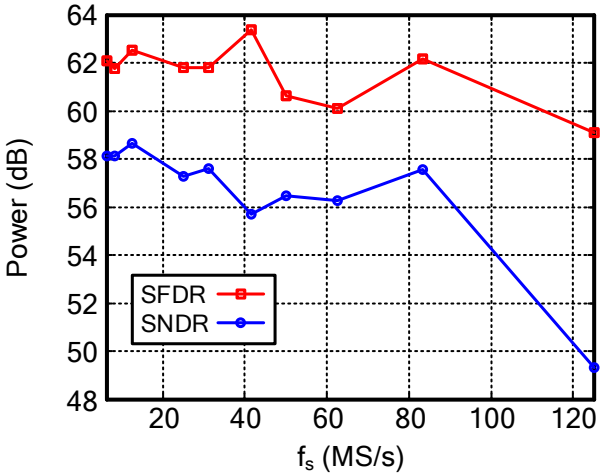


Fig. 6.23 SFDR and SNDR vs. sampling frequency for a 1.5MHz input.

At 85MS/s, the ADC consumes 582 μW from a 0.9 V supply whose breakdown is 267 μW consumed by the SAR logic, 219 μW by the CAP-DACs and 121 μW by the comparator, as illustrated in Fig. 6.24. The measurement of the average power was easily done by

reading the current reported on the power supply. For the CAP-DAC current consumption, the current reported on the power supply for the positive reference voltage (i.e. V_{REFP}) was used. However, the breakdown of power between the comparator and the SAR logic was not possible during measurement since they shared the same supply. To accomplish the breakdown showed on Fig. 6.24 the post-layout simulation results were used.

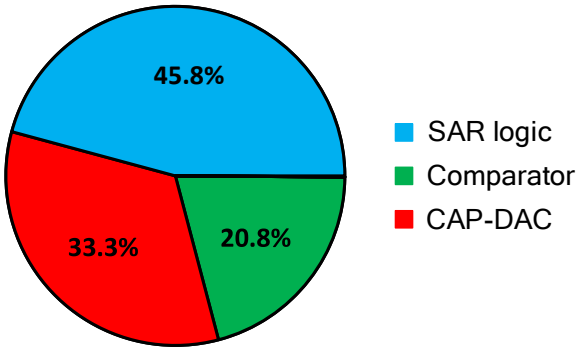


Fig. 6.24 ADC's power consumption breakdown. As can be seen, the SAR logic dissipates almost half the total power.

Table 6.1 Performance comparison with ADCs with mismatch calibration

	[171]	[172]	[168]	[173]	This work
Technology (nm)	90	65	28	55	28
Supply voltage (V)	1.2	1.2	1.1	3.3/1.2	0.9
Input swing ($V_{pp,d}$)	2	2.4	1.7	-	1.75
Resolution (bits)	12	12	12	16	10
Sampling rate (MS/s)	120	50	104	16	85
ENOB (bits)	10.39	10.9	10.2	12.7	9.14
Power (mW)	3.2	2.1	0.88	16.3	0.52
FoM (fJ/conv-step)	28	21.9	7.3	157	10.9
Calibration	Off-chip	Off-chip	On-chip	On-chip	On-chip

The measured Walden figure-of-merit (FOM) [2] at Nyquist rate is 10.9 fJ/conv-step. Table 6.1 compares the performance of our ADC with state-of-the-art mismatch-calibrated SAR ADCs. Only [168] appears to have a bit better FoM, but our ADC offers a more effective and accurate calibration technique. Table 6.2 draws comparison between a few reported 10b single-channel ADCs with high sampling rate ($>$ 50 MS/s) and this work. Fig. 6.25 also shows the comparison with the state-of-the-art as published in ISSCC and VLSI between 1997 and 2018.

Table 6.2 Performance comparison with 10b single-channel high-speed ADCs

	[71]	[86]	[169]	[170]	This work
Technology (nm)	40	65	40	40	28
Supply voltage (V)	1.2	1.2	1.2	0.9	0.9
Sampling rate (MS/s)	80	100	120	200	85
ENOB (bits)	8.71	9.51	8.83	9.2	9.14
Power (mW)	-	1.13	1.12	0.818	0.52
FoM (fJ/conv-step)	63	15.5	20.5	6.95	10.9

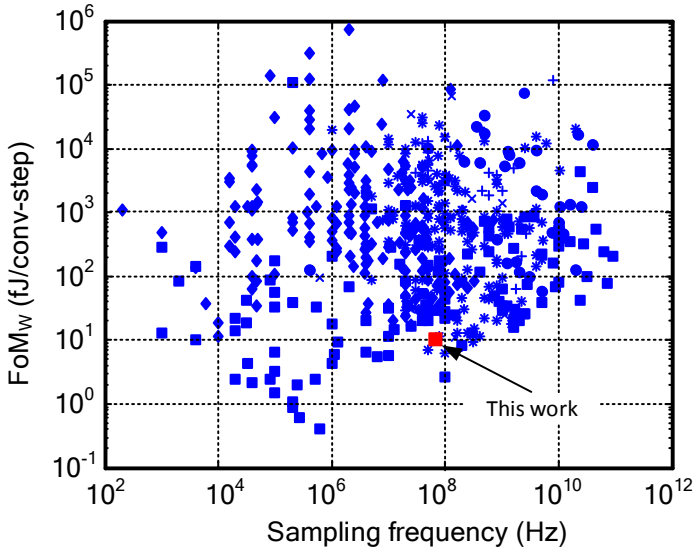


Fig. 6.25 Comparison of this work with the ADCs published in ISSCC and VLSI from 1997 until 2018.

6.11 Further Discussion on the Calibration Method

6.11.1 Area Overhead

As can be seen in Fig. 6.15, the area overhead of the calibration circuit is quite large in this design. One may argue that this extra area could be instead spent on a larger CAP-DAC to lower down the mismatch, so much so that a calibration might have not been even needed in the first place. In this section, we analytically show for what resolution of the ADC the choice of the proposed calibration scheme becomes economically feasible.

This analysis depends on the total value of the CAP-DAC, which directly determines the amount of mismatch as well as the kT/C sampling noise. Therefore, we only consider the random mismatch and the kT/C noise as the factors that affect the ADC's accuracy.

Two versions of the circuit are to be compared:

- The circuit *without* calibration with a CAP-DAC large enough to only cause a 0.5b ENOB loss due to ransom mismatch and kT/C noise together. Table 6.3 reports on the minimum required unit capacitor (C_{u1}) and the total CAP-DAC area (A_{DAC1}) for different ADC resolutions (N) for this version of the circuit.
- The circuit *with* calibration with a CAP-DAC large enough to result in a 0.4b ENOB loss due to only kT/C noise along with the calibration circuit that would correct for the random mismatch upto another 0.1 ENOB loss (making a total of 0.5b loss). Table 6.4 reports on the minimum required unit capacitor (C_{u2}) and the total CAP-DAC area (A_{DAC2}) for different ADC resolutions for this version.

The area estimation is based on the CAP-DAC area of the chip in Fig. 6.15, which is 100 μm$\times$10 μm=1000 μm^2, and rests on the fact that with the chosen inter-digitized lay-out (Fig. 6.5), the area is proportional to the unit capacitor (the length of the fingers) and the ADC resolution (the total number of the fingers) in a linear and exponential way, respectively, and hence is given by

$$A_{\mathrm{DAC}} \approx 10^{-3}\ \mathrm{mm}^2 \cdot \frac{C_u}{0.28\,\mathrm{fF}} \cdot \frac{2^N}{2^{10}}. \tag{6.2}$$

In (6.2), 0.28 fF represents the value of the unit capacitor used to implement the CAP-DAC for this work.

Table 6.3 Minimum unit capacitor for 0.5b ENOB loss due to random mismatch and kT/C sampling noise.

N	9	10	11	12	13	14
C_{u1} (fF)	0.14	0.32	0.6	1.7	3.1	16
A_{DAC1} (mm^2)	2.5×10^{-4}	1.1×10^{-3}	4.3×10^{-3}	2.4×10^{-2}	8.9×10^{-2}	3.4×10^{-1}

Table 6.4 Minimum unit capacitor for 0.4b ENOB loss due to kT/C sampling noise only.

N	9	10	11	12	13	14
C_{u2} (fF)	0.026	0.06	0.12	0.26	0.54	1.08
A_{DAC2} (mm^2)	4.6×10^{-5}	2.1×10^{-4}	8.6×10^{-4}	3.8×10^{-3}	1.5×10^{-2}	6.2×10^{-2}

The area estimation of the calibration circuit is a bit more complex. There are three parameters that determine the precision of the calibration algorithm and its total area:

- n_{comp}: number of comparisons
- s_{LUT}: size of the LUT for the inverse error function, i.e the number of rows
- *prec*: precision of the calculations, i.e. the number of binary fractional bits that are considered for the numerical calculations of the calibration algorithm

Since *prec* determines the resolution of the constituent digital blocks of the synthesized circuit, and since the size of main digital circuits (adder, multiplier, divider, register, comparator, etc) are proportional to their number of bits, we can surmise that the area of the calibration circuit is linearly proportional to the parameter *prec*. The same goes for s_{LUT} and the size of the LUT. However, n_{comp} only determines the size of a counter that counts the number of comparison, and therefore changing it would not affect the total size of the calibration circuit much. For this reason, for our analysis to estimate the area we keep n_{comp}=2^{14} as is. Our area estimation of the calibration circuit will be based on the one implemented in the chip of Fig. 6.15, which is also shown separately in Fig. 6.26. As can be seen, the total area is 0.01 mm^2 where the LUT, the μ/σ estimation and the mismatch calibration occupies, respectively, 1/4, 1/8 and 5/8 of the area. The precision parameters used for this circuit are *prec*=12 and s_{LUT}=1024. Following this breakdown and the earlier discussion, the area of the calibration circuit (A_{calib}) can be estimated as

$$A_{\text{calib}} \approx 0.01 \text{ mm}^2 \cdot \left[\frac{1}{4} \cdot \frac{prec \times s_{LUT}}{16 \times 1024} + \frac{prec}{12} \cdot \frac{1}{8} + \frac{5}{8} \cdot \frac{prec}{12} \right]. \tag{6.3}$$

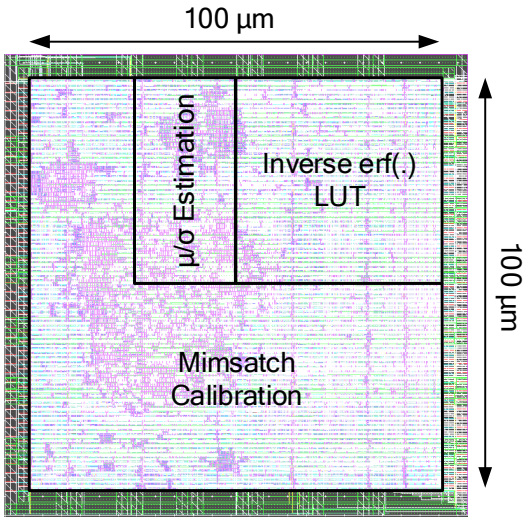


Fig. 6.26 Area breakdown of the calibration circuit.

The minimum area for a maximum of 0.5b ENOB loss (table 6.4) is worked out using a MATLAB Monte-Carlo simulation, and is reported in table 6.5.

Table 6.5 Estimation of minimum area of the calibration circuit for 0.5b ENOB loss due to mismatch and kT/C noise.

N	9	10	11	12	13	14
$prec$	4	4	5	6	7	8
s_{LUT}	16	16	32	128	256	512
A_{calib} (mm^2)	6.7×10^{-3}	6.7×10^{-3}	6.8×10^{-3}	7.0×10^{-3}	7.3×10^{-3}	7.7×10^{-3}

By adding the area of the calibration circuit from Table 6.5 with the area of the CAP-DAC from Table 6.4 and comparing it with the CAP-DAC area from Table 6.3, we can see if using the proposed calibration is area-wise justifiable. The result is shown in Table 6.6. As can be seen, calibration is economically acceptable beyond 11 bits of resolution. For a 12b ADC, in particular, employing the proposed calibration would reduce the area of the ADC core by almost 50%[4].

[4]This is based on the fact that the area of the CAP-DAC would be much larger that the rest of the ADC.

Table 6.6 Area of the ADC with and without calibration.

N	A_{DAC1} (mm^2)	$A_{\text{DAC2}} + A_{\text{calib}}$ (mm^2)	Is calibration justifiable?
9	2.5×10^{-4}	6.7×10^{-3}	No
10	1.1×10^{-3}	6.9×10^{-3}	No
11	4.3×10^{-3}	7.7×10^{-3}	No
12	2.4×10^{-2}	1.1×10^{-2}	Yes
13	8.9×10^{-2}	2.2×10^{-2}	Yes
14	3.4×10^{-1}	6.9×10^{-2}	Yes

6.11.2 Effect of Voltage/Temperature Variations

The calibration estimates the random mismatches of the capacitors of the CAP-DAC and presents this as a fractional number. The capacitors are made out of fringe capacitance between different metals (Fig. 6.5). Unless the *distance* between these metals changes, the initial value of the capacitance would stay unchanged. As such, any changes in the supply voltage is not expected to affect the capacitor values, and thereby their mismatches. This results was also confirmed during the measurements.

Simulation results show that the temperature variation changes the absolute value of the MOM capacitors by an amount of about 1% in a temperature range of -40°C to 125°C. The Monte-Carlo simulation also shows that the mismatch characteristic of a capacitor does not change with temperature. This has been approved by measuring the standard deviation of the mismatch, i.e. $\sigma(\Delta C/C)$, and observe that it stays unchanged over temperature. This result was expected as the random mismatch of an MOM capacitor is due to microscopic variations of the edges of the metal plates making it. We thus conclude that temperature variations change the absolute value of all the capacitors of the CAP-DAC by a certain percentage.

On the other hand, the estimated mismatches as fractional numbers represent the amount of surplus charge a certain capacitor within the CAP-DAC contributes to the output compared to its ideal value. As such, those estimated mismatches would still represent the mismatches of the new CAP-DAC of the new temperature. Let us clarify this by a simple example of a 4-bit ADC where the bit-weights of the CAP-DAC are $W_1 = \{1, 1, 1.96, 4.1, 7.94\}$. The calibration algorithm would then compute the mismatches as $E_1 = \{0, 0, -0.0025, +0.00625, -0.0025, \}$, where the mismatch of the third capacitor is, for instance, given by $e = (1.96-2)/(1+1+1.96+4.1+7.96) = -0.0025$ (Eq. (5.16)). Now, at a new temperature, the value of the capacitors would all shift by a certain amount, say 10%. This would result in a new set of capacitor values as $W_2 = \{1.1, 1.1, 2.156, 4.51, 8.734\}$. However the mismatch of the capacitors (normalized to the total capacitor) would still stay the same as

$e = (2.156 - 2.2)/(1.1 + 1.1 + 2.156 + 4.51 + 8.734) = -0.0025$. In other words, both the numerator and the denominator in (5.16) are multiplied by 1.1, leaving the normalized mismatch unchanged.

Chapter 7

Interchannel Mismatch Calibration in Time-Interleaved SAR ADCs

Time interleaving is a technique that allows multiple identical ADCs to operate in parallel and process the data at a faster rate than the operating speed of each individual converter. Fig. 7.1 depicts the block digram of a time-interleaved (TI) ADC . It is composed of M channels [1] that operate in parallel, thereby increasing the overall speed of the ADC by a factor M compared to the speed of one single sub-ADC. As such, the maximum speed of the sub-ADCs is a limiting factor to the maximum achievable speed of the TI ADC. Another limiting factor is the speed of the input sampler. It was proven in [174] that if the sampler is the speed bottleneck, interleaving beyond $0.44(N+1)$ offers little improvement.

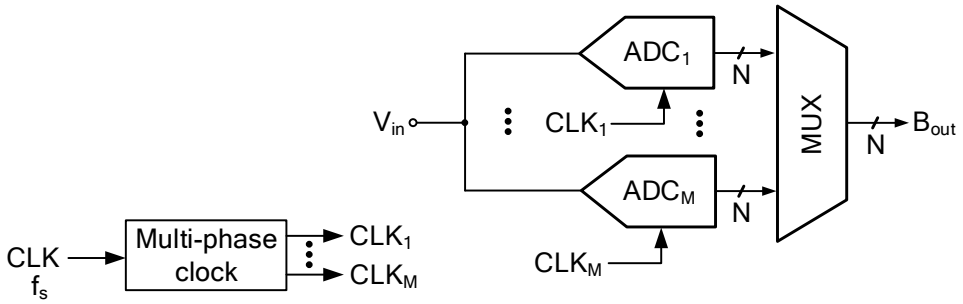


Fig. 7.1 Block digram of a TI ADC. It is composed of M channels in parallel, a multi-phase clock generator and an output multiplexer.

[1] Throughout this chapter, M is referred to the number of channels (sub-ADCs) of the TI ADC. Also, N, f_s, T_s and f_{in} are referred to the resolution of the TI ADC, the sampling rate, the sampling period and the maximum input frequency, respectively.

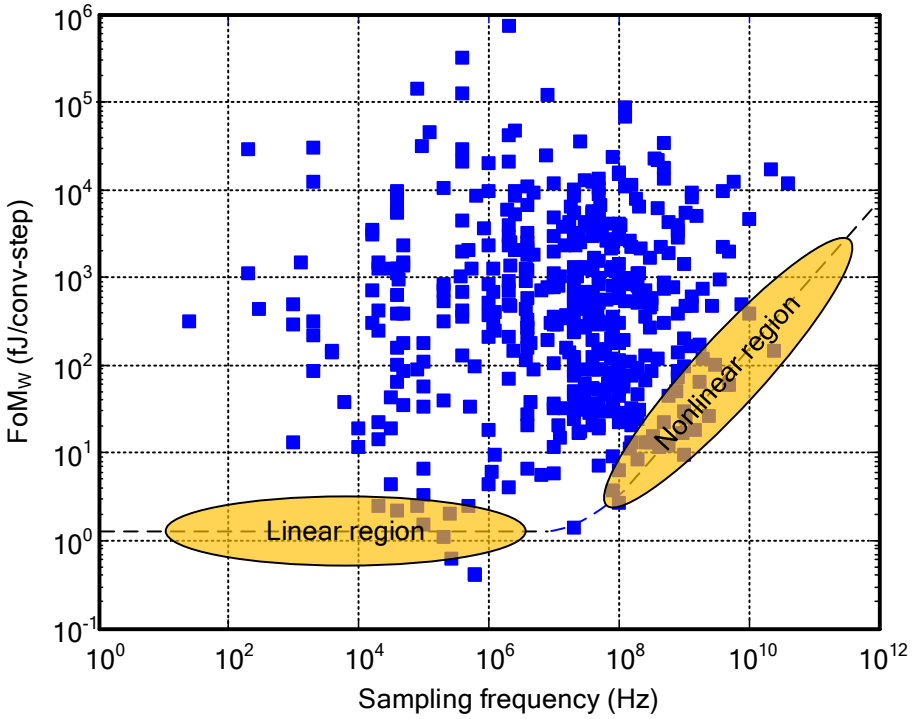


Fig. 7.2 Single-channel ADCs presented at the ISSCC and VLSI from 1997 until 2018.

Time-interleaving can also improve the power efficiency. Walden figure of merit (FoM$_W$) [2], which is defined as

$$\text{FoM} = \frac{P}{2^{\text{ENOB}} \cdot f_s}, \tag{7.1}$$

where P is the power consumption, ENOB is the effective number of bits and f_s is the Nyquist-rate sampling frequency, is typically used to report on the energy efficiency of an ADC. As long as the power consumption of a sub-ADC scales linearly with the sampling speed, the equivalent time-interleaved ADC will always be less energy-efficient than its constituent sub-ADCs because of the overhead associated with interleaving, e.g. the multi-phase clock generation and distribution and the correction of interchannel mismatches. However, as the conversion speed of a sub-ADC approaches the limits of the technology, the power-speed trade-off becomes nonlinear. This can be seen in Fig. 7.2 which illustrates the plot of FoM versus the maximum sampling frequency for all single-channel ADCs presented at the International Solid State Circuits Conference (ISSCC) and at the Symposia on VLSI Technology and Circuits (VLSI) from 1997 until 2018 [3]. Time-interleaving can improve the power-efficiency when the power-speed curve of a single-channel ADC enters the nonlinear region.

Despite the aforementioned advantages of time-interleaving, it comes with some disadvantages as well, most importantly the mismatches between the sub-channels. These mismatches include the gain and offset mismatch, the timing mismatch and the bandwidth mismatch.

A. **Offset Mismatch:** Offset between the sub-channels introduces a fixed pattern noise to the ADC. This pattern is independent of the input in the time domain while in the frequency domain it causes peaks at frequencies of [175]

$$f_{\mathrm{o}} = k \times \frac{f_s}{M}, \tag{7.2}$$

where k is an integer. The SNR degradation due to the offset mismatch does not depend on the input frequency or amplitude.

B. **Gain Mismatch:** Gain mismatch between channels stems from mismatches in reference voltages, transistors and passive components. For a TI SAR ADC in particular, this gain mismatch mainly originates from the mismatch between the total capacitance value of the CAP-DACs of different sub-ADCs. This gain mismatch causes spurs (images) in the ADC output spectrum at frequencies of

$$f_{\mathrm{g}} = \pm f_{\mathrm{in}} + k \times \frac{f_s}{M}. \tag{7.3}$$

The SNR degradation due to the gain mismatch does not depend on the input frequency but, unlike the offset mismatch, it *does* depend on the input amplitude (similar to an amplitude modulation).

C. **Timing Mismatch:** In a TI ADC, the k-th channel ideally samples the input at times $iMT_s + kT_s$. Due to systematic mismatches such as asymmetry of the sampling clock distribution to different channels as well as non-systematic mismatches such as random mismatch of the threshold voltage of the sampling switch, the actual sampling times are skewed from the ideal ones by an amount of, say, Δt_k for the k-th channel. This skew causes noise in the time domain, which is a function of the signal amplitude, just like gain mismatch, but with a phase shift of 90° (similar to a phase modulation). In the frequency domain, the timing skew of the sampling clock causes spurs in the ADC output spectrum at frequencies of

$$f_{\mathrm{t}} = \pm f_{\mathrm{in}} + k \times \frac{f_s}{M}, \tag{7.4}$$

similar to gain mismatch. Yet, unlike gain mismatch, the SNR degrades as f_{in} increases.

D. **Bandwidth Mismatch:** The input S&H circuit of every channel can be modelled by a linear filter with a certain frequency response [176]. Mismatch between the bandwidth of this filter for different channels causes an *ac gain mismatch* and an *ac phase mismatch* [175]. These gain and phase mismatches are different from the gain and mismatch discussed previously in that they are a *nonlinear* function of the input frequency f_{in}.

7.1 Offset Mismatch Calibration

The approach used to estimate the offset mismatch error assumes that the *mean value* of the output of each sub-ADC corresponds to its offset error [177]. Therefore, the simple implementation of Fig. 7.3 can be used to cancel out the offset mismatch of all sub-ADCs. Since offsets of the channels change slowly, a down-sampling can be performed, as can be seen in the figure, merely to reduce the power consumption of this implementation [178]. In extreme cases, e.g. when the input is a sine wave, the offset may not be distinguished from the input signal, in which case, randomization can be employed[177].

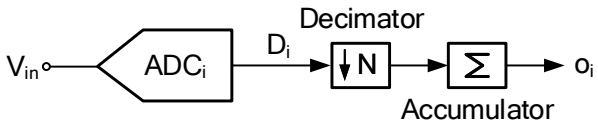


Fig. 7.3 Block diagram of the offset calibration for TI ADCs. This approach simply uses the average of sub-ADCs' output to estimate their offset.

7.2 Gain Mismatch Calibration

The idea for estimation of the gain mismatch error is that the *variance* of the output of each sub-ADC corresponds to its gain error [179, 180]. Fig. 7.4 shows the implementation of the gain error estimation. Again, to reduce the power dissipation, a down-sampling can also be performed.

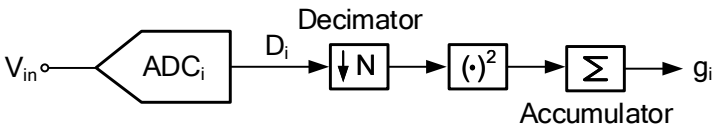


Fig. 7.4 Block diagram of the gain calibration for TI ADCs. This approach uses the variance of sub-ADCs' output to estimate their gain.

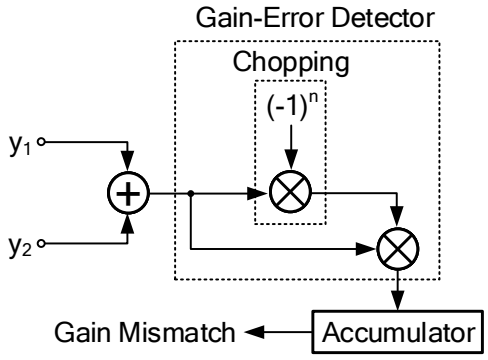


Fig. 7.5 Gain mismatch detection using a frequency-domain method.

Ref. [181] proposed a frequency-domain approach to cancel the gain mismatch in a (M=2)-channel TI ADC. This approach is based on the fact that gain mismatch between the channels in a TI ADC causes spurs (images) in the ADC output spectrum at frequencies of

$$f_{\mathrm{g}} = \pm f_{\mathrm{in}} + k \cdot \frac{f_s}{M}. \tag{7.5}$$

Specifically, for a single input tone at f_0, the gain mismatch causes an image at a certain frequency, say f_i. By applying a *chopping* at the ADC output, one could swap the frequencies at which the main input signal and the gain-mismatch induced image signal occur. This would cause the image to move to f_0 and the input to move to f_i. In the end, multiplying the original ADC output and the chopped output would result in a dc component that is proportional to the amount of the gain mismatch. A block diagram implementation of such a technique is shown in Fig. 7.5. In this figure, y_1 and y_2 are the ADC outputs of channel one and two, respectively.

7.3 Effect of Timing Mismatch

The SNR degradation due timing mismatch in a TI ADC can be analyzed both in the time and the frequency domains. A frequency-domain analysis was discussed in [174] for two channels. Here, we follow the same methodology to work out the effect of timing mismatch for the more general case of a multi-channel interleaved ADC with M channels.

For an input signal $x(t)$ with a bandwidth of f_{in}, the output of an M-channel time-interleaved ADC can be expressed as

$$y(t) = \sum_{i=1}^{M} \sum_{k=-\infty}^{\infty} x\left(kT + \frac{T}{M}i\right) \cdot \delta\left(t - kT - \frac{T}{M}i\right), \tag{7.6}$$

where $T = M/f_s$ is the sampling period of each channel. This means for a Nyquist-rate ADC, $T \leq M/(2f_{in})$. In the frequency domain, (7.6) becomes

$$Y(f) = \frac{1}{T}\sum_{i=1}^{M} X(f) * \left[\sum_{k=-\infty}^{\infty} \delta\left(f - \frac{k}{T}\right)\right] e^{-j2\pi\frac{k}{M}i}. \tag{7.7}$$

Specifically the output of channel i is

$$Y_i(f) = \frac{1}{T} X(f) * \sum_{k=-\infty}^{\infty} \delta\left(f - \frac{k}{T}\right) e^{-j2\pi\frac{k}{M}i}, \tag{7.8}$$

the magnitude of which comprises copies of $X(f)$ with a frequency shift of $1/T = 2f_{in}/M$, as shown in Fig. 7.6. The aliasing seen in the figure will be cancelled out by the re-interleaving multiplexer at the output, provided there is no timing mismatch between the channels.

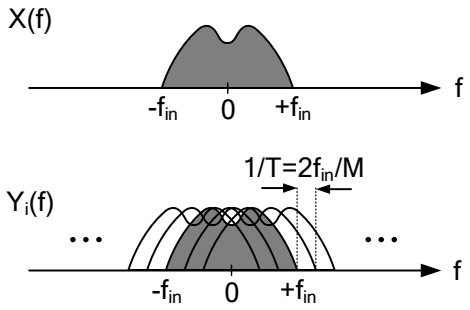


Fig. 7.6 Magnitude of the output spectrum of one channel of an M-channel TI ADC.

An error delay, ΔT_i, in the sampling moment of a specific channel i is equivalent to assuming that the input signal of the channel is delayed by that amount [i.e. $x_i(t) = x(t + \Delta T_i)$]. Assuming that ΔT_i is small, then $x(t + \Delta T_i) \approx x(t) + \Delta T_i \mathrm{d}x/\mathrm{d}t$. This means a (persistent) timing mismatch creates a magnitude error equal to $\Delta T_i \mathrm{d}x/\mathrm{d}t$ in the time domain, and $j2\pi f \Delta T_i X(f)$ in the frequency domain. Eq. (7.8) can thus be adjusted as

$$\begin{aligned} Y_i(f) = &\frac{1}{T}\left[X(f) + j2\pi f \Delta T_i X(f)\right] \\ &* \sum_{k=-\infty}^{\infty} \delta\left(f - \frac{k}{T}\right) e^{-j2\pi\frac{k}{M}i}, \end{aligned} \tag{7.9}$$

The error term in (7.9) can be rewritten as

$$E_i(f) = \frac{1}{T} \cdot j2\pi f \Delta T_i \sum_{k=-\infty}^{\infty} X\left(f - \frac{k}{T}\right) e^{-j2\pi\frac{k}{M}i}. \tag{7.10}$$

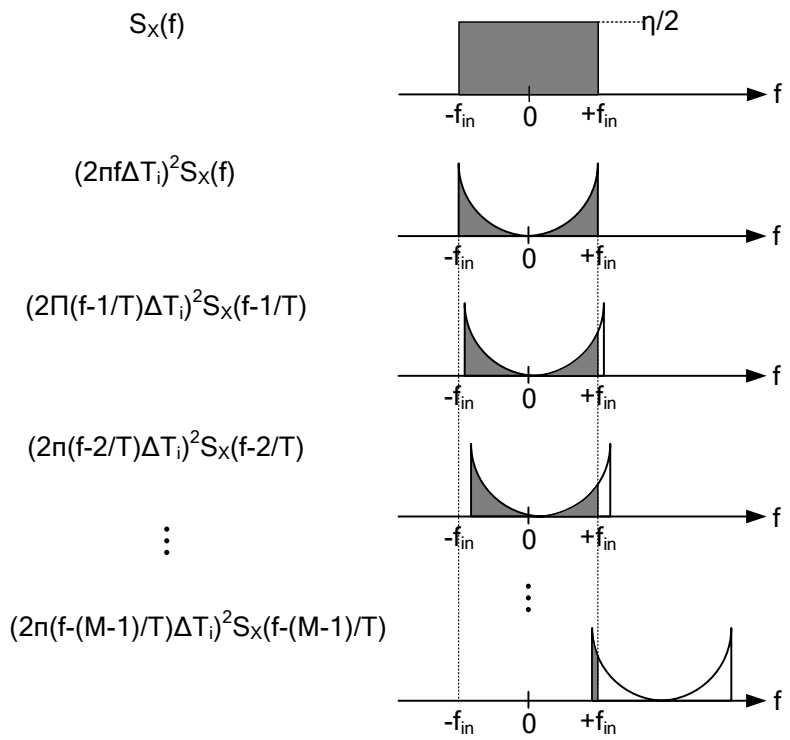


Fig. 7.7 Illustration of the timing mismatch error of one single channel as presented in (7.13) for a random input signal with flat power spectral density.

By integrating (7.10) from $-f_{in}$ to $+f_{in}$, the 'noise' power due to the timing mismatch for the i-th channel can be found as

$$P_{n,i} = \sum_{k=-\infty}^{\infty} \int_{-f_{in}}^{+f_{in}} \left[2\pi(f-\frac{k}{T})\Delta T_i\right]^2 S_X\left(f-\frac{k}{T}\right) df, \tag{7.11}$$

where $S_X(f)$ is the power spectral density of the input signal. Given that the input signal is band-limited within $\pm f_{in}$, and that $1/T \geq 2f_{in}/M$ to satisfy Nyquist condition, the term $S_X(f-k/T)$ in (7.11) can be excluded from the integration for $k \geq M$ and $k \leq M$. Thus, (7.11) becomes

$$P_{n,i} = \sum_{k=-(M-1)}^{M-1} \int_{-f_{in}}^{+f_{in}} [2\pi(f-\frac{k}{T})\Delta T_i]^2 S_X(f-\frac{k}{T}) df \tag{7.12}$$

$$= \int_{-f_{in}}^{+f_{in}} (2\pi f \Delta T_i)^2 S_X(f) df + 2\sum_{k=1}^{M-1} \int_{-f_{in}}^{+f_{in}} [2\pi(f-\frac{k}{T})\Delta T_i]^2 S_X(f-\frac{k}{T}) df. \tag{7.13}$$

Now assuming $S_X(f)$ is flat and equal to $\eta/2$, the terms in (7.13) for $k = 0, ..., M-1$ would look like what is shown in Fig. 7.7, where the shaded area is the area for integration. In order to compute the k-th integral in (7.13), we can write it as

$$P_{n,i}^k = \int_{-f_{in}}^{+f_{in}} [2\pi(f - \frac{k}{T})\Delta T_i]^2 S_X(f - \frac{k}{T}) df \tag{7.14}$$

$$= \frac{\eta}{2} \int_{-f_{in}+k/T}^{+f_{in}} [2\pi(f - \frac{k}{T})\Delta T_i]^2 df. \tag{7.15}$$

By substituting $1/T$ by $2f_{in}/M$ in (7.15) and working out the integral, we obtain

$$P_{n,i}^k = \frac{4}{3}\pi^2 \eta \Delta T_i^2 f_{in}^3 \left[1 - \frac{6k}{M} + \frac{24k^2}{M^2} - \frac{32k^3}{M^3}\right]. \tag{7.16}$$

Therefore,

$$P_{n,i} = P_{n,i}(0) + 2\sum_{k=1}^{M-1} P_{n,i}(k) \tag{7.17}$$

$$= \frac{4}{3}\pi^2 \eta \Delta T_i^2 f_{in}^3 \left[1 + 2\sum_{k=1}^{M-1} \left(1 - \frac{6k}{M} + \frac{24k^2}{M^2} - \frac{32k^3}{M^3}\right)\right] \tag{7.18}$$

$$= \frac{4}{3} M \pi^2 \eta \Delta T_i^2 f_{in}^3. \tag{7.19}$$

And, finally, the total timing mismatch 'noise' power for all the channels is

$$P_n = \sum_{i=1}^{M-1} P_{ni} \tag{7.20}$$

$$= \frac{4}{3} M \pi^2 \eta \left(\sum_{i=1}^{M-1} \Delta T_i^2\right) f_{in}^3. \tag{7.21}$$

The signal power is also equal to

$$P_{\text{sig}} = M^2 \int_{-f_{in}}^{+f_{in}} \frac{\eta}{2} \tag{7.22}$$

$$= M^2 \eta f_{in}. \tag{7.23}$$

From (7.21) and (7.23), the signal-to-noise ratio due to the timing mismatch ($\text{SNR}_{\Delta T}$) can be computed as

$$\mathrm{SNR}_{\Delta t} = \frac{3M}{4\pi^2 \left(\sum_{i=1}^{M-1} \Delta T_i^2\right) f_{in}^2}. \tag{7.24}$$

The term $\sum_{i=1}^{M-1} \Delta T_i^2$ in (7.24) is worth noting. It is known that the sum of the squares of k independent standard normal random variables has a so-called "chi-squared" distribution, $\chi^2(k)$, with k degrees of freedom. Fig. 7.8 depicts the cumulative distribution function (CDF) of the chi-squared distribution for some different values of k. Now, assuming that the timing errors ΔT_is are identical independent random variables with a Gaussian distribution of zero mean and standard deviation $\sigma_{\Delta T}^2$ (i.e. $\Delta T_i = \mathcal{N}(0, \sigma_{\Delta T}^2)$), then the term $\sum_{i=1}^{M-1} \Delta T_i^2$ is a random variable with a distribution of

$$\sum_{i=1}^{M-1} \Delta T_i^2 \sim \sigma_{\Delta T}^2 \chi^2(M-1). \tag{7.25}$$

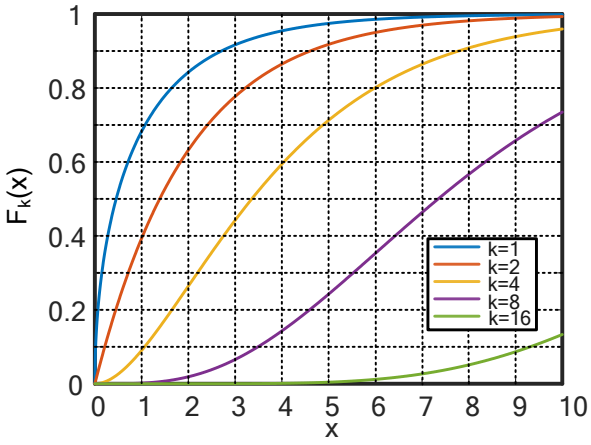


Fig. 7.8 CDF of chi-squared distribution with k degrees of freedom.

Now, we can replace the term $\sum_{i=1}^{M-1} \Delta T_i^2$ in (7.24) with its expected value, knowing that the expected value of a chi-squared random variable with k degrees of freedom is equal to k, that is

$$\mathbf{E}\left[\sum_{i=1}^{M-1} \Delta T_i^2\right] = (M-1)\sigma_{\Delta T}^2. \tag{7.26}$$

By substituting (7.26) into (7.24), we have

$$\mathrm{SNR}_{\Delta t} = \left(\frac{M}{M-1}\right)\left(\frac{3}{4\pi^2\sigma_{\Delta T}^2 f_{in}^2}\right). \tag{7.27}$$

In the end, to compute the SNR degradation due to the timing mismatch in an N-bit ADC, it should be considered along with the quantization noise. The signal-to-noise ratio due to an N-bit quantization (SNR_Q) is equal to

$$\mathrm{SNR}_Q = 3 \times 2^{2N-1}. \tag{7.28}$$

By merging (7.27) and (7.28), we can find the total SNR as

$$\mathrm{SNR}_{\mathrm{tot}} = 1/\left[\frac{1}{\mathrm{SNR}_Q} + \frac{1}{\mathrm{SNR}_{\Delta t}}\right] \tag{7.29}$$

$$= 1/\left[\frac{1}{3 \times 2^{2N-1}} + \frac{4(M-1)}{3M}\pi^2 f_{in}^2 \sigma_{\Delta T}^2\right]. \tag{7.30}$$

Fig. 7.9 depicts the maximum tolerable timing mismatch for a certain SNR degradation for a 16-channel TI ADC for input frequency of $f_{in} = 720$ MHz. For a 10-bit Nyquist-rate ($f_s = 2f_{in}$) TI ADC with a sampling frequency of $f_s = 1.4$ GS/s with 16 channels, (7.30) results in

$$\sigma_{\Delta T} < 160 \text{ fs} \tag{7.31}$$

for a 1-dB SNR penalty. The 16-channel TI ADC with the aforementioned specifications is outlined as the "case-study" throughout this chapter[2].

One method to deal with the timing skew error in interleaved ADCs is to employ a *master* sampler that is followed by the sampling switches of the sub-ADCs [57, 182–184], as shown in Fig. 7.10. The master sampling switch S_M samples the input voltage on the capacitor C_M, as shown in the figure, at a sampling rate of f_s and is driven by the master clock CLK_M. If the sub-ADCs' sampling switches sample while the voltage on C_M is stationary (i.e. when S_M is off), the skew in the sub-ADCs' clocks would not cause any harm. However, there are a few disadvantages associated with this method. First, the available acquisition time for the master switch S_M is half of that of the sub-ADC sampling switches, which makes the design of this master switch challenging, especially for high resolution. Second, and more importantly, upon the closing of the sub-ADC sampling switch, a charge sharing occurs between C_M

[2] As mentioned in the introduction section, these are typical specifications for 5G applications.

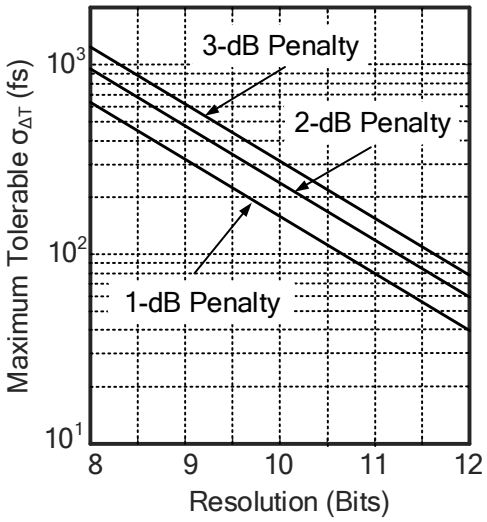


Fig. 7.9 Maximum tolerable timing error for different SNR penalties for a 16-channel TI ADC at $f_{in} = 720$ MHz.

and C_{Si}, where C_{Si} is the sampling capacitance of the i-th sub-ADC (which in the case of a capacitive ADC $C_S \approx C_{\mathrm{DAC}}$). This charge sharing causes input signal attenuation and loss of ADC dynamic range by almost a factor of $\alpha = C_{\mathrm{DAC}}/(C_{\mathrm{DAC}} + C_M)$. To make α much smaller than one, we should have $C_{\mathrm{DAC}} \ll C_M$. This either makes C_{DAC} very small or C_M very big. Due to the kT/C noise and random mismatch limitations, a small C_{DAC} does not easily lend itself to a high resolution ADC. A large C_M also makes the design of the master switch very challenging for high speed and/or high resolution.

Another way of dealing with timing error in TI ADCs is through calibration, which we will discussed in detail in the subsequent sections.

7.4 Timing Mismatch Calibration Techniques

The timing mismatch calibration in interleaved ADCs is performed in two phases: detection and correction. In the following, we discuss the various methods for timing error correction and detection that have been proposed in the literature.

7.4.1 Detection

Foreground

Foreground calibration of the timing mismatch is performed upon the start-up of the chip and before the actual ADC conversion kicks off.

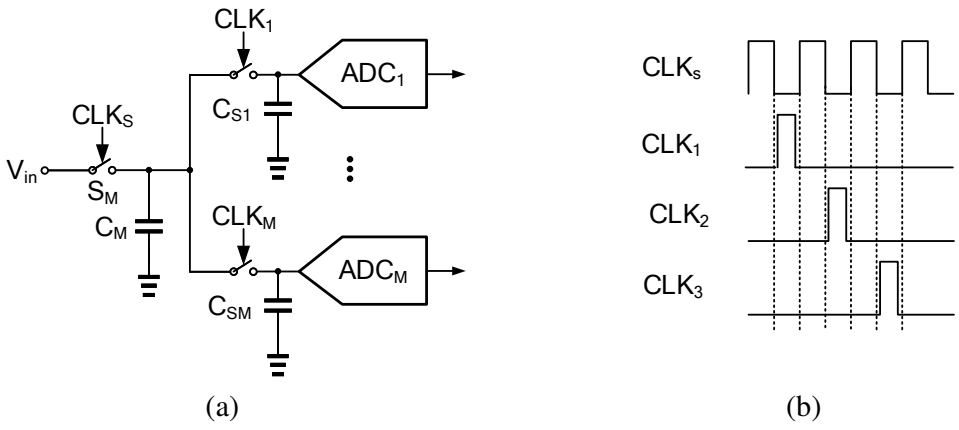


Fig. 7.10 Sampling using a master switch: (a) block diagram, (b) waveforms of the clocks for the first three channels.

One way of performing a timing mismatch calibration in the foreground is to apply a specific known input to the ADC through which the timing errors can be computed. In [185] a *linear ramp* signal was proposed for this purpose. Ideally, if there is no timing error, the sampling instances of the sub-ADCs would sample the input linear ramp in equal intervals, which would result in linearly-spaced ADC outputs. A timing skew in any of the sampling instances would cause these skewed outputs to deviate from the ideal ones. The timing error for channel i would be equal to

$$\Delta T_i = \frac{\Delta D_i}{m}, \tag{7.32}$$

where ΔD_i is the difference between the ideal output of the i-th sub-ADC and 'skewed' one, while m is the slope of the input ramp, as shown in Fig. 7.11. In order to have accuracy of at least 1 LSB, the minimum slope of the ramp signal has to be [185]

$$m_{\min} = \frac{\text{LSB}}{\Delta T}, \tag{7.33}$$

meaning that, in face of a maximum clock timing skew ΔT, the ramp should be steep enough such that the ADC output deviates of at least 1 LSB from its nominal value. However, it can be shown that such a minimum slope results in an input frequency that exceeds the ADC Nyquist frequency, i.e. $f_s/2$. In [47, 186], instead of a ramp, a sinusoidal input of frequency f_s was used. Although generating a sine signal is easier than a ramp, the problem of limited input bandwidth of the ADC still persists. In [187], an input signal of the form

$$V_{\text{in}}(t) = \sum_{i=0}^{K-1} A_i \sin(2\pi f_i t + \theta_i), \tag{7.34}$$

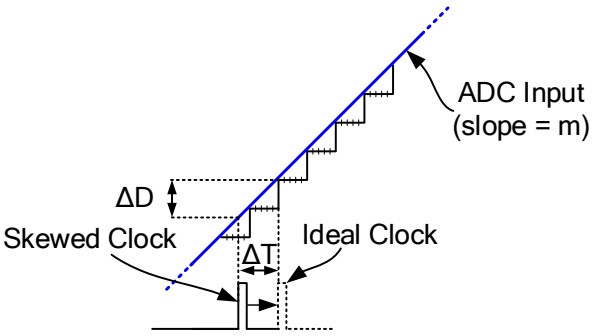


Fig. 7.11 Illustration of timing skew calibration with an input ramp.

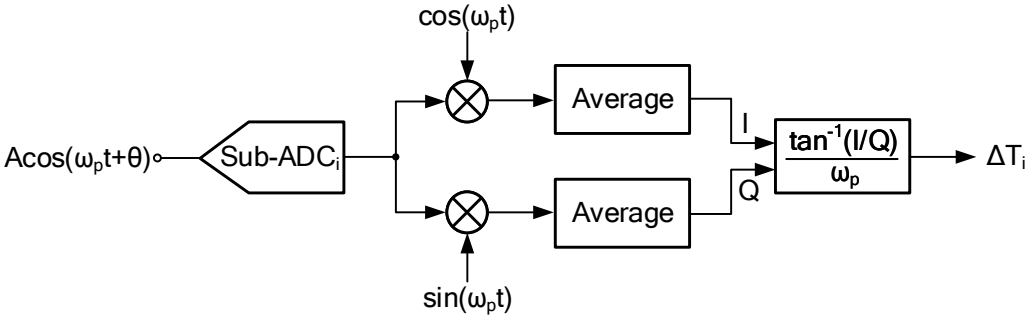


Fig. 7.12 Block diagram of a timing skew calibration technique using a pilot tone as the input.

which consists of K frequency components, was used for the calibration. It was shown in [187] that if $f_i \neq k/2MT_s$ for $k = 0,1,...,M/2$, the mean value of the multiplication of two adjacent sub-ADC outputs with the inputs of (7.34) is related with the timing skew. In particular, this mean value indicates the polarity of the timing error ΔT between the two channels, which is sufficient information to force ΔT to go to zero.

In [178], a pilot tone, $A\cos(2\pi f_p t + \theta)$, was used as the calibration signal, as shown in Fig. 7.12. The digital output of the sub-ADC is then multiplied by a digital I/Q sinusoidal signal with the same frequency as the input frequency f_p.It was shown in [178] that the relative timing mismatch (relative to some specific sub-ADC) of the i-th sub-ADC, ΔT_i, can be computed as

$$\Delta T_i = \frac{\tan^{-1}(Q/I)}{2\pi f_p}. \tag{7.35}$$

The accuracy of (7.59) depends on the accuracy of the digital I/Q sinusoidal waveforms. For high-speed ADCs, the I/Q signals are required to have a very low quantization noise, something that might be difficult to achieve.

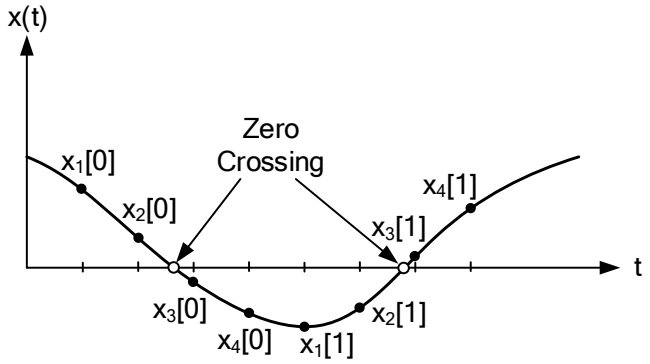


Fig. 7.13 Sampling voltages of a 4-channel TI ADC.

One major problem associated with most of the foreground timing skew calibration, where a specific input is applied to compute the timing error, and is usually overlooked, is the fact that using this technique the offset error of a sub-ADC cannot be distinguished from its timing error. This requires the sub-ADCs to have a very low offset and calls for an accurate offset mismatch calibration circuit.

Another method for foreground detection of timing mismatch is using *zero crossing* [188, 189]. For an input signal $x(t)$ that is continuous in both time and amplitude, a zero crossing occurs when $x(t)$ changes its polarity (at the net of its dc value) from positive to negative, and vice versa. For instance, in Fig. 7.13, which demonstrates the first sampling instances of a 4-channel TI ADC, there is a zero crossing between $x_2[0]$ and $x_3[0]$, and another between $x_2[1]$ and $x_3[1]$. We also define a zero crossing of two consecutive sub-ADCs with outputs $x_i[n]$ and $x_{i+1}[n]$ to be *odd* if $x_i[n]$ appears before the zero crossing and to be *even* if $x_{i+1}[n]$ appears before the zero-crossing. It was shown in [188] that if $x(t)$ is a sinusoidal signal of frequency f_i and if the ratio f_i/f_s is irrational, the average difference between the counts of odd zero crossings and even zero crossings is proportional to the timing mismatch. A simple circuit implementation to detect even and odd zero crossings is shown in Fig. 7.14(a). As already mentioned, by using this technique, the offset of the comparators cannot be distinguished from the timing error. To tackle this issue, [188] proposed the circuit shown in Fig. 7.14.(b), where two high-pass filters are added after the comparators to reduce the effect of the offset.

Background

A number of techniques for background calibration of timing error in interleaved ADCs have been proposed in the literature. Ref. [181] used a frequency-domain method. This is similar to the gain mismatch calibration method discussed in Section 7.2, since it is also based on

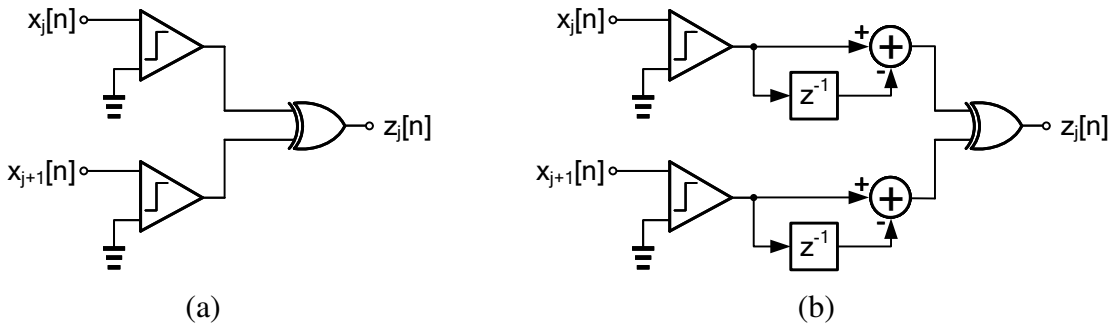


Fig. 7.14 Zero-crossing detector: (a) simple implementation, (b) with high-pass filters to reduce the effect of the comparator offset.

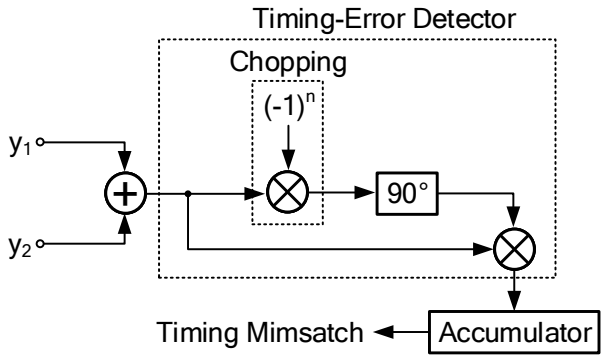


Fig. 7.15 Timing mismatch calibration in the frequency domain.

the fact that the timing mismatch between channels causes spurs at the same frequencies as the gain mismatch does, but with a 90° phase shift. As such, the circuit implementation is very similar to Fig. 7.5 with an addition of a 90° phase-shift all-pass filter, as shown in Fig. 7.15. The realization of the 90° phase-shift is non-trivial. Ref. [181] simply used $H(z) = z^{-1}$, which only has a 90° phase shift at $f = f_s/4$.

One category of the background timing-skew detection techniques is based on finding a *timing function* of the sub-ADCs outputs that is proportional to their corresponding timing error. One such timing function is simply the mean square difference between the outputs of two adjacent sub-ADCs, that is

$$\overline{D_{\Delta T}} = E\left[(y_{i+1}[k] - y_i[k])^2\right], \tag{7.36}$$

where $y_i[k]$ is the k-th output of the i-th channel. It was claimed in [179] that this timing function is proportional to the timing error between the two channels. Intuitively speaking, if the timing interval between the two adjacent sub-ADCs is shorter than $T_s = 1/f_s$, the signal

will change less on average between the samples compared to when this interval is greater than T_s. This claim is only true if the input signal is band limited to the Nyquist frequency, that is $f_s/2$. In [190] a similar timing function was proposed, that is

$$\overline{D_{\Delta T}} = E\left[(y_{i+1}[k] - y_i[k])^2\right] - E\left[(y_{i+2}[k] - y_{i+1}[k])^2\right], \tag{7.37}$$

and it was shown that

$$\overline{D_{\Delta T}} \approx -4\Delta T \frac{\mathrm{d}R_{\mathrm{in}}}{\mathrm{d}\tau}, \tag{7.38}$$

where $\mathrm{d}R_{\mathrm{in}}/\mathrm{d}\tau$ is the derivative of the autocorrelation of the input at the sampling moment. The average of the product of the outputs of two interleaved channels can also be used as a measure of the timing mismatch. This approach, however, fails for some specific cases, e.g. when the input is a single tone sinusoidal signal. In order to get around this problem, [174] proposed to use two products: the product of an odd sample of channel 1 and the next even sample of channel 2, and the product of an even sample of channel 1 and the next odd sample of channel 2. It was shown in [174] that the difference between the two products' average is proportional to the timing error ΔT, that is

$$\overline{D_{\Delta T}} = E\left[y_1[k]y_2[k-1] - y_1[k-1]y_2[k-1]\right] \tag{7.39}$$

$$\approx -2\Delta T \frac{\mathrm{d}R_{\mathrm{in}}}{\mathrm{d}\tau}. \tag{7.40}$$

The block diagram implementation of this technique is illustrated in Fig. 7.16.

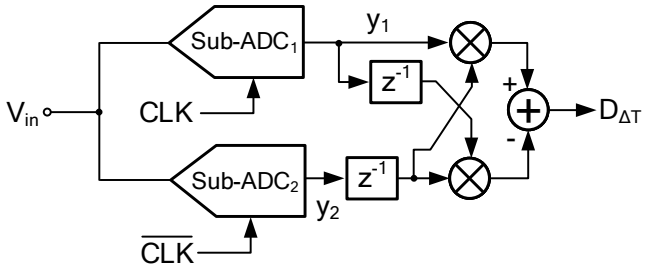


Fig. 7.16 Timing mismatch detection topology based on the output difference of two consecutive channels.

These type of timing functions all rely on one channel to be ideal and detect the timing mismatch of other channels with respect to that ideal channel. This would make the realization of timing function excessively complex as the number of channels increases, since the channels need to be calibrated one by one with making the previous calibrated channels as the reference channel. Also this technique is only possible when the number of channels is a

power of two. One solution to this difficulty in timing function realization is to employ an extra *reference* sub-ADC with respect to which all other sub-ADCs would be calibrated. For instance, [191] used a reference ADC which is identical to the sub-ADCs and its sampling time aligns with that of the sub-ADCs. With no timing error, the outputs of the reference ADC and the corresponding aligned sub-ADC are equal. Otherwise, the remaining code difference can be used as a measure of the timing mismatch for calibrating. In general, if the sampling frequency of the reference ADC is set to f_s/L and

$$\gcd(L,M) = 1, \tag{7.41}$$

where $\gcd(n,m)$ denotes the greatest common divisor of integers n and m, the sampling instance of the reference ADC coincides with that of a specific sub-ADC once for every $L \cdot M$ samples. One issue associated with this simple timing mismatch calibration method is the limited accuracy of the timing error detection. The minimum detectable timing error is equal to

$$\Delta T_{\min} = \mathrm{LSB}/m, \tag{7.42}$$

where m is the slope of the input voltage. For instance, for our case-study ADC with a full-scale range of $1\,V$, LSB is roughly 1 mV and m can be as high as $0.5 \times 2\pi f_{\mathrm{in}} \approx 2.3$ V/ns ($f_{\mathrm{in}} = 720\,\mathrm{MHz}$). This results in $\Delta T_{\min} \approx 430$ fs which is larger than the requirement of (7.31). Using the same technique, [192] achieved a finer resolution of timing error detection by dithering the sampling time of the reference ADC in a pseudorandom manner.

In [193], the cross-correlation between the outputs of the sub-ADCs and the output of the reference ADC at the coincidence of the sampling times (which is basically the autocorrelation of the output signal of each channel) was proposed as a measure of the timing mismatch. It was proved in [193] that for an interleaved ADC, the maximum output SNR is achieved when the autocorrelation of the output of the individual sub-ADCs is maximized, as graphically illustrated in Fig. 7.17. Therefore, to minimize the timing error of one sub-ADC, the cross-correlation of the output of that sub-ADC with the output of the reference ADC should be maximized. The block digram implementation of this technique is shown in Fig. 7.18. One advantage of this technique is that, since the cross-correlation between the outputs of sub-ADC and reference ADC does not require the transfer functions of both ADCs to be identical, it is possible to reduce the resolution of the reference ADC to only one bit (i.e. a single comparator). Moreover, the reference ADC can operate at a lower conversion rate, thus at a lower power consumption, compared to the sub-ADCs. It was mentioned in [194] that this timing mismatch calibration technique does not easily lend itself to high-resolution converters if the input source impedance is finite. Another drawback of this cross-correlation-

based timing mismatch calibration is that it is heavily reliant on the statistics of the input, and requires the input signal to be wide-band stationary (WBS). This is not necessarily the case for all communication channels.

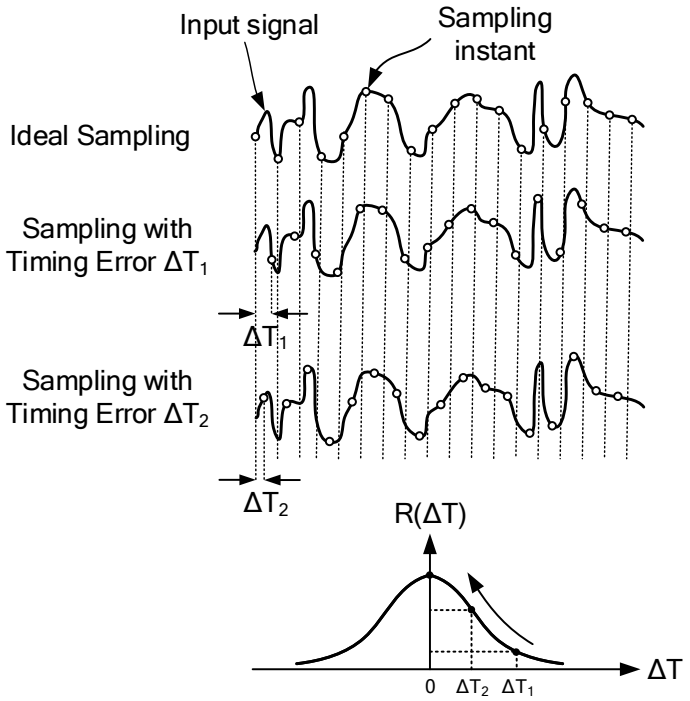


Fig. 7.17 Illustration of the autocorrelation as a function of timing mismatch.

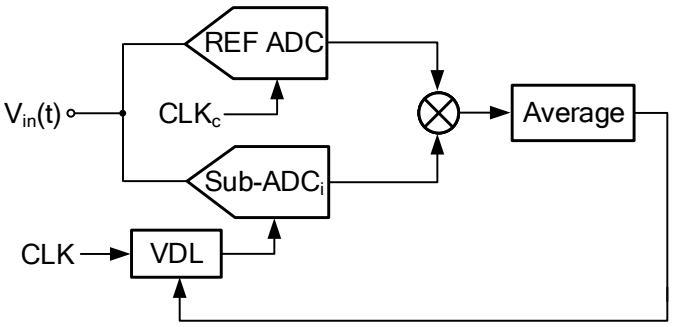


Fig. 7.18 Timing mismatch detection by using the cross-correlation between the sub-ADC and the reference ADC.

A category of the timing mismatch calibration techniques that does not depend on the statistics of the input signal uses a *derivative* of the input voltage to correct the timing error. The core idea of all derivative-based calibration algorithms is that, given an input signal $x(t)$, if the sampled values are skewed by ΔT from the ideal sampling timestamps kT $(k = 0, 1, \ldots)$,

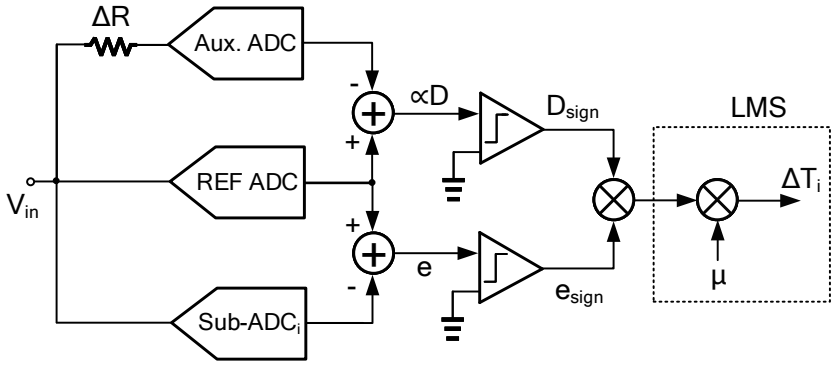


Fig. 7.19 A derivation-based timing mismatch detection scheme along with the LMS implementation.

the sampled signal values can be modeled by the first-order Taylor expansion approximation as

$$x(kT+\Delta T) \approx x(kT) + \Delta T \left.\frac{\mathrm{d}x}{\mathrm{d}t}\right|_{t=kT}. \tag{7.43}$$

The error term thus would be equal to

$$e = D \cdot \Delta T, \tag{7.44}$$

where D is the derivative of the input signal at the sampling moment. If D is known, the term $e \cdot D$ can be used to force ΔT to zero, e.g. via an LMS algorithm. For this technique to work, the sole knowledge of the sign of D would be sufficient (i.e. a sign-sign LMS algorithm). In order to obtain (the sign of) D, [195] proposed to use an auxiliary ADC with a different input RC time constant, as shown in Fig. 7.19. As can be seen, the bandwidth of the auxiliary ADC is intentionally lowered by putting a series resistance of value ΔR in the signal path in order to delay the input signal. By working out the transfer function from the input to the difference of the output voltages of the auxiliary and reference ADCs, it can be shown that the output is proportional to the derivative of the input [195].

Ref. [196] proposed a way of determining the derivative of the input without using an auxiliary ADC, by claiming that the slope

$$m_{\text{avg}} = \frac{y[k+1]-y[k-1]}{2T_s} \tag{7.45}$$

has the same sign as the ideal slope at the sampling instant, as shown in Fig. 7.20. Again, this is only valid as long as the input is within the Nyquist limit. The block diagram implementation of such a method is shown in Fig. 7.21. Here, the subtraction of the outputs

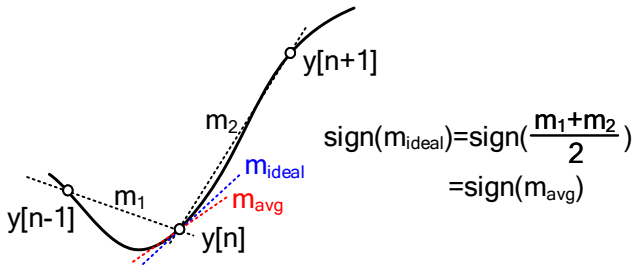


Fig. 7.20 Sign determination of the input derivative at the sampling moment.

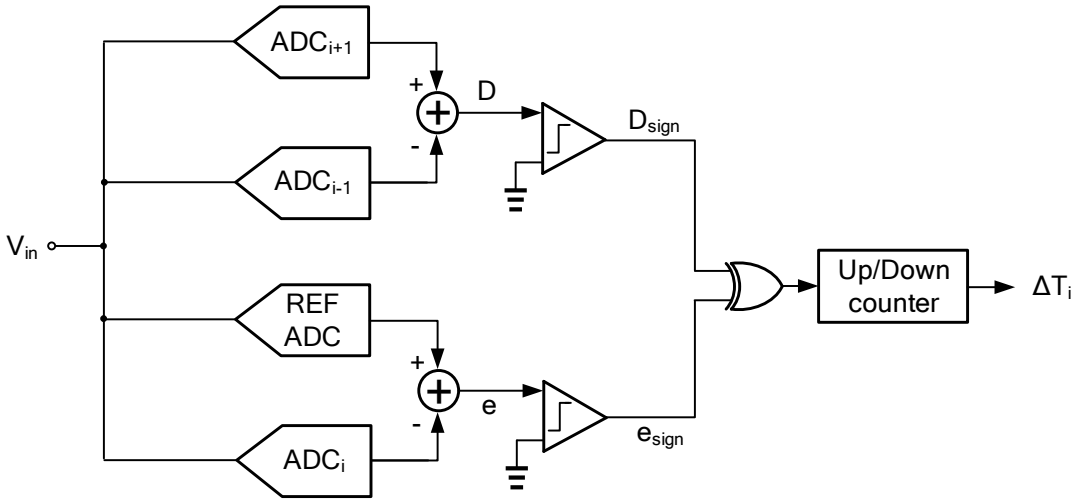


Fig. 7.21 Modification of Fig. 7.19 with the delayed auxiliary ADC replaced by two sub-ADCs and the LSM circuit by a counter.

of sub-ADC$_{i+1}$ and sub-ADC$_{i-1}$ produces an output whose sign is the same as that of the input signal derivative at the sampling instant of sub-ADC$_i$.

In [180], a product of the sub-ADCs output with its derivative is used for the timing error detection as follows. Each sub-ADC output $\tilde{D}_{\text{out}}$ can be seen as a sum of an ideal signal D_{out} and an error term ΔD_{out}: $\tilde{D}_{\text{out}} = D_{\text{out}} + \Delta D_{\text{out}}$. Therefore, the average product of the sub-ADC output with the derivative of the input signal at the sampling time is given by

$$\overline{\tilde{D}_{\text{out}} \times \frac{\mathrm{d}D_{\text{out}}}{\mathrm{d}t}} \approx \overline{(D_{\text{out}} + \Delta D_{\text{out}}) \times \frac{\mathrm{d}D_{\text{out}}}{\mathrm{d}t}} \tag{7.46}$$

$$= \overline{D_{\text{out}} \times \frac{\mathrm{d}D_{\text{out}}}{\mathrm{d}t}} + \overline{\Delta D_{\text{out}} \times \frac{\mathrm{d}D_{\text{out}}}{\mathrm{d}t}}. \tag{7.47}$$

Assuming the input signal is random, it would be orthogonal to its derivative, making the first term in (7.68) zero. Also, according to (7.44), $\Delta D_{\text{out}} \approx \Delta T \times \mathrm{d}D_{\text{out}}/\mathrm{d}t$. Hence, the timing

error can be estimated by

$$\Delta T = \overline{\tilde{D}_{\text{out}} \times \frac{\mathrm{d}D_{\text{out}}}{\mathrm{d}t}} \Big/ \overline{\left(\frac{\mathrm{d}D_{\text{out}}}{\mathrm{d}t}\right)^2}. \tag{7.48}$$

7.4.2 Correction

Analog

Analog correction of the timing error is usually performed using a variable-delay line (VDL). A VDL is basically a digital-to-time converter (DTC) that delays the input with a certain amount that is controlled by a digital code. To adjust the skewed clock in a TI ADC, the clock signal inputs the VDL, is delayed, and outputs as the actual sampling signal that drives the sampling switch. For a TI ADC, a VDL is usually implemented as an inverter with a digitally-controlled variable capacitor (varactor) in a binary-weighted structure at its output [195, 193, 189]. To achieve fine delay resolution for the VDL, the capacitors are implemented using MOSFETs, as shown in Fig. 7.22.(a). In [174], a degenerated inverter is used where the degeneration resistor can be adjusted by a variable resistor, as shown in Fig. 7.22.(b). The drawbacks of the analog correction include the feedback-induced stability hazard, jitter introduced by the VDL itself and the long convergence time.

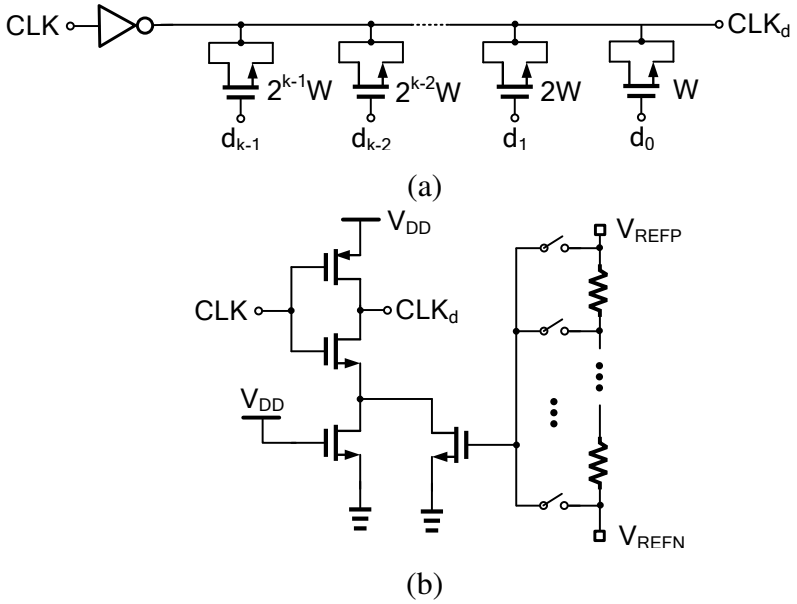


Fig. 7.22 Circuit implementation of a variable-delay line, using (a) binary-weighted MOSCAPs, (b) a resistive DAC.

Digital

Correction in the digital domain can be performed by means of a finite impulse-response (FIR) filter as follows. Let us assume that a sub-ADC is sampled with a sampling timing error of τ seconds, as shown in Fig. 7.23. The output $y[n]$ can be represented, in the z-domain as

$$Z\left\{x[n-\tau/T]\right\} = z^{-\tau/T}X(z), \tag{7.49}$$

and in the Fourier domain as

$$\mathscr{F}\left\{x[n-\tau/T]\right\} = e^{-j\omega\tau/T}X(e^{j\omega}). \tag{7.50}$$

where T is the sampling interval.

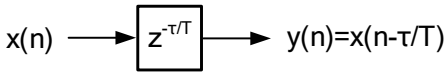


Fig. 7.23 An ideal discrete-time delay system.

In (7.50), the term $e^{-j\omega\tau/T}$ (denoted by H_d) is a 'filter' which represents the delay of the input signal. The impulse response of this filter is readily given by

$$h_d[n] = \frac{\sin\left[\pi\cdot(n-\tau/T)\right]}{\pi\cdot(n-\tau/T)} = \mathrm{sinc}(n-\tau/T). \tag{7.51}$$

When τ/T is an integer, the impulse response of (7.51) reduces to a single impulse at $n=\tau$. For noninteger values of τ, however, the impulse response is an infinitely long, shifted and sampled version of the sinc function, as shown in Fig. 7.24, and usually referred to as *fractional delay filter*. For a practical implementation, this impulse response needs to be approximated by a causal response of finite order. The order of the approximate filter determines the accuracy towards which the timing correction can be performed. The approximation can be done through the least-square-mean (LMS) method or by using Lagrange interpolation [197]. For an M-channel TI ADC, where there are M different delay filters, the correction of the input signal becomes a reconstruction problem of M-order filter bank [198–200]. It is well known that if the average sampling period is shorter than the signal Nyquist rate, a 'perfect' reconstruction to within the limits of the quantizers is possible.

The digital-domain correction takes advantage of the technology scaling but the complex digital reconstruction filter for error acquisition limits the signal bandwidth.

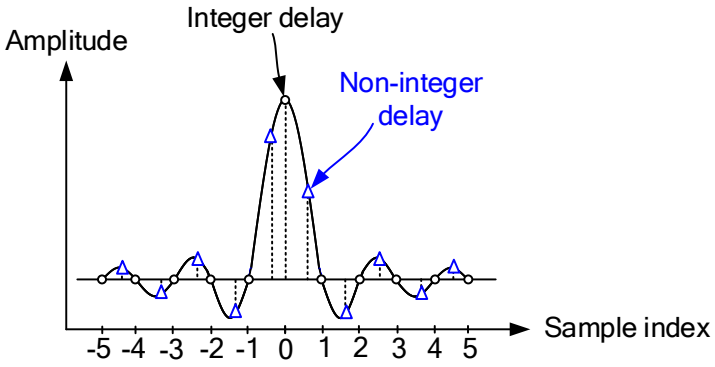


Fig. 7.24 Impulse response of an ideal delay filter with integer and non-integer delay.

7.5 Proposed Calibration Technique

We propose a calibration technique that is not only able to effectively calibrate the timing mismatches between different channels of an interleaved ADC but also, and at the same time, to correct for their offset mismatches. The proposed calibration method is foreground and detects the mismatches in the digital domain.

Fig. 7.25 shows a block diagram of the proposed calibration method for an N-bit M-channel TI ADC. It is composed of the calibration unit, M k-bit digital-to-time converters (DTC) and a dedicated signal generator which is used only during the calibration. The function of the DTCs is to delay the input mismatched clock of each channel by an amount proportional to its k-bit digital input coming from the calibration circuit. Assuming that the DTC is linear and with a resolution of ΔT_d, the maximum covered time range (i.e. the DTC full-scale) is

$$\Delta T_{\mathrm{DTC,FS}} = 2^k \Delta T_d. \tag{7.52}$$

The idea is to have a periodic input signal $V_{in}(t)$ whose frequency f_{in} is chosen such that, in the absence of timing mismatch (i.e. the sampling clocks CLK_1 to CLK_M are progressively phase-shifted by precisely $2\pi/M$ rad), the voltage sampled by *all of the channels* would be the same (denoted with V_0 from now onward). In the presence of timing mismatch, on the contrary, the voltage samples would differ from V_0. By denoting with B_0 the binary representation of V_0 (i.e. the ADCs output code for $V_{in} = V_0$), the calibration unit then checks the N-bit outputs of the ADCs sequentially, and if a given output is not equal to B_0, it adjusts the next sampling instance through the k-bit DTC until the output becomes B_0. The accuracy of such a calibration protocol obviously depends on the resolution of the DTCs (ΔT_d).

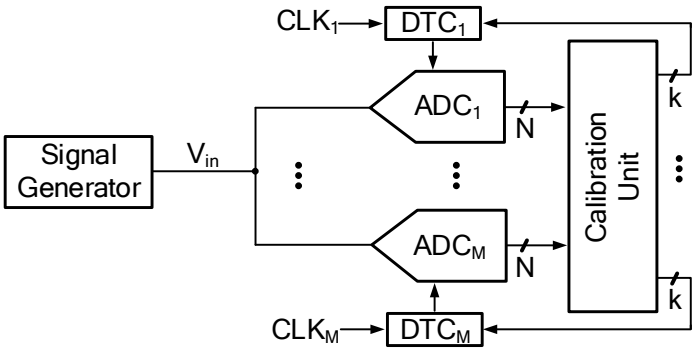


Fig. 7.25 Block diagram of an M-channel TI ADC along with the extra circuitry for the proposed timing-skew calibration.

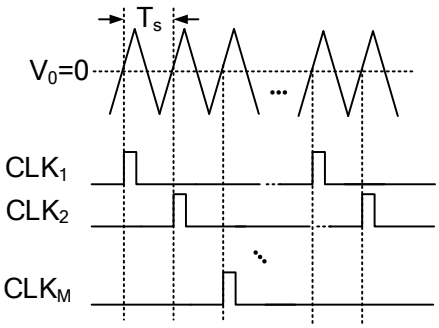


Fig. 7.26 Triangle signal for the purpose of calibration with period of T_s.

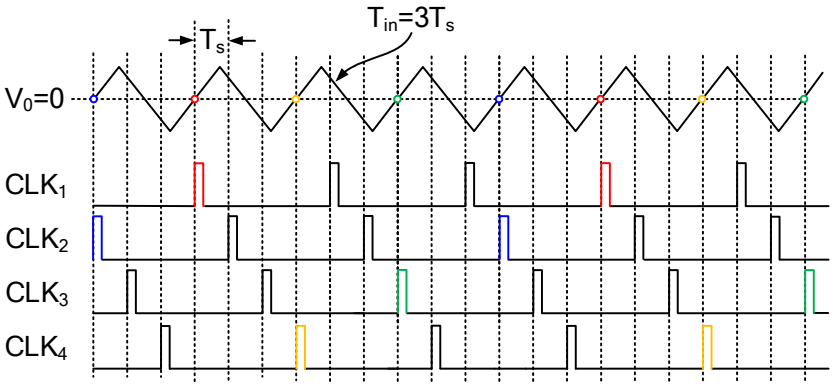


Fig. 7.27 Timing diagram for a 4-channel TI ADC with a triangle signal of period $3T_s$ as the input.

An example of a signal which can be employed for such a purpose is a triangle waveform with a period $T_{in} = 1/f_s$, as shown in Fig. 7.26. The sampling moments coincide with the zero-crossings of the triangle signal ($V_0 = 0$) in an orderly manner. For the ADC to be able to process such a high-frequency input signal, its bandwidth needs to be at least equal to

f_s, which may not represent a practical solution. Therefore, a signal with a lower frequency would be preferred. It can be proven that if $T_{in} = lT_s$, where l is an arbitrary integer and

$$\gcd(l, M) = 1, \tag{7.53}$$

then the triangle signal would still have the desired property. The most straightforward choice is $l = M - 1$, for which the sampling instant of each channel coincides with the zero-crossing once in every M samples in a periodic pattern. Fig. 7.27 illustrates this for the specific case of $M = 4$. The calibration unit then needs to only look at every M-th ADC output of a specific channel to check whether it is equal to B_0; if not, it will adjust the sampling clock through the corresponding DTC until the ADC output converges to B_0. In practice, however, this turns out to be a bit more complicated since the ADC may not change its output away from the nominal value B_0 even in the presence of timing mismatch errors which are (much) larger than the requirement imposed by (7.31). To understand this, let us assume the triangle input calibration signal has a peak-to-peak voltage amplitude of $V_{in,pp}$ and period $T_{in} = (M-1)/f_s$, thus a slope of

$$m = \frac{V_{in,pp}}{T_{in}/2} = \frac{2V_{in,pp}f_s}{M-1} \tag{7.54}$$

The output of an N-bit ADC with a full-scale of V_{FS} (hence, the LSB amplitude $V_{\mathrm{LSB}} = V_{\mathrm{FS}}/2^N$) and whose input is such a triangle signal would not change for a time duration of

$$\Delta T_{\mathrm{LSB}} = V_{\mathrm{LSB}}/m = \frac{V_{FS}}{V_{in,pp}} \cdot \frac{(M-1)}{2^{N+1} f_s} \tag{7.55}$$

For our case-study ADC ($N = 10$, $M = 15$, $f_s = 1.4\,\mathrm{GHz}$, $V_{FS} = 1\,\mathrm{V}$) and for $V_{in,pp} = 0.5\,\mathrm{V}$, ΔT_{LSB} would be equal to 10.4 ps, which is far larger than the requirement set by (7.31). In other words, this means that, in the worst case, any timing mismatch below this value would not be detected (thus corrected) by the calibration algorithm. This issue can be readily solved as follows. Let us assume that $\pm\Delta T_{\max}$ is the maximum timing-skew that can happen due to both systematic and non-systematic timing mismatches. According to (7.54), this is equivalent to

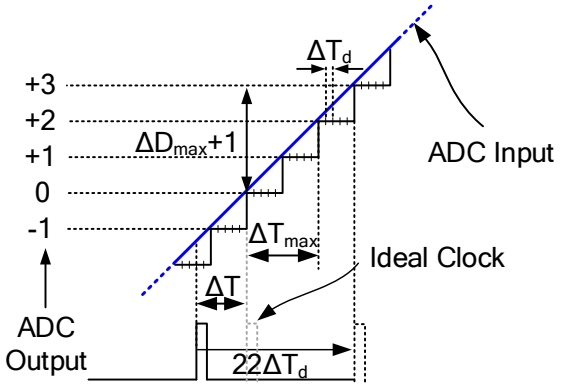
$$\Delta V_{\max} = m \cdot \Delta T_{\max} = \frac{\Delta T_{\max}}{T_{in}/2} V_{in,pp}. \tag{7.56}$$

This maximum timing mismatch should be known beforehand as it determines the full-scale range of the DTC's ($\Delta T_{\mathrm{DTC,FS}}$) which are employed during the calibration to properly skew the clock. Let us also denote with $\Delta D_{\max}$ the quantized ADC output which corresponds to

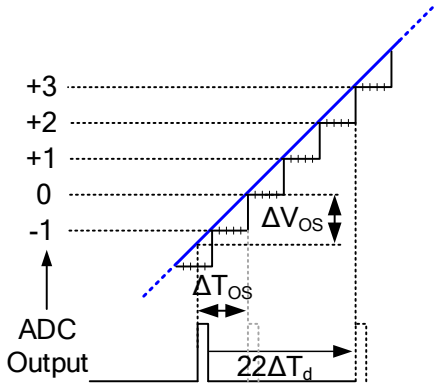
$\Delta V_{\max}$, i.e:

$$\Delta D_{\max} = Q(\Delta V_{\max}/V_{\text{LSB}}). \tag{7.57}$$

Clearly, if $\Delta T_{\max} < \Delta T_{\text{LSB}}$, $\Delta D_{\max} = 0$. Now, what the calibration unit must do to calibrate the timing mismatch of the i-th channel of the TI ADC is to progressively increase the delay of CLK_i by means of DTC_i so that the output of ADC_i changes by $\Delta D_{\max} + 1$ compared to its initial value. If this is done for all of the sub-ADCs, the sampling clocks will be equally distanced within an error of ΔT_d (the resolution of the DTC).



(a)



(b)

Fig. 7.28 Timing-skew calibration: (a) without offset, (b) with offset, where ΔT and ΔV_{os} are the timing and offset error, respectively.

Fig. 7.28(a) illustrates this for the case of $\Delta D_{\max} = 2$. As can be seen, the sampling signal is ΔT seconds skewed from its ideal point which results in an ADC output of -1 before

calibration. The calibration algorithm progressively delays the sampling clock until the ADC output becomes $\Delta D_{\max} + 1 = +3$. The amount of the time shift incurred by the clock in this example is $22\Delta T_d$.

Thus far, it has been assumed that the offset of all of the channels was zero. The offset of a channel is also translated into timing mismatch through the voltage-to-time conversion via the ramp slope, that is

$$\Delta T_{\text{os}} = \Delta V_{\text{os}}/m, \tag{7.58}$$

where ΔV_{os} and ΔT_{os} are the offset voltage and the offset-induced timing mismatch[3]. However, this equivalent timing error caused by the offset cannot be ultimately distinguished from the actual timing mismatch by the proposed calibration method. This is illustrated in Fig. 7.28(b), which shows the case where an offset mismatch of ΔV_{os} is present. As can be seen, this is exactly equivalent to the case of Fig. 7.28(a) even though no timing mismatch exists.

In order to solve this issue, a triangle signal with frequency of $f_{\text{in}} = f_s/(M+1)$ is employed. Such frequency still satisfies the above-mentioned requirements, thus ensuring the proper operation of the proposed calibration method. However, the TI-ADC channels now sample the zero-crossing points on both the rising and falling edges of the triangle signal, as shown in Fig. 7.29 for the specific case of $M = 3$. The key point here is that a timing mismatch would cause an error $\delta V_{e,r}$ in the sampled voltage on the rising edge which has *opposite* sign compared to the error $\delta V_{e,f}$ in the sampled voltage on the falling edge, i.e. $\delta V_{e,r} > 0$ and $\delta V_{e,f} < 0$ in the case the real clock timestamp is lagging the ideal one. On the contrary, the offset error would cause a sampling voltage error with the *same* sign on both edges (i.e. both $\delta V_{e,r}$ and $\delta V_{e,f} > 0$ in the presence of $\Delta V_{OS} > 0$), thus making it possible for the calibration algorithm to distinguish between them. In other terms, the difference x_{calib} between the ADC output associated with a sample on the rising edge of the ramp (henceforth called D_r) and a sample on the falling edge of the ramp (D_f) is free from the ADC offset and retains only the timing error information. This variable is used as an input of the calibration unit

Fig. 7.30 illustrates this new sampling scenario in the absence of offset mismatch, and for a timing mismatch $\Delta T = n_d \Delta T_d$, with n_d being an integer (in the example, the real clock timestamp occurs n_d times the DTC resolution step earlier than the nominal one). Under the assumption that the rising slope is exactly the opposite of the falling slope, $D_r = -D_f$ (with respect to the mid-code). The calibration would then delay the clock with steps of ΔT_d by progressively incrementing the DTC input code by 1, while reading out

[3]Note that ΔT_{os} is not the timing mismatch of the actual clock, but the equivalent timing mismatch which would cause the output of an offset-free ADC to be ΔV_{os}.

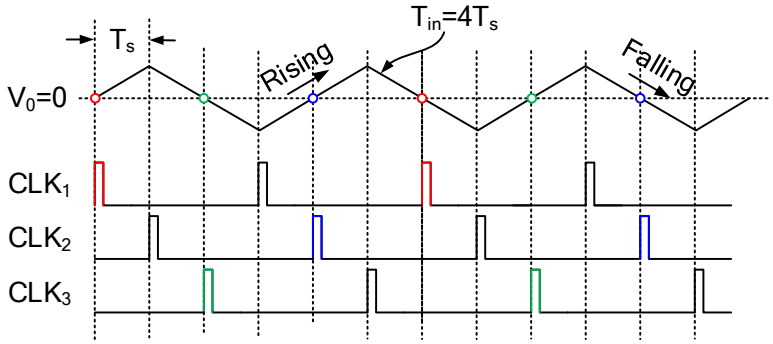


Fig. 7.29 Timing diagram for a 3-channel TI ADC with a triangle signal of period $4T_s$ as the input.

$x_{\text{calib}} = D_r - D_f = 2D_r$. Whenever the clock experiences enough delay such that the sampled ADC input crosses a quantization level ($D_r + 1$ in the case of rising slope of the triangle waveform and $D_f - 1$ for the falling slope), x_{calib} increments by 2 (or decrements by 2 in the case the real clock lags the ideal one). As before, the calibration stops when $D_r = \Delta D_{\max} + 1$ or, equivalently, when

$$x_{\text{calib}} = 2\Delta D_{\max} + 2. \tag{7.59}$$

Assuming that $\Delta T_{\text{LSB}} = n_{\text{LSB}} \Delta T_d$ is the minimum time shift of the clock such that the ADC output changes by 1 (n_{LSB} is an integer), the initial value of D_r would be

$$D_{r0} = \lfloor \frac{-n_d}{n_{\text{LSB}}} \rfloor, \tag{7.60}$$

where $\lfloor y \rfloor$ is the floor function of the real number y. As the calibration proceeds in the manner described above, a sequence of linearly progressive numbers would then appear at the ADC output, as shown in Fig. 7.30(b). Here, n_0 is given by

$$n_0 \equiv n_d \mod n_{\text{LSB}}, \tag{7.61}$$

meaning n_d is the modulo of n_0 divided by n_{LSB}.

On the other hand, in face of an input-referred offset of ΔV, as shown in Fig. 7.31(a), a different pattern of numbers appears at D_r and D_f. This is illustrated in Fig. 7.31(b), where

$$n_{os} = \lfloor \frac{m\Delta V}{\Delta T_d} \rfloor. \tag{7.62}$$

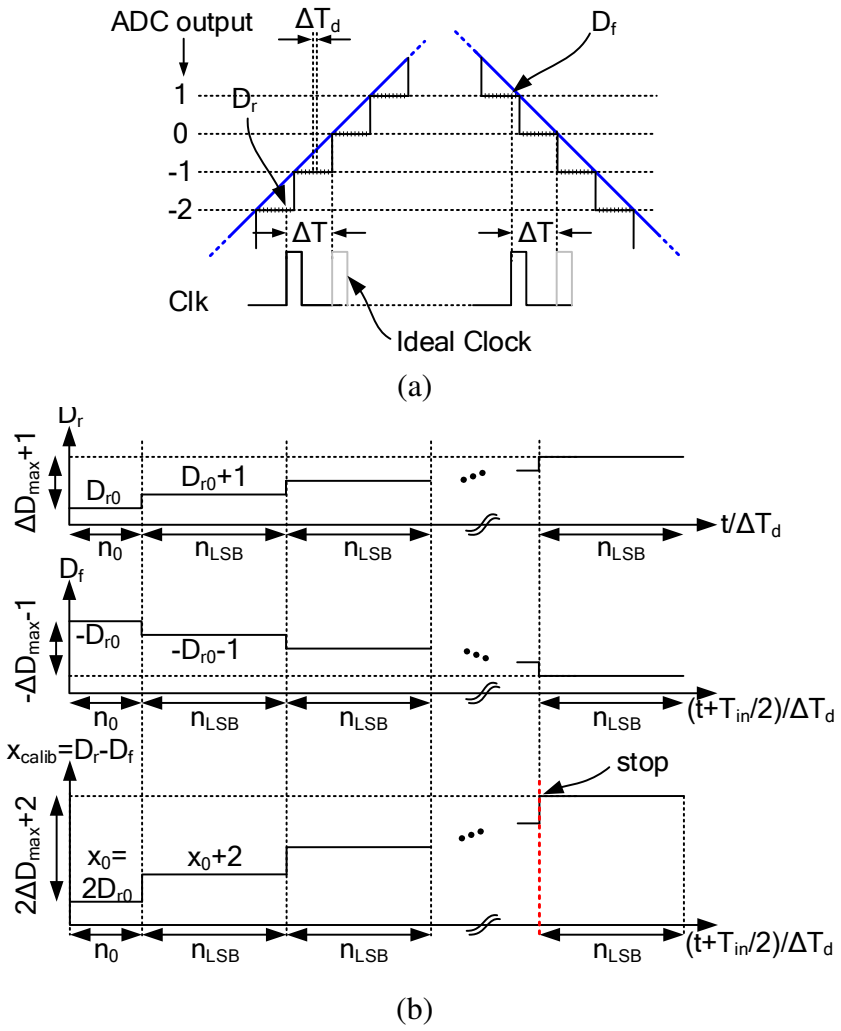
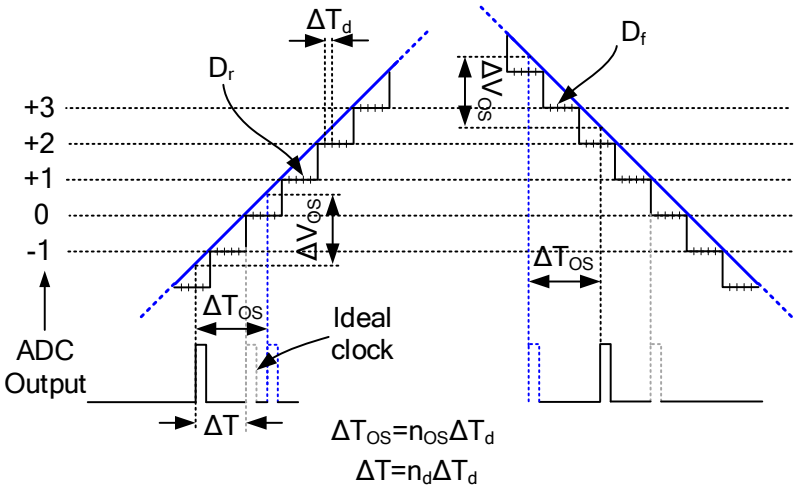
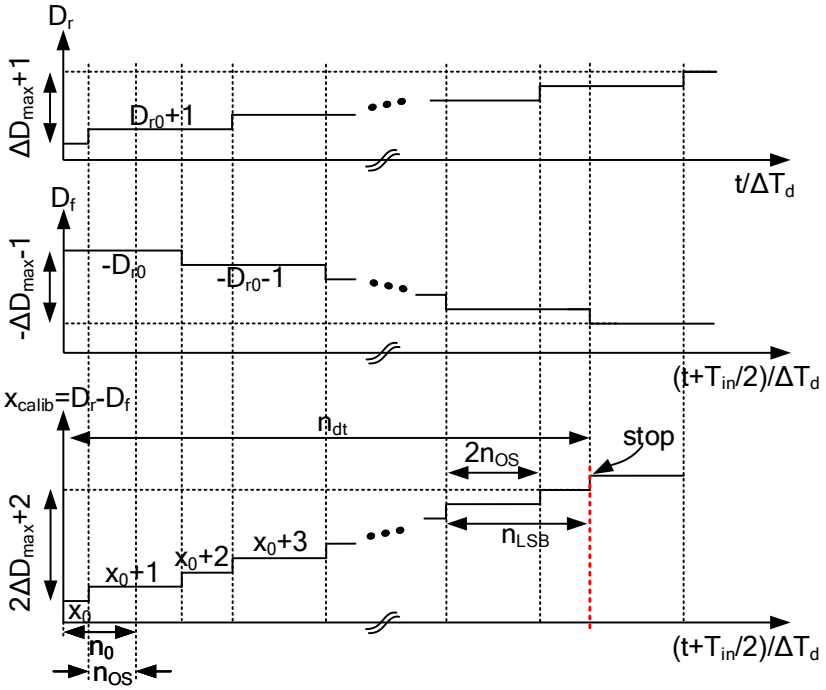


Fig. 7.30 (a) Position of the clock signal on the triangle waveform for a timing error of ΔT and zero offset error. (b) the sequence of numbers appearing at the ADC output during the calibration



(a)



(b)

Fig. 7.31 (a) Position of the clock signal on the triangle waveform for a timing error of ΔT and an offset error of ΔV. (b) the sequence of numbers appearing at the ADC output during the calibration

As can be observed, an asymmetry is formed in the sequences of D_r and D_f (i.e. D_r is not -D_f any more) by an amount of n_{os}, which then results in a sequence of $2\Delta D_{\max}$ with length $2n_{os}$ at x_{calib}. Therefore, the calibration unit should keep track of the last sequence of numbers before it terminates the calibration and register its length as $2n_{os}$, and the total length of the last two sequences of numbers as n_{LSB}, as depicted in Fig. 7.31(b). The calibration process ends when x_{calib} becomes equal to $2\Delta D_{\max}+1$. The calibration unit also keeps track of the number of DTC steps from the beginning until the end, which represents the total length of the entire sequence x_{calib}, hereinafter referred to as n_{dt}. The timing and offset mismatch can then be computed as:

$$\Delta T = \Delta D_{\max} \cdot n_{\text{LSB}} - (n_{dt} - n_{os}) \tag{7.63}$$

$$\Delta V = \pm n_{os}/n_{\text{LSB}}. \tag{7.64}$$

The sign of ΔV depends on which variable, whether D_r or D_f, becomes $\Delta D_{\max}$ first, which is also recorded.

In practice, the ideal clock may not be exactly aligned with the zero-crossings of the input triangle waveform upon the start of calibration. This, however, only represents a constant offset common to all of the TI-ADC channels and would therefore not affect the validity of equations (7.63) and (7.64).

7.6 Nonidealities

7.6.1 Triangle Signal Non-Linearity

The triangle signal can be generated by an integrator with a square wave input signal of frequency $f_{in} = f_s/(M+1)$. Thus far, it has been assumed that the positive slope of the triangle signal is equal to the negative slope. It has been also assumed that the positive and negative edges are straight lines. Yet, this requirement should only be fulfilled for a voltage range of $\Delta V_{\max} = m \cdot \Delta T_{\max}$. Since $\Delta T_{\max}$, which is the maximum timing skew of the ADC, is usually in the order of tens of ps, $\Delta V_{\max}$ is only few millivolts. It is then reasonable to assume that the triangle waveform is linear within this small voltage range. Moreover, given that the triangle waveform generated by the integrator is more linear around its mid-range (i.e. zero-crossings), it is convenient, before the calibration is asserted, to internally time-shift the waveform using coarse time steps, thus allowing the sampling clock timestamps to occur within such linear voltage region.

In high-speed ADCs, the design of a linear integrator, which demands a high-gain op-amp, is not trivial, especially in the case of low supply voltage (e.g. below 1 V). A full-scale sinusoidal signal can represent a more practical alternative to the triangle waveform, since it approximates well a ramp within the very small voltage range of calibration (compared to the full-scale range of the ADC). Of course, such an approximation introduces an error, which is however much smaller compared to the requirement set by (7.31) for the proposed case-study.

7.6.2 Capacitive DAC and DTC Non-Linearity

So far, we have assumed that the capacitive DAC (CAP-DAC) of each SAR-ADC channel is perfectly linear (i.e. zero DNL and INL). Unfortunately, and unlike with the offset mismatch, the non-linearity of the CAP-DAC, which directly impacts on the whole ADC non-linearity, cannot be distinguished from the timing mismatch errors. Quantitatively, a DAC INL error of ΔV_{INL} is equivalent to a timing mismatch of

$$\Delta T = \Delta V_{\mathrm{INL}}/m, \tag{7.65}$$

or equivalently

$$\sigma_{\Delta T} = \sigma_{\mathrm{INL}}/m, \tag{7.66}$$

which however cannot be calibrated by the proposed timing skew calibration algorithm, as explained in the following. Per our earlier discussion, the sampling clock timestamps occur around the zero-crossings of the input signal (either a triangle wave or a sinewave) during the calibration. This corresponds to the ADC's mid-scale, which is also the level at which the CAP-DAC exhibits greater nonlinearity (i.e. maximum INL). The standard deviation of this maximum INL is given by

$$\sigma_{\mathrm{INL}} = \frac{1}{2}\sigma_0\sqrt{2^N - 1}\, V_{\mathrm{LSB}}, \tag{7.67}$$

where σ_0 is the relative standard deviation of an LSB capacitor of the CAP-DAC [i.e. $\sigma_{(\Delta C_u/C_u)}$].

By substituting (7.67) into (7.66), we have

$$\sigma_{\Delta T} = \frac{1}{2}\sigma_0\sqrt{2^N-1}\,V_{\mathrm{LSB}}/m = \frac{1}{2}\sigma_0\sqrt{2^N-1}\,\Delta T_{\mathrm{LSB}}. \tag{7.68}$$

In advanced technologies, σ_0 is well below 1% for a 1 fF MOM capacitor. Let us now assume that, for the case-study ADC, the binary DAC has a unity capacitor of 1 fF; this would result in an equivalent timing mismatch [due to the INL error derived in (7.68)] of about 1.2 ps, which largely violates (7.31).

The proposed solution to overcome such an issue relies on the fact that only a small portion of the whole ADC full-scale range (and hence of the full-scale of the DAC) is exercised during the calibration. Indeed, assuming that

$$\Delta T_{\max} = \alpha \Delta T_{\mathrm{LSB}}, \tag{7.69}$$

an ADC with a resolution of $\log_2(\alpha+1)$-bit would be enough to cover the maximum timing error for the calibration. $\Delta T_{\max}$ is in order of a few tens of picoseconds, which makes α less than 4 for our case-study ADC, where ΔT_{LSB} = 7.6 ps. Therefore, a 3-bit ADC would be sufficient for the timing-skew calibration to be functional. The advantage here is that the equivalent timing error induced by the ADC INL error [as per (7.68)] would now be significantly lower. For the case of $N = 3$, the equivalent timing mismatch would only be equal to 100 fs, which satisfies the minimum timing requirement of (7.31). From a practical perspective, it is straightforward to convert a 10-bit SAR ADC to a 3-bit one, which is achieved by involving in the conversion only the 3 LSB bits of the DAC, while freezing the rest of the bits [4].

The DTC nonlinearity manifests itself in the form of an accumulated output error when its input is swept by the calibration unit. If the DTC has a maximum peak-to-peak INL of $\mathrm{INL}_{\mathrm{DTC,max}}$, the maximum output error of the DTC cannot be larger than $\mathrm{INL}_{\mathrm{DTC,max}}$. This essentially means that the timing mismatch is going to be calibrated by a timing resolution of $\Delta T_d + \mathrm{INL}_{\mathrm{DTC,max}}$ rather than ΔT_d

The noise of the ADC plus the noise of the waveform generator can be easily cancelled out by averaging. This means the input waveform can be read out by the ADC multiple times and then averaged out.

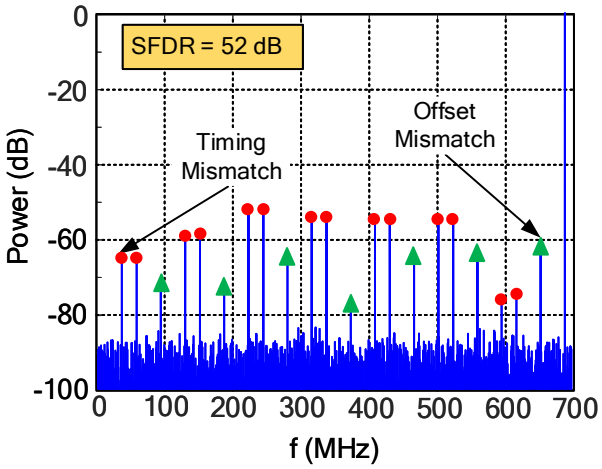
[4] The same can be done for a pipelined SAR ADC by freezing the first stages and the first few bits of the last stage.

7.7 Simulation Results

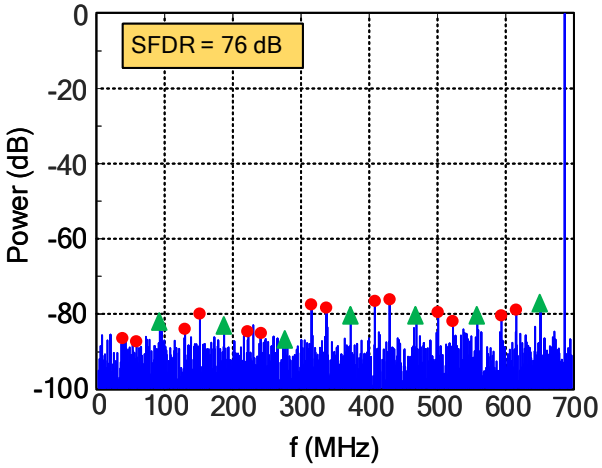
The proposed calibration technique has been verified through simulations by using algorithmic models for both the calibration unit and the constituent blocks of the core ADC. The outlined modeling case scenario is a 1.4 GS/s 10-bit 15-channel[5] time-interleaved ADC. The sub-ADCs are modeled as conventional SAR ADC converters with binary-weighted capacitive DAC. A random mismatch normal distribution is added to the DAC capacitors, assuming a standard deviation of the unit capacitance (i.e. the LSB capacitor) equal to 1% of its nominal value C_u. The standard deviation of the other capacitors (i.e. from the LSB + 1 up to the MSB) increases proportionally to the square root of their value, i.e. $\sigma_C = 1\% \cdot \sqrt{C \cdot C_u}$.

Timing and offset mismatches are modeled as normally-distributed random variables with standard deviation of 5 ps and 2 LSB, respectively. A noise of 1 $\mathrm{mV_{RMS}}$ is also added to the input so as to account for the input-referred noise of the ADC, the kT/C noise and the noise of the input signal. The input signal used for calibration is a *sinewave* with a frequency of $f_{\mathrm{in}} = f_s/16$. The resolution of the calibration, which is basically the resolution of the DTC, is set to 100 fs. Random mismatch is also added to the DTC, in such a way that the maximum peak-to-peak INL is about 1LSB (i.e. 100 fs). A η_{avg}=256× averaging is performed to diminish the effect of random noise. Finally, a 100-run Monte-Carlo simulation is executed based on the aforementioned characteristics and we report worse-case results in terms of the output SNDR. The results are depicted in Fig. 7.32 for an input frequency of f_{in}=689 MHz. As can be seen, the SFDR improves by 24 dB, thus proving the effectiveness of the proposed calibration technique. It is straightforward to demonstrate that the total calibration takes about $n_{\mathrm{LSB}} \cdot (\Delta D_{\max} + 1) \cdot M \cdot n_{\mathrm{avg}} \cdot T_s$ seconds, where $n_{\mathrm{LSB}} = \Delta T_{\mathrm{LSB}}/\Delta T_d$ and ΔT_{LSB}, $\Delta D_{\max}$ and ΔT_d are given by (7.55), (7.57) and (7.52), respectively. For the analyzed test-case, this time duration is below 1 ms.

[5] As mentioned in Sec.7.5, the calibration signal period T_{in} should be M+1 times the sampling clock period [i.e. $f_{in} = f_s/(M+1)$]. Since a frequency division by a factor of 2 is practically straightforward, the interleaving factor M was set to 15.



(a)



(b)

Fig. 7.32 Output SFDR (a) before calibration (b) after calibration.

Chapter 8

Conclusion

8.1 Thesis Contribution

This dissertation has investigated design, as well as precision techniques for designing high performance SAR ADCs, specifically for 5G applications. This requires an ADC with a resolution of 10 bits and an operation speed of $1 \sim 2$GS/s. This thesis makes three main contributions to this field.

Firstly, an extensive investigation of the literature on design techniques for high-performance SAR A/D converters. This includes the most important calibration techniques to correct the mismatch of the capacitors of the capacitive DAC in a SAR ADC, as well as to detect-correct the timing mismatch in a TI ADC.

Secondly, proposed a fully-automated mismatch calibration method that is able to precisely compute the mismatch of the capacitors used in the CAP-DAC of a SAR ADC. The mismatch calibration allowed us to scale down the unit capacitor to as low as 280aF for a 9-bit DAC, thereby saving energy and boosting speed. To validate the proposed idea, a 10-bit asynchronous SAR ADC was fabricated in 28-nm CMOS. Measurement results showed an ENOB of 9.14 bit at a sampling rate of 85 MS/s, resulting in a Walden FoM of 10.9fJ/c-s. The mismatch calibration proved effective by improving the INL, the SFDR and the SNDR by 6.4 LSB, 14.9dB and 11.5dB, respectively.

Thirdly, proposed a foreground timing-skew calibration technique for interleaved SAR ADCs. The proposed method is also able to compute the offset mismatch between the sub-ADCs as well. A MATLAB model for a 10-bit 85MS/s SAR ADC was used to build a 15-channel TI ADC in order to validate the proposed timing mismatch calibration technique. The simulation results showed that this calibration technique effectively suppressed the spurs due to both the timing mismatch and offset mismatch and improved the SFDR by 11.6dB.

8.2 Future Work

Although the prototype single-channel SAR ADC achieved good power efficiency, it was intended only to concept-prove the proposed mismatch calibration method. Further improvements can be made by employing the architectural techniques presented in section 2.3.7 , as well as the design techniques discussed in chapter 3. This would help to improve the power efficiency of our design even further and enable us to achieve higher speeds. This potentially includes one or more of the following approaches:

- Employing a two-step ADC architecture in order to increase the maximum achievable speed;
- Using a more power-saving switching scheme to further reduce the power dissipation of the CAP-DAC;
- Reducing the delay of a conversion by using the techniques presented in section 3.2.3. This ultimately means using the technique shown in Fig. 3.20 that would guarantee reaching the maximum achievable speed for a single-channel SAR ADC. However, a dedicated control logic and probably a complex switching network for the CAP-DAC are required in order to adopt this mismatch calibration method.

Moreover, the proposed timing-skew calibration technique has only been tested by simulation. For future work, the 15-channel Ti ADC can be fully designed and implemented for fabrication in order to verify the effectiveness of the proposed timing mismatch calibration method by measurements. This requires the following

- Designing a multi-phase clock generator for 15 channels;
- Designing a saw-tooth signal generator;
- Design, implement, layout and verification of the TI ADC;
- RTL synthesize and verify the proposed timing mismatch calibration method along with the actual Ti ADC.

This prototype ADC would then be a good candidate for the specifications of 5G.

References

[1] J. G. Andrews, S. Buzzi, W. Choi, S. V. Hanly, A. Lozano, A. C. K. Soong, and J. C. Zhang. What Will 5G Be? *IEEE Journal on Selected Areas in Communications*, 32(6):1065–1082, June 2014.

[2] R. H. Walden. Analog-to-digital converter survey and analysis. *IEEE Journal on Selected Areas in Communications*, 17(4):539–550, April 1999.

[3] B. Murmann. ADC Performance Survey 1997-2018. *http://www.stanford.edu/ murmann/adcsurvey.html*.

[4] B. Widrow. A study of rough amplitude quantization by means of nyquist sampling theory. *IRE Transactions on Circuit Theory*, 3(4):266–276, December 1956.

[5] M. Bossche, J. Schoukens, and J. Renneboog. Dynamic testing and diagnostics of a/d converters. *IEEE Transactions on Circuits and Systems*, 33(8):775–785, August 1986.

[6] B. Ginetti and P. Jespers. Reliability of code density test for high resolution adcs. *Electronics Letters*, 27(24):2231–2233, Nov 1991.

[7] P. J. A. Harpe, C. Zhou, K. Philips, and H. de Groot. A 0.8-mW 5-bit 250-MS/s Time-Interleaved Asynchronous Digital Slope ADC. *IEEE Journal of Solid-State Circuits*, 46(11):2450–2457, Nov 2011.

[8] C. Sandner, M. Clara, A. Santner, T. Hartig, and F. Kuttner. A 6-bit 1.2-gs/s low-power flash-adc in 0.13-/spl mu/m digital cmos. *IEEE Journal of Solid-State Circuits*, 40(7):1499–1505, July 2005.

[9] G. Tretter, M. Khafaji, D. Fritsche, C. Carta, and F. Ellinger. A 24 gs/s single-core flash adc with 3 bit resolution in 28 nm low-power digital cmos. In *2015 IEEE Radio Frequency Integrated Circuits Symposium (RFIC)*, pages 347–350, May 2015.

[10] S. Shahramian, S. P. Voinigescu, and A. C. Carusone. A 35-gs/s, 4-bit flash adc with active data and clock distribution trees. *IEEE Journal of Solid-State Circuits*, 44(6):1709–1720, June 2009.

[11] A. Zandieh, P. Schvan, and S. P. Voinigescu. A 2x-oversampling, 128-gs/s 5-bit flash adc for 64-gbaud applications. In *2018 IEEE BiCMOS and Compound Semiconductor Integrated Circuits and Technology Symposium (BCICTS)*, pages 52–55, Oct 2018.

[12] P. Scholtens and M. Vertregt. A 6b 1.6 gsample/s flash adc in 0.18 /spl mu/m cmos using averaging termination. In *2002 IEEE International Solid-State Circuits Conference. Digest of Technical Papers (Cat. No.02CH37315)*, volume 1, pages 168–457 vol.1, Feb 2002.

[13] K. Deguchi, N. Suwa, M. Ito, T. Kumamoto, and T. Miki. A 6-bit 3.5-gs/s 0.9-v 98-mw flash adc in 90nm cmos. In *2007 IEEE Symposium on VLSI Circuits*, pages 64–65, June 2007.

[14] H. Yu and M. F. Chang. A 1-v 1.25-gs/s 8-bit self-calibrated flash adc in 90-nm digital cmos. *IEEE Transactions on Circuits and Systems II: Express Briefs*, 55(7):668–672, July 2008.

[15] and A. A. Abidi. Spatial filtering in flash a/d converters. *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, 50(8):424–436, Aug 2003.

[16] C. W. Mangelsdorf. A 400-mhz input flash converter with error correction. *IEEE Journal of Solid-State Circuits*, 25(1):184–191, Feb 1990.

[17] C. L. Portmann and H. Y. Meng. Metastability in cmos library elements in reduced supply and technology scaled applications. *IEEE Journal of Solid-State Circuits*, 30(1):39–46, Jan 1995.

[18] M. Steyaert, R. Roovers, and J. Craninckx. A 100 mhz 8 bit cmos interpolating a/d converter. In *Proceedings of IEEE Custom Integrated Circuits Conference - CICC '93*, pages 28.1.1–28.1.4, May 1993.

[19] H. Kimura, A. Matsuzawa, T. Nakamura, and S. Sawada. A 10 b 300 mhz interpolated-parallel a/d converter. In *1992 Symposium on VLSI Circuits Digest of Technical Papers*, pages 94–95, June 1992.

[20] K. Kusumoto, A. Matsuzawa, and K. Murata. A 10-b 20-mhz 30-mw pipelined interpolating cmos adc. *IEEE Journal of Solid-State Circuits*, 28(12):1200–1206, Dec 1993.

[21] R. Roovers and M. S. J. Steyaert. A 175 ms/s, 6 b, 160 mw, 3.3 v cmos a/d converter. *IEEE Journal of Solid-State Circuits*, 31(7):938–944, July 1996.

[22] M. P. Flynn and D. J. Allstot. CMOS folding ADCs with current-mode interpolation. In *Proceedings ISSCC '95 - International Solid-State Circuits Conference*, pages 274–275, Feb 1995.

[23] and and K. Bacrania. An 8-b 100-MSample/s CMOS pipelined folding ADC. *IEEE Journal of Solid-State Circuits*, 36(2):184–194, Feb 2001.

[24] B. Nauta and A. G. W. Venes. A 70-MS/s 110-mW 8-b CMOS folding and interpolating A/D converter. *IEEE Journal of Solid-State Circuits*, 30(12):1302–1308, Dec 1995.

[25] F. Vessal and C. A. T. Salama. An 8-bit 2-Gsample/s folding-interpolating analog-to-digital converter in SiGe technology. *IEEE Journal of Solid-State Circuits*, 39(1):238–241, Jan 2004.

[26] M. P. Flynn and B. Sheahan. A 400-Msample/s, 6-b CMOS folding and interpolating ADC. *IEEE Journal of Solid-State Circuits*, 33(12):1932–1938, Dec 1998.

[27] A. N. Karanicolas, , and K. L. Barcrania. A 15-b 1-Msample/s digitally self-calibrated pipeline ADC. *IEEE Journal of Solid-State Circuits*, 28(12):1207–1215, Dec 1993.

[28] L. A. Singer and T. L. Brooks. A 14-bit 10-MHz calibration-free CMOS pipelined A/D converter. In *1996 Symposium on VLSI Circuits. Digest of Technical Papers*, pages 94–95, June 1996.

[29] I. E. Opris, L. D. Lewicki, and B. C. Wong. A single-ended 12-bit 20 Msample/s self-calibrating pipeline A/D converter. *IEEE Journal of Solid-State Circuits*, 33(12):1898–1903, Dec 1998.

[30] S. H. Lewis, H. S. Fetterman, G. F. Gross, R. Ramachandran, and T. R. Viswanathan. A 10-b 20-Msample/s analog-to-digital converter. *IEEE Journal of Solid-State Circuits*, 27(3):351–358, March 1992.

[31] L. Singer, S. Ho, M. Timko, and D. Kelly. A 12 b 65 MSample/s CMOS ADC with 82 dB SFDR at 120 MHz. In *2000 IEEE International Solid-State Circuits Conference. Digest of Technical Papers (Cat. No.00CH37056)*, pages 38–39, Feb 2000.

[32] B. Murmann and B. E. Boser. A 12-bit 75-MS/s pipelined ADC using open-loop residue amplification. *IEEE Journal of Solid-State Circuits*, 38(12):2040–2050, Dec 2003.

[33] Yun Chiu, C. W. Tsang, B. Nikolic, and P. R. Gray. Least mean square adaptive digital background calibration of pipelined analog-to-digital converters. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 51(1):38–46, 2004.

[34] Xiaoyue Wang, P. J. Hurst, and S. H. Lewis. A 12-bit 20-msample/s pipelined analog-to-digital converter with nested digital background calibration. *IEEE Journal of Solid-State Circuits*, 39(11):1799–1808, 2004.

[35] B. P. Brandt, D. E. Wingard, and B. A. Wooley. Second-order sigma-delta modulation for digital-audio signal acquisition. *IEEE Journal of Solid-State Circuits*, 26(4):618–627, April 1991.

[36] R. T. Baird and T. S. Fiez. Stability analysis of high-order delta-sigma modulation for ADC's. *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, 41(1):59–62, Jan 1994.

[37] B. E. Boser and B. A. Wooley. The design of sigma-delta modulation analog-to-digital converters. *IEEE Journal of Solid-State Circuits*, 23(6):1298–1308, Dec 1988.

[38] Y. Matsuya, K. Uchimura, A. Iwata, T. Kobayashi, M. Ishikawa, and T. Yoshitome. A 16-bit oversampling A-to-D conversion technology using triple-integration noise shaping. *IEEE Journal of Solid-State Circuits*, 22(6):921–929, Dec 1987.

[39] L. A. Williams and B. A. Wooley. A third-order sigma-delta modulator with extended dynamic range. *IEEE Journal of Solid-State Circuits*, 29(3):193–202, March 1994.

[40] Z. Cao, S. Yan, and Y. Li. A 32 mW 1.25 GS/s 6b 2b/Step SAR ADC in 0.13 μm CMOS. *IEEE Journal of Solid-State Circuits*, 44(3):862–873, March 2009.

[41] H. Hong, W. Kim, H. Kang, S. Park, M. Choi, H. Park, and S. Ryu. A Decision-Error-Tolerant 45 nm CMOS 7b 1 GS/s Nonbinary 2b/Cycle SAR ADC. *IEEE Journal of Solid-State Circuits*, 50(2):543–555, Feb 2015.

[42] Q. Liu, W. Shu, and J. S. Chang. A 400-MS/s 10-b 2-b/Step SAR ADC With 52-dB SNDR and 5.61-mW Power Dissipation in 65-nm CMOS. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 25(12):3444–3454, Dec 2017.

[43] H. Hong, H. Kang, D. Jo, D. Lee, Y. You, Y. Lee, H. Park, and S. Ryu. A 2.6b/cycle-architecture-based 10b 1 JGS/s 15.4mW 4×-time-interleaved SAR ADC with a multi-step hardware-retirement technique. In *2015 IEEE International Solid-State Circuits Conference - (ISSCC) Digest of Technical Papers*, pages 1–3, Feb 2015.

[44] S. Kiran, S. Cai, Y. Luo, S. Hoyos, and S. Palermo. A 32 Gb/s ADC-based PAM-4 receiver with 2-bit/stage SAR ADC and partially-unrolled DFE. In *2018 IEEE Custom Integrated Circuits Conference (CICC)*, pages 1–4, April 2018.

[45] J. Song, X. Tang, and N. Sun. A 10-b 2b/cycle 300MS/s SAR ADC with a single differential DAC in 40nm CMOS. In *2017 IEEE Custom Integrated Circuits Conference (CICC)*, pages 1–4, April 2017.

[46] C. Chan, Y. Zhu, S. Sin, S. Ben U, and R. P. Martins. A 6 b 5 GS/s 4 Interleaved 3 b/Cycle SAR ADC. *IEEE Journal of Solid-State Circuits*, 51(2):365–377, Feb 2016.

[47] J. Nam, M. Hassanpourghadi, A. Zhang, and M. S. Chen. A 12-Bit 1.6, 3.2, and 6.4 GS/s 4-b/Cycle Time-Interleaved SAR ADC With Dual Reference Shifting and Interpolation. *IEEE Journal of Solid-State Circuits*, 53(6):1765–1779, June 2018.

[48] C. Chan, Y. Zhu, W. Zhang, S. U, and R. P. Martins. A Two-Way Interleaved 7-b 2.4-GS/s 1-Then-2 b/Cycle SAR ADC With Background Offset Calibration. *IEEE Journal of Solid-State Circuits*, 53(3):850–860, March 2018.

[49] H. Wei, C. Chan, U. Chio, S. Sin, S. U, R. P. Martins, and F. Maloberti. An 8-b 400-MS/s 2-b-Per-Cycle SAR ADC With Resistive DAC. *IEEE Journal of Solid-State Circuits*, 47(11):2763–2772, Nov 2012.

[50] C. Chan, Y. Zhu, S. Sin, U. Seng-Pan, and R. P. Martins. 26.5 A 5.5mW 6b 5GS/S 4×-lnterleaved 3b/cycle SAR ADC in 65nm CMOS. In *2015 IEEE International Solid-State Circuits Conference - (ISSCC) Digest of Technical Papers*, pages 1–3, Feb 2015.

[51] W. Yang, D. Kelly, L. Mehr, M. T. Sayuk, and L. Singer. A 3-V 340-mW 14-b 75-Msample/s CMOS ADC with 85-dB SFDR at Nyquist input. *IEEE Journal of Solid-State Circuits*, 36(12):1931–1936, Dec 2001.

[52] B. Lee, B. Min, G. Manganaro, and J. W. Valvano. A 14-b 100-MS/s Pipelined ADC With a Merged SHA and First MDAC. *IEEE Journal of Solid-State Circuits*, 43(12):2613–2619, Dec 2008.

[53] S. Devarajan, L. Singer, D. Kelly, S. Decker, A. Kamath, and P. Wilkins. A 16-bit, 125 MS/s, 385 mW, 78.7 dB SNR CMOS Pipeline ADC. *IEEE Journal of Solid-State Circuits*, 44(12):3305–3313, Dec 2009.

[54] V. Tripathi and B. Murmann. A 160 MS/s, 11.1 mW, single-channel pipelined SAR ADC with 68.3 dB SNDR. In *Proceedings of the IEEE 2014 Custom Integrated Circuits Conference*, pages 1–4, Sep. 2014.

[55] Y. Hu, P. Huang, H. Tai, and H. Chen. A 12.5-fJ/Conversion-Step 8-Bit 800-MS/s Two-Step SAR ADC. *IEEE Transactions on Circuits and Systems II: Express Briefs*, 63(12):1166–1170, Dec 2016.

[56] H. Huang, H. Xu, B. Elies, and Y. Chiu. A Non-Interleaved 12-b 330-MS/s Pipelined-SAR ADC With PVT-Stabilized Dynamic Amplifier Achieving Sub-1-dB SNDR Variation. *IEEE Journal of Solid-State Circuits*, 52(12):3235–3247, Dec 2017.

[57] Y. Zhu, C. Chan, Z. Zheng, C. Li, J. Zhong, and R. P. Martins. A 0.19 mm210 b 2.3 GS/s 12-Way Time-Interleaved Pipelined-SAR ADC in 65-nm CMOS. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 65(11):3606–3616, Nov 2018.

[58] L. Kull, D. Luu, C. Menolfi, M. Braendli, P. A. Francese, T. Morf, M. Kossel, H. Yueksel, A. Cevrero, I. Ozkaya, and T. Toifl. 28.5 A 10b 1.5GS/s pipelined-SAR ADC with background second-stage common-mode regulation and offset calibration in 14nm CMOS FinFET. In *2017 IEEE International Solid-State Circuits Conference (ISSCC)*, pages 474–475, Feb 2017.

[59] J. Zhong, Y. Zhu, C. Chan, S. Sin, S. U, and R. P. Martins. A 12b 180MS/s 0.068mm2With Full-Calibration-Integrated Pipelined-SAR ADC. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 64(7):1684–1695, July 2017.

[60] Y. Zhu, C. Chan, S. P. U, and R. P. Martins. A 10-bit 500-MS/s Partial-Interleaving Pipelined SAR ADC With Offset and Reference Mismatch Calibrations. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 25(1):354–363, Jan 2017.

[61] E. Martens, B. Hershberg, and J. Craninckx. A 69-dB SNDR 300-MS/s Two-Time Interleaved Pipelined SAR ADC in 16-nm CMOS FinFET With Capacitive Reference Stabilization. *IEEE Journal of Solid-State Circuits*, 53(4):1161–1171, April 2018.

[62] J. A. Fredenburg and M. P. Flynn. A 90-MS/s 11-MHz-Bandwidth 62-dB SNDR Noise-Shaping SAR ADC. *IEEE Journal of Solid-State Circuits*, 47(12):2898–2904, Dec 2012.

[63] S. Li, B. Qiao, M. Gandara, and N. Sun. A 13-ENOB 2nd-order noise-shaping SAR ADC realizing optimized NTF zeros using an error-feedback structure. In *2018 IEEE International Solid - State Circuits Conference - (ISSCC)*, pages 234–236, Feb 2018.

[64] Z. Chen, M. Miyahara, and A. Matsuzawa. A 9.35-ENOB, 14.8 fJ/conv.-step fully-passive noise-shaping SAR ADC. In *2015 Symposium on VLSI Circuits (VLSI Circuits)*, pages C64–C65, June 2015.

[65] W. Guo and N. Sun. A 12b-ENOB 61μW noise-shaping SAR ADC with a passive integrator. In *ESSCIRC Conference 2016: 42nd European Solid-State Circuits Conference*, pages 405–408, Sep. 2016.

[66] Y. Lin, C. Tsai, S. Tsou, R. Chu, and C. Lu. A 2.4-mW 25-MHz BW 300-MS/s passive noise shaping SAR ADC with noise quantizer technique in 14-nm CMOS. In *2017 Symposium on VLSI Circuits*, pages C234–C235, June 2017.

[67] Y. Shu, L. Kuo, and T. Lo. An Oversampling SAR ADC With DAC Mismatch Error Shaping Achieving 105 dB SFDR and 101 dB SNDR Over 1 kHz BW in 55 nm CMOS. *IEEE Journal of Solid-State Circuits*, 51(12):2928–2940, Dec 2016.

[68] J. Liu, Y. Zhu, C. Chan, S. Sin, S. U, and R. P. da Silva Martins. Uniform Quantization Theory-Based Linearity Calibration for Split Capacitive DAC in an SAR ADC. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 24(7):2603–2607, July 2016.

[69] E. Alpman, H. Lakdawala, L. R. Carley, and K. Soumyanath. A 1.1V 50mW 2.5GS/s 7b Time-Interleaved C-2C SAR ADC in 45nm LP digital CMOS. In *2009 IEEE International Solid-State Circuits Conference - Digest of Technical Papers*, pages 76–77,77a, Feb 2009.

[70] J. Craninckx and G. van der Plas. A 65fJ/Conversion-Step 0-to-50MS/s 0-to-0.7mW 9b Charge-Sharing SAR ADC in 90nm Digital CMOS. In *2007 IEEE International Solid-State Circuits Conference. Digest of Technical Papers*, pages 246–600, Feb 2007.

[71] B. Malki, T. Yamamoto, B. Verbruggen, P. Wambacq, and J. Craninckx. A 70 dB DR 10 b 0-to-80 MS/s Current-Integrating SAR ADC With Adaptive Dynamic Range. *IEEE Journal of Solid-State Circuits*, 49(5):1173–1183, May 2014.

[72] T. Rabuske and J. Fernandes. A SAR ADC With a MOSCAP-DAC. *IEEE Journal of Solid-State Circuits*, 51(6):1410–1422, June 2016.

[73] M. Maddox, B. Chen, M. Coln, R. Kapusta, J. Shen, and L. Fernando. A 16 bit linear passive-charge-sharing SAR ADC in 55nm CMOS. In *2016 IEEE Asian Solid-State Circuits Conference (A-SSCC)*, pages 153–156, Nov 2016.

[74] B. P. Ginsburg and A. P. Chandrakasan. An energy-efficient charge recycling approach for a SAR converter with capacitive DAC. In *2005 IEEE International Symposium on Circuits and Systems*, pages 184–187 Vol. 1, May 2005.

[75] and and. A 8-bit 500-KS/s low power SAR ADC for bio-medical applications. In *2007 IEEE Asian Solid-State Circuits Conference*, pages 228–231, Nov 2007.

[76] C. Liou and C. Hsieh. A 2.4-to-5.2fJ/conversion-step 10b 0.5-to-4MS/s SAR ADC with charge-average switching DAC in 90nm CMOS. In *2013 IEEE International Solid-State Circuits Conference Digest of Technical Papers*, pages 280–281, Feb 2013.

[77] C. Liu, S. Chang, G. Huang, and Y. Lin. A 10-bit 50-MS/s SAR ADC With a Monotonic Capacitor Switching Procedure. *IEEE Journal of Solid-State Circuits*, 45(4):731–740, April 2010.

[78] C. Liu, S. Chang, G. Huang, and Y. Lin. A 0.92mW 10-bit 50-MS/s SAR ADC in 0.13μm CMOS process. In *2009 Symposium on VLSI Circuits*, pages 236–237, June 2009.

[79] H. Tai, Y. Hu, H. Chen, and H. Chen. 11.2 A 0.85fJ/conversion-step 10b 200kS/s subranging SAR ADC in 40nm CMOS. In *2014 IEEE International Solid-State Circuits Conference Digest of Technical Papers (ISSCC)*, pages 196–197, Feb 2014.

[80] Y. Zhu, C. Chan, U. Chio, S. Sin, S. U, R. P. Martins, and F. Maloberti. A 10-bit 100-MS/s Reference-Free SAR ADC in 90 nm CMOS. *IEEE Journal of Solid-State Circuits*, 45(6):1111–1121, June 2010.

[81] V. Hariprasath, J. Guerber, S. . Lee, and U. . Moon. Merged capacitor switching based SAR ADC with highest switching energy-efficiency. *Electronics Letters*, 46(9):620–621, April 2010.

[82] Z. Zhu, Y. Xiao, and X. Song. VCM-based monotonic capacitor switching scheme for SAR ADC. *Electronics Letters*, 49(5):327–329, February 2013.

[83] V. Giannini, P. Nuzzo, V. Chironi, A. Baschirotto, G. Van der Plas, and J. Craninckx. An 820μW 9b 40MS/s Noise-Tolerant Dynamic-SAR ADC in 90nm Digital CMOS. In *2008 IEEE International Solid-State Circuits Conference - Digest of Technical Papers*, pages 238–610, Feb 2008.

[84] T. Ogawa, H. Kobayashi, M. Hotta, Y. Takahashi, and and. SAR ADC algorithm with redundancy. In *APCCAS 2008 - 2008 IEEE Asia Pacific Conference on Circuits and Systems*, pages 268–271, Nov 2008.

[85] T. C. Verster. A Method to Increase the Accuracy of Fast-Serial-Parallel Analog-to-Digital Converters. *IEEE Transactions on Electronic Computers*, EC-13(4):471–473, Aug 1964.

[86] C. Liu, S. Chang, G. Huang, Y. Lin, C. Huang, C. Huang, L. Bu, and C. Tsai. A 10b 100MS/s 1.13mW SAR ADC with binary-scaled error compensation. In *2010 IEEE International Solid-State Circuits Conference - (ISSCC)*, pages 386–387, Feb 2010.

[87] S. Cho, C. Lee, J. Kwon, and S. Ryu. A 550-μW10-b 40-MS/s SAR ADC With Multistep Addition-Only Digital Error Correction. *IEEE Journal of Solid-State Circuits*, 46(8):1881–1892, Aug 2011.

[88] W. Liu, , , , , , , , and Y. Chiu. A 600MS/s 30mW 0.13μm CMOS ADC array achieving over 60dB SFDR with adaptive digital equalization. In *2009 IEEE International Solid-State Circuits Conference - Digest of Technical Papers*, pages 82–83,83a, Feb 2009.

[89] J. Yang, T. L. Naing, and B. Brodersen. A 1-GS/s 6-bit 6.7-mW ADC in 65-nm CMOS. In *2009 IEEE Custom Integrated Circuits Conference*, pages 287–290, Sep. 2009.

[90] J. Yang, T. L. Naing, and R. W. Brodersen. A 1 GS/s 6 Bit 6.7 mW Successive Approximation ADC Using Asynchronous Processing. *IEEE Journal of Solid-State Circuits*, 45(8):1469–1478, Aug 2010.

[91] C. P. Hurrell, C. Lyden, D. Laing, D. Hummerston, and M. Vickery. An 18b 12.5MHz ADC with 93dB SNR. In *2010 IEEE International Solid-State Circuits Conference - (ISSCC)*, pages 378–379, Feb 2010.

[92] W. Liu, P. Huang, and Y. Chiu. A 12b 22.5/45MS/s 3.0mW 0.059mm2 CMOS SAR ADC achieving over 90dB SFDR. In *2010 IEEE International Solid-State Circuits Conference - (ISSCC)*, pages 380–381, Feb 2010.

[93] C. P. Hurrell, C. Lyden, D. Laing, D. Hummerston, and M. Vickery. An 18 b 12.5 MS/s ADC With 93 dB SNR. *IEEE Journal of Solid-State Circuits*, 45(12):2647–2654, Dec 2010.

[94] K. Doris, E. Janssen, C. Nani, A. Zanikopoulos, and G. van der Weide. A 480 mW 2.6 GS/s 10b Time-Interleaved ADC With 48.5 dB SNDR up to Nyquist in 65 nm CMOS. *IEEE Journal of Solid-State Circuits*, 46(12):2821–2833, Dec 2011.

[95] W. Liu, P. Huang, and Y. Chiu. A 12-bit, 45-MS/s, 3-mW Redundant Successive-Approximation-Register Analog-to-Digital Converter With Digital Calibration. *IEEE Journal of Solid-State Circuits*, 46(11):2661–2672, Nov 2011.

[96] J. Guerber, H. Venkatram, M. Gande, A. Waters, and U. Moon. A 10-b Ternary SAR ADC With Quantization Time Information Utilization. *IEEE Journal of Solid-State Circuits*, 47(11):2604–2613, Nov 2012.

[97] W. Liu, P. Huang, and Y. Chiu. A 12-bit 50-MS/s 3.3-mW SAR ADC with background digital calibration. In *Proceedings of the IEEE 2012 Custom Integrated Circuits Conference*, pages 1–4, Sep. 2012.

[98] H. Tai, H. Chen, and H. Chen. A 3.2fJ/c.-s. 0.35V 10b 100KS/s SAR ADC in 90nm CMOS. In *2012 Symposium on VLSI Circuits (VLSIC)*, pages 92–93, June 2012.

[99] E. Janssen, K. Doris, A. Zanikopoulos, A. Murroni, G. v. d. Weide, Y. Lin, L. Alvado, F. Darthenay, and Y. Fregeais. An 11b 3.6GS/s time-interleaved SAR ADC in 65nm CMOS. In *2013 IEEE International Solid-State Circuits Conference Digest of Technical Papers*, pages 464–465, Feb 2013.

[100] H. Hong, H. Kang, B. Sung, C. Lee, M. Choi, H. Park, and S. Ryu. An 8.6 ENOB 900MS/s time-interleaved 2b/cycle SAR ADC with a 1b/cycle reconfiguration for resolution enhancement. In *2013 IEEE International Solid-State Circuits Conference Digest of Technical Papers*, pages 470–471, Feb 2013.

[101] R. Kapusta, J. Shen, S. Decker, H. Li, E. Ibaragi, and H. Zhu. A 14b 80 MS/s SAR ADC With 73.6 dB SNDR in 65 nm CMOS. *IEEE Journal of Solid-State Circuits*, 48(12):3059–3066, Dec 2013.

[102] I. Daubechies and O. Yilmaz. Robust and Practical Analog-to-Digital Conversion With Exponential Precision. *IEEE Transactions on Information Theory*, 52(8):3533–3545, Aug 2006.

[103] M. Furuta, M. Nozawa, and T. Itakura. A 0.06mm2 8.9b ENOB 40MS/s pipelined SAR ADC in 65nm CMOS. In *2010 IEEE International Solid-State Circuits Conference - (ISSCC)*, pages 382–383, Feb 2010.

[104] F. Kuttner. A 1.2V 10b 20MSample/s non-binary successive approximation ADC in 0.13/spl mu/m CMOS. In *2002 IEEE International Solid-State Circuits Conference. Digest of Technical Papers (Cat. No.02CH37315)*, volume 1, pages 176–177 vol.1, Feb 2002.

[105] M. Hesener, T. Eicher, A. Hanneberg, D. Herbison, F. Kuttner, and H. Wenske. A 14b 40MS/s Redundant SAR ADC with 480MHz Clock in 0.13pm CMOS. In *2007 IEEE International Solid-State Circuits Conference. Digest of Technical Papers*, pages 248–600, Feb 2007.

[106] L. Qiu, K. Tang, Y. Zheng, L. Siek, Y. Zhu, and S. U. A 16-mW 1-GS/s With 49.6-dB SNDR TI-SAR ADC for Software-Defined Radio in 65-nm CMOS. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 26(3):572–583, March 2018.

[107] L. Qiu, K. Tang, Y. Zheng, and L. Siek. A Flexible-Weighted Nonbinary Searching Technique for High-Speed SAR-ADCs. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 24(8):2808–2812, Aug 2016.

[108] B. P. Ginsburg and A. P. Chandrakasan. Dual Time-Interleaved Successive Approximation Register ADCs for an Ultra-Wideband Receiver. *IEEE Journal of Solid-State Circuits*, 42(2):247–257, Feb 2007.

[109] C. Chan, Y. Zhu, S. Sin, B. Murmann, S. U, and R. P. Martins. Metastablility in SAR ADCs. *IEEE Transactions on Circuits and Systems II: Express Briefs*, 64(2):111–115, Feb 2017.

[110] J. Nam and M. S. Chen. An Embedded Passive Gain Technique for Asynchronous SAR ADC Achieving 10.2 ENOB 1.36-mW at 95-MS/s in 65 nm CMOS. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 63(10):1628–1638, Oct 2016.

[111] T. Tsai, H. Tai, P. Tsai, C. Tsai, and H. Chen. An 8 b 700 MS/s 1 b/Cycle SAR ADC Using a Delay-Shift Technique. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 63(5):683–692, May 2016.

[112] Z. Cao, S. Yan, and Y. Li. A 32mW 1.25GS/s 6b 2b/step SAR ADC in 0.13μm CMOS. In *2008 IEEE International Solid-State Circuits Conference - Digest of Technical Papers*, pages 542–634, Feb 2008.

[113] D. Xu, L. Qiu, Z. Zhang, T. Liu, L. Liu, K. Chen, and S. Xu. A Linearity-Improved 8-bit 320-MS/s SAR ADC With Metastability Immunity Technique. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 26(8):1545–1553, Aug 2018.

[114] T. Jiang, W. Liu, F. Y. Zhong, C. Zhong, K. Hu, and P. Y. Chiang. A Single-Channel, 1.25-GS/s, 6-bit, 6.08-mW Asynchronous Successive-Approximation ADC With Improved Feedback Delay in 40-nm CMOS. *IEEE Journal of Solid-State Circuits*, 47(10):2444–2453, Oct 2012.

[115] L. Chen, K. Ragab, X. Tang, J. Song, A. Sanyal, and N. Sun. A 0.95-mW 6-b 700-MS/s Single-Channel Loop-Unrolled SAR ADC in 40-nm CMOS. *IEEE Transactions on Circuits and Systems II: Express Briefs*, 64(3):244–248, March 2017.

[116] F. M. Yaul and A. P. Chandrakasan. A 10 bit SAR ADC With Data-Dependent Energy Reduction Using LSB-First Successive Approximation. *IEEE Journal of Solid-State Circuits*, 49(12):2825–2834, Dec 2014.

[117] Y. Chung, C. Yen, P. Tsai, and B. Chen. A 12-bit 40-MS/s SAR ADC With a Fast-Binary-Window DAC Switching Scheme. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 26(10):1989–1998, Oct 2018.

[118] J. Montanaro, R. T. Witek, K. Anne, A. J. Black, E. M. Cooper, D. W. Dobberpuhl, P. M. Donahue, J. Eno, W. Hoeppner, D. Kruckemyer, T. H. Lee, P. C. M. Lin, L. Madden, D. Murray, M. H. Pearce, S. Santhanam, K. J. Snyder, R. Stehpany, and S. C. Thierauf. A 160-MHz, 32-b, 0.5-W CMOS RISC microprocessor. *IEEE Journal of Solid-State Circuits*, 31(11):1703–1714, Nov 1996.

[119] T. Kobayashi, K. Nogami, T. Shirotori, Y. Fujimoto, and O. Watanabe. 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*, pages 28–29, June 1992.

[120] B. Razavi. The StrongARM Latch [A Circuit for All Seasons]. *IEEE Solid-State Circuits Magazine*, 7(2):12–17, Spring 2015.

[121] C. Huang, J. Lin, Y. Shyu, and S. Chang. A Systematic Design Methodology of Asynchronous SAR ADCs. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 24(5):1835–1848, May 2016.

[122] B. Wicht, T. Nirschl, and D. Schmitt-Landsiedel. Yield and speed optimization of a latch-type voltage sense amplifier. *IEEE Journal of Solid-State Circuits*, 39(7):1148–1158, July 2004.

[123] D. Xing, Y. Zhu, C. Chan, S. Sin, F. Ye, J. Ren, S. U, and R. P. Martins. Seven-bit 700-MS/s Four-Way Time-Interleaved SAR ADC With Partial V_{cm} -Based Switching. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 25(3):1168–1172, March 2017.

[124] D. Schinkel, E. Mensink, E. Klumperink, E. van Tuijl, and B. Nauta. 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*, pages 314–605, Feb 2007.

[125] and and and. A low-noise self-calibrating dynamic comparator for high-speed ADCs. In *2008 IEEE Asian Solid-State Circuits Conference*, pages 269–272, Nov 2008.

[126] A 12-b 600 ks/s digitally self-calibrated pipelined algorithmic ADC. *IEEE Journal of Solid-State Circuits*, 29(4):509–515, April 1994.

[127] P. Amaral, J. Goes, N. Paulino, and A. Steiger-Garcao. An improved low-voltage low-power CMOS comparator to be used in high-speed pipeline ADCs. In *2002 IEEE International Symposium on Circuits and Systems. Proceedings (Cat. No.02CH37353)*, volume 5, pages V–V, May 2002.

[128] K. Uyttenhove and M. S. J. Steyaert. A 1.8-V 6-bit 1.3-GHz flash ADC in 0.25-/spl mu/m CMOS. *IEEE Journal of Solid-State Circuits*, 38(7):1115–1122, July 2003.

[129] Samgsuk Kim and Minkyu Song. An 8-bit 200 msps cmos a/d converter for analog interface module of tft-lcd driver. In *ISCAS 2001. The 2001 IEEE International Symposium on Circuits and Systems (Cat. No.01CH37196)*, volume 1, pages 528–531 vol. 1, 2001.

[130] Yun-Ti Wang and B. Razavi. An 8-bit 150-mhz cmos a/d converter. *IEEE Journal of Solid-State Circuits*, 35(3):308–317, 2000.

[131] L. Y. Nathawad, R. Urata, B. A. Wooley, and D. A. B. Miller. A 40-GHz-bandwidth, 4-bit, time-interleaved A/D converter using photoconductive sampling. *IEEE Journal of Solid-State Circuits*, 38(12):2021–2030, Dec 2003.

[132] M. van Elzakker, E. van Tuijl, P. Geraedts, D. Schinkel, E. A. M. Klumperink, and B. Nauta. A 10-bit Charge-Redistribution ADC Consuming 1.9μW at 1 MS/s. *IEEE Journal of Solid-State Circuits*, 45(5):1007–1015, May 2010.

[133] M. Dessouky and A. Kaiser. Very low-voltage digital-audio /spl Delta//spl Sigma/ modulator with 88-dB dynamic range using local switch bootstrapping. *IEEE Journal of Solid-State Circuits*, 36(3):349–355, 2001.

[134] S. Gambini and J. Rabaey. Low-Power Successive Approximation Converter With 0.5 V Supply in 90 nm CMOS. *IEEE Journal of Solid-State Circuits*, 42(11):2348–2356, Nov 2007.

[135] K. Lin, Y. Cheng, and K. Tang. A 0.5 V 1.28-MS/s 4.68-fJ/Conversion-Step SAR ADC With Energy-Efficient DAC and Trilevel Switching Scheme. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 24(4):1441–1449, April 2016.

[136] Y. Shen, Z. Zhu, S. Liu, and Y. Yang. A Reconfigurable 10-to-12-b 80-to-20-MS/s Bandwidth Scalable SAR ADC. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 65(1):51–60, Jan 2018.

[137] X. Zheng, Z. Wang, F. Li, F. Zhao, S. Yue, C. Zhang, and Z. Wang. A 14-bit 250 MS/s IF Sampling Pipelined ADC in 180 nm CMOS Process. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 63(9):1381–1392, Sep. 2016.

[138] R. Xu, B. Liu, and J. Yuan. Digitally Calibrated 768-kS/s 10-b Minimum-Size SAR ADC Array With Dithering. *IEEE Journal of Solid-State Circuits*, 47(9):2129–2140, Sep. 2012.

[139] J. Wu, A. Wu, and Y. Du. Dithering-based calibration of capacitor mismatch in SAR ADCs. *Electronics Letters*, 52(19):1598–1600, 2016.

[140] G. Wang, F. Kacani, and Y. Chiu. IRD Digital Background Calibration of SAR ADC With Coarse Reference ADC Acceleration. *IEEE Transactions on Circuits and Systems II: Express Briefs*, 61(1):11–15, Jan 2014.

[141] J. A. McNeill, K. Y. Chan, M. C. W. Coln, C. L. David, and C. Brenneman. All-Digital Background Calibration of a Successive Approximation ADC Using the "Split ADC" Architecture. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 58(10):2355–2365, Oct 2011.

[142] M. Ding, P. Harpe, Y. Liu, B. Busze, K. Philips, and H. de Groot. A 46 μW 13 b 6.4 MS/s SAR ADC With Background Mismatch and Offset Calibration. *IEEE Journal of Solid-State Circuits*, 52(2):423–432, Feb 2017.

[143] S. K. Sunter and N. Nagi. A simplified polynomial-fitting algorithm for DAC and ADC BIST. In *Proceedings International Test Conference 1997*, pages 389–395, Nov 1997.

[144] F. Adamo, F. Attivissimo, N. Giaquinto, and M. Savino. FFT test of A/D converters to determine the integral nonlinearity. *IEEE Transactions on Instrumentation and Measurement*, 51(5):1050–1054, Oct 2002.

[145] N. Csizmadia and A. J. E. M. Janssen. Estimating the integral non-linearity of A/D-converters via the frequency domain. In *International Test Conference 1999. Proceedings (IEEE Cat. No.99CH37034)*, pages 757–762, Sep. 1999.

[146] V. Kerzérho, V. Fresnaud, D. Dallet, S. Bernard, and L. Bossuet. Fast Digital Post-Processing Technique for Integral Nonlinearity Correction of Analog-to-Digital Converters: Validation on a 12-Bit Folding-and-Interpolating Analog-to-Digital Converter. *IEEE Transactions on Instrumentation and Measurement*, 60(3):768–775, March 2011.

[147] P. Handel, M. Skoglund, and M. Pettersson. A calibration scheme for imperfect quantizers. *IEEE Transactions on Instrumentation and Measurement*, 49(5):1063–1068, Oct 2000.

[148] S. Lloyd. Least squares quantization in PCM. *IEEE Transactions on Information Theory*, 28(2):129–137, 1982.

[149] F. H. Irons, D. M. Hummels, and S. P. Kennedy. Improved compensation for analog-to-digital converters. *IEEE Transactions on Circuits and Systems*, 38(8):958–961, Aug 1991.

[150] J. Tsimbinos, W. Marwood, A. Beaumont-Smith, and C. C. Lim. Results of a/d converter compensation with a vlsi chip. In *Final Program and Abstracts on Information, Decision and Control*, pages 289–293, Feb 2002.

[151] K. Tan. On board self-calibration of analog-to-digital and digital-to-analog converters. In *2009 IEEE Custom Integrated Circuits Conference*, Sep. 1982.

[152] and D. Hodges. Self-calibration technique for A/D converters. *IEEE Transactions on Circuits and Systems*, 30(3):188–190, March 1983.

[153] J. L. McCreary and D. A. Sealer. Precision Capacitor Ratio Measurement Technique for Integrated Circuit Capacitor Arrays. *IEEE Transactions on Instrumentation and Measurement*, 28(1):11–17, March 1979.

[154] A. Shikata, R. Sekimoto, T. Kuroda, and H. Ishikuro. A 0.5 V 1.1 MS/sec 6.3 fJ/Conversion-Step SAR-ADC With Tri-Level Comparator in 40 nm CMOS. *IEEE Journal of Solid-State Circuits*, 47(4):1022–1030, April 2012.

[155] H. . Lee, D. A. Hodges, and P. R. Gray. A self-calibrating 15 bit CMOS A/D converter. *IEEE Journal of Solid-State Circuits*, 19(6):813–819, Dec 1984.

[156] M. Yoshioka, K. Ishikawa, T. Takayama, and S. Tsukamoto. A 10b 50MS/s 820μW SAR ADC with on-chip digital calibration. In *2010 IEEE International Solid-State Circuits Conference - (ISSCC)*, pages 384–385, Feb 2010.

[157] C. C. Lee, C. Lu, R. Narayanaswamy, and J. B. Rizk. A 12b 70ms/s sar adc with digital startup calibration in 14nm cmos. In *2015 Symposium on VLSI Circuits (VLSI Circuits)*, pages C62–C63, June 2015.

[158] I. Jung and Y. Kim. A 12-bit 32MS/s SAR ADC using built-in self calibration technique to minimize capacitor mismatch. In *2014 IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT)*, pages 276–280, Oct 2014.

[159] B. Verbruggen, J. Tsouhlarakis, T. Yamamoto, M. Iriguchi, E. Martens, and J. Craninckx. A 60 db sndr 35 ms/s sar adc with comparator-noise-based stochastic residue estimation. *IEEE Journal of Solid-State Circuits*, 50(9):2002–2011, Sep. 2015.

[160] I. Banerjee and A. Sanyal. Statistical estimator for simultaneous noise and mismatch suppression in SAR ADC. *Electronics Letters*, 53(12):773–775, 2017.

[161] L. Chen, X. Tang, A. Sanyal, Y. Yoon, J. Cong, and N. Sun. A 0.7-V 0.6- μW 100-kS/s Low-Power SAR ADC With Statistical Estimation-Based Noise Reduction. *IEEE Journal of Solid-State Circuits*, 52(5):1388–1398, 2017.

[162] M. . E. Lee, W. J. Dally, and P. Chiang. Low-power area-efficient high-speed I/O circuit techniques. *IEEE Journal of Solid-State Circuits*, 35(11):1591–1599, Nov 2000.

[163] M. Yoshioka, K. Ishikawa, T. Takayama, and S. Tsukamoto. A 10-b 50-MS/s 820-μW SAR ADC With On-Chip Digital Calibration. *IEEE Transactions on Biomedical Circuits and Systems*, 4(6):410–416, Dec 2010.

[164] B. Verbruggen, J. Craninckx, M. Kuijk, P. Wambacq, and G. Van der Plas. A 2.2 mW 1.75 GS/s 5 Bit Folding Flash ADC in 90 nm Digital CMOS. *IEEE Journal of Solid-State Circuits*, 44(3):874–882, March 2009.

[165] C. Chan, Y. Zhu, S. Sin, S. U, R. P. Martins, and F. Maloberti. A 5-Bit 1.25-GS/s 4x-Capacitive-Folding Flash ADC in 65-nm CMOS. *IEEE Journal of Solid-State Circuits*, 48(9):2154–2169, Sep. 2013.

[166] N. Verma and A. P. Chandrakasan. An Ultra Low Energy 12-bit Rate-Resolution Scalable SAR ADC for Wireless Sensor Nodes. *IEEE Journal of Solid-State Circuits*, 42(6):1196–1205, June 2007.

[167] C. Lin, Y. Wei, and T. Lee. 27.7 A 10b 2.6GS/s time-interleaved SAR ADC with background timing-skew calibration. In *2016 IEEE International Solid-State Circuits Conference (ISSCC)*, pages 468–469, 2016.

[168] W. Tseng, W. Lee, C. Huang, and P. Chiu. A 12-bit 104 MS/s SAR ADC in 28 nm CMOS for Digitally-Assisted Wireless Transmitters. *IEEE Journal of Solid-State Circuits*, 51(10):2222–2231, Oct 2016.

[169] Y. Shen, X. Tang, L. Shen, W. Zhao, X. Xin, S. Liu, Z. Zhu, V. S. Sathe, and N. Sun. A 10-bit 120-MS/s SAR ADC With Reference Ripple Cancellation Technique. *IEEE Journal of Solid-State Circuits*, 55(3):680–692, 2020.

[170] G. Huang, S. Chang, Y. Lin, C. Liu, and C. Huang. A 10b 200MS/s 0.82mW SAR ADC in 40nm CMOS. In *2013 IEEE Asian Solid-State Circuits Conference (A-SSCC)*, pages 289–292, 2013.

[171] Y. Zhu, C. Chan, S. Wong, U. Seng-Pan, and R. P. Martins. Histogram-Based Ratio Mismatch Calibration for Bridge-DAC in 12-bit 120 MS/s SAR ADC. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 24(3):1203–1207, March 2016.

[172] A. H. Chang, H. Lee, and D. Boning. A 12b 50MS/s 2.1mW SAR ADC with redundancy and digital background calibration. In *2013 Proceedings of the ESSCIRC (ESSCIRC)*, pages 109–112, Sep. 2013.

[173] J. Shen, A. Shikata, L. D. Fernando, N. Guthrie, B. Chen, M. Maddox, N. Mascarenhas, R. Kapusta, and M. C. W. Coln. A 16-bit 16-MS/s SAR ADC With On-Chip Calibration in 55-nm CMOS. *IEEE Journal of Solid-State Circuits*, 53(4):1149–1160, April 2018.

[174] B. Razavi. Design Considerations for Interleaved ADCs. *IEEE Journal of Solid-State Circuits*, 48(8):1806–1817, Aug 2013.

[175] N. Kurosawa, H. Kobayashi, K. Maruyama, H. Sugawara, and K. Kobayashi. Explicit analysis of channel mismatch effects in time-interleaved ADC systems. *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*, 48(3):261–271, March 2001.

[176] T. . Tsai, P. J. Hurst, and S. H. Lewis. Bandwidth Mismatch and Its Correction in Time-Interleaved Analog-to-Digital Converters. *IEEE Transactions on Circuits and Systems II: Express Briefs*, 53(10):1133–1137, Oct 2006.

[177] J. . Eklund and F. Gustafsson. Digital offset compensation of time-interleaved ADC using random chopper sampling. In *2000 IEEE International Symposium on Circuits and Systems. Emerging Technologies for the 21st Century. Proceedings (IEEE Cat No.00CH36353)*, volume 3, pages 447–450 vol.3, May 2000.

[178] J. Fang, S. Thirunakkarasu, X. Yu, F. Silva-Rivas, C. Zhang, F. Singor, and J. Abraham. A 5-GS/s 10-b 76-mW Time-Interleaved SAR ADC in 28 nm CMOS. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 64(7):1673–1683, July 2017.

[179] J. Elbornsson, F. Gustafsson, and J. . Eklund. Blind adaptive equalization of mismatch errors in a time-interleaved A/D converter system. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 51(1):151–158, Jan 2004.

[180] N. Le Dortz, J. Blanc, T. Simon, S. Verhaeren, E. Rouat, P. Urard, S. Le Tual, D. Goguet, C. Lelandais-Perrault, and P. Benabes. 22.5 A 1.62GS/s time-interleaved SAR ADC with digital background mismatch calibration achieving interleaving spurs below 70dBFS. In *2014 IEEE International Solid-State Circuits Conference Digest of Technical Papers (ISSCC)*, pages 386–388, Feb 2014.

[181] S. M. Jamal, Daihong Fu, N. C. . Chang, P. J. Hurst, and S. H. Lewis. A 10-b 120-Msample/s time-interleaved analog-to-digital converter with digital background calibration. *IEEE Journal of Solid-State Circuits*, 37(12):1618–1627, 2002.

[182] S. K. Gupta, M. A. Inerfield, and J. Wang. A 1-GS/s 11-bit ADC With 55-dB SNDR, 250-mW Power Realized by a High Bandwidth Scalable Time-Interleaved Architecture. *IEEE Journal of Solid-State Circuits*, 41(12):2650–2657, Dec 2006.

[183] S. Le Tual, P. N. Singh, C. Curis, and P. Dautriche. 22.3 A 20GHz-BW 6b 10GS/s 32mW time-interleaved SAR ADC with Master T H in 28nm UTBB FDSOI technology. In *2014 IEEE International Solid-State Circuits Conference Digest of Technical Papers (ISSCC)*, pages 382–383, Feb 2014.

[184] K. Doris, E. Janssen, C. Nani, A. Zanikopoulos, and G. Van der Weide. A 480mW 2.6GS/s 10b 65nm CMOS time-interleaved ADC with 48.5dB SNDR up to Nyquist. In *2011 IEEE International Solid-State Circuits Conference*, pages 180–182, Feb 2011.

[185] Huawen Jin and E. K. F. Lee. A digital-background calibration technique for minimizing timing-error effects in time-interleaved ADCs. *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, 47(7):603–613, July 2000.

[186] J. Chu and H. Lee. A 450 MS/s 10-bit time-interleaved zero-crossing based ADC. In *2011 IEEE Custom Integrated Circuits Conference (CICC)*, pages 1–4, Sep. 2011.

[187] Q. Lei, Y. Zheng, D. Zhu, and L. Siek. A statistic based time skew calibration method for time-interleaved ADCs. In *2014 IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 2373–2376, June 2014.

[188] C. Wang and J. Wu. A multiphase timing-skew calibration technique using zero-crossing detection. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 56(6):1102–1114, June 2009.

[189] C. Huang, C. Wang, and J. Wu. A CMOS 6-Bit 16-GS/s Time-Interleaved ADC Using Digital Background Calibration Techniques. *IEEE Journal of Solid-State Circuits*, 46(4):848–858, April 2011.

[190] H. Wei, P. Zhang, B. D. Sahoo, and B. Razavi. An 8 Bit 4 GS/s 120 mW CMOS ADC. *IEEE Journal of Solid-State Circuits*, 49(8):1751–1761, Aug 2014.

[191] Ying-Zu Lin, Chih-Hou Tsai, Shan-Chih Tsou, and Chao-Hsin Lu. A 8.2-mW 10-b 1.6-GS/s 4× TI SAR ADC with fast reference charge neutralization and background timing-skew calibration in 16-nm CMOS. In *2016 IEEE Symposium on VLSI Circuits (VLSI-Circuits)*, pages 1–2, June 2016.

[192] B. Xu, Y. Zhou, and Y. Chiu. A 23-mW 24-GS/s 6-bit Voltage-Time Hybrid Time-Interleaved ADC in 28-nm CMOS. *IEEE Journal of Solid-State Circuits*, 52(4):1091–1100, April 2017.

[193] M. El-Chammas and B. Murmann. A 12-GS/s 81-mW 5-bit Time-Interleaved Flash ADC With Background Timing Skew Calibration. *IEEE Journal of Solid-State Circuits*, 46(4):838–847, April 2011.

[194] B. Razavi. Problem of timing mismatch in interleaved ADCs. In *Proceedings of the IEEE 2012 Custom Integrated Circuits Conference*, pages 1–8, Sep. 2012.

[195] D. Stepanovic and B. Nikolic. A 2.8 GS/s 44.6 mW Time-Interleaved ADC Achieving 50.9 dB SNDR and 3 dB Effective Resolution Bandwidth of 1.5 GHz in 65 nm CMOS. *IEEE Journal of Solid-State Circuits*, 48(4):971–982, April 2013.

[196] H. Kang, H. Hong, S. Park, K. Kim, K. Ahn, and S. Ryu. A Sign-Equality-Based Background Timing-Mismatch Calibration Algorithm for Time-Interleaved ADCs. *IEEE Transactions on Circuits and Systems II: Express Briefs*, 63(6):518–522, June 2016.

[197] T. I. Laakso, V. Valimaki, M. Karjalainen, and U. K. Laine. Splitting the unit delay [FIR/all pass filters design]. *IEEE Signal Processing Magazine*, 13(1):30–60, Jan 1996.

[198] J. Brown. Multi-channel sampling of low-pass signals. *IEEE Transactions on Circuits and Systems*, 28(2):101–106, February 1981.

[199] Won Namgoong. Finite-length synthesis filters for non-uniformly time-interleaved analog-to-digital converter. In *2002 IEEE International Symposium on Circuits and Systems. Proceedings (Cat. No.02CH37353)*, volume 4, pages IV–IV, May 2002.

[200] R. S. Prendergast, B. C. Levy, and P. J. Hurst. Reconstruction of band-limited periodic nonuniformly sampled signals through multirate filter banks. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 51(8):1612–1622, Aug 2004.

Verilog Code for the Mismatch Calibration

```
module mismatch_calib_RTL(Resetn, CLK, Enable, CMP, BUU, BDD, SW_res, ...
E3UU_1, E3UU_2 ,E4UU_1, E4UU_2, E5UU_1, E5UU_2, E6UU_1, E6UU_2, E7UU_1, E7UU_2, ...
E8UU_1, E8UU_2, E9UU_1, E9UU_2, E10UU_1, E10UU_2, ...
E3DD_1, E3DD_2, E4DD_1, E4DD_2, E5DD_1, E5DD_2, E6DD_1, E6DD_2, E7DD_1, E7DD_2,...
E8DD_1, E8DD_2, E9DD_1, E9DD_2, E10DD_1, E10DD_2, SIGMA_1,SIGMA_2, MU_1, MU_2, YYY, KK);

    parameter n = 11, logn = 4;
    parameter r = 4; // number of bits for maximum mismatch  of the cap array (in unit of LSB)
    parameter prec = 12; // prec of the mismatch correction numbers (Mem values) = (prec+r - 4) * 4mV

    parameter s_LUT = 1024, logs_LUT = 10; // number of rows for LUT
    parameter q = 16, logq = 4;// number of columns for LUT

    parameter nr = 8, log_nr = 3; // SW_reset

    //******** mu/sigma ********//
    parameter p1 = 128, logp1 = 7; // #runs1
    parameter m1 = 128, logm1 = 7; // #averaging1

    //******** calibration ********//
    parameter p2 = 128, logp2 = 7; // #runs2
    parameter m2 = 128, logm2 = 7; // #averaging2

    parameter ones2_85 =108, ones2_15 = 20; // ones2_85 = 0.85*p2 rounded DOWN, ones2_15 = 0.15*p2 rounded UP

    parameter n_SPI = 16;

    input Resetn, CLK, Enable;
    input CMP;
    wire [n-1:0] BU, BD;
    output [n-1:0] BUU, BDD;
    output SW_res;
    wire [prec+r-1:0] E3U, E4U, E5U, E6U, E7U, E8U, E9U, E10U, E3D, E4D, E5D, E6D, E7D, E8D, E9D, E10D;
    output [7:0] E3UU_1, E3UU_2 ,E4UU_1, E4UU_2, E5UU_1, E5UU_2, E6UU_1, E6UU_2,...
     E7UU_1, E7UU_2, E8UU_1, E8UU_2, E9UU_1, E9UU_2, E10UU_1, E10UU_2;
    output [7:0] E3DD_1, E3DD_2, E4DD_1, E4DD_2, E5DD_1, E5DD_2, E6DD_1, E6DD_2,...
     E7DD_1, E7DD_2, E8DD_1, E8DD_2, E9DD_1, E9DD_2, E10DD_1, E10DD_2;
    wire signed [q-1:0]  sigma, mu;
    output signed [7:0]  SIGMA_1,SIGMA_2, MU_1, MU_2;
    output [6:0] YYY;
    output [logn-1:0] KK;

    reg SW_res;
    wire signed [q-1:0] erfinv;
    reg adrs_ready;
    wire [logs_LUT-1:0] erfinv_adrs;
    wire data_ready;
    wire Clock;
    reg [6:0] y, Y;
    wire [6:0] YYY;
    wire [1:0] Sign;
    wire [n-3:0] bu_i1, bu_i2, bu_o, bd_i1, bd_i2, bd_o;
    reg calib_up;
    wire [r-1:0] Eu, Ed, EE;
```

```
55      reg E_k, L_k;
56      wire [logn-1:0] k;
57      reg E_ee, L_ee;
58      reg E_eu, L_eu , E_ed, L_ed, L_shift_bu, L0_shift_u, E_shift_bu, L_shift_bd, L0_shift_d, E_shift_bd, bu0, bd0, bu00, bd00;
59      wire bu5, bd5;
60      reg s_b5;
61      reg Clk_cmp, E_ones1, L_ones1, E_ones2, L_ones2;
62      wire Comp;
63      reg E_runs1, L_runs1, E_runs2, L_runs2;
64      wire [logp1:0] runs1;
65      wire [logp1+logm1-1:0] ones1;
66      wire [logp2:0] runs2;
67      wire [logp2:0] ones2;
68      wire [logs_LUT-1:0] erfinv_adrs1, erfinv_adrs2;
69      wire s_85, s_15;
70      reg E_sign;
71      wire [prec+r-1:0] AU1, AU2, AD1, AD2, E_sumU1, E_sumD1, E_sumU2, E_sumD2;
72      wire [prec+r-1:0] EE2, EE3, EE4, xa_ext, EE7;
73      wire [prec+r+logm2-1:0] EE5, EE5_reg, EE6;
74      reg L_EE5, E_EE5, E_avg1, L_avg1, E_avg2, L_avg2;
75      wire [logm1-1:0] N_avg1;
76      wire [logm2-1:0] N_avg2;
77      wire [prec+r-1:0] ErrorU1, ErrorU2, ErrorD1, ErrorD2;
78      reg Valid_mu_sigma;
79      reg E_erfinv1, E_erfinv0, E_erfinv_1;
80      wire [q-1:0] erfinv1, erfinv0, erfinv_1, a0, a0_abs, a1;
81      wire [q-1:0] a1_a0, a1_a0_abs;
82      wire [2*q-1:0] DataA1, DataA2, DataB, R1, R2;
83      wire signed [2*q-1:0] sigma_int, mu_int, mu_int2, xa_sigma_int;
84      wire signed [q-1:0]  xa_mu, xa_int, xa_sigma;
85      wire signed [prec-1:0] xa;
86      reg start_div, LA_div, EB_div;
87      wire Done_div1, Done_div2;
88      reg mu_sigma;
89      reg E_reset, L_reset;
90      wire [log_nr:0] N_reset;
91      wire [prec+r-1:0] E3U_2, E4U_2, E5U_2, E6U_2, E7U_2, E8U_2, E9U_2, E10U_2, E3D_2, E4D_2, E5D_2, E6D_2, E7D_2, E8D_2, E9D_2, E10D_2;
92      reg E_AU, E_AD, EEU, EED;
93
94      // control circuit
95      //****************** states for mu/sigma ******************//
96      parameter S0 = 7'b0000000, S1 = 7'b0000001, S2 = 7'b0000010, S3 = 7'b0000011, S4 = 7'b0000100;
97      parameter S5 = 7'b0000101, S6 = 7'b0000110, S7 = 7'b0000111, S8 = 7'b0001000, S9 = 7'b0001001;
98      parameter S10 = 7'b0001010, S11 = 7'b0001011, S12 = 7'b0001100, S13 = 7'b0001101, S14 = 7'b0001110;
99      parameter S15 = 7'b0001111, S16 = 7'b0010000, S17 = 7'b0010001, S18 = 7'b0010010, S19 = 7'b0010011;
100     parameter S20 = 7'b0010100, S21 = 7'b0010101, S22 = 7'b0010110, S23 = 7'b0010111, S24 = 7'b0011000;
101     parameter S25 = 7'b0011001, S26 = 7'b0011010, S27 = 7'b0011011, S28 = 7'b0011100, S29 = 7'b0011101;
102     parameter S30 = 7'b0011110, S31 = 7'b0011111;
103     parameter S6_1 = 7'b1011110, S12_1 = 7'b1011111, S18_1 = 7'b1100000;  // 94, 95, 96
104
105
106     //*************** states for calibration (Up) ****************//
107     parameter S32 = 7'b0100000, S33 = 7'b0100001, S34 = 7'b0100010, S35 = 7'b0100011, S36 = 7'b0100100;
108     parameter S37 = 7'b0100101, S38 = 7'b0100110, S39 = 7'b0100111, S40 = 7'b0101000, S41 = 7'b0101001;
109     parameter S42 = 7'b0101010, S43 = 7'b0101011, S44 = 7'b0101100, S45 = 7'b0101101, S46 = 7'b0101110;
110     parameter S47 = 7'b0101111, S48 = 7'b0110000, S49 = 7'b0110001, S50 = 7'b0110010, S51 = 7'b0110011;
111     parameter S43_1 = 7'b1001000, S43_2 = 7'b1001001; // 72,73
112
113     //*************** states for calibration (Down) ****************//
114     parameter S52 = 7'b0110100, S53 = 7'b0110101, S54 = 7'b0110110, S55 = 7'b0110111, S56 = 7'b0111000;
115     parameter S57 = 7'b0111001, S58 = 7'b0111010, S59 = 7'b0111011, S60 = 7'b0111100, S61 = 7'b0111101;
116     parameter S62 = 7'b0111110, S63 = 7'b0111111, S64 = 7'b1000000, S65 = 7'b1000001, S66 = 7'b1000010;
117     parameter S67 = 7'b1000011, S68 = 7'b1000100, S69 = 7'b1000101, S70 = 7'b1000110, S71 = 7'b1000111;
118     parameter S63_1 = 7'b1011100, S63_2 = 7'b1011101; // 92,93
119
120
121     always @*
122     begin: State_table
123         case(y)
124
125             //******** mu/sigma calculation ********//
126             S0: Y = S1; // reset everything
127             //*** VX = +LSB *** //
128             S1: if (N_reset !=  nr) Y = S1;
129                 else Y = S2;
```

```
130             S2: Y = S3;
131             S3: Y = S4;
132             S4: if (runs1 != {1'b1,{(logp1){1'b0}}}) Y = S4;
133                 else Y = S5;
134             S5: if (N_avg1 != {(logm1){1'b1}}) Y = S1;
135                  else Y = S6;
136             S6:if (!data_ready) Y = S6;
137                 else Y = S6_1;
138             S6_1: Y = S7;
139             //*** VX = 0 *** //
140             S7: if (N_reset != nr) Y = S7;
141                 else Y = S8;
142             S8: Y = S9;
143             S9: Y = S10;
144             S10: if (runs1 != {1'b1,{(logp1){1'b0}}}) Y = S10;
145                 else Y = S11;
146             S11: if (N_avg1 != {(logm1){1'b1}}) Y = S7;
147                  else Y = S12;
148             S12: if (!data_ready) Y = S12;
149                 else Y = S12_1;
150             S12_1: Y = S13;
151             //*** VX = -LSB *** //
152             S13:  if (N_reset != nr) Y = S13;
153                 else Y = S14;
154             S14: Y = S15;
155             S15: Y = S16;
156             S16: if (runs1 != {1'b1,{(logp1){1'b0}}}) Y = S16;
157                 else Y = S17;
158             S17: if (N_avg1 != {(logm1){1'b1}}) Y = S13;
159                  else Y = S18;
160             S18: if (!data_ready) Y = S18;
161                 else Y = S18_1;
162             S18_1: Y = S19;
163             S19: Y = S20;
164             S20: if (Done_div1 & Done_div2) Y = S21;
165                 else Y = S20;
166             S21: Y = S32;
167
168             //******** Calibration of Cap-ArrayU ********//
169             S32: Y = S33;
170             S33: Y = S34;
171             S34: if (N_reset != nr) Y = S34;
172                 else Y = S35;
173             S35: Y = S36;
174             S36: Y = S37;
175             S37: if (runs2 != {1'b1,{(logp2){1'b0}}}) Y = S37;
176                 else Y = S38;
177             S38: Y = S39;
178             S39: Y = S40;
179             S40: if (runs2 != {1'b1,{(logp2){1'b0}}}) Y = S40;
180                 else Y = S41;
181             S41: if (((s_85 & Sign == 2'b01) | (s_15 & Sign == 2'b10)) & (Sign != 2'b00)) Y = S42;
182                 else Y = S43;
183             S42: Y = S40;
184             S43: if (!data_ready) Y = S43;
185                  else Y = S43_1;
186             S43_1: if (N_avg2 != {(logm2){1'b1}}) Y = S43_2;
187                  else Y = S44;
188             S43_2: Y = S34;
189             S44: Y = S45;
190             S45: Y = S46;
191             S46: Y = S47;
192             S47: Y = S48;
193             S48: Y = S49;
194             S49: Y = S50;
195             S50: Y = S51;
196             S51: Y = S52;
197
198             //******** Calibration of Cap-ArrayD ********//
199             S52: Y = S53;
200             S53: Y = S54;
201             S54:  if (N_reset != nr) Y = S54;
202                 else Y = S55;
203             S55: Y = S56;
204             S56: Y = S57;
```

```
205             S57: if (runs2 != {1'b1,{(logp2){1'b0}}}) Y = S57;
206                 else Y = S58;
207             S58: Y = S59;
208             S59: Y = S60;
209             S60: if (runs2 != {1'b1,{(logp2){1'b0}}}) Y = S60;
210                 else Y = S61;
211             S61: if (((s_85 & Sign == 2'b01) | (s_15 & Sign == 2'b10)) & (Sign != 2'b00)) Y = S62;
212                 else Y = S63;
213             S62: Y = S60;
214             S63: if (!data_ready) Y = S63;
215                     else Y = S63_1;
216             S63_1: if (N_avg2 != {(logm2){1'b1}}) Y = S63_2;
217                     else Y = S64;
218             S63_2: Y = S54;
219             S64: Y = S65;
220             S65: Y = S66;
221             S66: Y = S67;
222             S67: Y = S68;
223             S68: Y = S69;
224             S69: if (k != 10) Y = S33;
225                 else Y = S70;
226             S70: Y = S71;
227             S71: Y = S71;
228 
229             default: Y = 7'bxxxxxxx;
230         endcase
231     end
232 
233     always @(posedge Clock or negedge Resetn)
234     begin: State_flipflops
235         if (!Resetn)
236             y <= S0;
237         else
238             y <= Y;
239     end
240 
241     always @*
242     begin: FSM_outputs
243         //defaults
244         SW_res = 0; E_k = 0; L_k = 0; E_ee = 0; L_ee = 0;
245         E_eu = 0; L_eu = 0; E_ed = 0; L_ed = 0; L_shift_bu = 0; L0_shift_u = 0;
246         E_shift_bu = 0; L_shift_bd = 0; L0_shift_d = 0; E_shift_bd = 0; bu0 = 0; bd0 = 0; s_b5 = 1;
247         bu00 = 0; bd00 = 0;
248         Clk_cmp = 0; E_ones1 = 0; L_ones1 = 0; E_ones2 = 0; L_ones2 = 0;
249         E_sign = 0;
250         E_runs1 = 0; L_runs1 = 0; E_runs2 = 0; L_runs2 = 0;
251         adrs_ready = 0;
252         E_erfinv1 = 0; E_erfinv0 = 0; E_erfinv_1 = 0;
253         start_div = 0; LA_div = 0; EB_div = 0;
254         L_EE5 = 0; E_EE5 = 0;
255         L_avg1 = 0; E_avg1 = 0; L_avg2 = 0; E_avg2 = 0;
256         L_reset = 0; E_reset = 0;
257         E_AU = 0; E_AD = 0; EEU = 0; EED = 0;
258         case(y)
259 
260             //******** mu/sigma calculation ********//
261             S0: begin // reset everything
262                 Valid_mu_sigma = 0;
263                 mu_sigma = 1;
264                 SW_res = 1;
265                 L_reset = 1;
266                 L_runs1 = 1;
267                 L_ones1 = 1;
268                 L_avg1 = 1;
269                 L_runs2 = 1;
270                 L_ones2 = 1;
271                 L_avg2 = 1;
272                 L_k = 1;
273                 L_ee = 1;
274                 L_eu = 1;
275                 L0_shift_u = 1;
276                 L_ed = 1;
277                 L0_shift_d = 1;
278                 Clk_cmp = 0;
279                 L_EE5 = 1;
```

```
280             end
281             //** VX = +LSB **//
282             S1: begin
283                 E_reset = 1;
284                 SW_res = 1;
285                 L_runs1 = 1;
286             end
287             S2: begin // releasing the reset
288                 L_reset = 1;
289                 SW_res = 0;
290             end
291             S3: begin
292                 bu00 = 1;
293                 bd00 = 0;
294             end
295             S4: begin
296                 bu00 = 1;
297                 bd00 = 0;
298                 E_runs1 = 1;
299                 if (Comp) E_ones1 = 1;
300                 else E_ones1 = 0;
301             end
302             S5: begin
303                 E_avg1 = 1;
304             end
305             S6: begin
306                 adrs_ready = 1;
307                 E_erfinv1 = 1;
308             end
309             S6_1: begin
310                 L_ones1 = 1;
311             end
312             //** VX = 0 **//
313             S7: begin
314                 E_reset = 1;
315                 SW_res = 1;
316                 L_runs1 = 1;
317             end
318             S8: begin // releasing the reset
319                 L_reset = 1;
320                 SW_res = 0;
321             end
322             S9: begin
323                 bu00 = 0;
324                 bd00 = 0;
325             end
326             S10: begin
327                 bu00 = 0;
328                 bd00 = 0;
329                 E_runs1 = 1;
330                 if (Comp) E_ones1 = 1;
331                 else E_ones1 = 0;
332             end
333             S11: begin
334                 E_avg1 = 1;
335             end
336             S12: begin
337                 adrs_ready = 1;
338                 E_erfinv0 = 1;
339             end
340             S12_1: begin
341                 L_ones1 = 1;
342             end
343             //** VX = -LSB **//
344             S13: begin
345                 E_reset = 1;
346                 SW_res = 1;
347                 L_runs1 = 1;
348             end
349             S14: begin // releasing the reset
350                 L_reset = 1;
351                 SW_res = 0;
352             end
353             S15: begin
354                 bu00 = 0;
```

```
355                 bd00 = 1;
356             end
357             S16: begin
358                 bu00 = 0;
359                 bd00 = 1;
360                 E_runs1 = 1;
361                 if (Comp) E_ones1 = 1;
362                 else E_ones1 = 0;
363             end
364             S17: begin
365                 E_avg1 = 1;
366             end
367             S18: begin
368                 adrs_ready = 1;
369                 E_erfinv_1 = 1;
370             end
371             S18_1: begin
372                 L_ones1 = 1; // ones1 = 0
373             end
374             S19: begin
375                 start_div = 1;
376                 LA_div = 1;
377                 EB_div = 1;
378             end
379             S20:; // mu/sigma calculation (division)
380             S21: Valid_mu_sigma = 1;
381
382             //******** Calibration of Cap-ArrayU ********//
383             S32: begin //reset
384                 mu_sigma = 0;
385                 SW_res = 1;
386                 L0_shift_u = 1;
387                 L0_shift_d = 1;
388                 L_ee = 1;
389                 L_ones2 = 1;
390                 Clk_cmp = 0;
391                 L_runs2 = 1;
392                 L_k = 1;
393                 calib_up = 1;
394             end
395             S33: begin
396                 L_ones2 = 1;
397                 L_ee = 1; // EE = 0
398                 SW_res = 1;
399                 E_shift_bu = 1;
400                 bu0 = 1;
401                 calib_up = 1;
402             end
403             S34: begin // BU = 00000000110
404                 E_reset = 1;
405                 SW_res = 1;
406                 bu0 = 1;
407                 s_b5 = 0;
408             end
409             S35: begin // releasing the reset
410                 L_reset = 1;
411                 SW_res = 0;
412                 bu0 = 1;
413                 s_b5 = 0;
414             end
415             S36: begin
416                 E_eu = 1;
417                 bu0 = 1;
418                 s_b5 = 0;
419             end
420             S37: begin // BU = 00000001000 // running the comparator to find Sign
421                 Clk_cmp = 1;
422                 E_runs2 = 1;
423                 if (Comp) E_ones2 = 1;
424                 else E_ones2 = 0;
425             end
426             S38:begin
427                 E_sign = 1;
428             end
429             S39: begin
```

```
430                 L_runs2 = 1;
431                 L_ones2 = 1;
432             end
433             S40: begin // running the comparator to find ones2
434                 Clk_cmp = 1;
435                 E_runs2 = 1;
436                 if (Comp) E_ones2 = 1;
437                 else E_ones2 = 0;
438             end
439             S41:L_runs2 = 1;
440             S42: begin
441                 E_ee = 1;
442                 if (Sign == 2'b01) E_ed = 1;
443                 else E_eu = 1;
444                 L_ones2 = 1; // added recently
445             end
446             S43: begin // xa evaluation from the erfinv-LUT
447                 adrs_ready = 1;
448             end
449             S43_1: begin
450                 E_EE5 = 1; // averaging
451                 E_avg2 = 1; // N_avg2++
452             end
453             S43_2: begin
454                 L_eu = 1;
455                 L_ed = 1;
456                 L_runs2 = 1;
457                 L_ones2 = 1;
458                 L_ee = 1;
459             end
460             S44: begin
461                 L_avg2 = 1; // N_avg2 = 0;
462             end
463             S45:; //waiting for the sum_array to finish
464             S46: ; //waiting for the additions to finish (EE7)
465             S47: begin // AU1 -> AU2
466                 E_AU = 1;
467             end
468             S48: begin // writting in the memory (AU2 -> EiU)
469                 EEU = 1;
470             end
471             S49: begin
472                 L_eu = 1;
473                 L_ed = 1;
474                 L_ones2 = 1;
475                 L_runs2 = 1;
476                 L_EE5 = 1;
477                 L_ee = 1;
478                 SW_res = 1;
479             end
480             S50: SW_res = 1;
481             S51: SW_res = 1;
482
483             //******** Calibration of Cap-ArrayD ********//
484             S52: begin //reset
485                 L_ones2 = 1;
486                 Clk_cmp = 0;
487                 L_runs2 = 1;
488                 calib_up = 0;
489                 SW_res = 1;
490             end
491             S53: begin
492                 L_ones2 = 1;
493                 L_ee = 1;
494                 SW_res = 1;
495                 E_shift_bd = 1;
496                 bd0 = 1;
497                 calib_up = 0;
498             end
499             S54: begin // BD = 00000000110
500                 E_reset = 1;
501                 SW_res = 1;
502                 bd0 = 1;
503                 s_b5 = 0;
504             end
```

```
505             S55: begin // releasing the reset
506                 L_reset = 1;
507                 SW_res = 0;
508                 bd0 = 1;
509                 s_b5 = 0;
510             end
511             S56: begin
512                 E_ed = 1;
513                 bd0 = 1;
514                 s_b5 = 0;
515             end
516             S57: begin // BD = 00000001000 // running the comparator to find Sign
517                 Clk_cmp = 1;
518                 E_runs2 = 1;
519                 if (Comp) E_ones2 = 1;
520                 else E_ones2 = 0;
521             end
522             S58:begin
523                 E_sign = 1;
524             end
525             S59: begin
526                 L_runs2 = 1;
527                 L_ones2 = 1;
528             end
529             S60: begin // running the comparator to find ones2
530                 Clk_cmp = 1;
531                 E_runs2 = 1;
532                 if (Comp) E_ones2 = 1;
533                 else E_ones2 = 0;
534             end
535             S61: begin
536                 L_runs2 = 1;
537             end
538             S62: begin
539                 E_ee = 1;
540                 if (Sign == 2'b01) E_ed = 1;
541                 else E_eu = 1;
542                 L_ones2 = 1; // added recently
543             end
544             S63: begin // xa evaluation from the erfinv-LUT
545                 adrs_ready = 1;
546             end
547             S63_1: begin
548                 E_EE5 = 1; // averaging
549                 E_avg2 = 1; // N_avg2++
550             end
551             S63_2: begin
552                 L_eu = 1;
553                 L_ed = 1;
554                 L_runs2 = 1;
555                 L_ones2 = 1;
556                 L_ee = 1;
557             end
558             S64: begin
559                 L_avg2 = 1; // added revcently;
560             end
561             S65:; //waiting for the sum_array to finish
562             S66:; //waiting for the additions to finish (EE7)
563             S67: begin // AD1 -> AD2
564                 E_AD = 1;
565             end
566             S68: begin //writting in the memory (AD1 -> EiD)
567                 EED = 1;
568             end
569             S69: begin
570                 E_k = 1;
571                 L_eu = 1;
572                 L_ed = 1;
573                 L_EE5 = 1;
574             end
575             S70:;
576             S71:;
577 
578             default:;
579         endcase
```

```
580     end
581
582     assign Clock = Enable & CLK;
583
584     //Comparator register
585     regne#(.n(1)) Regcmp (.R(CMP), .Clock(Clock), .Resetn(Resetn), .E(1'b1), .Q(Comp));
586    //Counter for SW_reset
587     upcount#(.n(log_nr+1)) Counter_reset(.Resetn(Resetn), .R({{(log_nr+1){1'b0}}),...
588      .Clock(Clock), .E(E_reset), .L(L_reset), .Q(N_reset));
589
590
591     //*************** mu/sigma **************//
592     // Counter to find number of runs1
593     upcount#(.n(logp1+1)) Counter_runs1(.Resetn(Resetn), .R({{(logp1+1){1'b0}}), .Clock(Clock), .E(E_runs1), .L(L_runs1), .Q(runs1));
594     // Counter to find ones1
595     upcount#(.n(logp1+logm1)) Counter_ones1(.Resetn(Resetn), .R({{(logp1+logm1){1'b0}}), .Clock(Clock),...
596      .E(E_ones1), .L(L_ones1), .Q(ones1)); // assumption: ones never becomes 2^(logp1+logm1)
597     // Averaging
598     upcount#(.n(logm1)) Counter_avg1(.Resetn(Resetn), .R({{(logm1){1'b0}}), .Clock(Clock), .E(E_avg1), .L(L_avg1), .Q(N_avg1));
599     // Adjusting ones to LUT
600     assign erfinv_adrs1 = ones1[logp1+logm1-1:logp1+logm1-logs_LUT]; //  logs_LUT bits
601
602     regne#(.n(q)) Reg_erfinv1 (.R(erfinv), .Clock(Clock), .Resetn(Resetn), .E(E_erfinv1), .Q(erfinv1));
603     regne#(.n(q)) Reg_erfinv0 (.R(erfinv), .Clock(Clock), .Resetn(Resetn), .E(E_erfinv0), .Q(erfinv0));
604     regne#(.n(q)) Reg_erfinv_1 (.R(erfinv), .Clock(Clock), .Resetn(Resetn), .E(E_erfinv_1), .Q(erfinv_1));
605
606     assign a0 = erfinv0;
607     assign a0_abs = a0[q-1] ? -a0 : a0;
608     assign a1 = erfinv0[q-1] ? erfinv1 : erfinv_1;
609     assign a1_a0 = a1 - a0; // a1-a0 < a1 -> always within q bits
610     assign a1_a0_abs = a1_a0[q-1] ? -a1_a0 : a1_a0;
611     assign DataA1 = {4'b0001,{(2*q-4){1'b0}}}; // DataA = 2^(2*q-4)
612     assign DataA2 = {a0_abs,{(q-2){1'b0}}};
613     assign DataB = {{(q){1'b0}},a1_a0_abs};
614
615     divider#(.n(2*q),.logn(logq+1)) divider_sigma (.Clock(Clock), .Resetn(Resetn),...
616      .start(start_div), .LA(LA_div), .EB(EB_div), .DataA(DataA1), .DataB(DataB), .R(R1), .Q(sigma_int), .Done(Done_div1));
617     divider#(.n(2*q),.logn(logq+1)) divider_mu (.Clock(Clock), .Resetn(Resetn),...
618      .start(start_div), .LA(LA_div), .EB(EB_div), .DataA(DataA2), .DataB(DataB), .R(R2), .Q(mu_int), .Done(Done_div2));
619
620     assign mu_int2 = a0[q-1] ? - mu_int : mu_int;
621
622     assign sigma = sigma_int[q:1]; // real = ./2^(q-3)
623     assign mu = mu_int2[q:1]; // real = ./2^(q-3)
624     //************************************//
625
626     //*************** Calibration **************//
627     // Counter to find number of runs2
628     upcount#(.n(logp2+1)) Counter_runs2(.Resetn(Resetn), .R({{(logp2+1){1'b0}}), .Clock(Clock), .E(E_runs2), .L(L_runs2), .Q(runs2));
629     // Counter to find ones2
630     upcount#(.n(logp2+1)) Counter_ones2(.Resetn(Resetn), .R({{(logp2+1){1'b0}}), .Clock(Clock), .E(E_ones2), .L(L_ones2), .Q(ones2));
631     // Averaging
632     upcount#(.n(logm2)) Counter_avg2(.Resetn(Resetn), .R({{(logm2){1'b0}}), .Clock(Clock), .E(E_avg2), .L(L_avg2), .Q(N_avg2));
633     // Adjusting ones to LUT
634     assign erfinv_adrs2 = {ones2,{(logs_LUT-logp2){1'b0}}};
635     // As erfinv is only chekced when ones2 != 10000, we can assume ones2 != 10000, this is for when logp2<=logs_LUT
636
637     // Counter for k
638     upcount#(.n(4)) Counter_k(.Resetn(Resetn), .R(4'b0011), .Clock(Clock), .E(E_k), .L(L_k), .Q(k));
639     // Counter for E
640     upcount#(.n(r)) Counter_E(.Resetn(Resetn), .R({{r}{1'b0}}), .Clock(Clock), .E(E_ee), .L(L_ee), .Q(EE));
641
642     //******** BLOCK1 ********//
643     // Counter for Eu
644     upcount#(.n(r)) Counter_eu(.Resetn(Resetn), .R({{r}{1'b0}}), .Clock(Clock), .E(E_eu), .L(L_eu), .Q(Eu));
645     // Register and right-shift for bu
646     shiftlne#(.n(n-2)) shift_bu(.R({{n-2}{1'b0}}), .L(L_shift_bu), .L0(L0_shift_u),...
647      .E(E_shift_bu), .w(1'b1), .Clock(Clock), .Resetn(Resetn), .Q(bu_i1));
648     assign bu_i2 = calib_up ? bu_i1 : {(n-3){1'b0}};
649     assign bu_o = bu_i2 + Eu;
650     assign bu5 = s_b5 ? bu_o[3] : 1'b0;
651     assign BU = {bu_o[n-3:4],bu5,bu_o[2:0],bu0,bu00};
652     // Counter for Ed
653     upcount#(r) Counter_ed(.Resetn(Resetn), .R({{r}{1'b0}}), .Clock(Clock), .E(E_ed), .L(L_ed), .Q(Ed));
654     // Register and right-shift for bd
```

```
655     shiftlne#(.n(n-2)) shift_bd(.R({(n-2){1'b0}}), .L(L_shift_bd), .L0(L0_shift_d),...
656      .E(E_shift_bd), .w(1'b1), .Clock(Clock), .Resetn(Resetn), .Q(bd_i1));
657     assign bd_i2 = !calib_up ? bd_i1 : {(n-3){1'b0}};
658     assign bd_o = bd_i2 + Ed;
659     assign bd5 = s_b5 ? bd_o[3] : 1'b0;
660     assign BD = {bd_o[n-3:4],bd5,bd_o[2:0],bd0,bd00};
661
662     //******** BLOCK2 ********//
663     assign s_85 = ($signed(ones2 - ones2_85) > 0 ) ;
664     assign s_15 = ($signed(ones2 - ones2_15) < 0 ) ;
665     regne#(.n(2)) Regsign (.R({s_15,s_85}), .Clock(Clock), .Resetn(Resetn), .E(E_sign), .Q(Sign));
666
667     assign xa_sigma_int = erfinv*sigma; //2*q bits
668     assign xa_sigma = xa_sigma_int[2*q-1:q]; // q bits , real = ./2^(q-5)
669     assign xa_mu = {mu[q-1],mu[q-1],mu[q-1:2]}; // q bits, real = ./2^(q-5)
670     assign xa_int = xa_sigma-xa_mu; // q bits, real = ./2^(q-5)
671     assign xa = xa_int[q-1:q-prec]; // prec bits, real = ./2^(prec-5) of 1.98mV
672
673     //******** BLOCK3 ********//
674     assign EE2 = {4'b0000, EE, {(prec-4){1'b0}}}; // ./2^(prec-4) of 1.98mV
675
676     assign EE3 = (Sign == 2'b01) ? +EE2 : -EE2;
677     assign xa_ext = {{(r){xa[prec-1]}}, xa};
678     assign EE4 = EE3 + xa_ext;
679     assign EE5 = {{(logm2){EE4[prec+r-1]}},EE4};
680     regne2#(.n(prec+r+logm2)) Reg_EE5 (.R(EE6), .Clock(Clock), .Resetn(Resetn), .L(L_EE5), .E(E_EE5), .Q(EE5_reg));
681     assign EE6 = EE5 + EE5_reg;
682     assign EE7 = EE5_reg[prec+r+logm2-1:logm2];
683
684     assign ErrorU1 = E_sumU1 + E_sumD2 + EE7;
685     assign ErrorU2 = E_sumU1 - E_sumU2 + EE7;
686     assign AU1 = (Sign == 2'b01) ? ErrorU1 : ErrorU2; // real = ./2^(prec-4)
687     regne#(.n(prec+r)) Reg_AU (.R(AU1), .Clock(Clock), .Resetn(Resetn), .E(E_AU), .Q(AU2));
688
689     assign ErrorD1 = E_sumD1 - E_sumD2 - EE7;
690     assign ErrorD2 = E_sumD1 +  E_sumU2 - EE7;
691     assign AD1 = (Sign == 2'b01) ? ErrorD1 : ErrorD2; // real = ./2^(prec-4)
692     regne#(.n(prec+r)) Reg_AD (.R(AD1), .Clock(Clock), .Resetn(Resetn), .E(E_AD), .Q(AD2));
693     //*****************************************
694
695
696     regne#(.n(prec+r)) Reg_E3U (.R(AU2), .Clock(Clock), .Resetn(Resetn), .E((k == 3) & EEU), .Q(E3U));
697     regne#(.n(prec+r)) Reg_E4U (.R(AU2), .Clock(Clock), .Resetn(Resetn), .E((k == 4) & EEU), .Q(E4U));
698     regne#(.n(prec+r)) Reg_E5U (.R(AU2), .Clock(Clock), .Resetn(Resetn), .E((k == 5) & EEU), .Q(E5U));
699     regne#(.n(prec+r)) Reg_E6U (.R(AU2), .Clock(Clock), .Resetn(Resetn), .E((k == 6) & EEU), .Q(E6U));
700     regne#(.n(prec+r)) Reg_E7U (.R(AU2), .Clock(Clock), .Resetn(Resetn), .E((k == 7) & EEU), .Q(E7U));
701     regne#(.n(prec+r)) Reg_E8U (.R(AU2), .Clock(Clock), .Resetn(Resetn), .E((k == 8) & EEU), .Q(E8U));
702     regne#(.n(prec+r)) Reg_E9U (.R(AU2), .Clock(Clock), .Resetn(Resetn), .E((k == 9) & EEU), .Q(E9U));
703     regne#(.n(prec+r)) Reg_E10U (.R(AU2), .Clock(Clock), .Resetn(Resetn), .E((k == 10) & EEU), .Q(E10U));
704
705     regne#(.n(prec+r)) Reg_E3D (.R(AD2), .Clock(Clock), .Resetn(Resetn), .E((k == 3) & EED), .Q(E3D));
706     regne#(.n(prec+r)) Reg_E4D (.R(AD2), .Clock(Clock), .Resetn(Resetn), .E((k == 4) & EED), .Q(E4D));
707     regne#(.n(prec+r)) Reg_E5D (.R(AD2), .Clock(Clock), .Resetn(Resetn), .E((k == 5) & EED), .Q(E5D));
708     regne#(.n(prec+r)) Reg_E6D (.R(AD2), .Clock(Clock), .Resetn(Resetn), .E((k == 6) & EED), .Q(E6D));
709     regne#(.n(prec+r)) Reg_E7D (.R(AD2), .Clock(Clock), .Resetn(Resetn), .E((k == 7) & EED), .Q(E7D));
710     regne#(.n(prec+r)) Reg_E8D (.R(AD2), .Clock(Clock), .Resetn(Resetn), .E((k == 8) & EED), .Q(E8D));
711     regne#(.n(prec+r)) Reg_E9D (.R(AD2), .Clock(Clock), .Resetn(Resetn), .E((k == 9) & EED), .Q(E9D));
712     regne#(.n(prec+r)) Reg_E10D (.R(AD2), .Clock(Clock), .Resetn(Resetn), .E((k == 10) & EED), .Q(E10D));
713
714     assign E_sumU1 = E3U + E4U + E6U + E7U + E8U + E9U + E10U; // E5U excluded
715     assign E_sumD1 = E3D + E4D + E6D + E7D + E8D + E9D + E10D;
716
717     assign E3U_2 = BU[3] ? E3U : 0;
718     assign E4U_2 = BU[4] ? E4U : 0;
719     assign E5U_2 = BU[5] ? E5U : 0;
720     assign E6U_2 = BU[6] ? E6U : 0;
721     assign E7U_2 = BU[7] ? E7U : 0;
722     assign E8U_2 = BU[8] ? E8U : 0;
723     assign E9U_2 = BU[9] ? E9U : 0;
724     assign E10U_2 = BU[10] ? E10U : 0;
725
726     assign E3D_2 = BD[3] ? E3D : 0;
727     assign E4D_2 = BD[4] ? E4D : 0;
728     assign E5D_2 = BD[5] ? E5D : 0;
729     assign E6D_2 = BD[6] ? E6D : 0;
```

```
730       assign E7D_2 = BD[7] ? E7D : 0;
731       assign E8D_2 = BD[8] ? E8D : 0;
732       assign E9D_2 = BD[9] ? E9D : 0;
733       assign E10D_2 = BD[10] ? E10D : 0;
734
735       assign E_sumU2 = E3U_2 + E4U_2 + E5U_2 + E6U_2 + E7U_2 + E8U_2 + E9U_2 + E10U_2;
736       assign E_sumD2 = E3D_2 + E4D_2 + E5D_2 + E6D_2 + E7D_2 + E8D_2 + E9D_2 + E10D_2;
737
738       assign erfinv_adrs = mu_sigma ? erfinv_adrs1 : erfinv_adrs2;
739       LUT_erfinv#(.n(q), .m(s_LUT), .logm(logs_LUT)) LUT(.Adrs(erfinv_adrs),...
740        .data_out(erfinv), .read_en(adrs_ready), .data_ready(data_ready), .Clock(Clock), .Resetn(Resetn));
741
742       assign YYY = y;
743
744       assign E3UU_1 = E3U[7:0];
745       assign E3UU_2 = E3U[15:8];
746       assign E4UU_1 = E4U[7:0];
747       assign E4UU_2 = E4U[15:8];
748       assign E5UU_1 = E5U[7:0];
749       assign E5UU_2 = E5U[15:8];
750       assign E6UU_1 = E6U[7:0];
751       assign E6UU_2 = E6U[15:8];
752       assign E7UU_1 = E7U[7:0];
753       assign E7UU_2 = E7U[15:8];
754       assign E8UU_1 = E8U[7:0];
755       assign E8UU_2 = E8U[15:8];
756       assign E9UU_1 = E9U[7:0];
757       assign E9UU_2 = E9U[15:8];
758       assign E10UU_1 = E10U[7:0];
759       assign E10UU_2 = E10U[15:8];
760
761       assign E3DD_1 = E3D[7:0];
762       assign E3DD_2 = E3D[15:8];
763       assign E4DD_1 = E4D[7:0];
764       assign E4DD_2 = E4D[15:8];
765       assign E5DD_1 = E5D[7:0];
766       assign E5DD_2 = E5D[15:8];
767       assign E6DD_1 = E6D[7:0];
768       assign E6DD_2 = E6D[15:8];
769       assign E7DD_1 = E7D[7:0];
770       assign E7DD_2 = E7D[15:8];
771       assign E8DD_1 = E8D[7:0];
772       assign E8DD_2 = E8D[15:8];
773       assign E9DD_1 = E9D[7:0];
774       assign E9DD_2 = E9D[15:8];
775       assign E10DD_1 = E10D[7:0];
776       assign E10DD_2 = E10D[15:8];
777
778
779
780       assign SIGMA_1 = sigma[7:0];
781       assign SIGMA_2 = sigma[15:8];
782       assign MU_1 = mu[7:0];
783       assign MU_2 = mu[15:8];
784
785       assign KK = k;
786
787       assign BUU = BD;
788       assign BDD = BU;
789
790
791   endmodule
```