Doctoral Dissertation

Doctoral Program in Electrical, Electronics and Communications Engineering
(35th cycle)

Energy-efficient Data-Processing Architectures

Coping with device and circuit-level nonidealities

By

Carmine Paolino

******

Supervisors:
Prof. Gianluca Setti, Supervisor
Prof. Fabio Pareschi, Co-Supervisor

Politecnico di Torino
2023

Declaration

* This dissertation is presented in partial fulfillment of the requirements for **Ph.D. degree** in the Graduate School of Politecnico di Torino (ScuDo).

Acknowledgements

My doctoral experience has been an unexpectedly invaluable occasion for personal growth, surpassing professional development. Navigating an uncharted research path presented numerous technical challenges, requiring extensive literature reviews, brainstorming sessions, trials, and occasional setbacks. However, the immense mental energy expended on contemplating desires, relationships, and the management of the human resource was beyond my expectations.

As this journey concludes, I have gained many valuable lessons and a deeper understanding of myself. I owe this primarily to the individuals who supported me during difficult times, those who stimulated my thoughts, made me laugh, helped me relax, and provided new, sometimes unwelcome, perspectives on life's realities.

For their contributions, I am sincerely grateful.

Rather than providing a list of names, I believe recounting memorable experiences will better convey a vivid image. Here is a glimpse into some of these situations that prompted profound self-reflection.

Enjoying a kebab while reminiscing about the times when spicy Chinese soups and underground billiard halls were the norm. Crafting an unnecessarily elaborate wrapping scheme for a perfume gift following a strenuous climbing session. Chewing a boring yet efficient homemade lunch while gazing out the windows of the department corridors. Engaging in a spirited debate from the comfort of a couch, comparing the Coven season of American Horror Story with the Asylum season. Enduring a poorly cooked "patan'e cangareddr"' while gazing at the majestic Cervino mountain from a snowy roadside. Celebrating a promotion with a warm hug in a Bolognese square, after months of virtual interactions. Beginning a day at the Ionian Sea by indulging in deliciously oily focaccia on a scorching summer day in Calabria. Constructing a miniature orchestral stage using sand and wooden sticks, eagerly anticipating a shot of Limoncello on a rooftop. Taking an impulsive plunge into the

sea, with shoes still on, after a rain-soaked walk and succumbing to the inevitable mud streams. Engaging in a lively debate on whether slippers should be shared or not, while sipping a 25-year-old Portuguese wine on the Basque coast. Lying on the side of a cliff, admiring the practicality of cut-off-fingertip handyman's gloves after an entertaining via ferrata climb (which, according to one, should not be attempted in winter!).

And if I may mention but a couple of names, I will forever be indebted to Andrea and Mariaelena. Life-changing encounters that persisted despite my initial inclination towards solitude. A friend turned family, and a climbing companion turned much, much more. Buddies in numerous memorable experiences, critically honest when needed, always supportive.

To Mariaelena, Andrea and all those who shared parts of this journey with me, I offer my heartfelt gratitude.

And now, let's get to the meat.

July 2023
White Mountains,
New Hampshire, USA

Abstract

This thesis focuses on designing and evaluating low-energy systems for signal acquisition and processing. The main theoretical foundation for the work is Compressed Sensing (CS), i.e., a theory that guarantees correct signal recovery despite violating the overarching constraints imposed by Shannon's Sampling Theorem. When a signal can be represented sparsely in a given vector basis, CS enables low-energy encoding and significantly fewer measurements than traditional sampling. The sparsity constraint is commonly satisfied in various fields, ranging from biological signals, such as electrocardiograms and electroencephalograms, to radar pulses and multi-tone RF communication systems. The thesis presents the detailed analysis of two CS encoders with appropriate strategies to address hardware nonidealities, either by ad-hoc circuit design or by appropriate decoding techniques.

The first encoder is a Charge-Redistribution (CR), Successive-Approximation-Register (SAR) Analog-to-Digital converter (ADC). Its architecture redefines the sampling operations of a typical CR SAR ADC by allowing a more granular decomposition of the capacitive array. The independent control of its capacitive elements enables the simultaneous storage of individual samples and the computation of the encoder output through entirely passive means. Therefore, energy consumption is significantly reduced compared to previously proposed architectures. Parasitic capacitances, leakage currents, and matching are some of the issues being addressed, together with effective mitigating strategies. Preliminary experimental evaluations are carried out on a physical implementation designed in UMC's 180 nm technology.

A second encoding platform is built on top of a Phase-Change Memory (PCM) Analog Array through numerical models of its steady-state response, programming variability, and conductance drift. Two techniques to drive the array, namely, voltage-encoded inputs and constant-amplitude time-encoded pulses, result in significantly different concerns from an applicative and design standpoint. Consequently, the resiliency of well-known CS reconstructors is tested against the observed device variability, and a new, iterative decoding strategy is proposed.

The numerical models are also used as the core computational kernel within PCM-based Neural Network Layers to verify the inherent redundancy and adaptability of Deep Neural Networks (DNNs) as an effective workaround to device nonidealities. Popular classification tasks (MNIST, Fashion-MNIST, CIFAR-10), and a spectral-content estimation task, are all addressed through PCM-based DNNs.

The work highlights the importance of the simultaneous optimization of system, circuit, and device-level parameters for the effectiveness of low-energy hardware implementation, especially for the success of emerging technologies such as Phase-Change Memories.

Contents

List of Figures **x**

List of Tables **xvii**

1 Introduction **1**

2 Compressed Sensing Fundamentals **3**

2.1 Why bother? 4

2.2 Sparse Signals Recovery 5

2.2.1 Uniqueness conditions 6

2.2.2 Numerically tractable recovery 8

2.2.3 Sensing matrix design 9

2.2.4 Noisy measurements 10

2.3 Enter Compressed Sensing 10

2.3.1 Rakeness-based Compressed Sensing 11

2.4 Relevant Reconstruction Algorithms 12

2.5 Reducing Complexity of the CS Acquisition Stage 13

2.5.1 Reducing encoding complexity: short windowing 14

2.5.2 Improving performance by adaptation: Rakeness CS 16

2.5.3 Improving performance through DNNs: TCSSO 17

2.5.4 Case study: synthetic ECG signals 18

2.6 Decoding by TCSSO with Short Windowing 21
2.6.1 The TCSSO approach 21
2.6.2 TCSSO with antipodal block-diagonal matrices 24
2.6.3 Numerical results with the modified TCSSO 26

3 CMOS Analog-to-Information Converter 28
3.1 Where to Apply Compression 29
3.2 Existing analog CS encoders 31
3.3 Proposed architecture 33
3.3.1 Successive Approximation Converters 34
3.4 Implementation details 42
3.4.1 Block diagram and I/Os 42
3.4.2 Capacitive Array and Sub-Arrays 42
3.4.3 Switch Architecture 45
3.4.4 Dynamic Comparator 49
3.4.5 Coefficient memory and SAR logic 58
3.5 Layout 61
3.5.1 Post-layout results 66
3.6 Testing the chip 66

4 Results on Nonidealities Estimation and Management 71
4.1 Parasitics-induced nonlinearities in the C-2C Array 72
4.1.1 Algorithmic derivation of the complete model 74
4.2 Preamplifier mismatch analysis 82
4.3 Mismatch variance in interdigitated geometries 87
4.3.1 Analytical Model of Mismatch Variance 88
4.3.2 Geometry and Model Approximation 90

4.3.3 Variance in Interdigitated Structures 92
4.3.4 Results 96
4.4 Stability analysis of a leakage compensator 97
4.4.1 Analysis of the nominal circuit 99
4.4.2 Effects of device mismatches 101
4.4.3 Stability with a dynamic transconductance 102
4.4.4 Numerical evaluations 105

5 Phase-Change Analog In-Memory Computing 108
5.1 Phase-Change Technology 109
5.2 Available HW Platform 113
5.2.1 PCM Programming 115
5.3 Conductance Measurements and Processing 117
5.4 Device Modeling 121
5.4.1 Application-based modeling 122
5.4.2 Numerical models for voltage-domain inputs 124
5.4.3 Numerical models for time-domain inputs 130
5.5 PCM-based Compressed Sensing Encoding 131
5.5.1 Voltage-domain inputs 132
5.5.2 Time-domain inputs 139
5.6 PCM-based Neural Network 145
5.6.1 Voltage-domain inputs 145
5.6.2 Time-domain inputs 150

6 Conclusion 155

References 157

List of Figures

2.1 Example of a block-diagonal sensing matrix A, with $n = 24$, $n_b = 6$, $m = 12$ and $m_b = 3$. White blocks correspond to zeroes. 15

2.2 Example of a synthetic ECG with its sparsified version ($\kappa = 24$). . . 19

2.3 Performance of a leaky CS-based signal processing chain for synthetic ECG signals. 20

2.4 Architecture of the TCSSO framework, with the CS encoder optimized along with the support oracle DNN during the training phase. 22

2.5 Equivalent views of the CS encoder in the case of block-diagonal sensing. 23

2.6 Reconstruction of the sparsified ECG waveform depicted in Figure 2.2, using an ideal setup (no leakage) with $n_b = 8$ and CR = 2.7. The different decoders result in a 16 dB RSNR for SPGL1 and 31 dB for the TCSSO. 25

2.7 Performance of a CS-based signal processing chain expressed in terms of ARSNR as a function of CR, for synthetic ECG signals at approx 60 beats/s, with $f_s = 256\,\mathrm{Hz}$ and $n = 128$. Ideal sensing. Sensing corrupted by leakage, with the presence of the compensator. 27

3.1 Compression applied at different locations of the signal processing chain. 29

3.2 Proposed entirely-passive CS encoder based on a charge-redistribution SAR ADC. The extra blocks enabling CS functionality are highlighted. 34

3.3 Circuit diagram and waveforms during conversion of a typical SAR A/D converter 34

3.4 Charge-redistribution SAR converter, combining sample-and-hold and D/A functionalities within the same block. 36

3.5 Schematic and working principle of a traditional charge-redistribution SAR ADC with a 4-bit binary-weighted array, and a 2-bit C-2C array. 37

3.6 Capacitive array during: sampling and conversion. 38

3.7 Schematic and working principle of the CS-based acquisition system, based on a charge-redistribution SAR ADC, proposed in this thesis. 41

3.8 I/Os of the converter and inner block diagram. 43

3.9 Split capacitive array with unequal sections and mixed-type array: scaled plus C-2C sections. 44

3.10 Charge injection and its effects on the sampled value 46

3.11 Charge injection compensation techniques: dummy transistor and transmission gate. T-switch configuration. 47

3.12 Selected switch topology. The four external inputs `IN`, `REFp`, `CM` and `REFn` can be routed to the shared `COM` node. Dummy elements compensate charge injection errors and a T-switch topology prevents coupling from the inputs and minimizes subthreshold leakage. . . . 49

3.13 Elements of the comparator: dynamic residual preamplifier and parallel-coupled regenerative latch. 50

3.14 Preamplifier circuit and equivalent circuit models in the active phase. 51

3.15 Preamplifier equivalent common- and differential-mode half-circuits during the linear growth phase. 53

3.16 Preamplifier in the regeneration phase. 55

3.17 Preamplifier simulated waveforms. Two transients with opposite input differential voltage. 57

3.18 Memory layout and its use during write and read operations 61

3.19 SAR logic block diagram, with a detailed view on the switch-state decoder. 62

3.20 Layout of the entire chip. 63

3.21 Layout of a single differential CS encoder, with its capacitor array and the dynamic comparator 64

3.22 Layout of the dynamic preamplifier and latch 65

3.23 Integral and Differential NonLinearity of the single-ended capacitive DAC used within the CS encoder. The test is performed by running through all the input digital codes. The reference linear characteristic is either through the endpoints of the actual one or a bestfit over the entire dataset. 67

3.24 Waveforms of the time-domain operations of the preamplifier within the dynamic comparator for different values of the differential-mode input voltage. The voltages shown maintain the nomenclature defined in Fig. 3.15. 68

3.25 Time-domain operations of the ADC, used as a Nyquist-rate converter 68

3.26 System-level schematic diagram of the test platform. The digital control lines all come from an FPGA evaluation board. 69

3.27 Layout of the support PCB. 70

4.1 Schematic of a 3-bit C-2C sub-array. 73

4.2 Capacitances in a generic section of a C-2C array 76

4.3 Integral and differential nonlinearities computed on a 5-input C-2C structure, without any scaled array. 81

4.4 Maximum INL and DNL for different widths of the C-2C section, amounts of scaled capacitance and amount of parasitics loading. Simulation vs model predictions. 82

4.5 Maximum number of bits as a function of the parasitics fraction and the amount of tolerable error γ. 83

4.6 Geometry of devices A and B. Interdigitated layout on a grid of N_r rows and N_c columns. Invariance of centroid distance on the number of columns N_c for a given number of rows N_r (negligible spacing between segments). Mirroring half of the array to obtain a common centroid structure for N_c even and N_r odd. 88

4.7 Mismatch variance computed through (4.17) and normalized to the value obtained for $N_r = N_c = 1$. ($w = 0.02$, $l = 0.001$) 93

4.8 Mismatch variance normalized with respect to the value for $N_r = N_c = 1$. Lines are obtained through a Monte Carlo simulation using the model in [1], markers using the asymptotic expressions derived in this work. Solid lines refer to the simple interdigitated layout, dotted lines to the mirrored geometry. ($w = 0.02$, $l = 0.001$, $s_x = s_y = 0$, $\Lambda_x = \Lambda_y = 10^3$, $\alpha = 1$) 95

4.9 Mismatch variance for two-finger alyouts, with respect to the single-finger reference case (N_r, N_c) = (1, 1). 96

4.10 Basic implementation of a sample-and-hold circuit. Implementation by means of a T-shaped switch. Equivalent circuit in the hold mode. Schematic of the leakage compensator 98

4.11 Complete block diagram of the compensator. 100

4.12 Root loci for the closed loop poles of the leakage compensator as a function of the g_{m0} parameter, in S. Nominal circuit. Circuit with a $\Delta g_{m0}/g_{m0} = -0.1$ mismatch term. Nominal circuit with a constant pole in $g_m(s)$. Nominal circuit with a $g_m(s)$ pole proportional to g_{m0}, $p_g = g_{m0}/C_g$, with $C_g \in \{C_h/50, C_h/100, C_h/1000\}$. 106

4.13 Transient behaviour of the voltage in the hold cell for several system configurations. The subplot on the right provides the zoomed-in view of the rectangular region highlighted on the left. 107

5.1 Mushroom-type PCM cell, typical PCM DC response and PCM crossbar structure. 110

5.2 Architecture of the Mk-I and the Mk-II PCM chips. 113

5.3 MAC accuracy as a function of $\frac{g_{\text{REF}}}{g_{\text{MAX}}}$. Continuous lines refer to simulated results after 2- and 18-hours room temperature and 90°C bake drift. Dashed lines represent the MAC accuracy in the same conditions when no compensation is adopted. Crosses report the results of experimental evaluation of MAC accuracy for different g_{REF} levels. 115

5.4 Comparison of the MAC output for compensated and uncompensated cells and distribution of the MAC errors. 116

5.5 Outline of the cells programming algorithm. 117

5.6 Cumulative distribution functions of cells programmed to 32 equispaced target conductances. 118

5.7 Alternative logical interpretation of PCM DC characteristics in different programming states 119

5.8 Average, normalized i/v characteristics of PCMs in three different states. Large-signal, normalized conductance $g = i/v$. 120

5.9 Accuracy of the PCM AIMC unit vs ideal MACs obtained using integrated resistances. MAC error distributions. 121

5.10 Effects of different values for the reference conductance g_{REF} on the output swing of the MACs. *i*) optimal swing (Equation (6)); *ii*) saturation due to low g_{REF} level; *iii*) swing reduction induced by PCM cells drift with constant g_{REF}. 122

5.11 Practical use cases and their respective PCM models. 125

5.12 Average, normalized *I–V* characteristics of PCM devices in four different states. Spline-based interpolation of the average, normalized PCM behaviours. Low-order and high-order polynomial fit. 127

5.13 Chebyshev grids to prevent the insurgence of Runge's phenomenon in high-order polynomial fit. 127

5.14 Comparison of the numerical representations of the same datapoints, using a spline interpolation and a polynomial one both on an equispaced and a Chebychev grid. 128

5.15 Level curves to appreciate the advantage of a Chebychev grid. (top) Level curve of the PCM current. Dashed lines refer to the spline model. (bottom) Level curve of the difference between the polynomial and spline model. Notice the stronger oscillations observed at the boundaries of the plots. 129

5.16 Datapoints and numerical models of PCM programming spread, uncompensated drift and compensated conductance drift. 131

5.17 Evolution over time of two batches of PCM devices, programmed to a target normalized conductance of 0.35 and 0.85. Both compensated and uncompensated cells are shown. 132

5.18 PCM encoder followed by a one-shot decoder, using an estimate of the input-dependent PCM conductance. 133

5.19 Empirical CDFs of RSNR values for the three conductance models and one-shot reconstruction 136

5.20 Architecture of a PCM-based system for CS applications with the proposed decoder. The encoder implements the ideal, binary sensing matrix A by PCM conductances. The decoder reconstructs the original signal by iteratively computing the estimate of the real sensing matrix $\hat{G}$. 136

5.21 Empirical CDFs of RSNR values for the three conductance models, at different dynamic ranges for the input signals. Mapping of RSNR values against ASNR. 138

5.22 Performance of different reconstruction algorithms under the sensing matrix uncertainty introduced by different levels of target conductance. 143

5.23 Reconstruction performance for different drift setups: after 2 hours, after 18 hours and after a 24-hours bake at 90 °C. 144

5.24 Traditional dense and PCM-based dense neural layer structures. . . 146

5.25 Examples of Fashion-MNIST instances. Spectrums for a selection of r with corresponding representative instances of length 128. Normalized histograms of input values for the two datasets being employed. 147

5.26 Results for Fashion-MNIST classification, and spectrum estimation regression. 149

5.27 Accuracy of the trained networks versus the programming spread scaling coefficient, both for the conventional (dotted line) and device-aware (DA) trainings 151

5.28 Accuracy achieved when drift is applied to the DNN weights at inference time, both with and without compensation, for the Lenet-5 and VGG-8 DNNs. 152

5.29 Classification accuracy when quantizing only the global network I/O vs the I/O at every layer. 154

List of Tables

2.1 Summary of the tradeoffs involving n or, equivalently, n_b, indicating the section where the topic is discussed. Whenever possible, an estimate of the dependence is given. 16

3.1 Summary of solutions implementing a CS encoder according to (2.7). The additional hardware with respect to a Nyquist-rate approach is highlighted. 32

4.1 Attenuation from an input node to the adjacent isolated node. Results shown for the first 4 inputs. 74

4.2 Attenuation from an isolated node to the adjacent one, moving towards N_0. Results shown for the first 3 nodes. 75

4.3 First order approximations of the attenuations from input i to node N_0 in the case of (a) $n = 2$, (b) $n = 3$ and (c) $n = 4$ 77

4.4 Approximate error on the analog value induced by the presence of parasitic loading on the inner isolated nodes. 79

4.5 Input-referred offset voltage due to parameter variations. Curly braces are used to compact the table when the same multiplicative factor affects a given term. 86

4.6 Routh-Hurwitz table for the characteristic polynomial when the dominant pole in $g_m(s)$ is proportional to g_{m0}. 104

5.1 Applications being evaluated in this work and corresponding requirements and properties of the models extracted from the conductance measurements. 126

5.2 Sequence of iterative estimations $\widehat{x_{|n}}$ generated by the iterative decoding strategy in Fig. 5.20. 137

Chapter 1

Introduction

Quo usque tandem...

Oratio I in Catilinam
CICERO

This thesis is the culmination of my Ph.D. experience. It builds on a series of publications where the ideas where originally outlined and analyzed. Two major themes can be identified, linked by the goal of constructing low-energy data processing schemes.

The first theme focuses on the design of an entirely-passive CMOS Compressed Sensing (CS) encoder, based on a Charge-Redistribution Successive Approximation Register A/D Converter. The architecture was initially described in [2] and its description was later expanded in [3, 4]. During the design phase, the need to quantify and mitigate circuit nonidealities, drove us into the deeper investigation of a few relevant topics, mainly the effect of layout strategies on the mismatch variance of nominally identical devices [5] and the stability analysis of a feedback leakage compensation architecture [6].

The second theme is based on Phase-Change technology, allowing the nonvolatile storage of information, which is encoded in the conductance of the device. The extraction of numerical models from actual measurement allowed us to evaluate the performance of the technology both in the context of Compressed Sensing [7, 8] and Deep Neural Networks [9, 10].

The thesis starts, in Chapter 2, with a theoretical explanation of the mathematics behind Compressed Sensing, with results on the use of hardware-friendly sensing matrices and showing how to leverage Deep Neural Networks for an additional performance boost. In Chapter 3 the CMOS architecture we have proposed is described in detail, with specific details on its elementary components (the capacitive array, switches, comparator and digital logic), ending with its layout and the design of an ad-hoc PCB for its testing. To preserve the flow of the discussion, more detailed analysis of specific topics has been collected in Chapter 4, covering the effect of parasitic capacitance on the nonlinearity of a C-2C capacitive DAC, the input-referred offset stemming from asymmetries of the dynamic preamplifier within the comparator, a general result on the relationship between matching and the layout strategy in integrated circuit, endind with the stability analysis of a feedback-based leakage compensator.

The final Chapter 5 introduces Phase-Change technology, presenting the numerical models we have constructed on two revisions of the hardware computing platform we could rely upon, the first having voltage inputs applied to the individual conductances, the second enabling time-domain encoding of the inputs, thus preventing issues with the nonlinear dependence of the conductance on the applied voltage, allowing us to focus on the issues of programming variability and conductance drift. Results on Compressed Sensing encoding are shown, with an iterative decoding strategy presented with the voltage-based platform and the evaluation of the resilience of existing decoders with the time-domain one. Additionally, the chapter ends with an evaluation of the performance obtainable when PCMs are used as synapses in analog neural layers, both in regression and classification tasks.

Conclusions are then drawn.

Chapter 2

Compressed Sensing Fundamentals

I would rather sit on a pumpkin and
have it all to myself, than be
crowded on a velvet cushion.

Walden
HENRY D. THOREAU

Compressed Sensing (CS) is a signal processing technique able to reconstruct particular signal families using far fewer measurements than what the Shannon sampling theorem suggests. This chapter will delve into the mathematical theory of CS, explaining the reasons of its effectiveness and discussing some theoretical bounds defining the limits of applicability.

The general principle guiding the construction of a CS-based signal acquisition chain is that of *adaptation*, i.e., tuning the process to the properties of the inputs. Along those lines, rakeness-based CS is then introduced as a way to further improve performance by properly selecting sensing matrices with a specific correlation profile. The use of hardware-friendly sensing matrices will be analyzed in order to simplify the design of a circuital implementation of a so called Analog-to-Information converter.

Finally, a strategy to leverage the power of Deep Neural Networks to additionally boost performance is presented, before delving, in later chapters, into a more hardware-centric discourse.

2.1 Why bother?

Whenever the need to acquire a continuous-time signal arises, Shannon sampling theorem enters the discussion [11]. The theorem states that functions whose Fourier spectrum is null above a certain frequency f_{max}, also known as the bandwidth or the Nyquist frequency, can be described exactly from samples collected at least at twice that frequency. As the operating speed of information-processing systems is increasingly higher, complying with this bound becomes unfeasible because of technological limitations. At the same time, the frequency domain is not necessarily the one that shows the most striking features of a signal, since the true information rate might not be readily observable (a fixed-frequency sine wave, does not provide new information over time, even if it is varying continuously), leading to an unnecessarily high Nyquist bound. Weakening such a constraint would lead to dramatic reduction of the resources employed.

The main advantage of the Nyquist-rate approach is that it is valid under general conditions, i.e. for any signal having limited bandwidth. Therefore, once a system has been designed to operate at a given sampling frequency, more often than not, it can manage any signal whose Nyquist frequency is low enough. Furthermore, if the sampling frequency f_s is sufficiently larger than the Nyquist one (as a rule of thumb $f_s > 10 \times f_{max}$), a reasonable approximation of the original information can be recovered by a simple linear interpolation.

In practice, any signal processing chain will be designed for a specific application. The operating conditions, like the kind of signals involved, will therefore be known in advance. Since some parameters already have to be matched to the specific properties of the signals (e.g. the sampling frequency), one might ask if it couldn't be possible to further tune the system properties to take advantage of more prior information. As an example, consider a pure sine wave at frequency f, Shannon theorem requires a sampling frequency greater than $2f$ to be able to recover the original information. However, having prior knowledge of the fact that the signal is indeed a sinusoid, only three measurements in total are sufficient to define its amplitude, frequency and initial phase. Therefore including in the design of a signal processing chain information on the properties of the signal family, the acquisition effort can be significantly reduced. What Compressed Sensing achieves is to observe the signal in a different domain and, using a property of the signal family called sparsity, work with far less scalars than what the Nyquist constraint would require.

Compressed Sensing has been developed as an extension of the theories on the recovery of sparse signals, with the distinction that CS processes the samples in a domain where the signal is not sparse. Therefore we will first analyze the properties and bounds involved in sparse signal recovery, later moving to the features of CS.

2.2 Sparse Signals Recovery

While we typically work on continuous-time signals, Compressed Sensing theory is most simply understood in a discrete, finite-dimensional setting. In that context, signals can be thought of as sequences of n Nyquist-rate samples, i.e. vectors in $\mathbb{R}^n$. A basis of such a space is the smallest set of vectors able to represent any other element by a proper linear combination. Although the number of basis for a given space is unlimited, each signal has a unique representation in any of them. If the matrix $\Phi \in \mathbb{R}^{n \times n}$ contains, on its columns, the basis vectors, a generic element x of the space can be expressed as

$$x = \Phi\xi. \tag{2.1}$$

The vector $\xi \in \mathbb{R}^n$ is an equivalent way to look at the original vector. In geometrical terms, the product in (2.1) is a change of coordinates. It preserves the informative content, with the possibility to observe other features of the original vector.

The fundamental property required by the theory underlying Compressed Sensing is *sparsity* of the signal. A vector in $\mathbb{R}^n$ is κ-sparse if the number of non-null coefficients is $\kappa \ll n$. In general, the linear combination of two κ-sparse vectors is at most 2κ-sparse, since the non-null coefficients may occupy different positions. Knowing the sparsity level of a vector, the quest is to reduce the number m of observations required to describe it uniquely such that $\kappa < m \ll n$, providing sufficient information to recover the original data.

Consider $\xi \in \mathbb{R}^n$ as a κ-sparse sequence of length n. A *measurement* corresponds to the linear combination of the n scalars in ξ, weighted by some a_i coefficients, and resulting in a single number

$$y_j = \sum_{i=1}^{n} a_i \xi_i \qquad j = 1, \dots, m.$$

In Nyquist-rate sampling, a measurement would correspond to a single sample, while in this context it involves the processing of an entire window of length n. The evaluation of m such measurements can be compactly written as

$$y = A\xi, \tag{2.2}$$

where the a_i coefficients are placed on the m rows of $A \in \mathbb{R}^{m\times n}$. The mapping described by A is from the n-dimensional space to a lower dimensional one.

Recovering ξ from y is, in general, not possible since the dimensionality reduction implies that multiple ξ map to a single y. Equivalently, this process corresponds to solving an underdetermined system of equations which, according to Rouché-Capelli's theorem [12], has ∞^{n-p} solutions, with p the rank of the matrix (the number of linearly independent columns) and $n-p$ being the number of free variables. Since $p \leq m$, the number of free variables is greater than or equal to $n-m$. In reality, by having prior knowledge on the sparsity of ξ, uniqueness of the solution can in fact be guaranteed.

2.2.1 Uniqueness conditions

Intuitively, knowing that the ξ's of interest are κ-sparse, any of them should be distinguishable from the others after the projection into the lower dimensional space. That is, for any ξ_1, ξ_2, the difference $\Delta\xi = \xi_1 - \xi_2$ observed in the target space has to be

$$\Delta y = A\Delta\xi \neq 0.$$

The term $\Delta\xi$ is, in general, 2κ-sparse, and the non-null positions are unknown. Considering Δy as the linear combination of columns of A, weighted by the coefficients in $\Delta\xi$, we have to guarantee that any 2κ columns of A are linearly independent. This way, there is no possibility of obtaining the null vector from the weighted sum of the matrix columns and, as a result, any ξ is still distinguishable in the smaller, m-dimensional space. Identifying uniquely the original ξ, starting from y, is then possible, with the advantage of having to perform only $m \ll n$ observations.

This qualitative description can be formalized by introducing the concept of *spark* of a matrix [13]. It is the maximum cardinality c such that any subset of c columns of A contains only linearly independent elements Therefore, recovering uniquely a κ-sparse vector $\xi \in \mathbb{R}^n$ from m linear observations ($m < n$) is possible if

$$\kappa = \|\xi\|_0 < \frac{1}{2}\text{spark}(A), \tag{2.3}$$

where $\|\cdot\|_0$ is the ℓ_0 (pseudo)norm, equal to the number of non null coefficients in the vector.

Under this uniqueness condition, the solution of (2.2) can be found through a minimization process that looks for the sparsest ξ such that $y = A\xi$, i.e.:

$$\begin{aligned} &\operatorname*{argmin}_{\xi \in \mathbb{R}^n} \|\xi\|_0 \\ &s.t. \quad y = A\xi. \end{aligned} \tag{2.4}$$

However, computing the spark requires a combinatorial search (NP-hard) over all possible subsets to evaluate the independence of their elements. A more easily computable index resulting in a uniqueness condition for the solution of (2.2) is based on the concept of *mutual coherence* of the columns of A, defined as:

$$\mu(A) \overset{\text{def}}{=} \max_{i<j} \frac{|A_i^T A_j|}{\|A_i\|\|A_j\|},$$

where A_i represents the i-th column of matrix A. Mutual coherence is constrained in the range $0 \leq \mu(A) \leq 1$. It is equivalent to the maximum cosine of the angle between any two columns, corresponding to the smallest acute angle between them. A high coherence is therefore equivalent to aligned vectors. From a different point of view, considering the columns as random vectors, mutual coherence represents the maximum correlation coefficient between the columns.

Since the mapping represented by A should store as much information as possible with the lowest redundancy, we expect a low coherence to result in better performance. Equivalently, the columns should be as orthogonal as possible or, looking at them as random vectors, as uncorrelated as possible. Indeed the uniqueness condition in (2.3) can be stated in terms of the mutual coherence index, becoming

$$\|\xi\|_0 < \frac{1}{2}\left(1 + \frac{1}{\mu(A)}\right).$$

This theoretical bound, representing the range of sparsity for which a unique solution can be found, is effectively increased by a lower coherence of A. However, since it can be shown [13] that

$$\text{spark}(A) \geq 1 + \frac{1}{\mu(A)},$$

the new bound is lower than the one based on the spark, thus it is applicable on a smaller subset of vectors, which have to be even sparsest.

2.2.2 Numerically tractable recovery

Up to now we have used the sparsity level of the vectors, quantified by their ℓ_0 norm, to find the correct solution to the minimization problem. Such a computation requires, however, a combinatorial search across all possible candidates and is not suitable for an efficient numerical implementation. An important consequence implied by the mutual coherence number is the equivalence of the result found by using the ℓ_1 norm instead [13]. The ℓ_1 norm corresponds to the summation of the absolute value of all vector coefficients and it is the smallest-rank norm that is convex, therefore suitable for use in a numerical implementation. The reconstruction in (2.4) then becomes:

$$\begin{aligned} &\underset{\xi \in \mathbb{R}^n}{\text{argmin}} \, \|\xi\|_1 \\ &s.t. \quad y = A\xi, \end{aligned} \tag{2.5}$$

where the only change is the rank of the norm. In turn, this forces stricter conditions on the sensing matrix A, which has to be generated with greater care.

Although this formulation leads to a (possibly) efficient numerical implementation, empirical evidence shows that the sparsity range for which the unique solution can be found is actually larger than what the mutual coherence implies. New indexes have been proposed, the most popular one being the *isometry constant*, defined as the smallest number such that, for all κ-sparse vectors ξ, the following property holds:

$$(1 - \Delta_\kappa)\|\xi\|_2^2 \leq \|A\xi\|_2^2 \leq (1 + \Delta_\kappa)\|\xi\|_2^2.$$

This condition is also referred to as the *Restricted Isometry Property* (RIP). It implies that the projection into the m-dimensional space approximately preserves the norm of the original vector.

Similarly to the reasoning on the meaning of the spark, the goal is to distinguish any sparse ξ, therefore the distance $\Delta\xi$ should be preserved and the isometry constant of interest is actually $\Delta_{2\kappa}$. It can be shown that for $\Delta_{2\kappa} < 1$, the ℓ_0 minimization can find the unique κ-sparse solution. The bound becomes $\Delta_{2\kappa} < \sqrt{2} - 1$ in the case of the ℓ_1 minimization, highlighting the fact that using a more convenient norm in the optimization process requires a more careful design of the measurement operator A.

The three indexes introduced thus far can be linked. Indeed if $\Delta_{2\kappa} < 1$, any subset of 2κ columns of A has linearly independent elements, therefore $\text{spark}(A) > 2\kappa$. Moreover, the RIP of order κ is satisfied, with $\Delta_\kappa \leq (\kappa - 1)\mu(A)$.

2.2.3 Sensing matrix design

The construction of the sensing operator A in order to achieve a small coherence or a small isometry constant is of great concern. This would ensure a wide range of sparsity in which a unique solution can be found. A deterministic process leading to matrices with suitable properties would involve the solution of

$$\operatorname*{argmin}_{A \in \mathbb{R}^{m \times n}} \mu(A) \qquad \text{or} \qquad \operatorname*{argmin}_{A \in \mathbb{R}^{m \times n}} \Delta_{2\kappa}(A).$$

Both these problems do not lend themselves to an easy evaluation, being of combinatorial nature. Indeed it can be shown that these properties hold with probability close to 1 for random matrices of size $m \times n$ [14]. In such matrices, each entry is a realization of some random variable, with a chosen probability density function (e.g. Gaussian, Uniform, Bernoulli). The simplest case is for the entries to be independent and identically distributed (i.i.d.). Moreover, the number of measurements m cannot be too low. Indeed

$$m = \mathcal{O}\left(\kappa \ln \frac{n}{\kappa}\right),$$

where the proportionality depends on Δ_κ. Still, since the lower bound on m could be higher than what is actually required for correct reconstruction, the empirical study of a reasonable range for m might be a suitable way to approach the problem at the beginning.

An alternative way to look at the use of random variables is that this ensures a high degree of spreading of the matrix columns in the signal space, with the further advantage of being robust against the loss of part of the measurements.

2.2.4 Noisy measurements

All the previous theoretical guarantees have been obtained in the context of noiseless measurements and exact κ-sparsity of the original vector. If any of these two conditions is not met, we need to ensure that the tools developed so far can still provide the expected results. Indeed, if the measurements vector is affected by a noise term η

$$y = A\xi + \eta,$$

the solution to the minimization problem is no more exact and (2.5) becomes

$$\begin{aligned} &\min_{\xi \in \mathbb{R}^n} \|\xi\|_1 \\ \text{s.t.} \quad &\|y - A\xi\|_2^2 \leq \varepsilon \end{aligned}$$

The solution $\hat{\xi}$ has to result in a projection $\hat{y}$ close to the measured one y, with its uncertainty depending on the amount of noise $\varepsilon(\eta)$. The same is true if the original signal is not sparse but *compressible*, in which case most of its coefficients are small but non-zero. Reconstruction based on sparsity won't be able to result in the same exact measurement acquired from the signal, but will be close enough.

Moreover, the uniqueness of the solution and the equivalence of the ℓ_0 and ℓ_1 minimization procedures no longer apply, but it can be shown that the RIP condition guarantees robustness of the formulation against noise [13].

2.3 Enter Compressed Sensing

In the previous discussion, measurements were computed directly from sparse vectors. In reality, sparsity is not necessarily observed in time domain, therefore if one wants to apply the previous results, the observation have to be performed on $x = \Phi\xi$, where $x \in \mathbb{R}^n$ is a signal window containing n Nyquist-rate samples, Φ is the $n \times n$ sparsity basis whose columns are the coordinate vectors of the space and ξ is κ-sparse. The measurements vector y is obtained by applying a linear projection $S \in \mathbb{R}^{m \times n}$ to the vector x (containing the time-domain samples), such that

$$y = Sx = S\Phi\xi \,. \tag{2.6}$$

It is the composition $S\Phi$ that has to satisfy the RIP of order 2κ. If the number of observations m is sufficient, then a unique solution ξ can be found by the solving

$$\begin{aligned} &\underset{\xi\in\mathbb{R}^n}{\operatorname{argmin}} \|\xi\|_1 \\ \text{s.t.} \quad &\|y - S\Phi\xi\|_2^2 \leq \varepsilon \end{aligned}$$

Then x can be recovered knowing Φ. The advantage introduced by Compressed Sensing is the early reduction of the number of measurements to be acquired, thus avoiding the need to collect every single sample. The *Compression Ratio* quantifies this gain, being expressed as

$$\mathrm{CR} \overset{\text{def}}{=} \frac{m}{n}.$$

Since the conditions on A are actually posed on the product $S\Phi$, but the matrix to be designed is S, it is necessary to look into how the properties of S are carried over to the compound operator. If Φ is orthonormal, then building S according to a Gaussian distribution and satisfying the RIP will result in a product $S\Phi$ with the same properties [13]. The same is true if the columns of S have low coherence to the columns of Φ. These consideration can also be extended to sub-gaussian distributions [14].

2.3.1 Rakeness-based Compressed Sensing

Continuing along the path of specialization of the signal processing chain, a significant improvement in reconstruction performance can be achieved by observing another feature of the signal family. Intuitively, once the sparsity basis has been identified, not necessarily the coefficients associated to each basis vector have the same average length, i.e. energy. Equivalently, the signal instances in the n-dimensional space are not uniformly spread, but concentrate along some of the directions. By focusing the measurements on the more energetic directions, the average energy collected (*raked*) by each measurement can be maximized, with a significant gain in the quality of the reconstructed signal.

The quantity measuring the distribution of energy across the basis vectors is named *localization* [15] and it represents, together with the sparsity level, an additional prior to the reconstruction process. Its effect is observed on the correlation profile of the sensing matrix rows:

$$C_S = \frac{1}{2}\left(\frac{C_x}{\mathrm{tr}(C_x)} - \frac{I_n}{n}\right),$$

where I_n is the $n \times n$ identity matrix, $\mathrm{tr}(\cdot)$ the trace operator and $C_x = \mathbb{E}[xx^{\mathrm{T}}]$ the expected correlation profile of the signal. C_x can be evaluated either having a model of the signal (and running some Monte Carlo simulations) or having acquired a large dataset of signal waveforms.

This simple modification to the generation of the sensing matrix leads to dramatic improvements in the reconstruction quality, therefore becoming essential to minimize the necessary resources.

2.4 Relevant Reconstruction Algorithms

A wide variety of techniques has been proposed for the solution of the optimization problem in (2.5). A brief introduction on the algorithms employed in this work is given here.

SPGL1 [16] solves the Basis Pursuit Denoise problem by recasting it as a root-finding for an equivalent nonlinear equation. In doing so, it administers the trade trade-off between the least-squares fit characterizing the constraints in (2.5) and the $|\xi|_1$ as an alternative objective function representing a good proxy of the sparsity-level of ξ.

The Orthogonal Matching Pursuit (OMP) is a greedy, iterative algorithm for sparse approximation. At each iteration, it looks for the component of the product AD which best explains the measurements. Given a sparsity level k for the family of input signals being acquired, it completes in at most k iterations, making it extremely fast. The variant used in this work is the Generalized OMP (GOMP), which selects more than one vector in each iteration thus reducing the computational complexity and making the decoder more robust to measurement non-idealities [17].

Generalized Approximate Message Passing (GAMP) is based on the concept of Belief Propagation, whereby the posterior distribution for each ξ_i is updated based on the observations. It allows flexibility in the selection of prior models both on x and the measurement process. In the case of uniform standard deviation of the sensing matrix weights A it performs at a level comparable to its variants specifically designed to manage sensing matrix uncertainties [18], hence the reason to rely on its non-specialized implementation.

Although, SPGL1 is designed to find the optimal solution for (2.5), we expect lower performances if compared with ones that could be obtained by means of both GOMP and GAMP since they posses the capabilities to manage sources of non-idealities in the computation of y.

2.5 Reducing Complexity of the CS Acquisition Stage

The CS paradigm was originally introduced to reduce the energy requirements of the encoder stage in a signal processing chain, thus additional reductions still align with the general goal of CS.

A possible strategy is to constrain the admissible values of the sensing matrix A. This method requires less resources to compute (2.6), and is adopted by almost all hardware implementations of CS-based acquisition systems proposed in the literature [19–23]. As an example, antipodal values, i.e. $A \in \{-1,+1\}^{m\times n}$, reduce multiplications to simple sign inversions, that come almost for free in a differential implementation and shows no performance loss [24].

Binary matrices, i.e. $A \in \{0,1\}^{m\times n}$, or ternary, i.e., $A \in \{-1,0,+1\}^{m\times n}$, allow further energy saving since the zero entries of A do not contribute to the whole energy cost. Several classes of matrices A belong to the latter class have been investigated in the literature [25–28]. Authors of [28] explore the trade-off between zeroing at random the entries of A and the CS performance in terms of signal recovery. The same topic is also discussed in [29], where the position of the non zero entries is not anymore randomly drawn, but tuned according to the statistical characterization of the class of acquired signals. In that case, authors did not observe any performance degradation even when 87.5% of the entries in the sensing matrix A are zeroed.

In this section we will go through a few other techniques developed in the literature to ease the hardware implementation of (2.6) and that are of relevance in the proposed architecture. In particular, we focus on two complementary approaches. The first one is focused on effectively reducing the complexity of the circuitry required to implement the linear projection. In the second one we instead aim at improving the performance of the CS system. This may be interpreted either as increasing reconstruction quality at a given CR, or as increasing CR at a given reconstruction quality. The latter corresponding to a reduction the number of measurements m and therefore of the complexity of the encoder generating y.

2.5.1 Reducing encoding complexity: short windowing

In a CS-based acquisition, the signal is processed in batches of n consecutive samples. If the total length of the signal is $N > n$, it has to be partitioned into N/n contiguous and non-overlapping time windows, each containing n samples[1]. CS is then applied separately to each window.

To reduce the computational complexity of the encoder, the system should be designed to operate with the smallest possible n. To understand why, we have to consider that the application of (2.6) to a single time window requires $\mathscr{O}(n \cdot m)$ multiply-and-accumulate operations. Extending the computation over all N/n time windows, the total number of operations increases to $\mathscr{O}(n \cdot m \cdot N/n) = \mathscr{O}(n \cdot N/\mathrm{CR})$. If CR is chosen to guarantee a target reconstruction quality and assuming, reasonably, that the input signal length N is a constraint, the computational complexity increases linearly with n.

However, some other effects have to be considered as well. Let us reformulate (2.6) component-wise as

$$y_j = \sum_{k=1}^{n} A_{j,k} x_k + \nu_j, \quad j = 1, \ldots, m \tag{2.7}$$

where $A_{j,k}$ is the element of A at the intersection of the j-th row and k-th column, and ν_j the j-th component of ν.

First, the hardware resources needed to compute (2.7) are increasing[2] with m, and a large n implies a large $m = n \cdot \mathrm{CR}$. At the same time, the noise on y_j increases, since

[1]We are implicitly assuming that N is an integer multiple of n in order to keep the description simple. This assumption holds for the rest of the thesis.

[2]This is certainly true when the circuit to compute y_j is replicated m times. A possible alternative to the simple hardware replication is to use the same circuit in an interleaved way to compute all m measurements [19]. In this case, the speed (hence the power consumption) has to be increased by a factor m, leading again to an increase in the required resources.

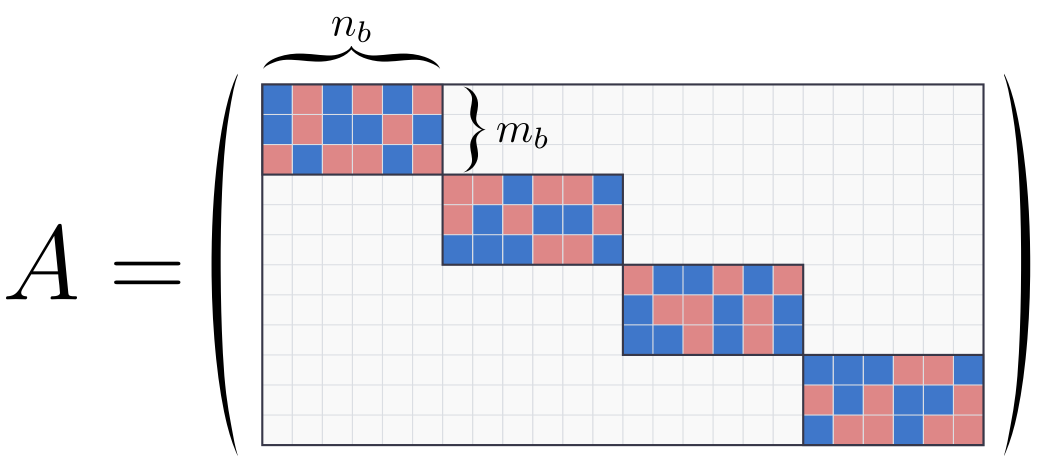


Fig. 2.1 Example of a block-diagonal sensing matrix A, with $n = 24$, $n_b = 6$, $m = 12$ and $m_b = 3$. White blocks correspond to zeroes.

detrimental effects such as clock feedthrough or charge injection cause a degradation that depends linearly on n.

Finally, as the individual x_k are available at different times, they must be sampled and held by the circuit to allow the computation of y_j. For slowly varying signals, leakage currents become a concern, as their effect increases with the hold time, which is proportional to n.

Regrettably, in many CS applications a straightforward reduction of n is not a valid option, since the sparsity properties (hence the ability of correctly retrieving the original signal) are only observed for values of n sufficiently large [30]. A workaround is the design of A as an *antipodal block-diagonal matrix* as in Figure 2.1, i.e. where the $m_b \times n_b$ blocks lying on the main diagonal of A have antipodal-valued entries, while the rest are set to zero. The aspect ratio of the blocks is the same of the original matrix, with $m_b/m = n_b/n$ or, in terms of compression ratio, $m_b = \mathrm{CR} \cdot n_b$.

With this, there are only n_b non-null elements in every row of A, and they are consecutive. This reduces to n_b the effective number of terms in (2.7). Furthermore, non-null elements in different blocks do not overlap, so that the corresponding measurements can actually be computed reusing the same hardware. As a consequence, the number of physical channels required to compute all measurements is reduced to m_b.

Table 2.1 Summary of the tradeoffs involving n or, equivalently, n_b, indicating the section where the topic is discussed. Whenever possible, an estimate of the dependence is given.

Tradeoff	Dependency	Section
Encoder computational complexity	$\mathcal{O}(n \cdot m)$	Sec. 2.5.1
No. of encoder hardware channels	$\mathcal{O}(n \cdot N/\mathrm{CR})$	
Noise injected in measurements	$\mathcal{O}(n)$	
Extension of the acquisition window	$\mathcal{O}(n)$	
Observed signal sparsity	$\uparrow$ with n	
Achievable reconstruction quality	$\uparrow$ with n_b	Sec. 2.5.4
Leakage-induced degradation	$\uparrow$ with n_b	
Compensator dissipated power	$\uparrow$ with n_b	Sec. 3.4.3
Number of leakage compensators	$\mathcal{O}(n_b)$	

The effectiveness of this matrix structure is confirmed by the empirical results published in the literature [31, 32]. For an exhaustive discussion on the consequences of employing a block-diagonal sensing matrix, we refer to [33].

Yet, this approach actually introduces a tradeoff. Values of n_b too low result in structured sensing matrices having a large number of structured zeroes, violating the requirements of the standard CS theory for some classes of signals [24]. Therefore, even if noise and degradation of the measurements are reduced, also the theoretical performance achievable by the CS reconstruction decreases. An example of this trade-off will be analyzed in Section 2.5.4.

As a convenient aid to the reader, Table 2.1 collects all the parameters and properties having a dependence on n and n_b described in the text.

2.5.2 Improving performance by adaptation: Rakeness CS

Many optimization techniques have been proposed with the aim of improving CS performance by adapting the statistical properties of A to those of the signals under consideration [34–36]. For an overview of these techniques, we refer the reader to [37] and we focus here on the approach known as *rakeness* [36, 29].

To the best of the authors' knowledge, this technique ensures a large performance improvement in comparison with other optimization approaches when applied in a realistic setting [37]. Due to this, we will consider a rakeness-based CS system as a reference case for all the analyses conducted in this thesis.

The rakeness approach exploits an additional prior on the input signal named *localization*. Intuitively speaking, the higher the localization of a signal, the farther is the generating process from being white. Formally, x is localized if the $n \times n$ matrix C_x describing the correlation profile of the instances of x, computed as $\mathbf{E}\left[xx^{\mathrm{T}}\right]$, is not the $n \times n$ identity matrix I_n.

Let us indicate with $a \in \mathbb{R}^n$ the generic row of A. According to the rakeness approach, the elements of a are not drawn as instances of i.i.d. Gaussian or sub-Gaussian (e.g., Bernoulli distributed, as in the antipodal case) random variables, but using instead a multivariate random process defined by a $n \times n$ correlation matrix C_a, computed as follows:

$$C_a = \frac{1}{2}\left(\frac{C_x}{\mathrm{tr}(C_x)} - \frac{I_n}{n}\right),$$

where $tr(\cdot)$ is the trace operator.

The generation of the rows of A according to such a correlation profile maximizes the expected value of the energy collected by each measurement, $\rho = \mathbf{E}\left[ax\right]$, as well as the performance of the CS system.

A method to generate rows of A according to the rakeness approach under the constraint of random ternary values can be found in [29]. Both the case of non-structured zeroing (i.e., when the position of the zero elements is a degree of freedom) and of structured zeroing (when the position of the zero elements is constrained a priori) are considered. The latter approach fits perfectly the use of the antipodal block-diagonal matrices depicted in Figure 2.1.

2.5.3 Improving performance through DNNs: TCSSO

Joint use of DNNs and CS can be found in many recent works [38–45]. Typically, DNNs replace the classic BPDn approach of reconstructing the input signal either as $\hat{x}$ or as $\hat{\xi}$ directly from the measurement vector y.

Several examples are focused on compressed images, such as the 3-layer neural network proposed in [42], or the DNN described in [46]. As for other application fields, fast recovery and improved reconstruction quality of videos is obtained in [43], while a joint optimization of the encoding and decoding stages in the CS-based acquisition of EEG signals is presented in [45].

Interestingly, in some works (e.g. [45]) the training process of the DNN also generates the sensing matrix A, ensuring optimal results when applied to the signals contained in the training set, and also when using the DNN for signal reconstruction.

Here, we focus on the innovative approach introduced in [30], where the reconstruction process is split in two consecutive steps. First, a DNN is used to divine the signal support $\hat{s}$ of the reconstructed signal. After that, standard linear algebra is employed to get $\hat{x}$ and $\hat{\xi}$ from $\hat{s}$ and y. The training process of the support oracle network also determines the optimal sensing matrices A to be used for signal acquisition. According to the results presented in [30], the approach ensures a largely improved reconstruction quality. However, the application to antipodal block-diagonal matrix as that in Figure 2.1 is not straightforward, and requires a major modification of the DNN training algorithm. Details on how to make TCSSO work with antipodal block-diagonal matrices, as well as numerical results on its performance, will be provided in Section 2.6.

2.5.4 Case study: synthetic ECG signals

The numerical evaluation of the solutions previously discussed has been carried out on synthetic ECG signals. The generator employed is thoroughly described in [47][3] and the experimental setup is the following.

The heart-beat rate is randomly selected from a uniform distribution between 60 beats/min and 100 beats/min. Signals are generated as noiseless waveforms, sampled at 256 samples/s and split into chunks of length n.

Moreover, in order to obtain a better fit of the signal properties with respect to the requirements imposed by CS, each signal window is modified as follows. *i*) The sparsity level is set to $\kappa = 24$ by zeroing the less energetic components of every signal instance, under the assumption of an orthonormal Symlet-6 wavelet sparsity basis D [48]. This operation introduces a sparsification-induced noise between 30 dB and 55 dB. An example is given in Figure 2.2, where a 32.4 dB signal-to-noise ratio

[3]The MATLAB code is freely available for downloaded from the Physionet website at http://physionet.org/content/ecgsyn/

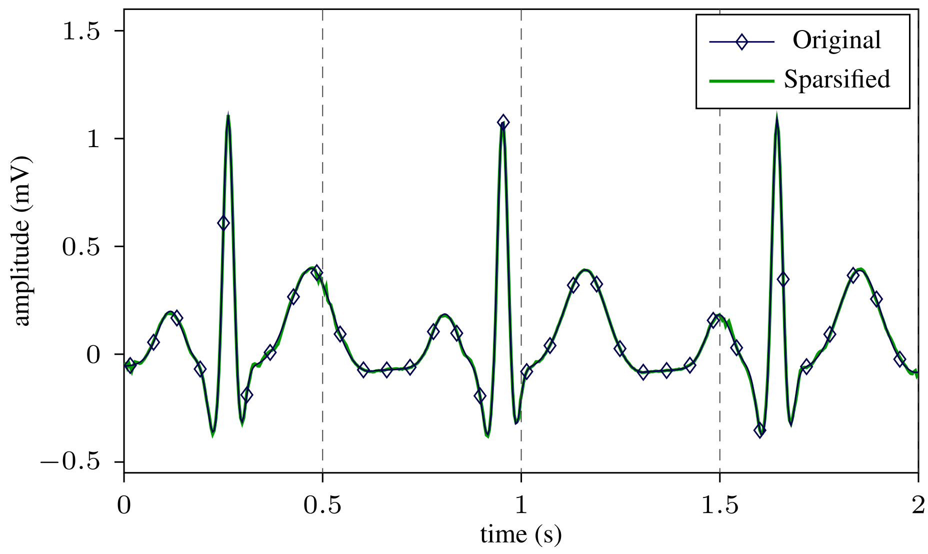


Fig. 2.2 Example of a synthetic ECG with its sparsified version ($\kappa = 24$). The separation in consecutive time windows of length 0.5 s (i.e., $n = 128$ at 256 samples/s) is also highlighted.

(SNR) is observed. *ii*) In order to define a target for an ideal reconstruction, we superimpose white Gaussian noise to the sparsified signal so that its SNR is 50 dB.

Our benchmark is a CS system with $n_b = n = 128$, i.e. full antipodal sensing matrix, in which A is drawn according to the rakeness approach and where the signal is reconstructed by a standard BPDn technique through the SPGL1 algorithm as in [49]. In Figure 2.3 we compare the benchmark results with those obtained for block-diagonal matrices A with different values of n_b, as a function of the desired CR. The figure of merit under consideration is the reconstruction SNR (RSNR), defined as

$$\mathrm{RSNR[dB]} = 20\log_{10}\left(\frac{\|x\|_2}{\|\hat{x}-x\|_2}\right)$$

The plot shows the average value of the RSNR (ARSNR) observed over 1000 Monte-Carlo trials.

According to the data depicted in the figure, a full sensing matrix (i.e., $n_b = 128$), for values of CR ≈ 2 achieves performance close to the theoretical limit of 50 dB. As n_b is decreased, the number of zeroes in the matrix grows and less energy is collected from the signal. As a consequence, performance noticeably drops.

A significant decrease is also observed when the degradation problems addressed in Section 2.5.1 concerning the analog implementation of the CS encoder are consid-

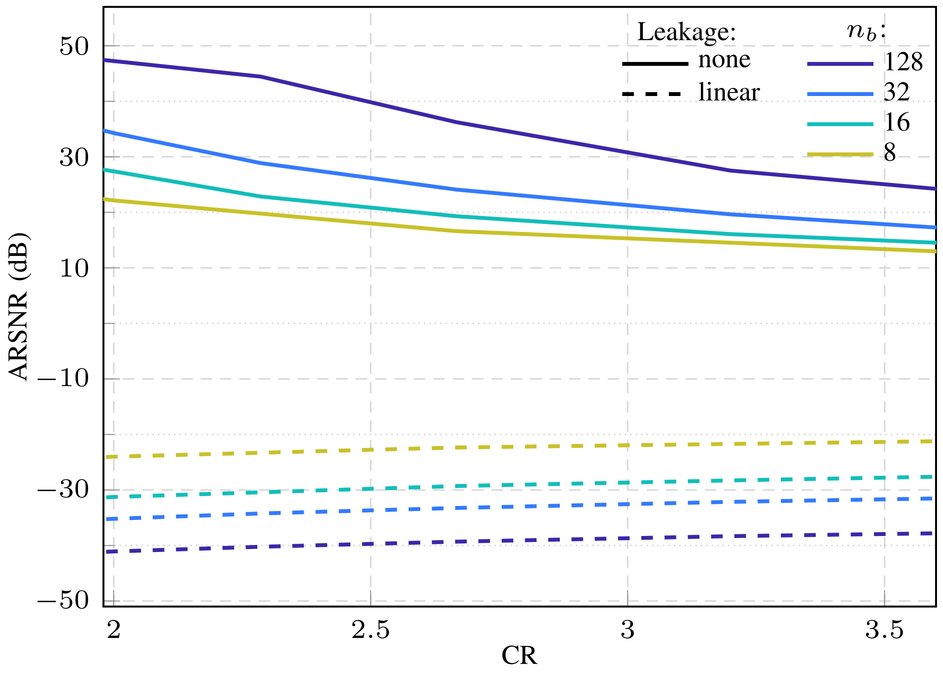


Fig. 2.3 Performance of a CS-based signal processing chain for synthetic ECG signals at approx 60 beats/s, with $f_s = 256\,\text{Hz}$ and $n = 128$. Solid lines refer to an ideal system; dashed lines to measurements corrupted by a constant-current leakage discharge.

ered. Targeting the sensing of low-frequency biomedical signals, leakage currents are indeed the predominant source of noise. In the example, using $n_b = 128$ with a sampling frequency $f_s = 256\,\text{Hz}$ implies that the value of $A_{j,1}x_1$ in (2.7) has to be sampled and preserved for a time period of almost 0.5 s, a hold time typically unaffordable even for pF-range hold capacitances.

In Figure 2.3, the solid lines obtained by an ideal system where no measurement degradation is considered are compared with the dashed curves, representing an acquisition process affected by a constant-current discharge of the hold capacitors due to leakage currents. The discharge model is based on actual data from a 180 nm CMOS technology. It considers a realistic configuration of four minimum size switches for each hold capacitor, leading to a 300 pA discharge current in the unfavorable condition of 85 °C. Moreover, the total sampling capacitance C_{tot} of the SAR array is kept constant to emulate equal area occupation and conversion power. Each hold capacitance is therefore equal to $C_h = C_{tot}/n_b$. For a fair comparison with respect to the solution proposed in [21], the value of C_{tot} has been set to 16 pF.

It is clear from the observation of the dashed curves in Figure 2.3 that block size has an opposite effect on reconstruction quality as compared to the ideal setup: a lower n_b shortens the acquisition window, reducing the degradation of measurements, with a positive effect on the reconstruction performance. However, it is also evident that performance is not sufficient for a decent reconstruction, requiring ad-hoc measures to counter the leakage-induced discharge (Section 3.4.3).

2.6 Decoding by TCSSO with Short Windowing

The application of the TCSSO approach proposed in [30] to the architecture considered in this thesis is not straightforward. In this section we first introduce the original method, then we show how to modify it to handle antipodal block-diagonal sensing matrices. Finally we apply the modified TCSSO to the case study introduced in section 2.5.4 and provide numerical results.

2.6.1 The TCSSO approach

The architecture of the TCSSO approach is depicted in Figure 2.4 and is described in the following.

The computation of the reconstructed signal $\hat{x}$ starts from a set of measurements y coming from the CS encoder. The vector y is fed into a DNN, named “support oracle”, trained to predict the support $\hat{s}$ of the input signal that generated y. Once the support $\hat{s}$ has been estimated, it is possible to reconstruct the original input signal from y ,i.e., to invert (2.6), using standard linear algebra.

Consider $\xi_{|\hat{s}} \in \mathbb{R}^{\kappa}$ as the vector containing only the non-null elements of ξ, whose positions are identified by $\hat{s}$. Similarly, we can define $D_{|\hat{s}} \in \mathbb{R}^{n\times\kappa}$ as the matrix containing only the columns of D selected by $\hat{s}$. Neglecting the noise term ν, the sensing equation (2.6) can be rewritten as

$$y = AD_{|\hat{s}}\,\xi_{|\hat{s}} \tag{2.8}$$

While the inversion of (2.6) gives rise to an underdetermined system of equations whose solution is a complex task involving a minimization problem (2.7), the

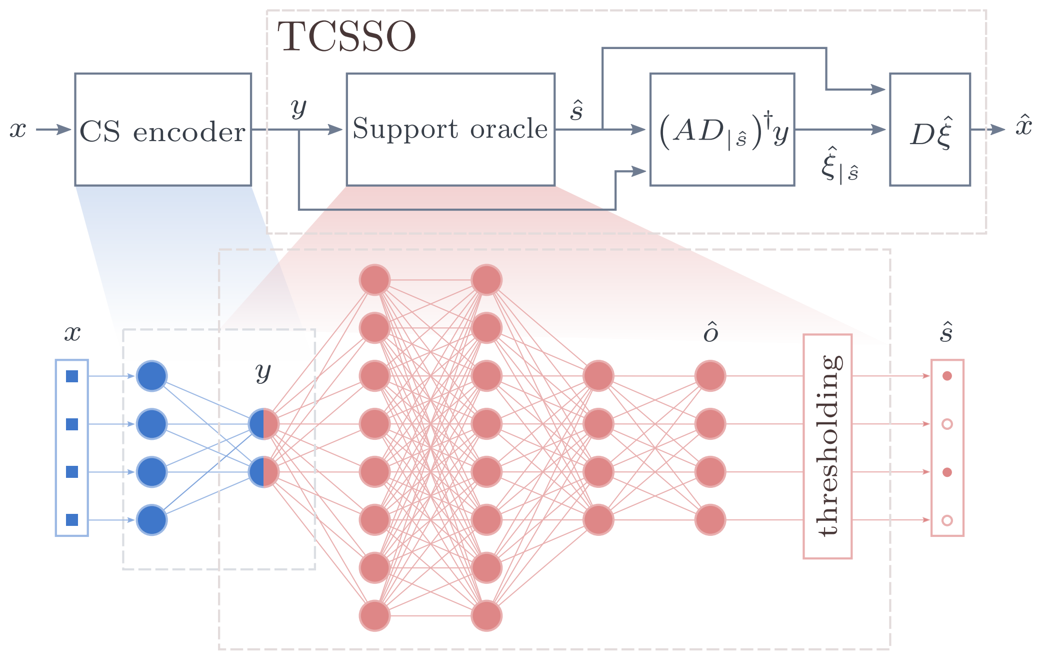


Fig. 2.4 Architecture of the TCSSO framework, with the CS encoder optimized along with the support oracle DNN during the training phase. The recovered signal $\hat{x}$ is computed using the estimated $\hat{s}$.

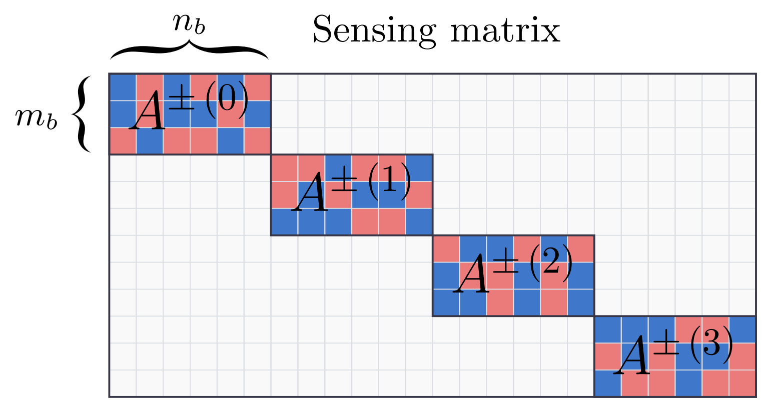
inversion of (2.8) corresponds to an overdetermined system. Several known methods, such as the least-squares based ones, can be used to find its approximate solution. In [30] the authors consider the Moore-Penrose pseudoinverse operator $(\cdot)^{\dagger}$, so that

$$\hat{\xi}_{|\hat{s}} = \left(AD_{|\hat{s}}\right)^{\dagger} y \tag{2.9}$$

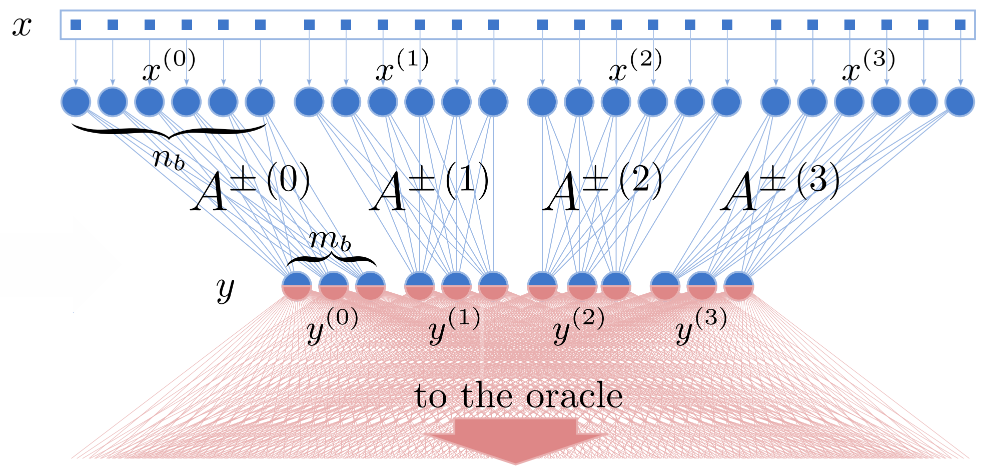
and the original signal is finally reconstructed either as $\hat{x} = D_{|\hat{s}}\,\hat{\xi}_{|\hat{s}}$ or as $\hat{x} = D\hat{\xi}$.

Figure 2.4 also details the internal structure of both the CS encoder and the support oracle DNN. The encoder is considered as part of the neural network only during the training phase, to generate the optimal A. It *emulates* the linear projection $y = Ax$, having n inputs (i.e., the dimensionality of x) and m outputs (dimensionality of y). With no bias, using as interconnection weights the actual elements of A, and employing a linear activation function, its behavior is equivalent to (2.6).

The actual oracle starts from the second layer of nodes, which are shared with the encoder during training. It is built with m inputs and three hidden layers with $2n$, $2n$ and n neurons each, and ReLU activation functions. The n outputs use a sigmoid



(a)



(b)

Fig. 2.5 Equivalent views of the CS encoder in the case of block-diagonal sensing. (a) Antipodal block-diagonal sensing matrix A. (b) DNN layer modeled as several parallel, independent, fully-connected sub-layers. Each block $A^{\pm(l)}$ is mapped to a subset of the weight matrix of a DNN layer so that it can be optimized during the training phase of the support oracle. In the example, $n_b = 6$ and $m_b = 3$, with $\mathrm{CR} = 2$.

activation function $\alpha(a) = 1/(1+e^{-a})$ and generate the output vector $\hat{o} \in \mathbb{R}^n$, which can be interpreted as the probabilities of the coefficients of ξ being non-zero. The divined support $\hat{s}$ is obtained from $\hat{o}$ by simply thresholding it elementwise, so that (ideally) its κ largest elements are set to 1 and the rest to 0 [4].

2.6.2 TCSSO with antipodal block-diagonal matrices

In the TCSSO approach, as well as in many other frameworks where the CS reconstruction relies on a DNN which is jointly trained with the sensing matrix A [42, 45], the key point is to model the matrix-vector product required by the CS acquisition in (2.6) as an extra layer of the DNN.

It is evident that any constraint on A (e.g., forcing it to be antipodal, ternary or block-diagonal) corresponds to an equivalent constraint on such a layer. In reality, when the neural network is used for inference, sensing matrices of any kind can be used with the support oracle. However, during the training phase, obtaining an optimized A with a specific structure requires particular care.

In the original TCSSO framework [30], a full antipodal sensing matrix $A^{\pm} \in \{-1,+1\}^{m\times n}$ is considered. The algorithm used for the optimization of the neural network is the Stochastic Gradient Descent (SGD), which consists of two phases, namely forward- and back-propagation. The convergence of the SGD algorithm requires tiny variations of A, which are not possible if training is performed directly on the antipodal matrix $A^{\pm}$, whose values are either -1 or 1. Therefore a real valued $A^{\mathbb{R}} \in \mathbb{R}^{m\times n}$ sensing matrix is employed since, during back-propagation, its values can be finely adjusted to minimize the error at the output. In the forward-propagation phase, the corresponding antipodal matrix $A^{\pm}$ is extracted from $A^{\mathbb{R}}$ by evaluating the sign of each matrix element.

At the same time, if the sensing matrix has to be block-diagonal, we may actually observe that such a matrix can be split into smaller antipodal matrices $A^{\pm(l)} \in \{-1,1\}^{m_b\times n_b}$ as in Figure 2.5, where l is the index of the l-th block. Each elementary block acts only on a portion of the input signal $x^{(l)} \in \mathbb{R}^{n_b}$, contributing to a subset

[4]In practice the number of ones in the recovered support is on average no less than κ.

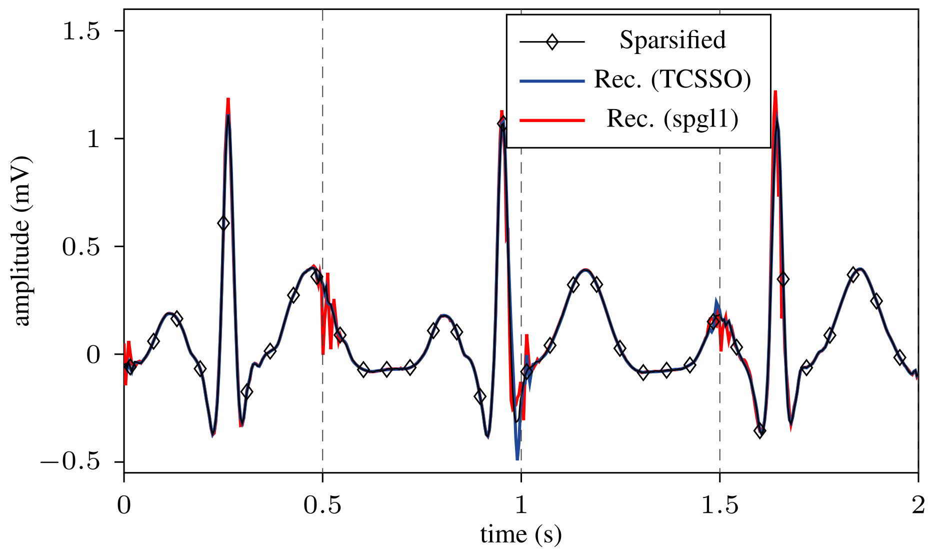


Fig. 2.6 Reconstruction of the sparsified ECG waveform depicted in Figure 2.2, using an ideal setup (no leakage) with $n_b = 8$ and CR $= 2.7$. The different decoders result in a 16 dB RSNR for SPGL1 and 31 dB for the TCSSO.

$y^{(l)} \in \mathbb{R}^{m_b}$ of the measurements vector y. Hence, the sensing operation becomes

$$y_j^{(l)} = \sum_{k=1}^{n_b} A_{j,k}^{\pm(l)} x_k^{(l)}. \quad \text{for} \begin{cases} l = 1, \ldots, n/n_b \\ j = 1, \ldots, m_b \end{cases}$$

In other words, we have decomposed the encoding process (2.7) into n/n_b independent and parallel operations, each of them defined by an antipodal matrix $A^{\pm(l)}$. This is illustrated in Figure 2.5. From the point of view of the DNN, the input layer is no more a single, fully-connected layer, but it is the composition of n/n_b mutually independent, fully-connected sub-layers. All the zeroes of the sensing matrix and the corresponding interconnections between neurons are thus neglected altogether.

Equivalently to what is done in the original framework, here the SGD algorithm is applied to multiple full-precision matrices $A^{\mathbb{R}(l)} \in \mathbb{R}^{m\times n}$. The corresponding antipodal sensing matrices $A^{\pm(l)}$ are obtained by extracting the sign from each element of every $A^{\mathbb{R}(l)}$. Finally, the desired antipodal block-diagonal matrix $A^{\pm}$ is composed by a proper arrangement of the individual blocks on the main diagonal.

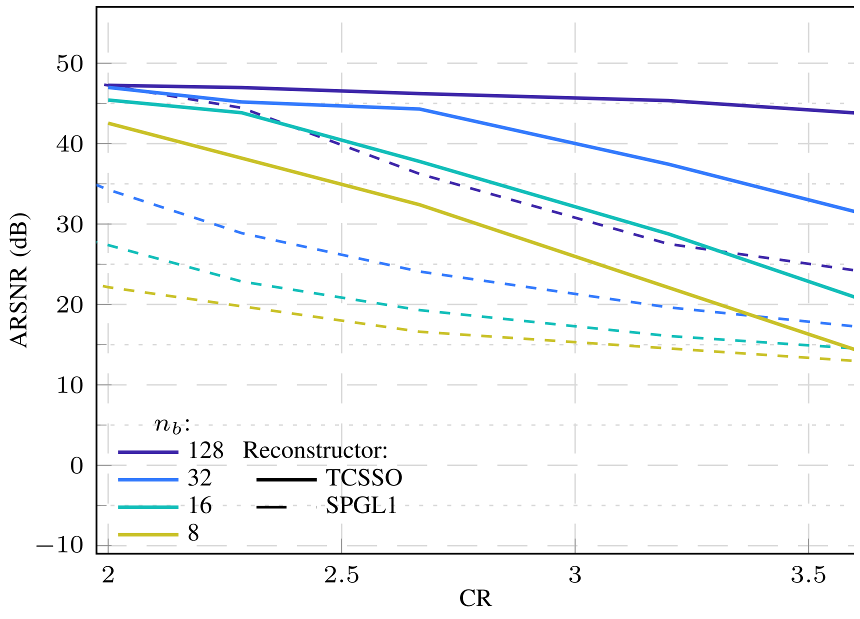
2.6.3 Numerical results with the modified TCSSO

Simulations using the same input signals defined in Section 2.5.4 prove that the TCSSO approach is extremely effective. When properly modified to work with antipodal block-diagonal sensing matrices, it ensures better performance with respect to the BPDn approach boosted by the rakeness optimization. A time-domain comparison of the two decoding is shown in Figure 2.6, where the techniques are applied to the signals depicted in Figure 2.2. The behavior is noticeably improved at the window boundaries, which are the largest noise contributors.

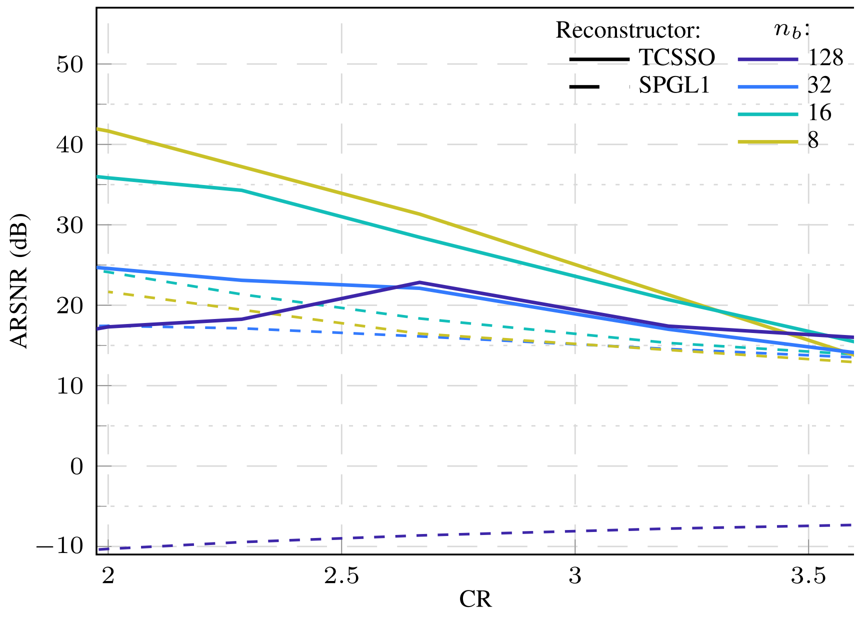
The ideal setup without the effects of the leakage currents is considered in details in Figure 2.7a. Similarly to what is shown in Figure 2.3 for the BPDn reconstruction, a higher value of n_b results in an improved performance. Furthermore, for all considered values of CR, reconstruction quality using the TCSSO is higher with respect to the reference case given by the SPGL1 algorithm with rakeness-optimized sensing matrices.

In Figure 2.7b measurements are degraded by leakage. It can be readily observed that TCSSO achieves up to 20 dB of increased ARSNR with respect to the reference case. As already described for the curves Figure 2.3, the value of n_b sets a trade-off. However, whereas for a BPDn-based reconstruction the optimal performance is obtained for $n_b = 16$, using TCSSO the optimum is found for $n_b = 8$. This is extremely important from the hardware point of view, since it allows at the same time an improvement in performance and a reduction of the complexity of the architecture, requiring a coarser decomposition of the SAR capacitive array (reduced number of hold capacitors C_h).

Leakage discharge currents represent one of the most detrimental factors for the maximum hold time in analog sample-and-hold circuits. Apart from the obvious passive solution of enlarging the sampling capacitor, alternatives based on active circuits have been proposed. We focus here on an existing solution which has proven to be effective in reducing the leakage discharge, hence extending the hold time, by a factor of 20. Being based on a feedback circuit built around the hold capacitor, it is paramount to understand its stability properties. This work tries to close the gap by analyzing the closed-loop stability of the nominal circuit. Classical control systems techniques are employed to thoroughly analyze the dynamic behaviour of the feedback circuit, highlighting the detrimental effect of device mismatches.



(a)



(b)

Fig. 2.7 Performance of a CS-based signal processing chain expressed in terms of ARSNR as a function of CR, for synthetic ECG signals at approx 60 beats/s, with $f_s = 256$ Hz and $n = 128$. Solid lines refer to the TCSSO reconstruction, dashed lines to the SPGL1 algorithm. (a) Ideal sensing (b) Sensing corrupted by leakage, with the presence of the compensator introduced in Section 3.4.3 and the same parameters used to obtain Figure 2.3.

Chapter 3

CMOS Analog-to-Information Converter

Quannu si' marteddru vatt'.
Quannu si' 'ncudine statt'.

Old Calabrian saying

In the previous chapter, we have outlined the requirements and properties of an adapted signal acquisition process, tuned to specific signal families. This Compressed-Sensing-based approach has been presented in a specific configuration, with compression being applied after sampling the input. This chapter will start from a more general viewpoint, so that tradeoffs can be better understood when designing ad-hoc hardware. The features of notable architectures published in the Literature will also be described, to introduce our proposed low-power analog implementation of the sensing process, based on a Successive Approximation Register (SAR) A/D converter. Minor modification of the textbook architecture, namely the addition of a few extra switches and a revised control logic, enable the circuit to act as a Compressed Sensing encoder, performing the matrix-vector product $y = Ax$ in the analog domain, maximizing the use of the passive structures already found in the A/D converter.

The chapter will then continue with implementation details, such as the structure of the capacitive array, the topology of the selection switches, the operations of the dynamic comparator and the description of the necessary digital logic.

Finally, we will go through the layout of the entire integrated circuit, show some post-layout simulations and introduce the design of the PCB to physically test the chip. Unfortunately, due to delays in the fabrication of the chip, no actual measurements are included in this work.

Additional analysis that have been helpful in the design but are somewhat tangential have been collected in Chapter 4 not to interrupt the flow of the following discussion.

3.1 Where to Apply Compression

Compressed Sensing condenses all the meaningful information carried by a signal in as few measurements as possible, fewer than what the Shannon sampling theorem would require. Along the signal acquisition chain, the compressed encoding can be performed in different locations, affecting the hardware requirements and potential benefits. Fig. 3.1 depicts the main blocks of a typical A/D signal processing chain, along with the sections where compression can be performed to obtain the desired Analog-to-Information conversion.

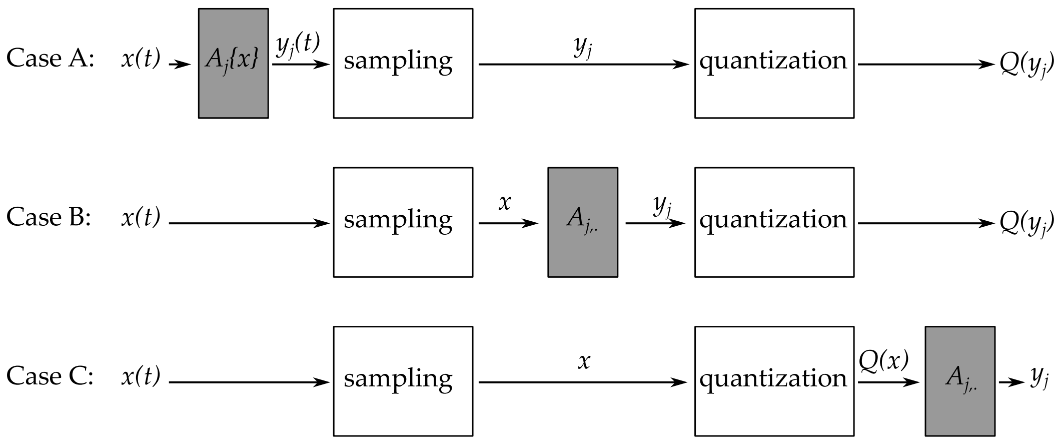


Fig. 3.1 Compression applied at different locations of the signal processing chain.

The first possibility (Case A) is to work in the continuous-time analog domain, where the continuous-time operator $\mathscr{A}_j\{x\}$ modulates the input. In general, active circuits[1] are required to implement the analog multiplication and continuous-time integration, leading to significant additional power consumption. This "CS-first" signal processing chain becomes interesting at extremely high frequencies, where even sampling the input at Nyquist-rates becomes difficult. In that case the compression introduced by CS directly translates to a reduction of the switches' operating frequency.

If instead the input is sampled first (Case B), compression can still be applied in the analog domain. Having discrete-time samples, the transformation becomes a modulation of the sampled input x by some coefficient vectors A_j. With the same considerations as in Section 2.5, the modulating coefficients can be limited to 0 and ± 1, so that the product effectively becomes a much simpler sign inversion. The sum of the partial terms $A_{j,k}x_k$ involved in the matrix-vector product can be implemented, other than using a discrete time integrator, with an entirely passive solution, as we will presented in this chapter. The input switches still work at Nyquist-rate, but A/D conversions are performed only at the end of an acquisition window and extremely low power consumption can be obtained.

The final solution (Case C) involves a digital compression, with no modifications of the blocks before the ADC, which generate quantized samples of the inputs as usual. This could be intended as a plugin solution attached to an existing ADC chain to compress the acquired data before sending them out to some receiving platform. The benefits are only in terms of the transmission data-rate and no more in terms of acquisition energy.

Our discussion will be focused on the second solution, with an in-depth analysis of how the traditional architecture of a SAR ADC can be adapted to include the processing steps required by a CS encoder. We will initially go through some significant architectures proposed in the Literature, in order to gain a better understanding of the tradeoffs and design choices available when implementing CS encoders in the analog domain.

[1]We denote as active circuits any circuit block that draws current continuously from the power supplies

3.2 Existing analog CS encoders

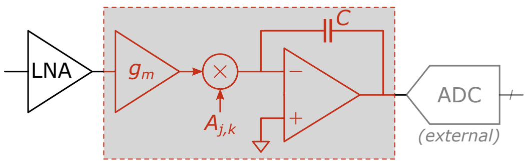
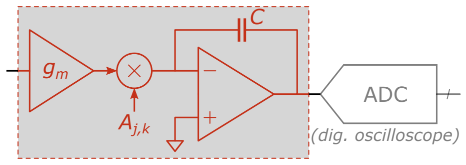
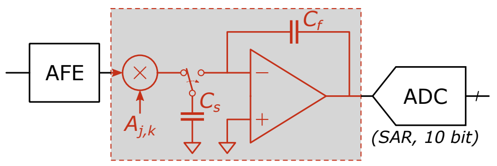
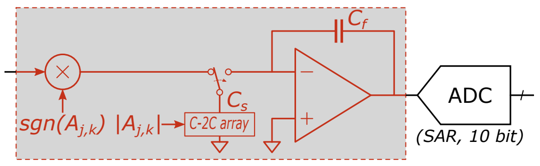
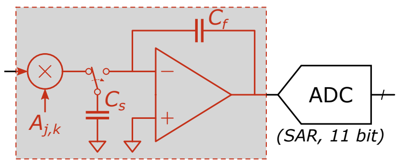
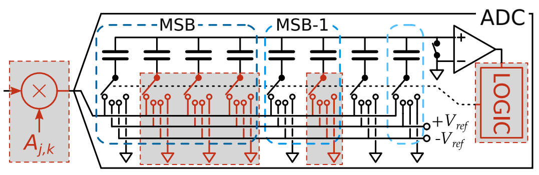
Table 3.1 summarizes the main features and performance metrics of integrated solutions recently proposed in the Literature. For each architecture being considered, a simplified schematic highlights the additional hardware blocks required with respect to a straightforward Nyquist-rate acquisition, given by the direct A/D conversion of the input signal samples. The table includes the energy required by the additional active elements to compute a single measurement, as well as the signal bandwidth and the ADC resolution. Of course, a fair comparison would look also at more CS-centric capabilities (mainly, the possibility to work at different levels of compression by controlling m or the availability of a multi-channel input). The aim of the comparison is to highlight how existing architectures require significant additional energy as compared to the that of a mere A/D converter circuit.

In [23] a sub-Nyquist rate receiver for radar pulse signal is presented. A single input, amplified by an LNA drives 8 parallel channels. Each channel has its own modulator, operating at Nyquist rate, its output is integrated over a fixed time and then digitized. The sparsity basis for the signals of interest, i.e., radar pulses, is a multi-scale Gabor dictionary, as the pulses are sparse in the time-frequency domain. Each channel is a modified direct-conversion receiver, working in the current domain to maximize the dynamic range.

Authors of [22] presented a data acquisition front-end for RF communication assuming a multi-tone input signal. Both solutions embed a passive mixer that exploits $A_{j,k} \in \{-1,1\}$ by exchanging the two wires of the differential input signal. However, they also require a power-hungry continuous-time integrator.

Solutions designed for lower bandwidths typically rely on a switched-capacitors integrator architecture. In [50] an analog front-end for ECG signals is presented, with a passive mixer designed for approximating $A_{j,k} \in \mathbb{R}$. It exploits the differential architecture to implement the sign change, and a 6-bit multiplying DAC embedded in the integrator. In [19] the target application is given by intracranial EEG signals, and also in this case a passive mixer is adopted, implementing $A_{j,k} \in \{0,1\}$ by means of simple pass-transistors. The last considered architecture is that described in [21], where a passive mixer is obtained constraining $A_{j,k} \in \{-1,+1\}$ and exploiting the fully differential architecture for sign inversion via pass-transistors. Anyway, all

Table 3.1 Summary of solutions implementing a CS encoder according to (2.7). The additional hardware with respect to a Nyquist-rate approach is highlighted.

Schematic and Description	Figures of Merit
	[23] (2012, 90 nm). Continuous-time g_mC integrator with passive mixer ($A_{j,k} \in \{-1,1\}$, fully differential). 2 GHz BW, 506.4 mW ($n = 100$, $m = 8$), 1.58 nJ/conv, 15.8 pJ/conv/n. –[1]
	[22] (2012, 90 nm). Continuous-time g_mC integrator with passive mixer ($A_{j,k} \in \{-1,1\}$, fully differential). 500 MHz BW, 30 mW 0.083 nJ/conv, ($n_b = 22$, $m_b = 8$)[2], 3.75 pJ/conv/n_b, 8 bit[2]
	[19] (2014, 180 nm). Switched-cap. integrator with passive mixer ($A_{j,k} \in \{0,1\}$, pass-transistors). 10 kHz/m BW[3], 8.4 µW, ($n = 16$), 0.42 nJ/conv, 9.2 bit
 sgn($A_{j,k}$) $\|A_{j,k}\|$ C-2C array C_s C_f ADC (SAR, 10 bit)	[50] (2014, 130 nm). Switched-cap. integ. with passive mixer ($A_{j,k} \in \mathbb{R}$, module of $A_{j,k}$ via modulation of the sampling capacitance). 1 kHz BW, 1.8 µW, ($n = 256$, $m = 64$), 3.58 nJ/conv, 14 pJ/conv/n, 6.5 bit
	[21] (2016, 180 nm). Switched-cap. integrator with passive mixer ($A_{j,k} \in \{-1,1\}$, fully differential). 60 kHz BW, 430 µW, ($n = 128$, $m = 16$), 13.2 nJ/conv, 216 pJ/conv/n, 9.0 bit
 $A_{j,k}$ MSB MSB-1 ADC LOGIC $+V_{ref}$ $-V_{ref}$	This work, 180 nm. Integration by passive charge redistrib. within the ADC. Passive mixer ($A_{j,k} \in \{-1,1\}$ through fully diff. implementation). Negligible extra energy

[1]Uses an external ADC, whose resolution is not indicated.
[2]Uses a block-diagonal sensing matrix. 8 bit assumed for the oscilloscope resolution.
[3]BW decreases with m (resource sharing). Energy indep. of n (integration over space).
[4]Includes the power consumption of the ADC (not declared for the other analog blocks).

these solutions require an operational amplifier as additional active circuit to execute the integration.

The last line of the table presents the architecture to be discussed. We can already see how the integrator is absent from its diagram, while the only significant addition are the extra switches required to collect and store individual samples to be compressed in a single measurement. The extra energy is considered to be negligible as compared to the other circuits, since the integration is performed by an entirely passive structure and the control logic only requires a counting mechanism to keep track of the samples acquired within a window. As a reference case, a 10-bit SAR converter [51] in a 90 nm technology, employing the conversion technique described in the previous section (also known in the literature as *VCM-based method*), shows a power consumption of 3 mW at 100 MS/s, equivalent to 30 pJ per conversion, with more than 9 effective bits. This energy is almost negligible when compared with the additional energy required by all previous solutions in Table 3.1.

An additional drawback of the active integrators is the possibility of saturating the output of the operational amplifiers. As shown in [24], this is disruptive in terms of performance, unless some ad-hoc strategy is applied to signal the saturation to the decoder. The passive charge redistribution in our proposed architecture is unaffected by such an issue. This comes, however, with two major drawbacks, leakage of the capacitive elements and a reduced dynamic range, as we will see in more details in the following.

3.3 Proposed architecture

An enlarged view of the last schematic of Table 3.1 is shown in Figure 3.2. It depicts the key elements of the proposed architecture, which is based on a traditional charge-redistribution SAR ADC [52], suitably modified to simultaneously hold different signal samples at the same time. This is indeed the key idea behind the circuit. Once different signal samples are stored, either acquired sequentially from a single input channel or sampled from multiple channels, their linear combination can be obtained by entirely passive means.

Before delving into the details of the architecture, we will briefly describe the general structure of SAR converters.

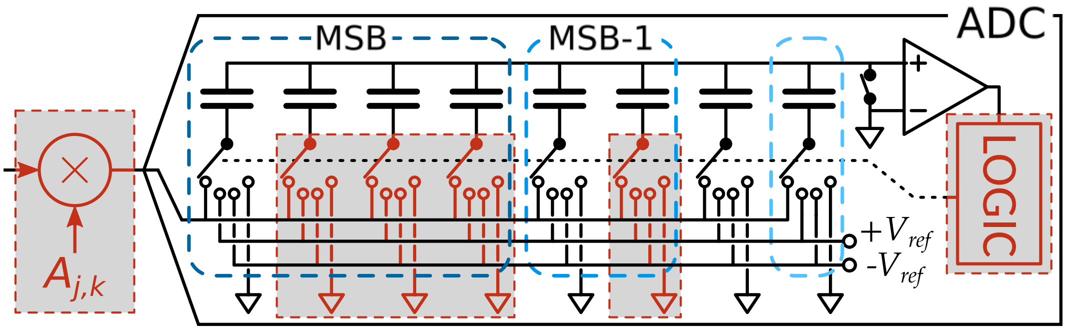


Fig. 3.2 Proposed entirely-passive CS encoder based on a charge-redistribution SAR ADC. The extra blocks enabling CS functionality are highlighted.

3.3.1 Successive Approximation Converters

The successive approximation algorithm allows the conversion of an analog value in digital form by performing a sequence of comparisons of the input against an adaptive reference [53]. The reference is updated across several cycles, until the required accuracy is reached. If the steps' height decreases as a power of two, a resolution of N bits is achieved in N cycles. A high-level view of the circuit blocks involved in such a conversion is depicted in Fig. 3.3a, with the time-domain behaviour of the most important signals in Fig. 3.3b.

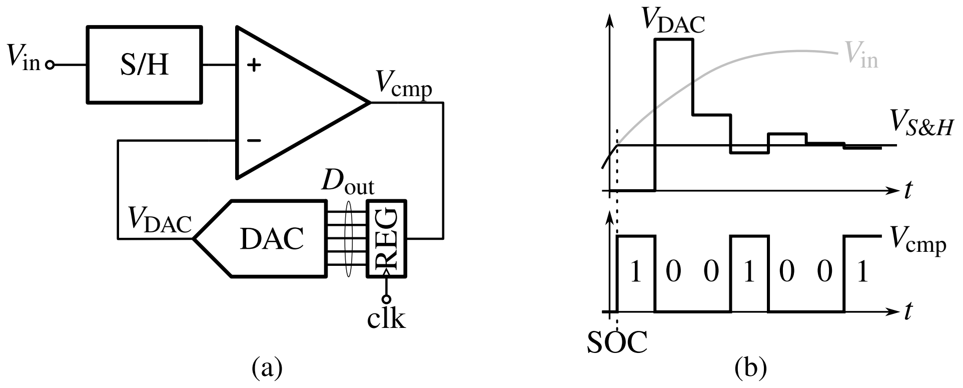


Fig. 3.3 SAR A/D converter. a) Circuit diagram and b) Signal waveforms during conversion.

The input signal is sampled and held constant during the entire conversion and the adaptive reference voltage evolves according to the previous comparison. If the reference voltage was lower than the input sample, the comparator output would go high and the reference is increased. The opposite happens if the reference voltage becomes higher than the input sample.

The outcome of the comparison is stored in a digital register, starting from the Most Significant Bit (MSB) at cycle 1 and moving towards the least significant ones as conversion goes on. The register content is initially reset and represents, over time, the approximation of the input signal, hence the name of Successive Approximation Register (SAR) converter. The analog equivalent of the register content becomes, through a D/A conversion, the adaptive comparator reference. Fig. 3.3b clearly shows how the difference between the DAC voltage and the signal sample progressively decreases, beginning from the assertion of the Start-of-Conversion (SOC) signal.

Over time, the reference voltage converges to within 1 LSB of the input sample. It is also possible, as it happens in the figure, that the final error is not the minimum one obtained during the conversion (which in the example is achieved at the cycle before the last), though it is guaranteed to be below the quantization error by the end of the conversion.

The most critical element required by the SAR A/D converter is actually the D/A converter embedded in it. A popular solution in CMOS technology is to employ switched capacitors [53], both because of the good technological properties of switches and capacitors in CMOS processes, for the absence of static currents and the ability to combine the Sample-and-Hold block and the DAC into a single structure, as depicted in Fig. 3.4, freeing one of the comparator inputs. In single ended systems, the remaining input is connected to a constant voltage. Differential implementations take advantage of it by duplicating the capacitive array to process the inverted replica of the input, for increased dynamic range and noise performance. In Fig. 3.4b the capacitive arrays are divided into identical banks at sampling time, to process independent input channels or different samples from the same input, if the channels are shorted together and the sampling cells are activated in different instants. This principle underlies our proposed architecture and will be explained in more detail in Section 3.3. Note however that it represents a natural extension of the original circuit.

Notwithstanding the energy efficiency of the charge redistribution solution, different SAR algorithms have been proposed in the Literature to lower the energy consumption even more [54, 55].

We will now describe analytically how a CR SAR operates, both in traditional sampling and when adapted to work as a CS encoder.

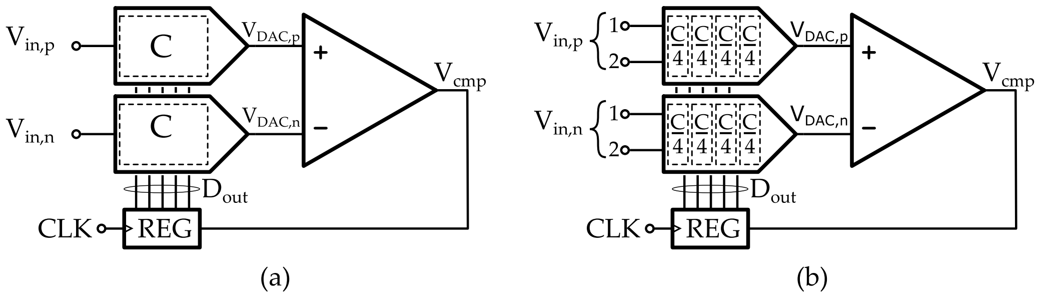


Fig. 3.4 Charge-redistribution SAR converter, combining sample-and-hold and D/A functionalities within the same block. (a) Typical circuit diagram for a differential implementation. (b) Decomposition of the capacitive arrays enabling the acquisition of different samples from either the same input, over time, or from independent inputs in multi-channel systems. Here 4 samples can be stored, from 2 input channels.

Charge-Redistribution SAR converter

A single-ended implementation is depicted in Figure 3.5, where the capacitive array is split, for generality, into a 4-bit binary-weighted array and a 2-bit C-2C structure. This solution is usually preferred to limit the total size of the array, which becomes critical at high-resolutions. To first order, the time-domain dynamics of the capacitive array are independent of the array implementation, and would be unchanged if considering a binary-weighted secondary array, or even a single main array exclusively.

To sample the input, the top plates of the capacitors are grounded by SW_0, while the bottom plates are *all* connected to the input signal V_{in} (selectors in Fig. 3.5 on the rightmost position), tracking the input. This technique, commonly referred to as bottom-plate sampling is advantageous when considering the effect of parasitic capacitances, which introduce a constant attenuation that can be easily compensated for and does not affect the linearity of the conversion.

SW_0 is then opened, isolating the top plates and all the capacitances are grounded. This stored charge makes the D/A operations of the array signal dependent, as required by the SAR algorithm, but using a single structure. Therefore only one comparator input is occupied and the other can be set to a constant voltage. The actual conversion starts right after driving all bottom plates to ground, with the top plates allowed to float.

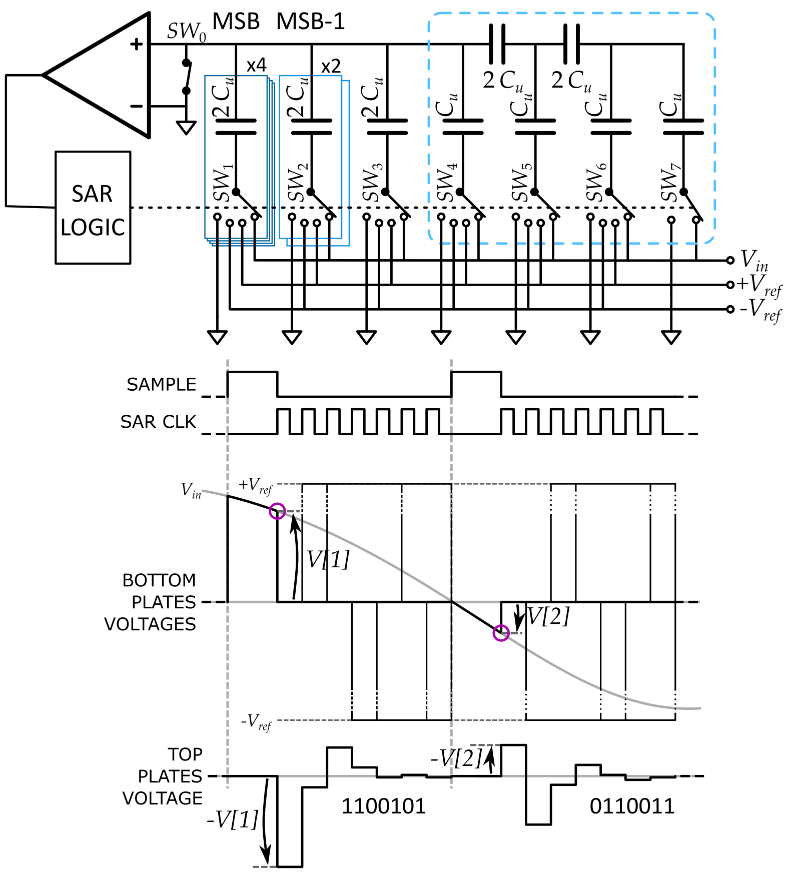


Fig. 3.5 Schematic and working principle of a traditional charge-redistribution SAR ADC with a 4-bit binary-weighted array, and a 2-bit C-2C array. Notice that, having the comparator reference in the middle of the range $[-V_{ref}, +V_{ref}]$, the overall ADC gains an extra bit, reaching a total of 7 bits.

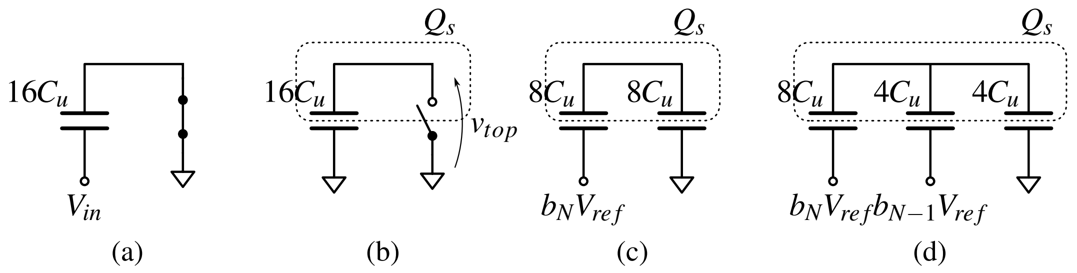


Fig. 3.6 Capacitive array during: a-b) sampling, c-d) conversion. The charge in the isolated node remains constant across the operations.

Using the notation $v[n] \overset{def}{=} v(nt_{\text{clk}})$, where n represents the conversion cycle and t_{clk} the time required by each step, the initial array voltage can be expressed as

$$v_{\text{top}}[0] = -V_{in}$$

and the most significant bit b_N, considering a bipolar representation where $b_i \in \{-1, +1\}$, is obtained as

$$b_N = \text{sign}\left(v_{\text{top}}[0] - V_{\text{cm}}\right).$$

In the circuit in Fig. 3.6 V_{cm} corresponds to the ground potential. If v_{top} is lower than the common mode level, $b_N = -1$ and the array voltage has to be increased to get closer to the common mode. The value of b_N determines the connection of the MSB capacitor to either V_{ref}^{+} or V_{ref}^{-} resulting in a new array voltage. This value can be derived by considering the fixed charge stored on the top plates and the new connection of the largest capacitor.

The array can be grouped into two elements, $C_{MSB} = 8C_u$ and $C_{rem} = 8C_u$, such that $C_{MSB} + C_{rem} = C_{tot} = 16C_u$ (Fig. 3.6c). Forcing the conservation of charge, we obtain

$$-V_{\text{in}}C_{\text{tot}} = \left(v_{\text{top}}[1] - b_N V_{\text{ref}}\right) C_{\text{MSB}} + C_{\text{rem}} v_{\text{top}}[1]$$

and consequently

$$v_{\text{top}}[1] = -V_{\text{in}} + b_N \frac{C_N}{C_{\text{tot}}} V_{\text{ref}} = -V_{\text{in}} + \frac{b_N}{2} V_{\text{ref}}.$$

Since the MSB capacitor represents half the total capacitance, the array voltage changes by half the reference voltage.

For the second bit:

$$-V_{\text{in}}C_{\text{tot}} = \left(v_{\text{top}}[2] - b_N V_{\text{ref}}\right) C_{\text{MSB}} + \left(v_{\text{top}}[2] - b_{N-1} V_{\text{ref}}\right) C_{\text{MSB}-1} + C_{\text{rem}} v_{\text{top}}[2].$$

Coherently with the fact that the (MSB-1) capacitor has one fourth of the array capacitance, the voltage becomes

$$v_{\text{top}}[2] = -V_{\text{in}} + \left(\frac{b_N}{2} + \frac{b_{N-1}}{4}\right) V_{\text{ref}}.$$

As conversion proceeds, new bits are generated, each of them determining the position of one selector. Since the capacitors have values that decrease as powers of two, every new bit leads to an increasingly smaller variation of the array voltage. At the end of the conversion

$$\begin{aligned} v_{\text{top}}[N] &= -V_{\text{in}} + \frac{b_N C_{\text{N}} + b_{N-1} C_{N-1} + \cdots + b_0 C_0}{C_{\text{tot}}} V_{\text{ref}} \\ &= -V_{in} + \left(\frac{b_N}{2} + \frac{b_{N-1}}{4} + \cdots + \frac{b_0}{2^{N+1}}\right) V_{\text{ref}}. \end{aligned} \tag{3.1}$$

The fact that the number of terms in (3.1) is $N+1$ whereas the capacitors are N stems from the fact that the first bit is generated by grounding all the array elements, and the remaining N by acting on the capacitors (the closure capacitance of value C_u is excluded from the count since it is introduced only to have power-of-two coefficients and is unused during conversion). The quantization error is therefore bounded by

$$|\varepsilon_q| < \frac{1}{2^{N+1}} V_{\text{ref}}.$$

As N is increased, the error is reduced and the approximation improves. However, every additional bit doubles the array capacitance, with an exponential increase of both the area and the power consumption. The total capacitance is in fact

$$C_{\text{tot}} = 2^N C_u.$$

As a consequence, also the impedance of both the source and the switches has to be extremely low to avoid slow transients.

Charge-Redistribution CS Encoder

The proposed architecture modifies the behavior of the traditional converter when sampling occurs, as shown in Figure 3.7. Additional switches allow a finer-grained control of the largest array capacitors, and through the presence of SW_{in} at the input the signal modulation described by equation (2.7) can be implemented directly in the analog domain. The modulation is achieved by the switch SW_{in}, which selects either $+V_{in}$ or $-V_{in}$, inherently available in differential implementations. The structure depicted in the figure corresponds to one row of the sensing matrix A and is used to compute the j-th measurement y_j.

The original capacitive array can be used as n_b identical sampling capacitors (8 in Fig. 3.7), each of size $C_h = C_{tot}/n_b$, so that n_b samples of the modulated input can be stored independently (in the figure, with $C_h = 2C_u$ we are able to obtain 8 hold cells). This operation requires the decomposition of the largest capacitors in Figure 3.5, i.e., $8C_u = 4 \times 2C_u$ and $4C_u = 2 \times 2C_u$, into smaller elements, hence the requirement for additional switches. At the same time, the smallest capacitors have to be driven simultaneously, so that their combined sampling capacitance is equal to C_h.

Before each sampling instant, only one among the switches SW_A to SW_H connects the input to a sampling capacitor. By the end of the acquisition window, each of them will hold a value equal to $A_{j,k}V[k]$, with $k = 1, 2, \ldots, 8$, in agreement with the notation of the figure.

Finally, SW_0 opens, and all other switches move to the ground position. All the sampling capacitors are thus connected in parallel, sharing the accumulated charge and generating a voltage level at the input of the comparator equal to the average of the individual capacitor voltages:

$$V_y[j] = -\sum_{k=1}^{n_b} \frac{C_h A_{j,k} V[k]}{n_b C_h} = -\frac{1}{n_b} \sum_{k=1}^{n_b} A_{j,k} V[k] \,.$$

Apart from the scaling factor $-1/n_b$, this is equivalent to the measurement y_j described by (2.7).

The acquisition phase is now complete and the A/D conversion can start by *logically* reconfiguring the capacitive array in its original shape. The largest of the binary-weighted capacitors, initially split into smaller elements, can be re-obtained

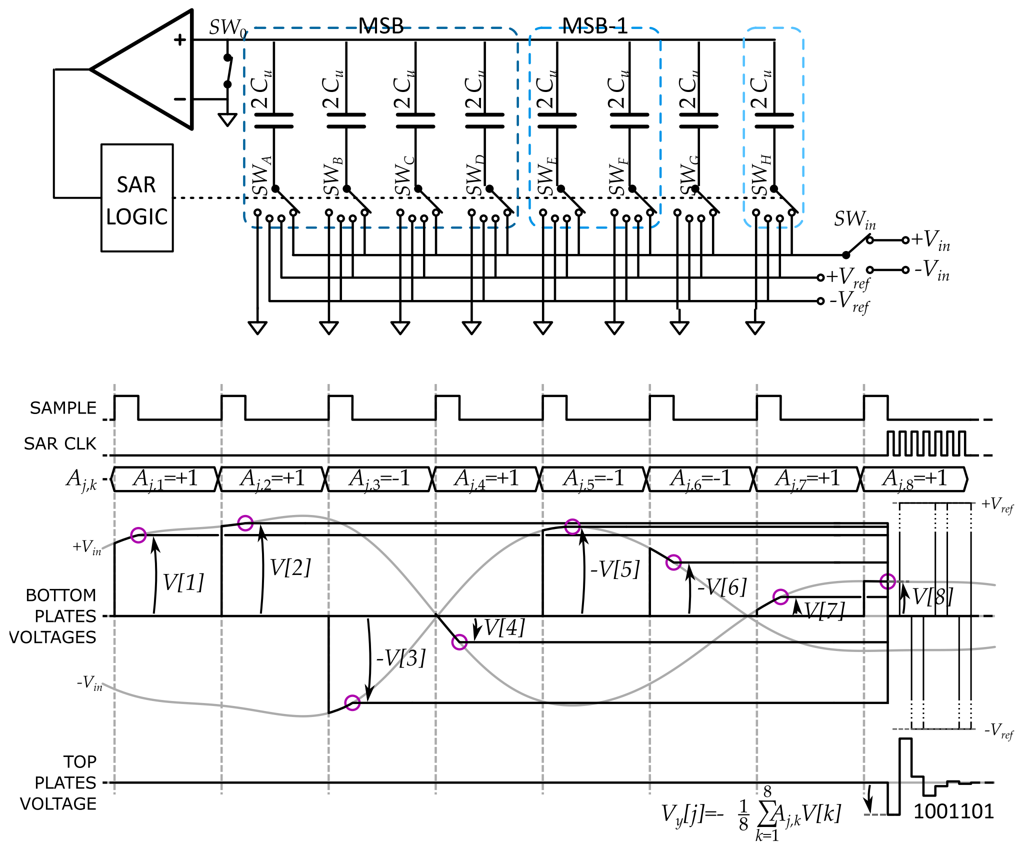


Fig. 3.7 Schematic and working principle of the CS-based acquisition system, based on a charge-redistribution SAR ADC, proposed in this thesis. The structure has $C_{tot} = 16C_u$, $n_b = 8$ and $C_h = 2C_u$. The $+V_{ref}$ and $-V_{ref}$ levels in the timing diagrams are not to scale.

by driving its switches simultaneously. Equivalently, the smallest capacitors, jointly driven during acquisition, have to be controlled independently.

Note that, as we will analyze in Section 4.1, the C-2C sub-array requires particular care because of its sensitivity to parasitic loading of the internal isolated nodes. Hence it is preferable not to change its topology during the acquisition phase, using it as a whole. This determines a minimum value for C_h, which has to be $C_h \geq C_u$.

3.4 Implementation details

A practical realization of the proposed Charge-Redistribution CS encoder requires the estimation of unwanted nonidealities, their mitigation through component sizing and an accurate selection of the employed circuit blocks. Here we will go through some of the considerations made during the design.

3.4.1 Block diagram and I/Os

The overall structure of the converter is shown Fig. 3.8. Each integrated circuit is split in two identical halves, each containing a coefficient memory and two CS encoders. The coefficient memory controls the state of the input modulators, as well as the activation of the sampling capacitors within the capacitive arrays. The following subsection will go through additional details of the specific blocks.

3.4.2 Capacitive Array and Sub-Arrays

The greatest concern as resolution is increased is the matching accuracy of the capacitive elements, since the operations of the DAC depend inescapably on exact ratios of capacitance. As larger capacitors are added to introduce new bits, guaranteeing the accuracy of the ratios becomes problematic. Two structures typically employed to mitigate the issue are the split array and the C-2C sub-array, both shown in Fig. 3.9.

In the scaled capacitive array considered in the previous section, all the capacitors shared the top node. If a series element (bridge) is introduced, the array is split. Between the two top nodes now present, only one needs to be grounded during

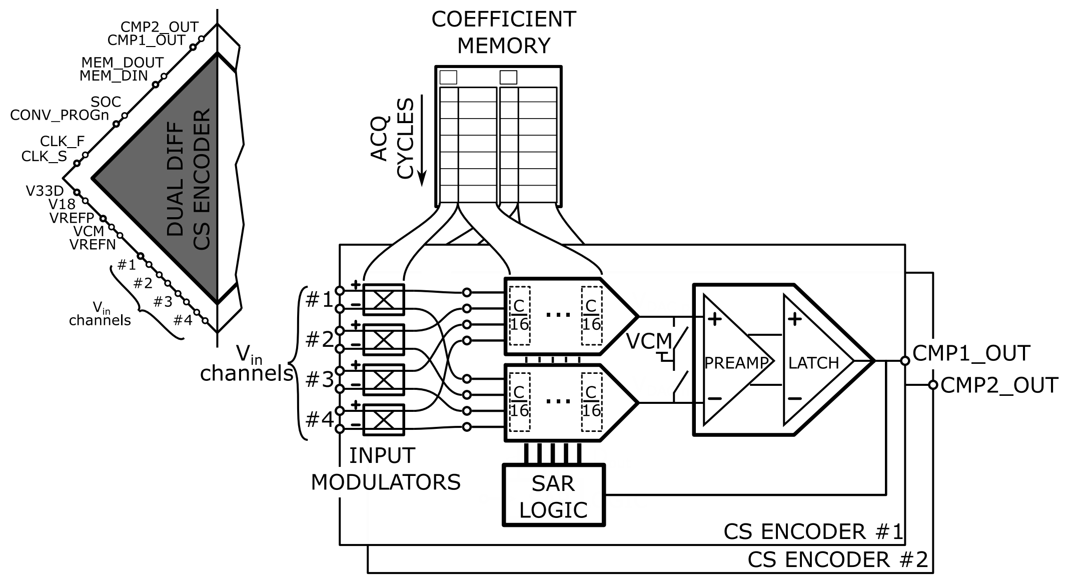


Fig. 3.8 I/Os of the converter and inner block diagram.

sampling and subsequently observed by the comparator. In Fig. 3.9a it is the left node, since the closure element C_u is on the right-hand side.

Looking from the comparator input, the capacitance on the other side of the bridge is attenuated, acting as an equivalent smaller elements, even though their size shares the range observed in the original array. The range of capacitances is thus limited and matching can be guaranteed more easily. Eventually the entire primary array, having N_p bits could be replicated, doubling the resolution while doubling the area and total capacitance $2^{N_p+1}C_u + C_b$. The same resolution increase, using exclusively a scaled array, would require a capacitance $2^{2N_p}C_u$, the square of the original one.

The value of bridge capacitance making the secondary array look as an extension of the scaled array is derived by considering that its total capacitance $C_{sec} = 2^{N_s}C_u$ in series with C_b has to equal C_u (to have a total capacitance, as seen from the input, expressed as a power of two). Therefore

$$C_b = \frac{2^{N_s}}{2^{N_s}-1}C_u.$$

The values of bridge capacitance are not integer multiples of the C_u, requiring a careful layout. The procedure could be repeated even more than once, adding several

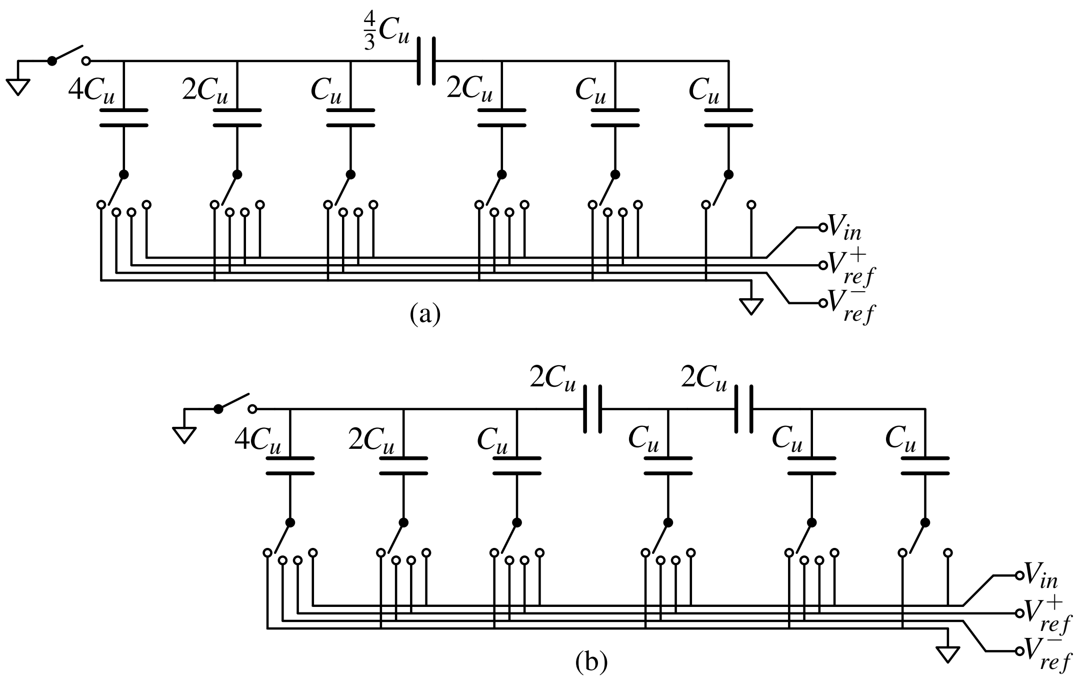


Fig. 3.9 a) Split array with unequal sections b) Mixed-type array: scaled plus C-2C sections. The comparator in both cases would be connected to the same node as leftmost switch.

bridges. However, having non-integer multiples of C_u and introducing at the same time many isolated nodes reduces the achievable accuracy, making the structure extremely sensitive to injected noise and parasitic loading, with detrimental effects on the conversion quality.

Notwithstanding this last consideration, the structure that minimizes the total capacitance and that has been actually considered in the proposed solution is the mixed-type array, employing a C-2C topology. It involves the cascade of several capacitive dividers, with values $2C_u$ and C_u and loading the original scaled array. One cell per each new bit is introduced, with a final closure capacitance equal to C_u. In Fig. 3.9b, two C-2C cells are added to the original scaled array. The total capacitance expressed as a function of the number of bits in the sub-array is:

$$C_{C2C} = (3N+1)C_u$$

This is lower than in a scaled implementation if $N \geq 4$. Moreover, requiring only two values of capacitance, high matching can be easily achieved.

The downside of this solution is the presence of the many isolated nodes. What will be shown in Section 4.1 is that the amount of parasitics loading the internal nodes determines the maximum number of bits achievable with such a structure. This is the main limitation preventing the realization of the entire array as a cascade of C-2C cells.

Evaluation of the array voltage in both the split and C-2C solutions, as already done in (3.1) for the scaled topology, is slightly more complicated. However, using superposition and considering the expression of a capacitive voltage divider under the assumption of no net charge in the hidden isolated nodes, the result can be obtained. Indeed the expressions in the case of the C-2C-based mixed-type array will be derived in Section 4.1, where the effect of the parasitic loading of the inner isolated nodes is evaluated for the resulting conversion nonlinearity.

3.4.3 Switch Architecture

The selection of an effective switch topology and sizing has to constraint a few limitations our architecture may suffer from, namely, charge injection, clock and input feedthrough, and leakage currents.

Charge injection errors

The so called “pedestal error” is a common phenomenon in CMOS sample-and-hold circuits whereas a voltage step is observed across the hold capacitor upon switch turn-off. Being a voltage-dependent effect, it introduces a nonlinearity. Extensive studies have been conductive to model the effect and find mitigation strategies [56]. The general rule is that a slow gate turn-off transient allows charge to flow towards the input source, provided that its output impedance is sufficienty low. Conversely a fast transient results in a more predictable, albeit larger charge injection, which becomes closer to half the charge held within the channel during the ON time. The use of a complementary switch, i.e., a “transmission gate”, is helpful, as each device injects capacitance of a different sign. Though the different depths of the channel below the gate electrode requires a specific ratio of the PMOS to NMOS widths.

In order for a MOS transistor to change its conduction state, the conductive channel has to be created/destroyed, implying a transfer of charge with the surrounding elements. When the transistor is driving a hold capacitor, the exchange leads to a voltage drop on the capacitor whose amount depends on the charge and the size of the capacitor itself. Since the charge stored inside the channel is a function of the local potential, modeling the injection effect is challenging as it would require a pointwise, time-dependent description of the channel potential as the gate voltage is modified [53].

If one of the switch terminals is connected to a low impedance source, a sufficiently slow gate voltage transition allows the charge to be gradually removed from the channel to the source. Conversely, if the transition of the gate voltage is fast enough, a reasonable approximation is to consider the charge equally divided between source and drain (Fig. 3.10).

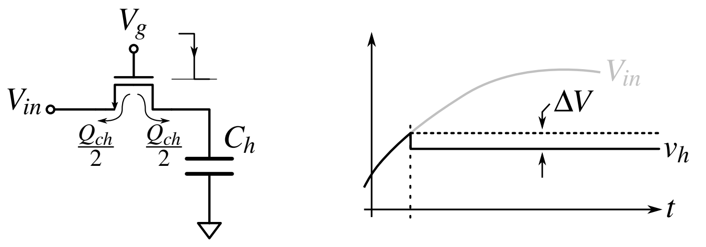


Fig. 3.10 Charge injection and its effects on the sampled value

The voltage drop induced in the hold capacitor C_h is

$$\Delta V = \frac{WLC'_{ox}}{2C_h}|V_{g,on} - V_{in}|.$$

Here W and L are the transistor dimensions and C'_{ox} the oxide capacitance per unit area. Being dependent on the input voltage, it leads to a distortion of the reconstructed waveform.

Other than increasing the value of the hold capacitor and minimizing the channel area, a couple of techniques can be employed to limit the errors. The first and most effective one involves the use of dummy switches (Fig. 3.11a). The dummy element is driven in phase opposition with respect to the main switch, so that when the inversion layer is removed from the main switch, the charge is absorbed entirely by the dummy.

The technique is effective only if the clock transitions are fast enough, so that the half-splitting approximation holds. Thus the dummy size has to be half that of the main transistor (considering that the effective size of small-size devices is different from the design dimensions, it means the main switch should be built as the parallel of two fingers, while the dummy element by a single finger). Since it is important to ensure that the injected charge is captured by the dummy element a small delay in the driving signal of the dummy has to be introduced so that the inversion layer in it is formed after the main switch is opened.

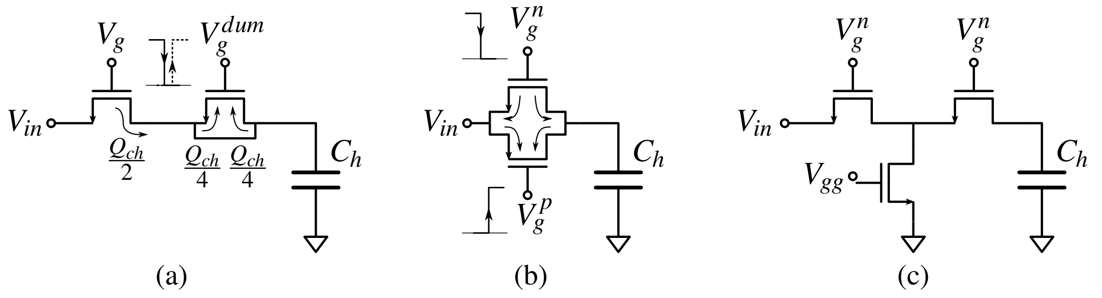


Fig. 3.11 (a-b) Charge injection compensation techniques: (a) dummy transistor and (b) transmission gate. (c) T-switch configuration.

A second solution is the use of a transmission gate (Fig. 3.11b). This is typically employed when the input signal varies in the entire supply range. Since transistor of opposite polarity have inversion charges of opposite kind, when both of them are turned off, they inject charges that mutually compensate. However the solution is not as effective as the first one, since the channel charge depends on V_{gs} and the two transistors are turned on by opposite voltages.

Clock and input feedthrough

The other effect induced by commutations of the switches is due to the capacitive coupling from the gate terminal to the hold capacitance. A capacitive partition takes effect, so that a voltage variation is observed on the hold capacitor.Also in this case, minimum size switches minimize the problem. The techniques previously described to mitigate the charge injection errors are also beneficial from a clock-feedthrough standpoint, as they require opposite voltage transitions to drive the dummy elements or the complementary pairs. This leads to injection of an opposite charge that partially compensates the original one.

Input-feedthrough requires a different technique to better isolate the input from the output of the switch. One of the easiest is to introduce an intermediate node, shorted to a low-impedance constant voltage source whenever the switch is in the OFF state. This topology, also known as a “T-switch”, prevents capacitive coupling through the source-drain capacitance of the MOSFET. It is shown in Fig. 3.11c.

Leakage currents

The mere presence of the switches leads to a continuous discharge of the hold capacitors because of subthreshold conduction through the channel as well as the reverse current through the source/drain diffusions. Minimizing the junction area is the first step to reduce the loss. However, the hold time in this CS-based applications is a significant fraction of one ECG period, i.e. 1 s. In Section 2.5.4 we have already shown how reconstruction may become impossible operating on a time scale this large if the capacitors discharge significantly. The search for a leakage compensation scheme in the Literature lead us to the in-depth analysis of the most promising architecture we have found, in terms of efficacy and robustness. The analysis is shown later, in Section 4.4 and our conclusion is that it is unsuitable for real world applications, as its stability is extremely sensitive to asymmetries of its circuit elements.

The T-switch topology used to decouple the input from the output can be used to minimize the subthreshold channel conduction. If the intermediate node created to decouple input and output is brought to the most positive (negative) voltage for the NMOS (PMOS) branch, the source node is the one on the side of the capacitor, which is the best we can do.

The selected switch topology, accounting for all the effects we want to mitigate, is shown in Fig. 3.12.

As an additional remark, the solutions proposed here to minimize the noise injected by the switches are detrimental from the point of view of leakage, since the number of junctions seen by the hold capacitors increases. The solution using the dummy element is the worse, with four junctions of the same kind acting on the hold node. Even though junction leakage is not modeled accurately in the simulation models provided by the foundry of our selected technology, its tabulated values derived from experimental characterization are reasonably low for our use case.

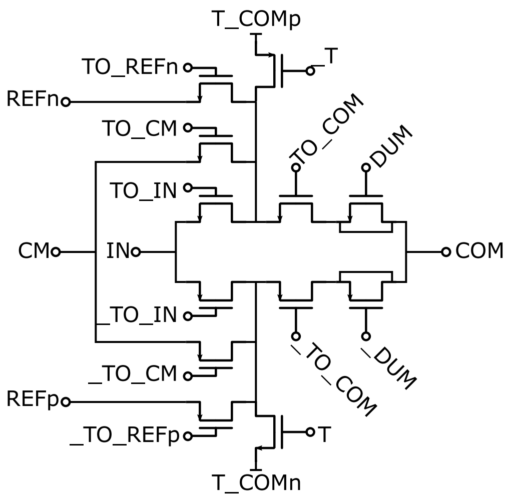


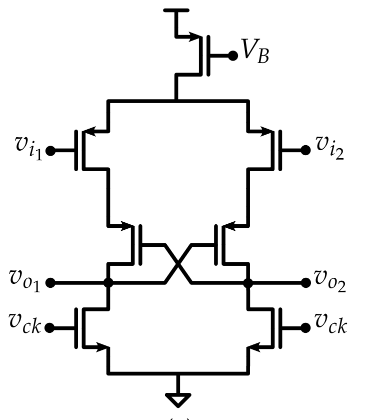
Fig. 3.12 Selected switch topology. The four external inputs IN, REFp, CM and REFn can be routed to the shared COM node. Dummy elements compensate charge injection errors and a T-switch topology prevents coupling from the inputs and minimizes subthreshold leakage.

3.4.4 Dynamic Comparator

The clocked nature of the SAR converter enables the use of dynamic comparators, enabled on-demand to perform the required conversion and preventing continuous current drawn from the supplies. Other than the power consumption, additional metrics to be considered are the response time from the stimulation of the comparator until a decision is made, since it determines the speed at which bits can be generated, the so called kickback effect, where charge is injected towards the input because of rapid voltage transients within the comparator, as well as the ability to resolve an input voltage smaller than the expected quantization error of the D/A converter.

All these constraints drove the selection of a suitable topology. The candidate architecture we have analyzed in detail and that will be described in the following is shown in Fig. 3.13 and has been taken from [57]. It is built as the cascade of a dynamic residual preamplifier and a parallel-coupled regenerative latch.

Each stage will be analyzed separately, deriving the analytic expression of the transient response in the different operating phases. Then, asymmetries of the circuits will be quantified as an input referred offset voltage.



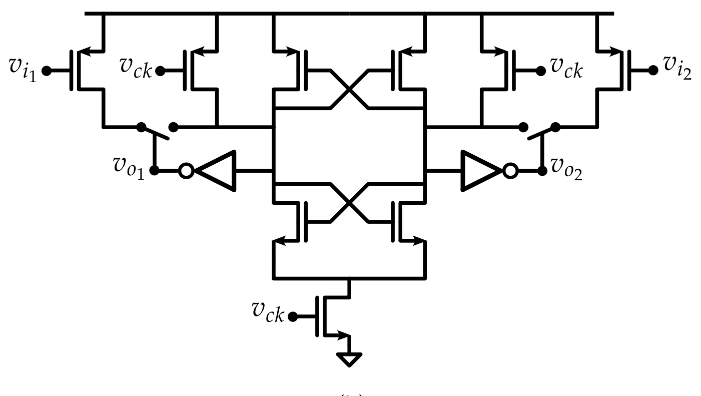


Fig. 3.13 Elements of the comparator: (a) dynamic residual preamplifier and (b) parallel-coupled regenerative latch.

Preamplifier

The first stage of the comparator is a dynamic residual amplifier. Its main purpose is to decouple the fast transitions of the comparator outputs from the highly sensitive inputs. It works by unbalancing the current flowing in two identical, capacitively loaded branches. The output voltage difference, observed on the capacitors, grows over time, resulting in a time dependent voltage gain. The cross-coupled transistors in the middle of each branch introduce positive feedback, increasing the gain until one branch saturates.

The circuit shown in Fig. 3.14a, requires steady inputs, which the capacitive DAC is able to provide. They are applied to the differential couple M_1, M_2, which charges the parasitic capacitance C_l of two clocked MOS transistors M_5, M_6. When the clock signal is active, the output nodes are shorted to ground, removing any memory of the previous cycle. Releasing the clock, the capacitances between drain and ground become the load for each branch (Fig. 3.14).

The cross-coupled pair (M_3, M_4), is initially current-biased and contributes a little gain. Its most significant effect is the positive feedback introduced during the regenerative phase, turning the weakest branch off and resulting in further amplification.

Assumptions in the analysis are:

- quadratic MOSFET model, neglecting channel length modulation

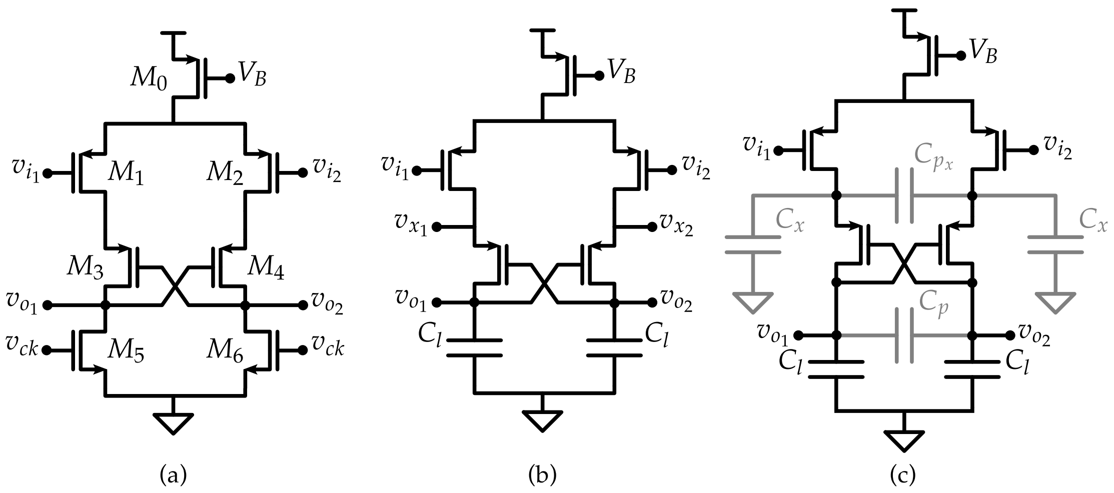


Fig. 3.14 (a) Preamplifier circuit. (b) Equivalent circuit in the active phase. (c) Equivalent circuit with parasitic capacitances.

- exact symmetry of the circuit (mismatches will be considered separately)
- small input differential voltage, as this is the most critical operating condition for the amplifier
- parasitic capacitive loading at the inner nodes $X_{\{1,2\}}$, as well as across the branches

Under these assumptions, the circuit can be linearized around its DC operating point, obtaining Common-Mode (CM) and Differential-Mode (dm) equivalent circuits. Given a couple of signals y_1 and y_2, they can be described in an equivalent form as:

$$y_1 = y^{\mathrm{CM}} + \frac{y^{\mathrm{dm}}}{2} \qquad \text{where} \qquad y^{\mathrm{CM}} = \frac{y_1 + y_2}{2}$$
$$y_2 = y^{\mathrm{CM}} - \frac{y^{\mathrm{dm}}}{2}, \qquad\qquad y^{\mathrm{dm}} = y_2 - y_1.$$

The analysis is simplified if the branches are decoupled, removing the crossed connection involving M_3 and M_4 in such a way as to preserve the behaviour of the original circuit.

In the initial phase of linear output voltage growth, assuming all devices in saturation, the crossed couple acts as a voltage shifter from the output on one branch, to the X node on the opposite branch. Therefore it can be represented as an equivalent voltage source V_{sh} acting on a single branch, with a proper value. Let us first write down the voltage at nodes X:

$$
\begin{aligned}
v_{x_1} &= v_{o_2} + V_{thp} + \sqrt{\frac{2i_1}{\beta_{xcp}}} \\
v_{x_2} &= v_{o_1} + V_{thp} + \sqrt{\frac{2i_2}{\beta_{xcp}}}
\end{aligned}
$$

We want to express it in terms of the output voltage on the same branch:

$$
\begin{aligned}
v_{x_1} &= v_{o_1} + V_{sh_1} \\
v_{x_2} &= v_{o_2} + V_{sh_2}
\end{aligned}
$$

Solving for V_{sh} and linearizing the square root of the current, considering a small differential voltage v_i^{dm} applied to the input of the amplifier, we obtain

$$
\begin{aligned}
V_{sh_1} &= v_{o_2} - v_{o_1} + V_{\mathrm{thp}} + \sqrt{\frac{2i_1}{\beta_{\mathrm{xcp}}}} \\
&= v_o^{\mathrm{dm}} + V_{\mathrm{thp}} + \sqrt{\frac{I_B}{\beta_{\mathrm{xcp}}}}\left(1 + \frac{g_m^{\mathrm{diff}} v_i^{\mathrm{dm}}}{I_B}\right) \\
&= V_{\mathrm{thp}} + \sqrt{\frac{I_B}{\beta_{\mathrm{xcp}}}} + v_o^{\mathrm{dm}} + \sqrt{\frac{\beta_{\mathrm{diff}}}{\beta_{\mathrm{xcp}}}} v_i^{\mathrm{dm}} \\
V_{sh_2} &= V_{\mathrm{thp}} + \sqrt{\frac{I_B}{\beta_{\mathrm{xcp}}}} - v_o^{\mathrm{dm}} - \sqrt{\frac{\beta_{\mathrm{diff}}}{\beta_{\mathrm{xcp}}}} v_i^{\mathrm{dm}} \qquad (3.2)
\end{aligned}
$$

In these expressions, V_{th} is the threshold voltage of the devices, I_B the bias current determined by M_0 and β the transconductance of the MOS devices.

The first two terms in the result belong to the common mode, the last two are instead differential terms. Since the expressions are mutually-decoupled, the CM-DM half circuits can be easily obtained (Fig. 3.15). In the first, parasitic 'across' capacitances C_p and C_{p_x} see the same voltage on both terminals, therefore they can be safely removed. In the latter, those same capacitances see the positive differential voltage on one side, and the negative one on the other, therefore, using a single branch, the equivalent capacitive loading has to double.

Linear growth regime

Common mode: Applying only the common mode input voltage, the differential

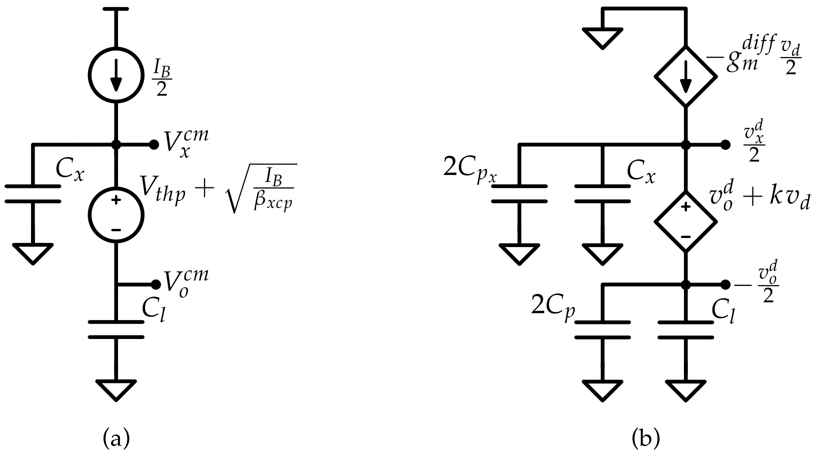


Fig. 3.15 Preamplifier equivalent half-circuits during the linear growth phase. (a) CM circuit (b) DM circuit.

couple injects half of the bias current in each branch. The crossed couple introduces a voltage shift of value $V_{\mathrm{thp}} + \sqrt{I_B/\beta_{\mathrm{xcp}}}$ between the output and the inner node. The output voltage is then

$$V_o^{\mathrm{CM}}(t) \simeq \frac{I_B}{2(C_x + C_l)}t.$$

The linear increase in voltage goes on until a time t_1, when the differential couple leaves saturation. Assuming $v_o^{\mathrm{dm}}(t_1)$ still small compared to V_o^{CM}, the time t_1 can be approximated by imposing the condition $V_x^{\mathrm{cm}}(t_1) = V_{\mathrm{in}}^{\mathrm{CM}} + V_{\mathrm{thp}}$, obtaining

$$t_1 \simeq \frac{V_{\mathrm{in}}^{\mathrm{CM}}}{I_B}2(C_x + C_l),$$

where the overdrive voltage of the crossed couple has been neglected and the threshold voltages of the crossed and differential couples have been considered equal. Under the same assumption of small $v_o^{\mathrm{dm}}(t_1)$, then both branches of the differential couple have similar voltages, therefore they leave saturation almost at the same time.

Differential mode: In differential mode, the input transistors are ground-referred transconductors, being their source terminal equivalent to virtual ground (Fig. 3.15b). The crossed couple, according to (3.2), is modeled as a voltage-controlled voltage source, depending on both the input and output differential voltages. This substitution unveils the origin of the positive feedback. While the output changes as $-v_o^{\mathrm{dm}}/2$, the inner node goes in the opposite direction, being $+v_o^{\mathrm{dm}}/2 + kv_i^{\mathrm{dm}}$. If v_{o_1} is the lowest output voltage, v_{x_1} is the highest inner voltage, weakening the differential couple on its branch and further decreasing the growth of v_{o_1}. The differential output voltage is then

$$v_o^{\mathrm{dm}}(t) = \frac{g_m v_i^{\mathrm{dm}}}{C_l - C_x + 2(C_p - C_{p_x})} t$$

At time t_1, when the linear growth ends, its value is

$$\begin{aligned} v_o^{\mathrm{dm}}(t_1) &\simeq g_m v_i^{\mathrm{dm}} \frac{V_{\mathrm{in}}^{\mathrm{CM}}}{I_B} \frac{2(C_l + C_x)}{C_l - C_x} \\ &\simeq 2 g_m v_i^{\mathrm{dm}} \frac{V_{\mathrm{in}}^{\mathrm{CM}}}{I_B} \end{aligned}$$

where last approximation holds if C_x is small compared to C_l.

Regenerative regime

Let us reconsider the original circuit (Fig. 3.14a) with a trioded differential couple. The effects of v_i^{dm} can be neglected and the couple can be modeled as two equal-valued resistors R acting as source degeneration for the crossed pair. The differential voltage across the internal nodes is then

$$\begin{aligned} v_x^{\mathrm{dm}} &= v_o^{\mathrm{dm}} + \sqrt{\frac{I_b}{\beta_{\mathrm{xcp}}}} \left(-\frac{i_1 - i_2}{I_B} \right) \\ &= v_o^{\mathrm{dm}} - \sqrt{\frac{I_b}{\beta_{\mathrm{xcp}}}} \frac{v_x^{\mathrm{dm}}}{RI_B} \\ &= \frac{v_o^{\mathrm{dm}}}{1 + \sqrt{\frac{I_b}{\beta_{\mathrm{xcp}}}} \frac{v_x^{\mathrm{dm}}}{RI_B}} \\ &\simeq \frac{g_m^{\mathrm{xcp}} R}{1 + g_m^{\mathrm{xcp}} R} v_o^{\mathrm{dm}} \end{aligned}$$

Being the resistance in the order of $10\,\mathrm{k}\Omega$, and the transconductance a few to tens of µS, v_x^{dm} is much smaller than v_o^{dm}. Neglecting it, the sources of the crossed couple are both at virtual ground. The equivalent circuit then is the one in Fig. 3.16a.

Transistors M_3 and M_4 act as transconductors, driven by the voltage on the opposite branch. Since the output voltages at the end of linear growth are slightly unbalanced, the currents through the branches, depending on the conduction of the

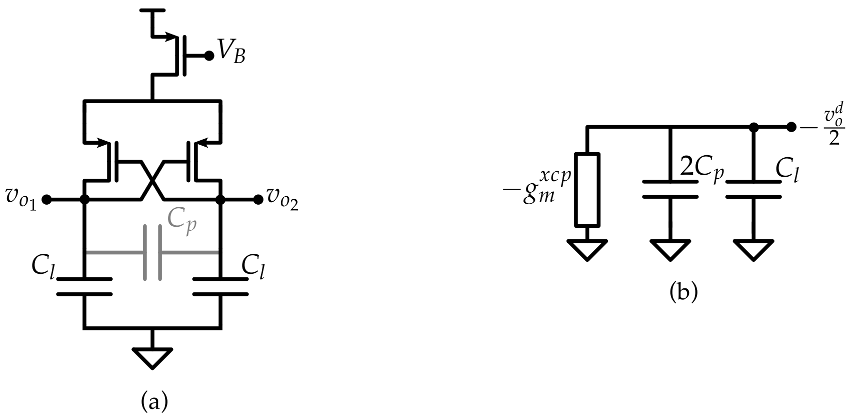


Fig. 3.16 Preamplifier in the regeneration phase.

cross-coupled transistors, are unbalanced, too. For simplicity, let us assume the transconductance to be constant.

Since the control voltage of each transconductor is the output on the other branch, the transistors behave as a negative conductance, as shown in Fig. 3.16b. The time constant of the circuit is negative, resulting in an exponential growth of the differential output voltage:

$$v_o^{\mathrm{dm}}(t) = v_o^{\mathrm{dm}}(0)\exp\left(\frac{g_m^{\mathrm{xcp}}}{C_l + 2C_p}t\right)$$

Being the total capacitance relatively small, the transient immediately determines a voltage sufficient to turn one branch off (by reducing the source/drain voltage) and bring the transistor on the other side into triode region. Such a condition can be expressed as $v_o^{\mathrm{dm}}(t) = V_{\mathrm{thp}}$ and it is reached after

$$\Delta t = \frac{C_l + 2C_p}{g_m^{\mathrm{xcp}}}\ln\frac{V_{\mathrm{thp}}}{v_o^{\mathrm{dm}}(t_1)}$$

In reality the transconductance decreases rapidly during the transient, causing the transient to last longer than predicted.

The fundamental result is that even a small v_o^{dm} is rapidly amplified to a value close to one threshold voltage, large enough to make the second stage of the comparator insensitive to mismatches. To achieve that, we have to guarantee the correct sign of $v_o^{\mathrm{dm}}(t_1)$ with respect to v_i^{dm}, thus enforcing constraints on the asymmetries that can be tolerated.

Saturation

After regeneration only one branch is still active and receive the entire bias current. Until the tail transistor is in saturation, both CM and dm output voltages continue to rise linearly; when also M_0 enters triode, an exponential transient (the resistive tail charging the capacitive load) ends the evolution.

At the very end of the transient, one voltage reaches the supply while the other is still close to $V_{\text{in}}^{\text{CM}}$. If the latch that follows is turned on after this transient has completed, then only one of its input transistors will be on, causing a huge unbalance and making sure the decision of the regenerative circuit goes in the expected direction.

Signal waveforms

Fig. 3.17 shows the most significant waveforms describing the operations of the preamplifier. The bottom plot represents the differential input voltage applied to the circuit. Its values are $\pm$ 5 mV in the two halves of the simulation.

The preamplifier is activated just after instants 0 ns and 250 ns and it is reset after 200 ns and 450 ns. Looking at the v_{o_1} and v_{o_2} waveforms, it is clear how their difference increases over time until a point where the two curves diverge (regeneration, around 90 ns and 340 ns). As already remarked before, on the branch of the highest output voltage, v_x is lower.

The common mode output voltage increases steadily towards the supply, with a slope that decreases slightly at the end of the transient. A reasonable approximation might be to consider it constant, therefore having a lower bound on the duration of the transient. The differential mode output voltage is also varying monotonically.

Regenerative latch

The output of the preamplifier is coupled to the regenerative latch through a couple of parallel transistors that unbalance the internal nodes of the latch by injecting unequal currents.

Modeling the inverters as constant transconductances G_m driven by the voltage on the opposite branch and capacitively loaded, the behaviour of the output voltages

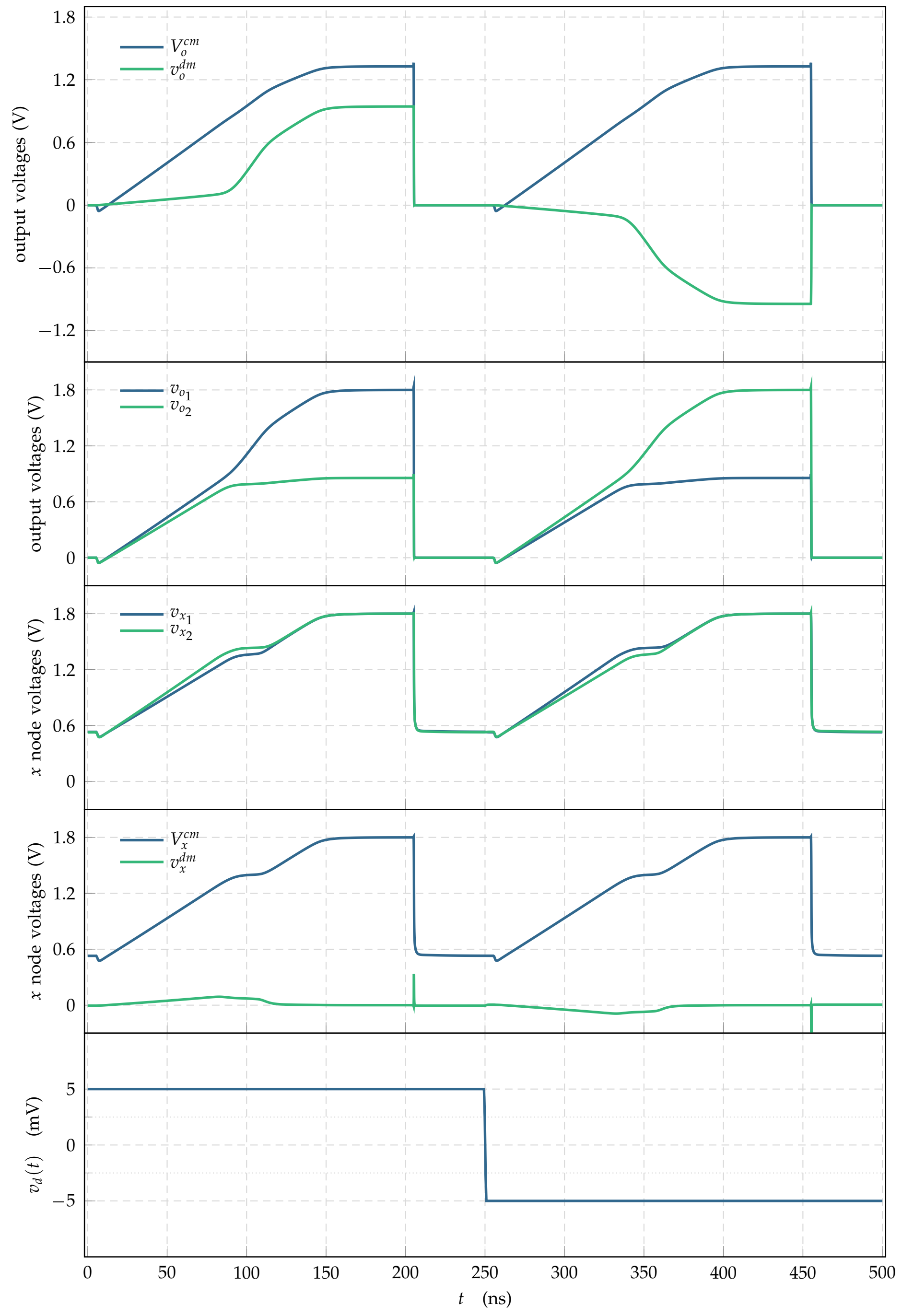


Fig. 3.17 Preamplifier simulated waveforms. Two transients with opposite input differential voltage.

can be derived. Performing the analysis with respect to the common and differential modes, the descriptive equations are:

$$V_o^{\mathrm{CM}} = V_o^{\mathrm{CM}}(0)\exp\left(-\frac{G_m}{C_l}t\right) + \frac{I^{\mathrm{CM}}}{G_m}\left[1 - \exp\left(-\frac{G_m}{C_l}t\right)\right]$$
$$v_o^{\mathrm{dm}} = v_o^{\mathrm{dm}}(0)\exp\left(\frac{G_m}{C_l + 2C_p}t\right) + \frac{i^{\mathrm{dm}}}{G_m}\left[1 - \exp\left(\frac{G_m}{C_l + 2C_p}t\right)\right]$$

The result shows a CM voltage that decays exponentially, while the dm signal grows with the same time constant, up to the point where the devices change operating region. The differential output response, ideally, should only stem from the injected currents. However, the fact of having an exponential growth implies careful evaluation of possible noise injected in the output nodes.

An interesting perspective on the evolution of the CM and dm output voltages of a simpler regenerative latch is provided [58]. The model for the latch analyzed here has been compared to the more simple one, however, the results have not been included. What has been observed is that the curves get closer to the middle of the plot, as the injected current speeds up the transient of both modes.

3.4.5 Coefficient memory and SAR logic

The need to constrain the number of digital control pins has driven us to include a digital memory block into the system. Its logical layout is depicted in Fig. 3.18. The memory holds the coefficient of two independent CS encoders. Focusing on the left half, the small blocks on the top hold two bits controlling the delay elements along the latch clock signal. The rows just below contain the configuration of the input modulators (left-hand side) and the capacitive sampling cells (right-hand side).

The memory element can be populated through a serial stream that gradually fills its content, entering from the top as shown in Fig. 3.18a. This enables the possibility of programming exclusively the rows to be used during acquisition, speeding up the initialization phase.

Once the memory is programmed, linear indexing is implemented through an address counter, which scans through the memory to make use of the stored coefficients. Up to 32 `ACQ_CYCLES`, i.e., acquisition cycles, can be programmed representing, according to the required functionality, any condition from 32 acquisitions-then-conversion, to 4 8-sample acquisitions with conversion of the 4 measurements.

Through serial programming, few I/O pins are needed:

- `CLK`: clock one more serial bit in (write mode) and increment the address counter (read mode).
- `DIN`: serial input to program the memory.
- `WRITE_EN_n`: enable memory programming (when `'0'`) or data output to the encoders (when `'1'`)
- `CNT_RESET_n`: reset the address counter to reuse the memory coefficients
- `DOUT`: verify the memory works correctly by overfilling it and checking the output stream, which should be identical to the input one, although delayed

the latter allows verification that data can propagate through the entire memory, seen as a shift-register during programming.

A few possible configurations of the coefficient memory are shown below. The first behavior uses the CS encoder as a standard SAR ADC, the entire capacitive array is used to simultaneously sample the input[2]. After sampling, the A/D conversion can be triggered, by appropriately driving the chip I/Os. Resetting the address counter of the memory, the same configuration is reused over and over.

```
-- Behavior: Sample all inputs simultaneosly, without inversion,
--           then convert and repeat
--          INPUT_CHANNEL_STATE        SAMPLING_CELL_STATE
-- ACQ_CYCLE   0   1   2   3      0   1   2   3   4   5   6   7
-- =============================================================
-- #1          0   0   0   0      1   1   1   1   1   1   1   1
```

This second snippet shows how the converter can alternate between two different configurations, with a different state of the input multiplexers. The first input is acquired, then converted, if the address counter is not reset, during the next cycle we will use the following row, with a different state of the input muxes. The newly acquired sample is converted and the address counter is now reset, restarting from `ACQ_CYCLE #1`.

[2]Multiple channels are available, but if they can potentially be driven by the same signal. If the input channels are driven independently, this already returns a CS encoding measurement

```
-- Behavior: Sample all inputs simultaneosly, inverting channels
--           2 and 3, then convert. Next samples to be acquired
--           by inverting channels 1 and 2, then convert and
--           repeat
--          INPUT_CHANNEL_STATE        SAMPLING_CELL_STATE
-- ACQ_CYCLE   0   1   2   3      0   1   2   3   4   5   6   7
-- ============================================================
-- #1          0   0   1   1      1   1   1   1   1   1   1   1
-- #2          1   1   0   0      1   1   1   1   1   1   1   1
```

Finally, true CS encoder operations, collecting four input samples over time, while changing the input modulation.

```
-- Behavior: Capture four samples in different time instants,
--           inverting the first and last sample, then convert
--           and repeat
--          INPUT_CHANNEL_STATE        SAMPLING_CELL_STATE
-- ACQ_CYCLE   0   1   2   3      0   1   2   3   4   5   6   7
-- ============================================================
-- #1          1   0   0   0      1   1   0   0   0   0   0   0
-- #2          0   0   0   0      0   0   1   1   0   0   0   0
-- #3          0   0   0   0      0   0   0   0   1   1   0   0
-- #4          0   0   0   1      0   0   0   0   0   0   1   1
```

The digital memory has been described in VHDL, then synthesized and placed in the design with the so called Analog-on-Top [], methodology, where the design phase is schematic-driven as the amount of digital logic within the design is limited[3].

The logic driving the evolution of the SAR algorithm has been also synthesized from a VHDL description. The corresponding block diagram is depicted in Fig. 3.19. It is based on a thermometric encoding, implemented by the chain of flip-flops, denoted as `thermo`, with a `'1'` input. The comparator output, representing the different bit values are stored at the boundary where the thermometric encoding goes from `'1'` to `'0'`, through gate `A1`. The bit value then drives the connection of the

[3]this holds true in our case, even if the digital memory occupies more than a third of the entire layout area

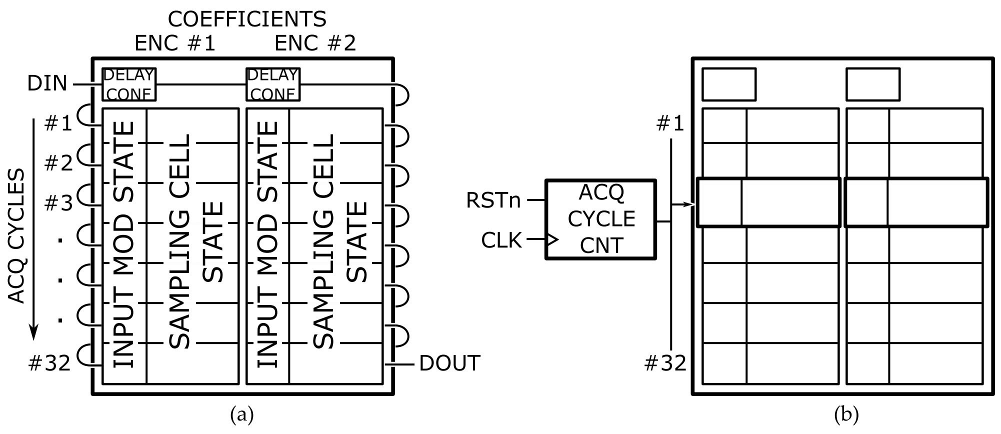


Fig. 3.18 Memory layout and its use during (a) write and (a) read operations.

converter switches for the subsequent evaluation, through a state decoder, whose details are depicted in the enlarged inset.

3.5 Layout

Fig. 3.20 depicts the layout of the entire chip. It highlights the division in two identical half, each having its own digital memory and two independent CS encoders. A more detailed view of a CS encoder is shown in Fig. 3.21, where the elementary blocks are highlighted. A few relevant aspects can be noted.

Looking at the structure of the capacitive arrays, in Fig. 3.21b, dummy elements are placed on the outer boundaries so that every unitary capacitance sees the same surroundings. A common centroid layout has been employed to try to cancel variability due to linear gradients, at least on the scaled section of the array. As the selected switch topology requires ad-hoc driving logic, being able to place it close to the capacitive cells has driven us to place it below the capacitive cells. Since this usually leads both to variability of the geometrical features of the plates, as well as electromagnetic coupling, the capacitors above digital logic have been used as supply and reference decoupling.

The layout of the dynamic comparator, including its bias resistor and the latch delay selection logic is shown in Fig. 3.22 The input differential couple of the its preamplifier has been constructed by splitting each transistor in two fingers, placed in a common-centroid structure. In Section 4.3 we will indeed motivate this selection

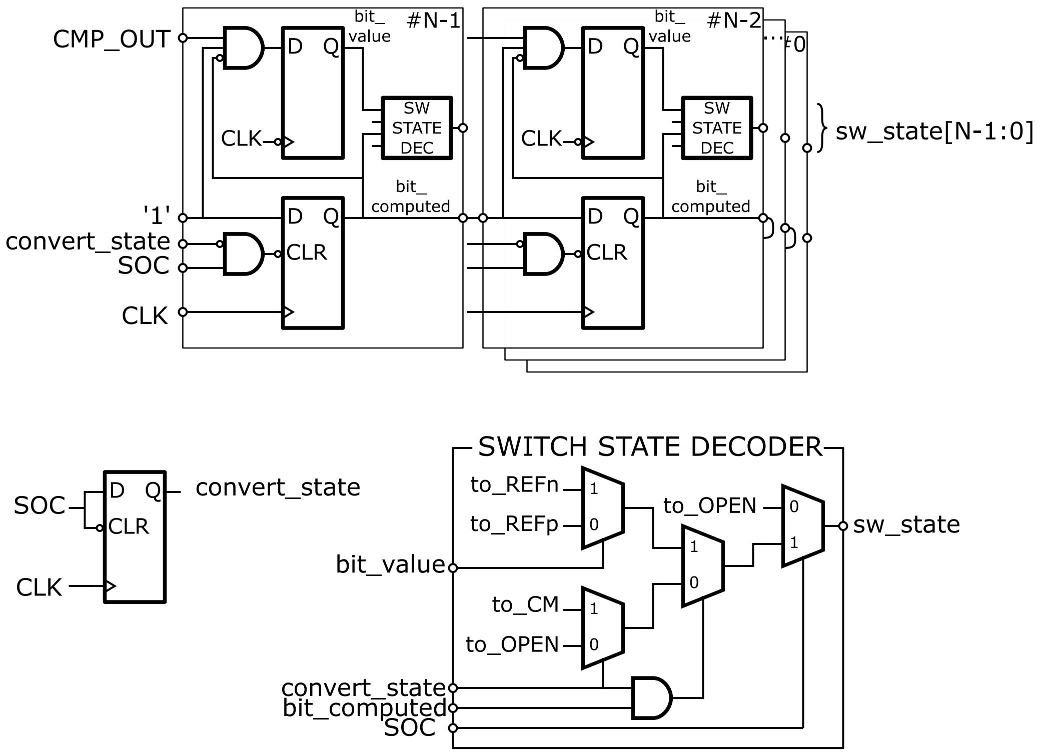


Fig. 3.19 SAR logic block diagram, with a detailed view on the switch-state decoder.

by a theoretical analysis conducted on the effect of interdigitated geometries in terms of matching performance of nominally identical elements.

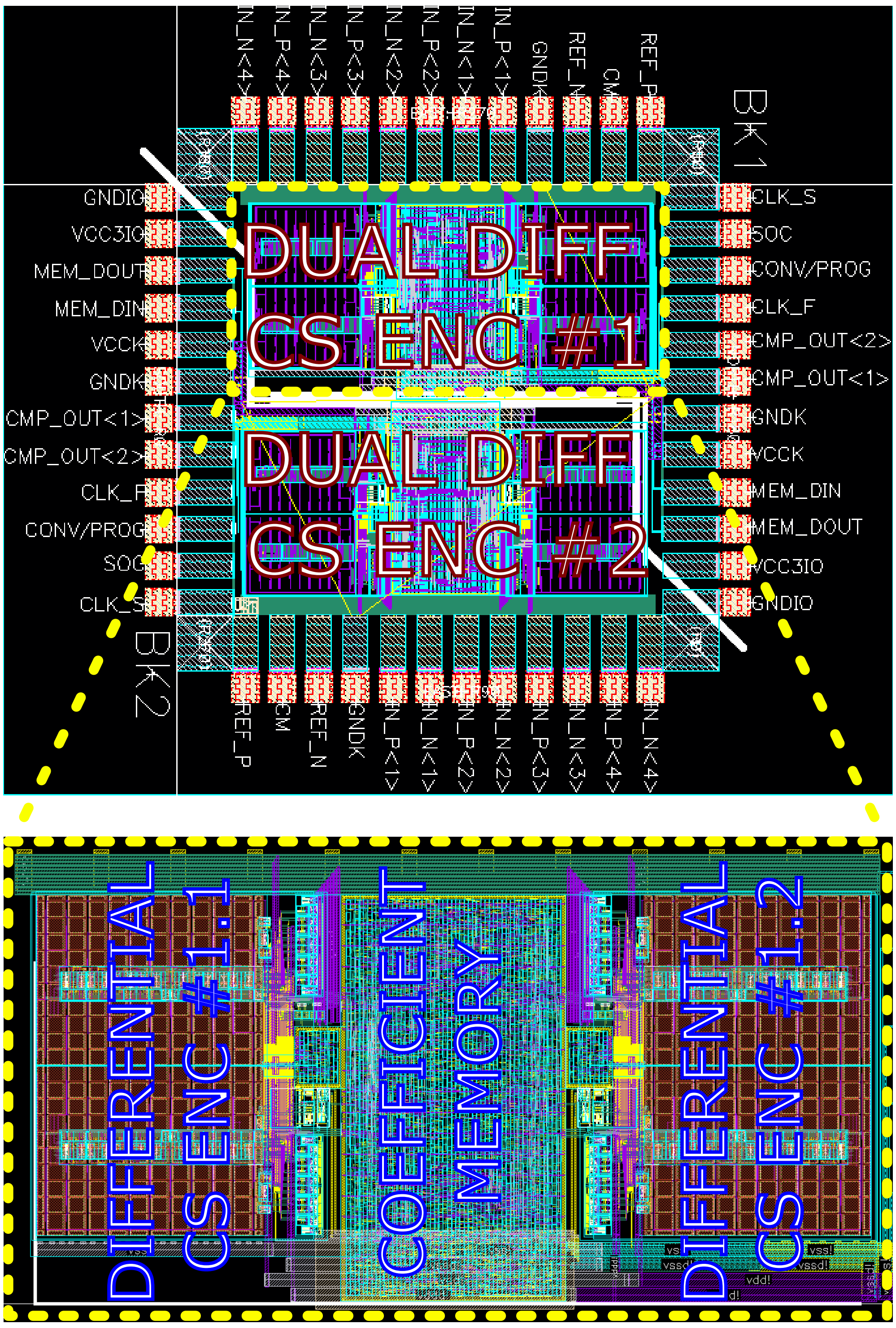


Fig. 3.20 Layout of the entire chip. The zoomed-in view highlights the two independent differential CS encoders whose behavior is defined by the programming of the coefficient memory.

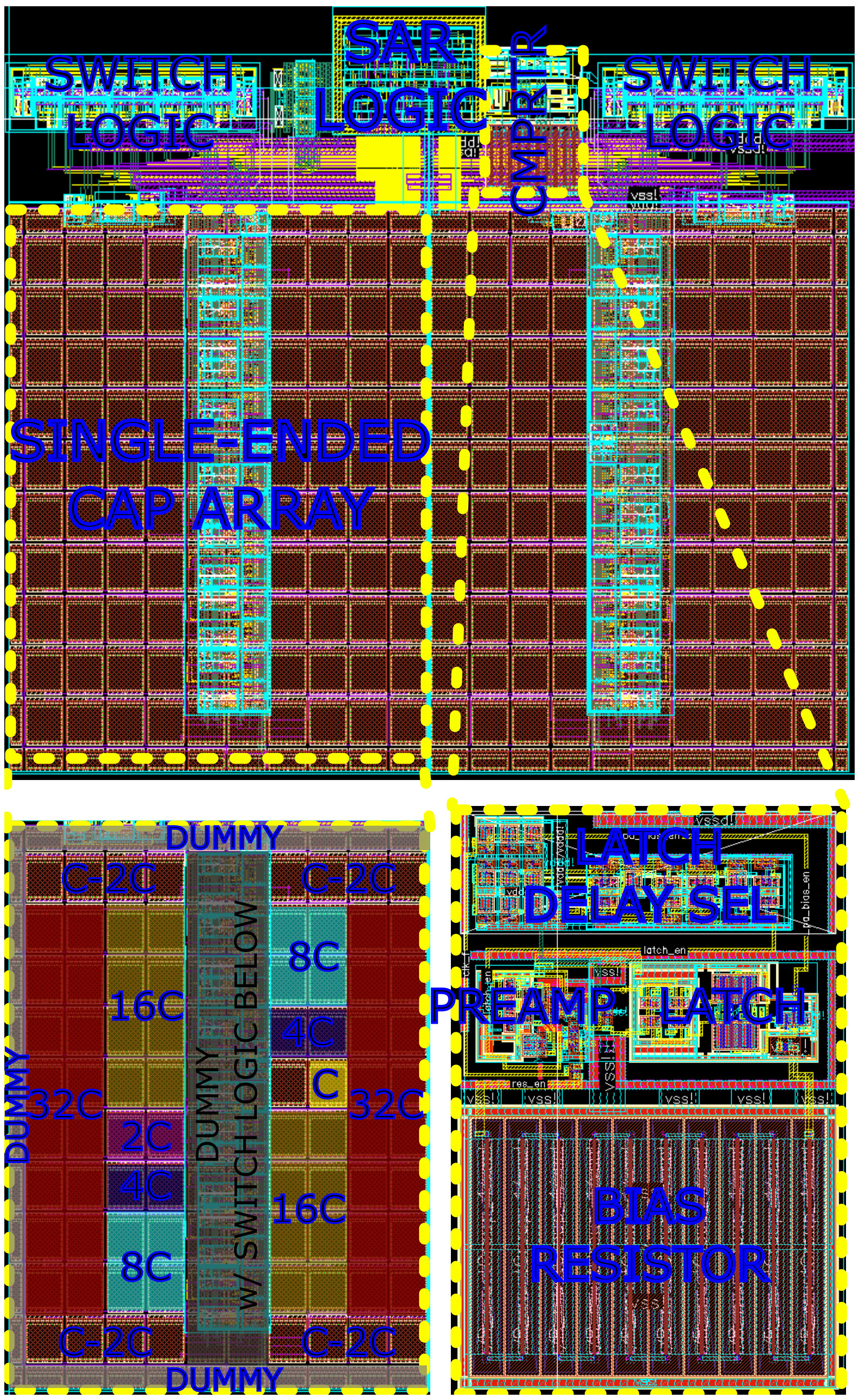


Fig. 3.21 (a) Layout of a single differential CS encoder. (b) Common centroid layout of the capacitor array, with dummy elements on the outer boundary and dummy elements in the middle to allow a closer placement of the driving switches to the capacitive elements. (c) Layout of the dynamic comparator.

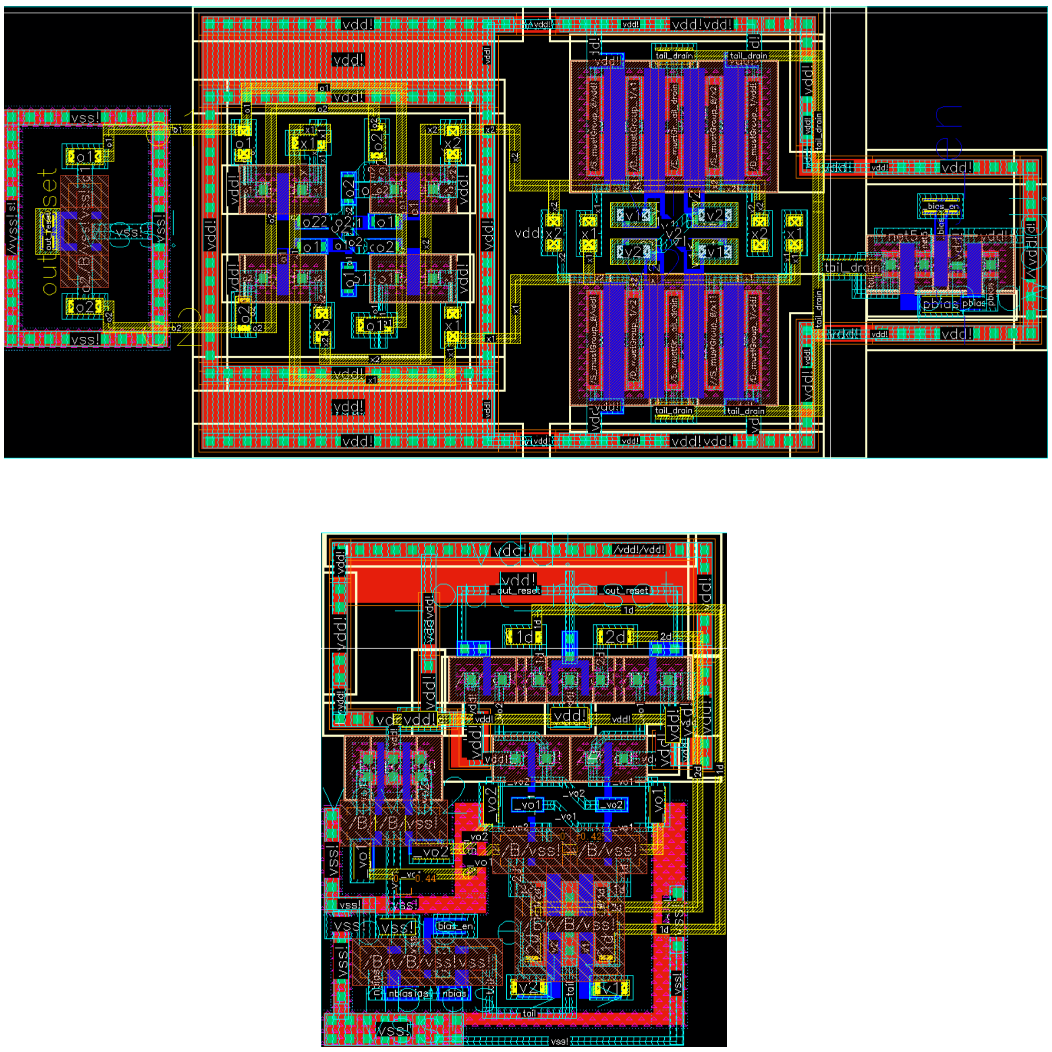


Fig. 3.22 Layout of the (a) preamplifier and (b) latch

3.5.1 Post-layout results

A few results on the post-layout performance of the main circuit blocks within the chip are shown here. Fig. 3.23 depicts the nonlinearity metrics of the capacitive DAC within the CS encoder, separating the contribution of the scaled and C-2C subarrays. As expected, the mid-range transition is the most critical for both structures, limiting the maximum achievable resolution of the block with respect to the nominal one. A possible solution to be employed in a future revision of the chip is a thermometric driving of the capacitive cells. Once the `'01...1'` value has been reached, going to the `'10...0'` with the current driving strategy means switching back all the previously-active capacitive cells, and activating the MSB one. In a thermometric strategy, the active cells are retained, and a single unitary cell is added, reaching the desired value. This effectively removes the jump in DNL observed in the current setup, at the cost of a larger area of the digital logic.

Time-domain operations of the most critical block within the dynamic comparator are shown in Fig. 3.24, together with the computation of the time-dependent voltage gain. The input-referred voltage offset has been computed as $(1.23 \pm 0.01)\,\mathrm{mV}$, independent of the common mode input voltage, which has been varied from 0.7 V to 1.1 V, with the nominal value for our differential implementation equal to 0.9 V.

The global operation of the chip, in a simulation which included both configuration of the coefficients memory, input sampling and A/D conversion is depicted in Fig. 3.25.

At the time of writing, we had just started the process of testing the physical chip, whose manufacturing was delayed multiple times. For this reason, this section does not contain more detailed analysis. An overview of the support PCB is given in the following section, with additional details to be found in [59].

3.6 Testing the chip

The requirements of versatility and reconfigurability of the SAR-AIC testbed drove the selection of the test platform to a Field Programmable Gate Array (FPGA) evaluation board. Indeed, the testbed has been designed as an addon PCB hosting the SAR-AIC, and to be plugged into the General Purpose I/O (GPIO) headers of an

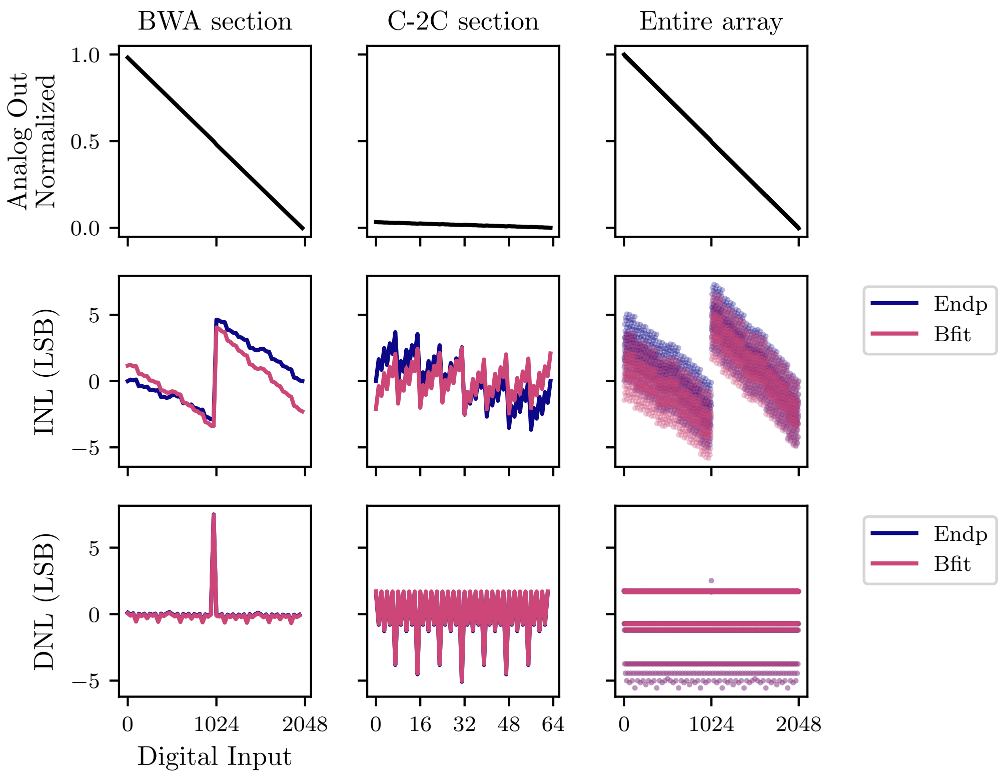


Fig. 3.23 Integral and Differential NonLinearity of the single-ended capacitive DAC used within the CS encoder. The test is performed by running through all the input digital codes. The reference linear characteristic is either through the endpoints of the actual one or a bestfit over the entire dataset.

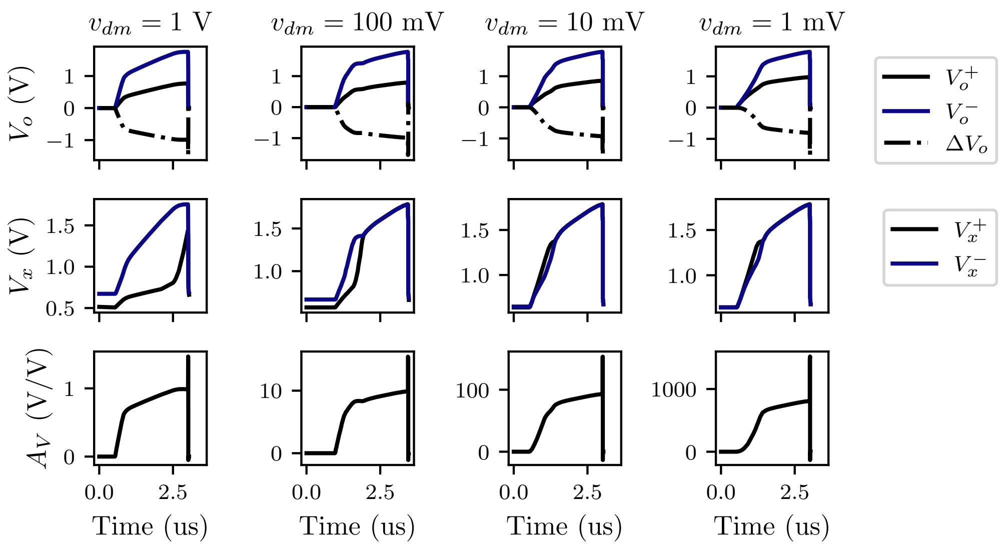


Fig. 3.24 Waveforms of the time-domain operations of the preamplifier within the dynamic comparator for different values of the differential-mode input voltage. The voltages shown maintain the nomenclature defined in Fig. 3.15.

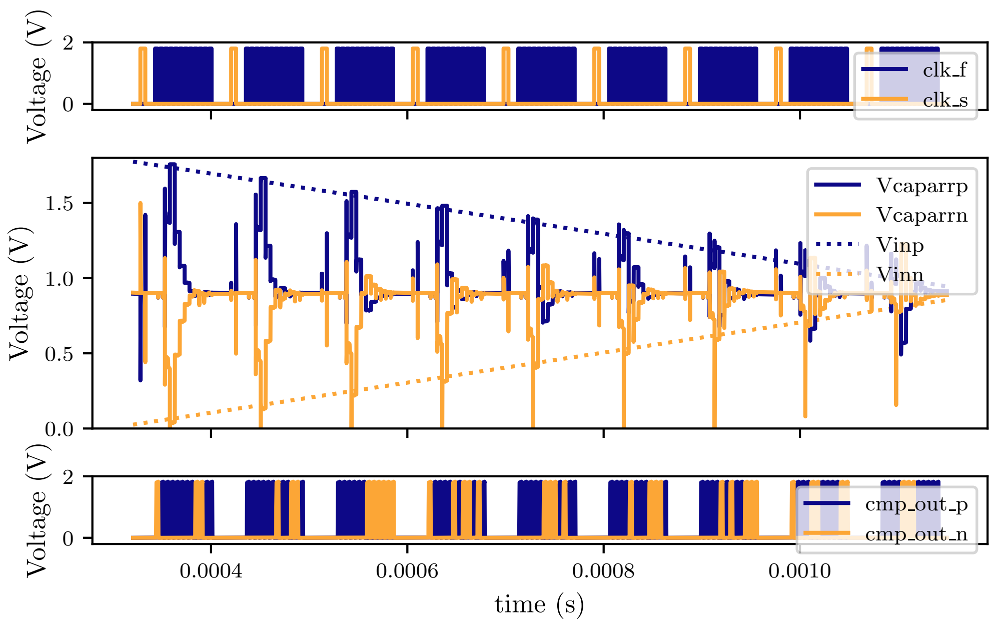


Fig. 3.25 Time-domain operations of the ADC, used as a Nyquist-rate converter

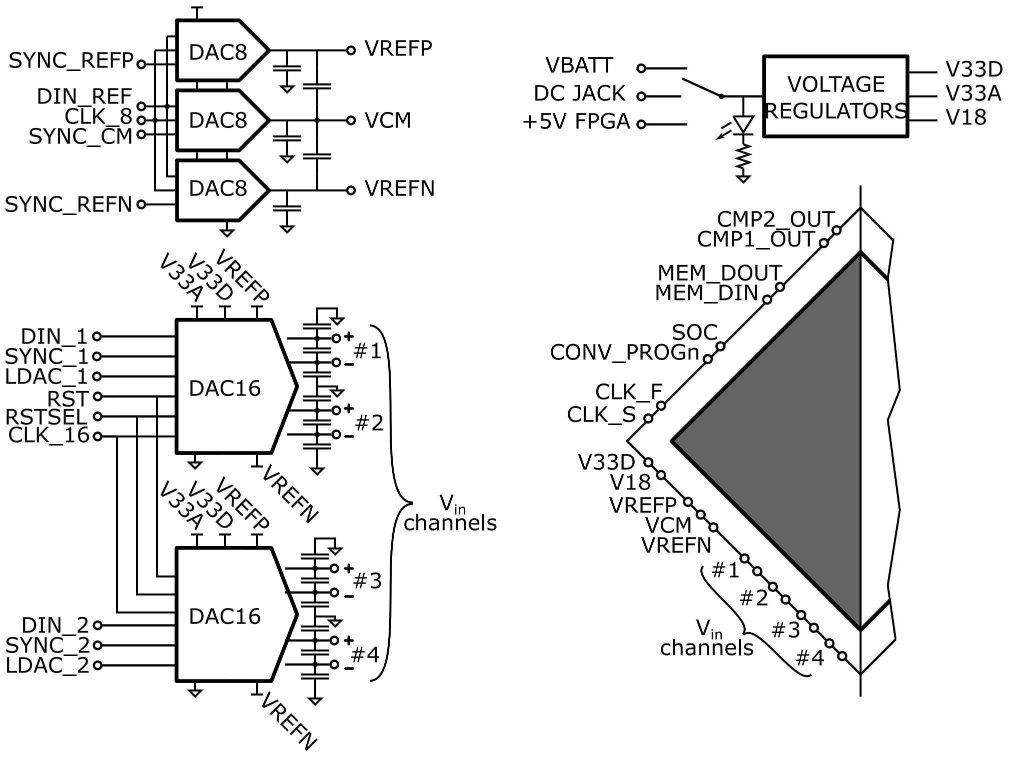


Fig. 3.26 System-level schematic diagram of the test platform. The digital control lines all come from an FPGA evaluation board.

FPGA board. The system-level block diagram of the test platform is illustrated in Fig. 3.26.

Three energy sources have been foreseen: directly from the evaluation board GPIO headers, from a DC-jack or three AAA batteries. From the 5 V `VCC` external power supply, two `ADP3300` regulators generate independent 3.3 V lines, `V33A` and `V33D`. The latter powers the I/O cells of the SAR-AIC. The former powers the DACs onboard the PCB, as well as a 1.8 V `V18` line by means of an `LT3085` regulator.

Three, 8-bit `DAC081` DACs are used to generate the `VREF_P`, `VREF_CM` and `VREF_N` references. They are all provided to the SAR-AIC, which needs them during conversion. `VREF_P` and `VREF_N` are also sourced to the high-resolution signal DACs. The three DACs are controlled through an SPI bus, with shared `SCLK` and `DIN` lines and independent $\overline{\texttt{SYNC}}$. Ideally these would be only configured at the beginning of a test and left as is for the rest of the system operations.

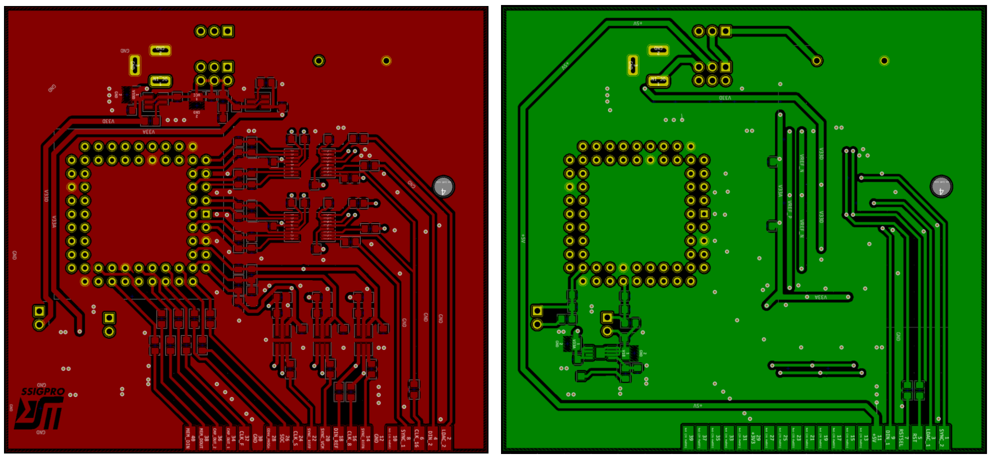


Fig. 3.27 Layout of the support PCB.

Two 16-bit 4-channel `DAC8555` generate the inputs to the SAR-AIC, which expects 4 differential input channels. Using the SAR-AIC in single channel mode, its 4 positive lines would be driven by the same signal, and so will be the negative lines. The most convenient way to operate the 16-bit DACs would then be to use one for the positive signals, and the other for the negative ones. But convenience of routing was preferred, therefore each DAC drives two differential lines. The DACs communicate through an SPI bus, independent from the one of the low-resolution DACs. Since their programming time limits the operating speed of the entire circuit, separate `DIN` lines have been provided for each DAC.

Passive input filters have been placed on the differential signal lines, to reject both common-mode and differential-mode high-frequency noise. Decoupling capacitors have been inserted wherever specified by the technical documentation or needed, to obtain steady, well-behaved signals.

The layout of the discrete components has tried to follow the signal paths as much as possible, as depicted in Fig. 3.27.

Chapter 4

Results on Nonidealities Estimation and Management

Don't step over dollars to pick up a dime.

AUTHOR UNKNOWN

In the previous chapter we have dealt with the proposed architecture, analyzing the intended behavior of its components. From the system level description of a SAR converter used as a CS encoder, down to the transistor level operations of the dynamic comparator. However, to design a circuit that can withstand the unavoidable nonidealities of an actual implementation, the effect of parasitic elements, as well as parameter variations has to be accounted for.

In this section, we will isolate some of the components of the proposed architecture and analyze the most critical deviations to be observed on a physical implementation. We will derive analytical expressions to guide the design process, for a more resilient circuit.

First, the resolution and linearity limits of the capacitive array will be analyzed, observing in particular the effects of parasitics and matching. Then, a proper switch configuration is defined, to minimize their injected charge and leakage currents. Since the ability of the circuit to operate properly relies on the storage of samples for the entire duration of an acquisition window, the stability and mismatch-robustness of a known leakage current compensation circuit is introduced. Its stability margins are derived, so that the absence of unwanted oscillations can be guaranteed. Indeed the deep study of the compensator reveals unexpected flaws that prevent its use in practice, to the point that we decided to not include it in the final circuit.

We will then analyze the dynamic comparator, computing the input-referred offset due to different asymmetries within the circuit.

Going one step further, towards the actual layout of the individual transistors, a general result on the effects of different layout strategies on the mismatch of topologically symmetric devices has been quantified, obtaining approximate expressions of the long-distance parameter mismatch. Instead of the well known mismatch model presented by Pelgrom et al. in [60], we construct our analysis on a more general statistical model already published in the Literature, for it more easily allows us to include the geometry of Manhattan-style structures in the computation.

Finally, the stability properties of a leakage compensation circuit is studied in detail. Though initially considered a robust candidate to solve the leakage-induced issues, our analysis proves its extreme sensitivity to mismatch terms, making it unsuitable for practical applications.

4.1 Parasitics-induced nonlinearities in the C-2C Array

As described in Chapter 3.3, the capacitive array is at the core of the proposed A/I converter. It is used to collect and store several modulated samples of the input signal, combine them and convert the result. All these operations rely on the redistribution of the charge stored in the isolated nodes.

To increase the resolution of the conversion, we have shown that the most efficient solution is to cascade a C-2C sub-array to the smallest scaled capacitor. This structure, however, introduces secondary isolated nodes which are particularly sensitive to injected noise and parasitic loading. Fig. 4.1 shows a 3-bit C-2C structure loading a scaled array, which has been compacted into one single capacitance C_{sc}. The smallest element of the scaled array is considered as belonging to the C-2C structure, so that scaled array starts with a $2C_u$ capacitor. Each isolated node has been named as N_i and all the parasitics have been identified in gray.

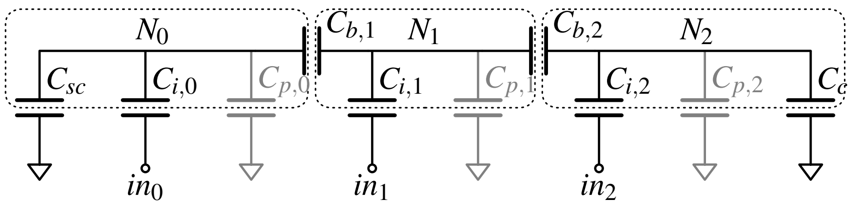


Fig. 4.1 Schematic of a 3-bit C-2C sub-array. The elements are (nominal value in parenthesis): input capacitors C_i (C_u); bridge capacitors C_b ($2C_u$); parasitic capacitors C_p (0); equivalent capacitance of the scaled array C_{sc} ($2C_u + 4C_u + \cdots$); closure capacitance C_c (C_u)

In the following we will construct an approximate analytical model to estimate nonlinearities of the capacitive D/A converter from the amount of parasitic loading at the intermediate nodes.

Analytical modeling of capacitive parasitics

Since the evolution of the SAR A/D algorithm depends on the voltage of the main isolated node N_0, all the issues affecting the array capacitors have to be observed from such node.

Let us consider the parasitic loading observed at every isolated node due to the capacitive coupling from the physical plates to the external environment. Having a unitary capacitance C_u, the amount of parasitics can be expressed as a fraction αC_u. The parameters involved in the analysis are summarized below:

- n_{sc}: Total # of inputs of the scaled section of the array $C_{sc} = 2C_u + 4C_u + \ldots$
- n: Total # of inputs of the C-2C section
- $i = [0 : n-1]$: Index of each array section
- N_i: Isolated nodes
- in$_i$: Input nodes
- $Cp, i = \alpha_i C_u$: Parasitic loading of section i

Note that, for convenience, the smallest capacitor in the scaled array is considered part of the C-2C section. This allows the analysis to be valid also for the limit condition of an independent C-2C structure, i.e., without a scaled section.

When observed from the N_0 node, voltages applied at different C-2C inputs experience an attenuation increasing with its i index, expressed as a power of 2. The presence of non-idealities introduces errors in the nominal attenuation that we will try to estimate analytically.

4.1.1 Algorithmic derivation of the complete model

We want to derive an expression that links the perturbations within the capacitive array to the attenuation from the input, to the N_0 isolated node, the one where the C-2C and the scaled array merge, as well as the one observed by the comparator. Let us define the attenuation between the input node A and the output node B as $R_{A,B}$. We want to obtain the ratios R_{in_i,N_0}, i.e. from the external inputs in_i to the N_0 node.

The R_{in_i,N_0} attenuation can be decomposed in two terms, the R_{in_i,N_i} capacitive partition from input in_i to the closest isolated node N_i, and the R_{N_i,N_0} term. This, in turn, is due to the cascade of attenuations from N_i to N_{i-1} to. Therefore:

$$R_{i,0} \overset{\text{def}}{=} R_{in_i,N_i} R_{N_i,N_{i-1}} R_{N_{i-1},N_{i-2}} \cdots R_{N_1,N_0}. \tag{4.1}$$

Nominal setup

For the ideal, unperturbed structure the partial attenuations terms appearing in (4.1) are shown in Table 4.2 and Table 4.1. The $\oplus$ operator describes the harmonic sum of the two operands, defined as:

$$x \oplus y \overset{\text{def}}{=} \frac{1}{\frac{1}{x}+\frac{1}{y}} = \frac{xy}{x+y}$$

and refers to the series connection of capacitors.

Table 4.1 Attenuation from an input node to the adjacent isolated node. Results shown for the first 4 inputs.

Index	Attenuation	Definition	Result
0	R_{in_0,N_0}	$\frac{C}{2C+C_{sc}}$	$\frac{1}{2^{n_{sc}+1}}$
1	R_{in_1,N_1}	$\frac{C}{C+C+2C\oplus(C+C_{sc})}$	$\frac{1+2^{n_{sc}+1}}{2^{n_{sc}+3}}$
2	R_{in_2,N_2}	$\frac{C}{C+C+2C\oplus(C+2C\oplus(C+C_{sc}))}$	$\frac{1+5\cdot 2^{n_{sc}+1}}{2^{n_{sc}+5}}$
3	R_{in_3,N_3}	$\frac{C}{C+C+2C\oplus(C+2C\oplus(C+2C\oplus(C+C_{sc})))}$	$\frac{1+21\cdot 2^{n_{sc}+1}}{2^{n_{sc}+7}}$

The overall attenuation $R_{i,0}$ defined in (4.1) is given by the product of the index-i row of Table 4.1 by all the rows in Table 4.2 from the first, up to i-th. This cumulative product across the rows results in the cancellation of the denominator in one row

Table 4.2 Attenuation from an isolated node to the adjacent one, moving towards N_0. Results shown for the first 3 nodes.

Index	Attenuation	Definition	Result
1	R_{N_1,N_0}	$\frac{2C}{2C+C+C_{sc}}$	$\frac{2}{1+2^{n_{sc}+1}}$
2	R_{N_2,N_1}	$\frac{2C}{2C+C+2C\oplus(C+C_{sc})}$	$2\frac{1+2^{n_{sc}+1}}{1+5\cdot 2^{n_{sc}+1}}$
3	R_{N_3,N_2}	$\frac{2C}{2C+C+2C\oplus(C+2C\oplus(C+C_{sc})}$	$2\frac{1+5\cdot 2^{n_{sc}+1}}{1+21\cdot 2^{n_{sc}+1}}$

with the numerator of the adjacent one, thus obtaining 2^i divided by the denominator of the last term. In turn, this is exactly equal to the numerator of the row selected in Table 4.1. The overall $R_{i,0}$ attenuation from a generic input to the 0-th isolated node is then:

$$R_{i,0} = \frac{2^i}{2^{n_{sc}+2i}} = \frac{1}{2^{n_{sc}+i}}, \tag{4.2}$$

as one would expect. The weight of each C-2C input decreases as a power of two, with the absolute attenuation depending on how big the scaled array is.

Parasitic-loaded setup

A similar procedure can be used to construct the mathematical description for a nonideal setup. However, obtaining a meaningful result requires us to approximate the terms so that the parasitic loading, which is assumed to be small, is only evaluated up to a first-order deviation.

Considering the modularity of the C-2C structure, the i-th section depicted in Fig. 4.2 becomes the starting point of the analysis. To its left and right-hand sides, the section sees capacitances C_{L_i} and C_{R_i}, respectively. Bridge capacitances C_{b_i} and $C_{b_{i+1}}$, of value $2C_u$ isolate the section, which is loaded by the nominal capacitance $C_{\text{in}_i} = C_u$ and $C_{p_i} = \alpha_i C_u$.

The equations describing the elementary section are:

$$C_{L_i} = \begin{cases} C_{sc} & \text{if } i = 0 \\ (C_{L_{i-1}} + C_{\text{in}_{i-1}} + C_{p_{i-1}}) \oplus C_{bi} & \text{if } i > 0 \end{cases}$$

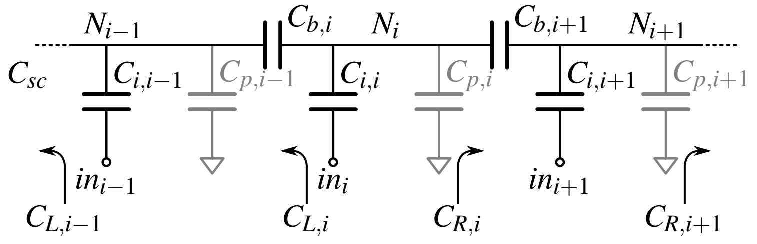


Fig. 4.2 Capacitances in a generic section of a C-2C array

and:

$$C_{R_i} = \begin{cases} (C_{R_{i+1}} + C_{\text{in}_{i+1}} + C_{p_{i+1}}) \oplus C_{b_{i+1}} & \text{if } i < n \\ C_u & \text{if } i = n \end{cases}$$

The $R_{i,0}$ attenuation can then be computed, according to (4.1), through a recursive equation for the adjacent isolated nodes:

$$R_{N_i,N_0} = R_{N_i,N_{i-1}} \frac{C_{b_i}}{C_{b_i} + C_{L_{i-1}} + C_{\text{in}_{i-1}} + C_{p_{i-1}}},$$

leading to:

$$R_{\text{in}_i,N_0} = R_{N_i,N_0} \frac{C_{\text{in}_i}}{C_{in_i} + C_{L_i} + C_{R_i} + C_{p_i}}.$$

The pseudocode describing how to evaluate every term is shown in Algorithm 1. Any symbolic manipulation library can be used to implement it (e.g. sympy for Python).

Algorithm 1: Pseudocode to compute R_{in_i,N_0}

```
Given $C_{\text{in}_i} = 1$, $C_{b_i} = 2$, $C_{p_i} = \alpha_i$, $C_{L_0} = C_{sc}$, $C_{R_n} = 1$, $R_{N_0,N_0}$ = 1
// Backward pass
for i = n-2:0
    compute $C_{R_i}$ according to (4.1.1)
// Forward pass
for i = 1:n-1
    compute $C_{L_i}$ according to (4.1.1)
    compute $R_{N_i,N_0}$ according to (4.3)
    compute $R_{\text{in}_i,N_0}$ according to (4.3)
```

The resulting attenuations will be expressed as fractions whose numerator and denominator can be approximated to first order in α_i, as shown in Table 4.3 for different sizes of the C-2C array, where C_{sc} is normalized with respect to the unitary capacitance, i.e. $\gamma_{sc} = C_{sc}/C_u = 2(2^{n_{sc}} - 1)$.

Table 4.3 First order approximations of the attenuations from input i to node N_0 in the case of (a) $n = 2$, (b) $n = 3$ and (c) $n = 4$

i	R_{in_i,N_0}
0	$\frac{4+\alpha_1}{8+4C_{sc}+4\alpha_0+3\alpha_1+C_{sc}\alpha_1}$
1	$\frac{2}{8+4C_{sc}+4\alpha_0+3\alpha_1+C_{sc}\alpha_1}$

(a)

i	R_{in_i,N_0}
0	$\frac{16+4\alpha_1+5\alpha_2}{32+16C_{sc}+16\alpha_0+12\alpha_1+11\alpha_2+4C_{sc}\alpha_1+5C_{sc}\alpha_2}$
1	$\frac{8+2\alpha_2}{32+16C_{sc}+16\alpha_0+12\alpha_1+11\alpha_2+4C_{sc}\alpha_1+5C_{sc}\alpha_2}$
2	$\frac{4}{32+16C_{sc}+16\alpha_0+12\alpha_1+11\alpha_2+4C_{sc}\alpha_1+5C_{sc}\alpha_2}$

(b)

i	R_{in_i,N_0}
0	$\frac{64+16\alpha_1+20\alpha_2+21\alpha_3}{128+64C_{sc}+64\alpha_0+48\alpha_1+44\alpha_2+43\alpha_3+16C_{sc}\alpha_1+20C_{sc}\alpha_2+21C_{sc}\alpha_3}$
1	$\frac{32+8\alpha_2+10\alpha_3}{128+64C_{sc}+64\alpha_0+48\alpha_1+44\alpha_2+43\alpha_3+16C_{sc}\alpha_1+20C_{sc}\alpha_2+21C_{sc}\alpha_3}$
2	$\frac{16+4\alpha_3}{128+64C_{sc}+64\alpha_0+48\alpha_1+44\alpha_2+43\alpha_3+16C_{sc}\alpha_1+20C_{sc}\alpha_2+21C_{sc}\alpha_3}$
3	$\frac{8}{128+64C_{sc}+64\alpha_0+48\alpha_1+44\alpha_2+43\alpha_3+16C_{sc}\alpha_1+20C_{sc}\alpha_2+21C_{sc}\alpha_3}$

(c)

Assuming $\alpha_j = \alpha$ for each j, the numerical coefficients can be collected and their patterns exploited. The overall contribution can be expressed by a simple formula dependent only on n and i, resulting in a description of the effects of parasitics for a generic size of the array and from any input, just as desired. Expressing each fraction in Table 4.3 as

$$R_{in_i,N_0} = \frac{U_i + \alpha \cdot u_i}{V_i + \alpha \cdot v_i + C_{sc}(Z_i + \alpha \cdot z_i)},$$

where U, V and Z represent the nominal terms and u, v and z the first-order error coefficients. The nominal terms are equal to

$$U_i = \frac{4^{n-1}}{2^i}$$
$$V_i = 2 \cdot 4^{n-1}$$
$$Z_i = 4^{n-1}$$

resulting in the nominal value of the attenuation already found in (4.2):

$$R^{nom}_{in_i,N_0} = \frac{U_i}{V_i + C_{sc} \cdot Z_i} = \frac{1}{2 + C_{sc}} \frac{1}{2^i} = \frac{1}{2^{n_{sc}+1}} \frac{1}{2^i}. \tag{4.3}$$

Concerning the coefficients of the errors, through the analysis of the patterns we can compute their closed-form expression:

$$\begin{aligned} u_i &= \frac{4^{n-2}}{2^i} \sum_{j=0}^{n-i-2} \frac{1}{4^j}(n-i-j-1) = \frac{2^i}{9}\left[1 + 4^{n-i}\left(\frac{3}{4}(n-i)-1\right)\right] \\ v_i &= \quad -\sum_{j=1}^{n-1} j4^{j-1} + n4^{n-1} \qquad = \frac{1}{9}\left[\left(\frac{3}{2}n+1\right)4^n - 1\right] \\ z_i &= \quad \sum_{j=1}^{n-1} j4^{j-1} \qquad = \frac{1}{9}\left[\left(\frac{3}{4}n-1\right)4^n + 1\right] \end{aligned} \tag{4.4}$$

Using Taylor's expansion of R_{in_i,N_0} for small α:

$$R_{in_i,N_0} \simeq R^{nom}_{in_i,N_0}\left(1 + \frac{\alpha u_i}{U_i}\right)\left(1 - \frac{\alpha v_i}{V_i + C_{sc}Z_i} - \frac{\alpha C_{sc} z_i}{V_i + C_{sc}Z_i}\right) \tag{4.5}$$
$$= R^{nom}_{in_i,N_0}\left[1 + \alpha\left(\frac{u_i}{U_i} - \frac{v_i + C_{sc}z_i}{V_i + C_{sc}Z_i}\right)\right] \tag{4.6}$$

Plugging (4.4) into (4.6) we obtain:

$$\begin{aligned} &\frac{4}{9}\frac{4^i-1}{4^n} - \frac{1}{3}\left(i - \frac{1-4^n}{4^n}\frac{4}{C_{sc}+2}\right) \\ &\qquad = \frac{4}{9}\frac{4^i-1}{4^n} - \frac{1}{3}\left(i - \frac{1-4^n}{4^n}\frac{1}{2^{n_{sc}-1}}\right) \\ &\qquad \simeq \frac{4}{9}\frac{4^i-1}{4^n} - \frac{1}{3}\left(i + \frac{1}{2^{n_{sc}-1}}\right) \qquad \text{if(n > 2)} \end{aligned}$$

These attenuations can be used to compute a few notable values, shown in Table 4.4, as well as to compute the converter output for any digital input value, as shown in the top left subplot of Fig 4.3.

Relevant metrics for the static performance of D/A converters are its offset and gain errors. While the former is 0, the latter can be computed from the error observed at a digital input of $2^n - 1$ (last row of the Table 4.3. The error is proportional to α, the amount of parasitic loading, since the C-2C sub-array weighs more than the unitary capacitor it is originally replacing.

As a consequence, nonlinearity metrics have to be computed by first compensating the potentially significant gain error. Two common strategies are shown in the second and third subplots at the top of Fig 4.3, namely, a least-squares best fit and a linear interpolation through the end points.

In terms of estimation of Integral and Differential Non-Linearities (INL and DNL), they result in approximately equal worst case values. The endpoints interpolation, guarantees however the derivation of an analytical approximation, using the quantities derived in this section, and allowing, as a final result of this analysis, the designer to use them as guides based on the technology at hand.

INL and DNL parameters are defined as follows [53]:

$$\mathrm{INL}(x) \overset{\mathrm{def}}{=} \frac{A(x) - x\mathrm{LSB}}{\mathrm{LSB}} \tag{4.7}$$

$$\mathrm{DNL}(x) \overset{\mathrm{def}}{=} \frac{A(x+1) - A(x)}{\mathrm{LSB}} - 1 \tag{4.8}$$

Digital value	Analog value
`00...01`	$-\frac{2}{3}\frac{1}{2^n}\left(n + \frac{2}{2^{n_{sc}}} - \frac{4}{3}\right)$
`01...11`	$-\frac{2}{3}\frac{2^{n_{sc}}+1}{2^{n_{sc}}}$
`10...00`	$-\frac{2}{3}\frac{1}{2^{n_{sc}}}$
`11...11`	$-\frac{2}{3}\frac{2^{n_{sc}}+2}{2^{n_{sc}}}$

Table 4.4 Approximate error on the analog value induced by the presence of parasitic loading on the inner isolated nodes.

where x is the digital code being represented and $A(x)$ the corresponding analog value. Both parameters are normalized with respect to one ideal step (LSB). Let us first express the INL as a function of the error associated to each bit. The term $A(x)$ in (4.8) is substituted with (4.7) and one LSB with (4.3).

The maximum deviation using the endpoints model, for this setup is always observed around the mid-scale transition, with the input code changing from `01...1` to `10...0`, leading to:

$$\begin{aligned}
\max_x\{\mathrm{INL}(x)\} &= \mathrm{INL}(2^{n-1}) \\
&\simeq \frac{\alpha}{3}\frac{1}{2^{n_{sc}+1}}\frac{2^n}{1-2^n}\left(1-\frac{n+5/3}{2^n}+\frac{2}{4^n}-\frac{4/3}{8^n}\right) && (4.9)\\
&\simeq \frac{\alpha}{3}\frac{1}{2^{n_{sc}+1}}\frac{2^n}{1-2^n} && (4.10)\\
\max_x\{\mathrm{DNL}(x)\} &= \mathrm{DNL}(2^{n-1}-1)\\
&= 2\max_x\{\mathrm{INL}(x)\}
\end{aligned}$$

Equations (4.9) and (4.10) are our accurate and coarse model, respectively. Their prediction is shown in the right-hand subplots of Fig. 4.3. The coarse model, easier to manipulate, provides a reasonably-conservative estimate of the errors, as desired.

A more extensive characterization is shown in Fig. 4.4, where the different columns refer to different width of the C-2C subarrays. The crosses represent INL and DNL points evaluated in SPICE simulations, for varying amount of scaled capacitance. The two models provide identical answers as the C-2C subarray, i.e., n, gets larger, with the more accurate description being valid also for extremely small arrays. The first order approximation being applied at the very beginning of the section in terms of the parameter α is valid up to a value of 0.1. Above that, the errors are lower than the prediction.

From the coarse model one can derive an upper limit on the number of C-2C bits based on the amount of tolerable error $\gamma < 1$:

$$\begin{aligned}
\frac{\alpha}{3}\frac{4^n}{2^n-1} &< \gamma \\
\frac{\alpha}{3}2^n &< \gamma \\
n &< \log_2 3 + \log_2 \gamma - \log_2 \alpha\,.
\end{aligned}$$

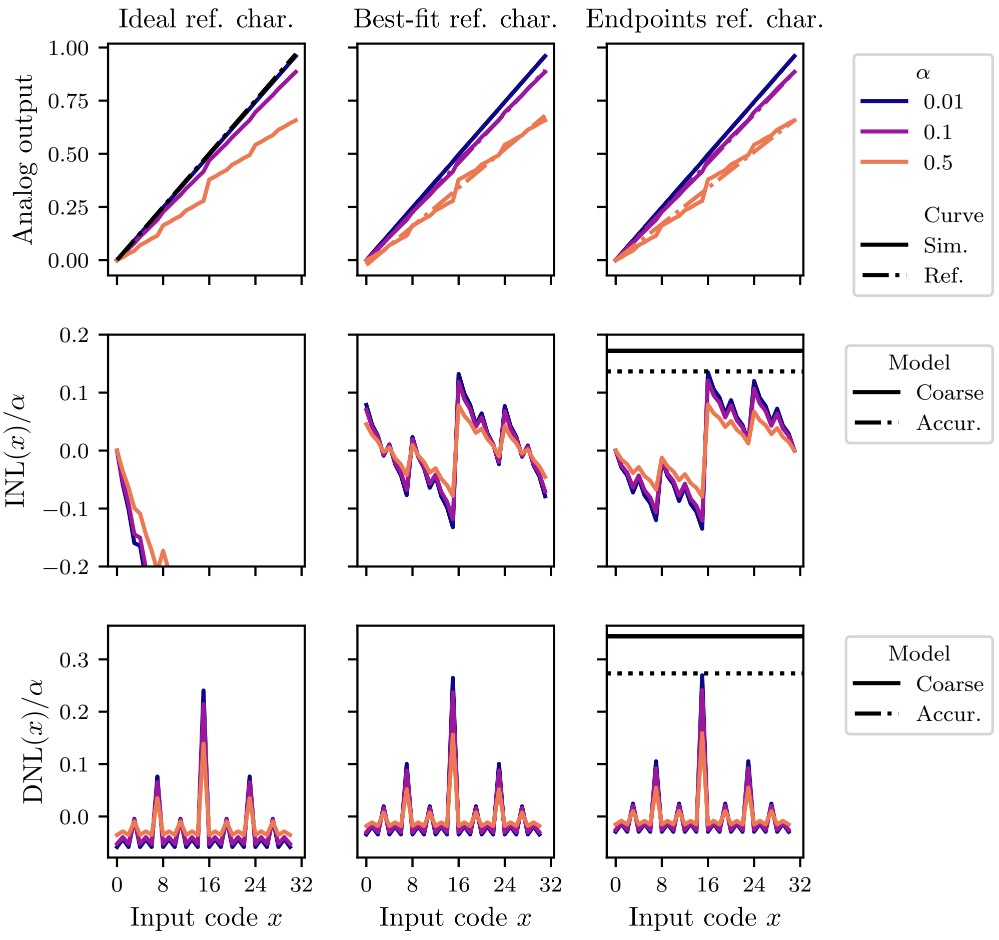


Fig. 4.3 Integral and differential nonlinearities computed on a 5-input C-2C structure, without any scaled array. Different columns consider different reference transfer curves (ideal, best-fit, endpoints). For the latter, the estimates of our accurate and coarse models for the maxima are shown as horizontal lines.

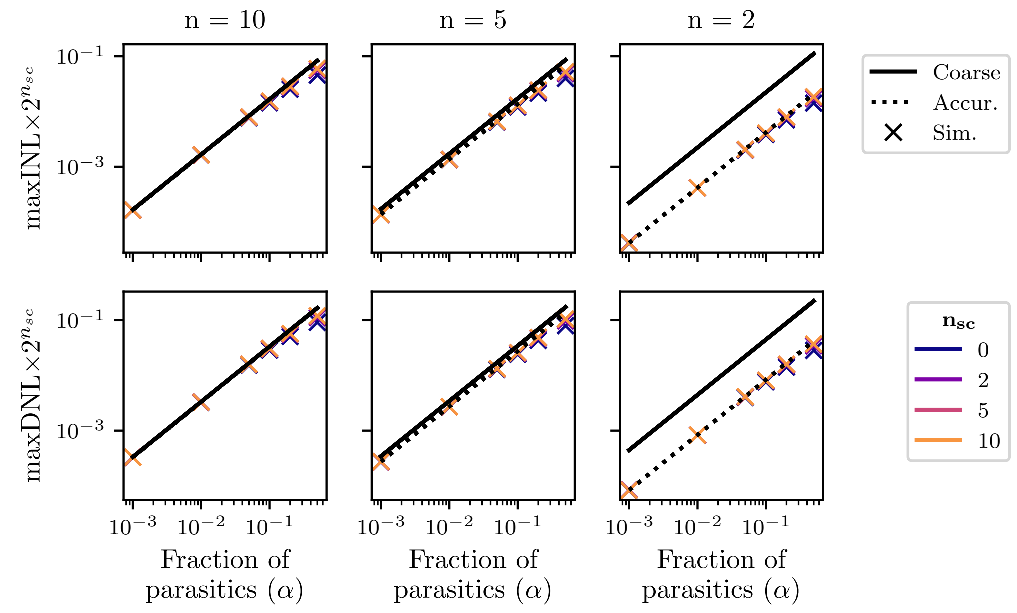


Fig. 4.4 Maximum INL and DNL for different widths of the C-2C section (along the columns), amounts of scaled capacitance (different colours) and amount of parasitics loading. Comparison between the simulated datapoints and the model predictions.

As an example, with $\alpha = 0.1$ and $\gamma = 0.5$ (maximum error of LSB/2), n should be lower than 4. The result is depicted in Fig. 4.5.

4.2 Preamplifier mismatch analysis

Mismatches are small deviations of a device parameter from its nominal value. Accounting for such deviations, parameters of topologically symmetric devices can be decomposed into an average and a differential term. The analysis in Section 3.4.4 can be considered as based on the average values of the parameters. The differential term is responsible for the coupling of CM and DM modes [61]. This phenomenon can be understood by examining the case of mismatched capacitive loads.

Consider the capacitances expressed as an average and a differential term:

$$C_{L_1} = C_L + \frac{\Delta C}{2}$$
$$C_{L_2} = C_L - \frac{\Delta C}{2}$$

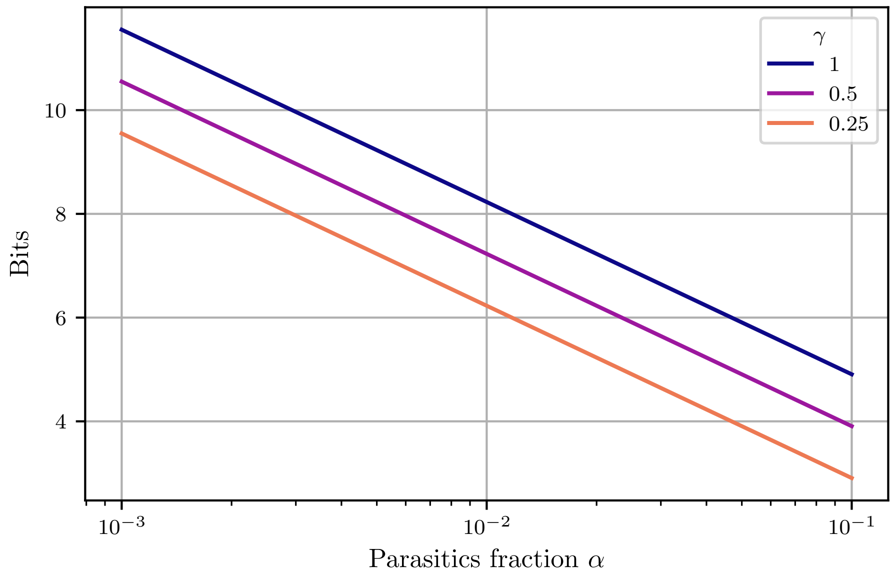


Fig. 4.5 Maximum number of bits as a function of the parasitics fraction and the amount of tolerable error γ.

If a constant current I is equally applied to them, the voltage across each capacitor becomes:

$$\begin{aligned} v_{C_1} &= \frac{I}{C_L + \frac{\Delta C}{2}} t \\ &\simeq \frac{I}{C_L} t - \frac{I}{C_L} \frac{\Delta C}{2C_L} t \\ v_{C_2} &= \frac{I}{C_L - \frac{\Delta C}{2}} t \\ &\simeq \frac{I}{C_L} t + \frac{I}{C_L} \frac{\Delta C}{2C_L} t \end{aligned}$$

The response due to the average value of the parameter is common to both branches. The mismatch determines a differential component. If such a component is small with respect to the common one, then the operating point of the circuit is not affected. In this example this is meaningless since we are considering passive components and an ideal current source. In an active circuit, where the device parameters depend on the physical quantities in the circuit, this allows us to approximate the response for small differential terms and use superposition to work separately with the common mode and differential mode circuits. The alternative would be to keep track of the coupling by solving the exact differential equations. If the approximation holds, the effort can be spared.

The differential response that stems from a mismatched parameter with a common mode input signal can be moved to the differential mode half circuit. Since the effect depends only on common mode parameters it becomes an independent source.

Conversely, by considering the effects when a differential input is applied to the circuit, the response becomes:

$$\begin{aligned}
v_{C1} &= \frac{i}{C_L + \frac{\Delta C}{2}} t \\
&\simeq \frac{I}{C_L} t + \frac{I}{C_L} \frac{\Delta C}{2C_L} t \\
v_{C_2} &= \frac{-i}{C_L - \frac{\Delta C}{2}} t \\
&\simeq \frac{-I}{C_L} t + \frac{I}{C_L} \frac{\Delta C}{2C_L} t
\end{aligned}$$

The average parameter value results in a differential component, as desired, while the mismatch generates a common mode response. This becomes an independent source in the common mode equivalent half circuit. However, since this is typically negligible with respect to the original common mode signal, this part of the analysis is not performed.

The technique can be applied to the preamplifier, considering a small variation associated to each parameter in the CM circuit and evaluating the resulting differential response. This, in turn, is placed into the DM circuit as an independent source and referred to the input in order to determine the equivalent offset voltage in the amplifier transcharacteristic. A few notable cases are shown here, the remaining values in Table 4.5 are derived with the same methodology.

Capacitor C_l

Consider a variation on the load capacitor. It can be represented as a parallel element of value $\frac{\Delta C}{2}$ on one branch, and the opposite on the other. The current it sources is

$$\begin{aligned} i_\Delta &= \frac{\Delta C_l}{2}\frac{\mathrm{d}V_o^{\mathrm{CM}}}{\mathrm{d}t} \\ &= \frac{\Delta C_l}{2}\frac{I_B}{2}\frac{1}{C_l+C_x+2(C_p+C_{p_x})} \\ &\simeq \frac{\Delta C_l}{2}\frac{I_B}{2}\frac{1}{C_l+C_x} \end{aligned}$$

The effect can be compared to the other DM quantities, by introducing it as an independent generator in the DM equivalent circuit.

If the differential current forced in the branch by v_d is exactly equal to the one just evaluated, then no differential output voltage is generated. In this condition an offset has appeared at the input.

$$\begin{aligned} -g_m\frac{v_d}{2} &= \frac{\Delta C_l}{2}\frac{I_B}{2}\frac{1}{C_l+C_x} \\ v_d^{\mathrm{off}}\big|_{\Delta C_l} &\simeq \frac{\Delta C_l}{C_l+C_x}\frac{I_B}{2g_m} \end{aligned}$$

Cross-coupled pair V_{off}

Suppose on the gate connection of the cross-coupled pair appears a differential offset voltage V_{off}. Since the XCP is current-biased, the voltage appears as a shift of the source voltage. If this constant differential voltage is so large to change the sign of $v_x^{\mathrm{dm}}(t_1)$, then the wrong decision is taken.

$$\begin{aligned} v_x^{\mathrm{dm}}(t_1) &= \left(\frac{C_l+C_x}{C_l-C_x}\frac{2g_mV_{\mathrm{in}}^{\mathrm{CM}}}{I_B}+2\sqrt{\frac{(W/L)_1}{(W/L)_2}}\right)v_d \\ &\simeq \left(\frac{2g_mV_{\mathrm{in}}^{\mathrm{CM}}}{I_B}+2\sqrt{\frac{(W/L)_1}{(W/L)_2}}\right)v_d \\ v_d^{\mathrm{off}}\big|_{V_{\mathrm{off}}} &\simeq \frac{V_{\mathrm{off}}}{\frac{2g_mV_{\mathrm{in}}^{\mathrm{CM}}}{I_B}+2\sqrt{\frac{(W/L)_1}{(W/L)_2}}} \end{aligned}$$

Clock transition

The release of the reset condition may not happen simultaneously on both branches. The variation is modeled differentially by a delay t_d applied to the start of the amplification in the CM circuit. Its differential effect is:

$$v_o^{\mathrm{dm}} = \frac{I_B}{2C_L} t_d$$

since it stems from the charging of one load capacitor at half the bias current.

The input-referred offset is evaluated as the v_d that would result in the opposite differential output voltage reached at the end of the linear growth.

$$\begin{aligned} v_d^{\mathrm{off}}\Big|_{t_d} &= -\frac{I_B^2}{g_m V_{\mathrm{in}}^{\mathrm{CM}}} \frac{C_l - C_x}{4C_l(C_l + C_x)} \\ &= -\frac{I_B^2}{g_m V_{\mathrm{in}}^{\mathrm{CM}}} \frac{t_d}{4C_l} \end{aligned}$$

Table 4.5 Input-referred offset voltage due to parameter variations. Curly braces are used to compact the table when the same multiplicative factor affects a given term.

Component	Parameter	Offset (absolute)
Differential couple	V_{th}	ΔV_{th}
	β	$\frac{\Delta\beta}{\beta} \frac{I_B}{2g_m^{\mathrm{diff}}}$
	R_{ds}	$\frac{V_{\mathrm{in}}^{\mathrm{CM}}}{g_m R_{\mathrm{ds}}} \frac{\Delta R}{R_{\mathrm{ds}}}$
Crossed couple	$\{V_{\mathrm{th}}, \beta, V_{\mathrm{off}}\}$	$\frac{\left\{V_{\mathrm{off}}, \Delta V_{\mathrm{th}}, \frac{\Delta\beta}{\beta}\sqrt{\frac{I_B}{\beta}}\right\}}{\frac{2g_m V_{\mathrm{in}}^{\mathrm{CM}}}{I_B} + 2\sqrt{\frac{(W/L)_1}{(W/L)_2}}}$
Capacitances	C_l, C_x	$\frac{\{\Delta C_l, \Delta C_x\}}{C_l + C_x} \frac{I_B}{2g_m^{\mathrm{diff}}}$
	t_{soc}	$\frac{I_B^2}{g_m V_{\mathrm{in}}^{\mathrm{CM}}} \frac{t_d}{4C_l}$

4.3 Mismatch variance in interdigitated geometries

Variability in electronic devices is one of the key issues limiting performance of analog integrated circuits [62]. To get the most out of a given technological process, designers are used to adopt layout strategies allowing at least a partial compensation of this unwanted effect. Commonly accepted guidelines are based both on rules-of-thumb and analytical considerations, and typically require each device to be split into several identical segments (also knows as *fingers*) placed according to specific patterns. An example is the interdigitated layout, which alternates segments of different devices.

However, these layout strategies are often analyzed only with respect to deterministic effects such as gradients along the wafer, neglecting any stochastic variation. The well known work by Pelgrom *et al.* [60], considers random variations only in the case of adjacent devices always being uncorrelated. Under this simplifying assumption, we will actually prove that geometry does not play any role on the spreading of device parameters caused by stochastic variations.

In recent years, many works have proposed an improvement over the original Pelgrom's model for mismatch variance [63–66]. In [65] the hypothesis of short correlation length is removed, and in [67], the authors analyze numerically the effects of different layout strategies. The same model has been extended by Poiroux *et al.* in [66], accounting for wide sense stationary (WSS) stochastic processes whose second-order statistic (i.e., correlation) can be expressed as a linear combination of Gaussian functions. WSS processes are characterized by statistical properties that do not depend on absolute positions but are only distance-dependent.

Here, starting from the theoretical model in [66], we derive an approximated model of the effects of geometry on mismatch variance, under the assumption of strong device correlation. This allows us to extend the theoretical analysis conducted on pairs of devices to complex multi-finger structures, deriving closed-form expressions in the case of interdigitated layouts.

We will initially introduce the analytical model of mismatch variance on which the analysis is built, taken directly from the original paper [66] and only adapting the notation when required. We will then describe the geometry of interest and the approximations of the model both for extremely slow and fast decay of device correlation over distance, observing that, in the latter condition, the resulting variance

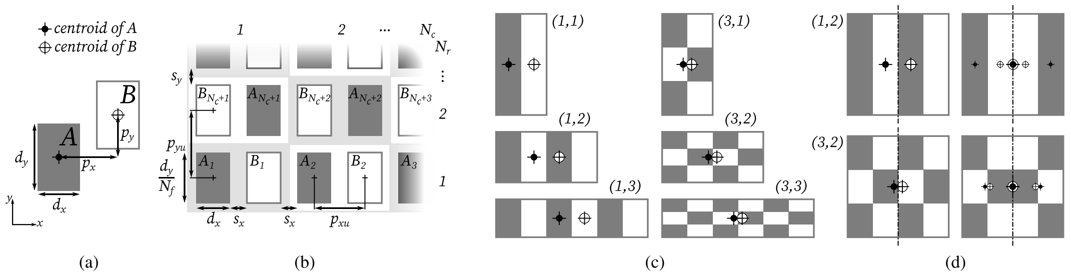


Fig. 4.6 (a) Geometry of devices A and B. (b) Interdigitated layout on a grid of N_r rows and N_c columns. Device segments are named linearly, starting from the bottom left corner. (c) Invariance of centroid distance on the number of columns N_c for a given number of rows N_r (negligible spacing between segments). In parenthesis, the couple (N_r, N_c) for each layout. (d) Mirroring half of the array to obtain a common centroid structure for N_c even and N_r odd. The centroids of the half arrays are also highlighted in the mirrored structures.

is independent of the layout of the segments. Simplified expression of the variance in terms of device size and placement for the multi-finger structures considered will be finally obtained and validated in simulation.

4.3.1 Analytical Model of Mismatch Variance

Mismatch is defined as the difference in the parameters describing two nominally identical devices. It is convenient to model mismatch as a random variable, with a probability distribution that depends on its original causes. The mean of such a variable stems from deterministic effects, whereas random fluctuations define its variance [68]. In this work we focus on the spreading of device parameters due to the latter effect. Since a large spreading might result in circuits not satisfying their design specifications, hence limiting the yield, quantifying variance becomes of paramount importance. The analysis proposed here is based on the results obtained in [66]. Let us consider two nominally identical devices occupying rectangular regions A and B on the Euclidean plane with Cartesian coordinates (x,y) as schematized in Figure 4.6a. Both devices are characterized by dimensions D_x and D_y, and their centroids are separated by distances P_x and P_y. The lowercase names appearing in the figure refer to these dimensions normalized by a suitable parameter introduced later in this section. We investigate the variation of a device parameter p that is defined pointwise on the euclidean plane by means of the WSS process $p(x,y)$. The actual value of the parameter for the two structures is indicated with $\bar{p}_A$ and $\bar{p}_B$, and is computed as the average value $\bar{p}$ of the process $p(x,y)$ over the area occupied by the two devices. Finally, let $\langle p \rangle$ be the average value of $p(x,y)$ over the entire area of interest (i.e., the whole die, or the region where the structure is placed).

The original model in [66] represents processes $p(x,y)$ defined by means of a linear combination of Gaussian autocorrelations, here we will limit our analysis to a single multivariate normal component, with independent x and y contributions. This assumption allows us to mathematically define the correlation lengths Λ_x and Λ_y along the two plane directions as $\sqrt{2}\sigma_x$ and $\sqrt{2}\sigma_y$, respectively, with σ_x and σ_y being the standard deviations of the Gaussian functions under consideration.

Instead of the actual values of D_x, D_y, P_x and P_y, it is convenient to handle their dimensionless counterparts d_x, d_y, p_x, and p_y, obtained after a normalization by Λ_x or Λ_y. Employing this notation, the main result of [66] is to express, for A and B, the autocorrelation of the difference $\bar{p} - \langle p \rangle$ as

$$\Gamma_{\bar{p}-\langle p \rangle}(p_x, p_y) = \frac{\alpha}{4d_x^2 d_y^2 \Lambda_x \Lambda_y} \gamma(p_x, d_x)\gamma(p_y, d_y), \tag{4.11}$$

where α is a scaling parameter expressed in the appropriate units, and the remaining dimensionless factors are

$$\gamma(p,d) = \theta(p+d) - 2\theta(p) + \theta(p-d),$$

being

$$\theta(u) = u \times \mathrm{erf}(u) + e^{-u^2}/\sqrt{\pi}.$$

The variance of $\Delta\bar{p} = \bar{p}_A - \bar{p}_B$, representing the spread of the difference between the parameters describing devices A and B, can then be expressed as

$$\sigma_{\Delta\bar{p}}^2 = 2\left[\Gamma_{\bar{p}-\langle p \rangle}(0,0) - \Gamma_{\bar{p}-\langle p \rangle}(p_x, p_y)\right]. \tag{4.12}$$

Table 1 in the original work [66] collected asymptotic approximations for this formula under the limiting conditions of p_x, p_y, d_x and d_y. In the following, this procedure will be extended to multi-finger devices placed on a generic 2D grid.

4.3.2 Geometry and Model Approximation

Consider a 2D grid of N_r rows and N_c columns. The original devices A and B, with physical dimensions $D_y = W$ and $D_x = L$, are divided into $N_f = N_r N_c$ segments each, preserving the total area. The segment length is assumed to be the same of the original device L, hence the height has to be scaled by $1/N_f$. Each grid cell contains one segment of A and one of B, as depicted in Figure 4.6b, so that segments of different devices alternate on the plane (interdigitation), separated by distances S_x and S_y. Given the extension of the grid, the size of the devices, and the spacing between their segments, our goal is to obtain an approximation of the mismatch variance simple enough to be used for paper-and-pencil calculations.

The generalization of (4.12) to multi-finger structures is:

$$\sigma^2_{\Delta\bar{p}} = \frac{2}{N_f^2} \sum_{i=1}^{N_f} \sum_{j=1}^{N_f} \Big[\Gamma_{\bar{p}-\langle p \rangle} \left(p_x^{AA}(i,j), p_y^{AA}(i,j) \right) \\ - \Gamma_{\bar{p}-\langle p \rangle} \left(p_x^{AB}(i,j), p_y^{AB}(i,j) \right) \Big], \tag{4.13}$$

where i and j are linear indices defined on the grid (e.g. starting from the bottom-left corner with segments A_1 and B_1 as in Figure 4.6b) and $p_x^{AA}(i,j)$ is the distance between segments i and j of device A, while $p_x^{AB}(i,j)$ is computed between segment i of device A and segment j of device B. Similar definitions stand for $p_y^{AA}(i,j)$ and $p_y^{AB}(i,j)$. Closed-form expressions for the double summation can be obtained for asymptotic values of the normalized dimension, as shown in the following.

Small correlation lengths ($d_x, d_y, p_x, p_y \gg 1$)

Under the assumption of small correlation lengths, normalized sizes and distances assume values much greater than 1, therefore we can express $\theta(u)$ as

$$\theta(u) \simeq \begin{cases} |u| & \text{if } u \neq 0 \\ \dfrac{1}{\sqrt{\pi}} & \text{if } u = 0\,. \end{cases} \tag{4.14}$$

The term for $u = 0$ is necessary since, whenever in (4.13) we have $i = j$, i.e. we consider the couple made of a segment of A and itself, the distance terms $p_x^{AA}(i,j)$

and $p_y^{AA}(i,j)$, which are arguments of θ, are equal to 0 no matter the correlation lengths.

Assuming $d \neq 0$, $p \geq 0$ and $p > d$ whenever $p \neq 0$, then

$$\gamma(p,d) \simeq \begin{cases} 2d - \dfrac{2}{\sqrt{\pi}} \simeq 2d & \text{if } p = 0 \\ 0 & \text{otherwise.} \end{cases}$$

Hence the autocorrelation of $\bar{p} - \langle p \rangle$ becomes

$$\Gamma_{\bar{p}-\langle p \rangle}(p_x, p_y) \simeq \begin{cases} \dfrac{\alpha}{d_x d_y \Lambda_x \Lambda_y} & \text{if } p_x = p_y = 0 \\ 0 & \text{otherwise .} \end{cases}$$

Plugging this expression into (4.13) and considering that $p_x^{AB}(i,j)$ and $p_y^{AB}(i,j)$ are always nonzero, we obtain

$$\begin{aligned} \sigma^2_{\Delta\bar{p}} &\simeq \frac{2}{N_f^2} \sum_{i=1}^{N_f} \sum_{j=1}^{N_f} \Gamma_{\bar{p}-\langle p \rangle} \left(p_x^{AA}(i,j), p_y^{AA}(i,j) \right) \\ &= \frac{2}{N_f^2} \sum_{i=1}^{N_f} \left(\sum_{j \neq i} 0 + \Gamma_{\bar{p}-\langle p \rangle}(0,0) \right) \\ &= \frac{2\alpha}{\Lambda_y \Lambda_x} \frac{1}{wl} = \frac{2\alpha}{WL}. \end{aligned}$$

This result corresponds to the traditional model derived by Pelgrom in [60], where variance depends on the inverse of device area. The expression depends neither on the position of the fingers nor on their number hence, under the assumption of correlation lengths much shorter than device sizes, geometry has no effect on the variance whatsoever.

Large correlation lengths ($d_x, d_y, p_x, p_y \ll 1$)

Large correlation lengths result in normalized device sizes and distances tending towards 0. The asymptotic approximation of (4.11) under this assumption is trivial. Its expression is not included here for compactness reasons, but will be used in the following sections to derive our main results.

In any case the linear indexing method employed in (4.13) is not convenient when working on a 2D grid since the expressions of the distance terms ($p_x^{AA}(i,j)$ and the others) become unmanageable. Indexing the segments through their row and column indices we can write, equivalently:

$$\begin{aligned}\sigma^2_{\Delta\bar{p}} = \frac{2}{N_r^2 N_c^2} & \sum_{r_i=1}^{N_r} \sum_{c_i=1}^{N_c} \sum_{r_j=1}^{N_r} \sum_{c_j=1}^{N_c} \\ & \left[\Gamma_{\bar{p}-\langle p\rangle}(p_x^{AA}(r_i,c_i,r_j,c_j), p_y^{AA}(r_i,c_i,r_j,c_j)) \right. \\ & \left. - \Gamma_{\bar{p}-\langle p\rangle}(p_x^{AB}(r_i,c_i,r_j,c_j), p_y^{AB}(r_i,c_i,r_j,c_j)) \right] . \end{aligned} \tag{4.15}$$

In general, (4.15) can be used to compute the variance for any 2D grid-based geometry, once the distance terms have been determined. For simplicity, the notation will be simplified in the following by not writing the explicit dependence of distances on the row and column indices.

4.3.3 Variance in Interdigitated Structures

In an interdigitated layout as the one depicted in Figure 4.6b, segment distances can be expressed as:

$$\begin{aligned} p_x^{AA} &= p_{xu} \left| 2(c_i - c_j) + \mathbf{1}_{\text{odd}}(r_i + r_j) \right| \\ p_x^{AB} &= p_{xu} \left| 2(c_i - c_j) + \mathbf{1}_{\text{even}}(r_i + r_j) \right| \\ p_y^{AA} &= p_{yu} \left| r_i - r_j \right| \\ p_y^{AB} &= p_{yu} \left| r_i - r_j \right| , \end{aligned} \tag{4.16}$$

where $p_{xu} = l + s_x$ and $p_{yu} = w/N_f + s_y$ are the distances between adjacent segments. The indicator function $\mathbf{1}_A(n)$ checks the membership of its argument to the set S and is defined as:

$$\mathbf{1}_S(x) = \begin{cases} 1 & \text{if } x \in S \\ 0 & \text{otherwise .} \end{cases}$$

It is employed in this context to account for the alternation of segment positions across consecutive rows of the grid. In the specific case of $\mathbf{1}_{\text{odd}}(n)$, the function evaluates to 1 whenever n is odd. Equivalently for $\mathbf{1}_{\text{even}}(n) = 1 - \mathbf{1}_{\text{odd}}(n)$. An

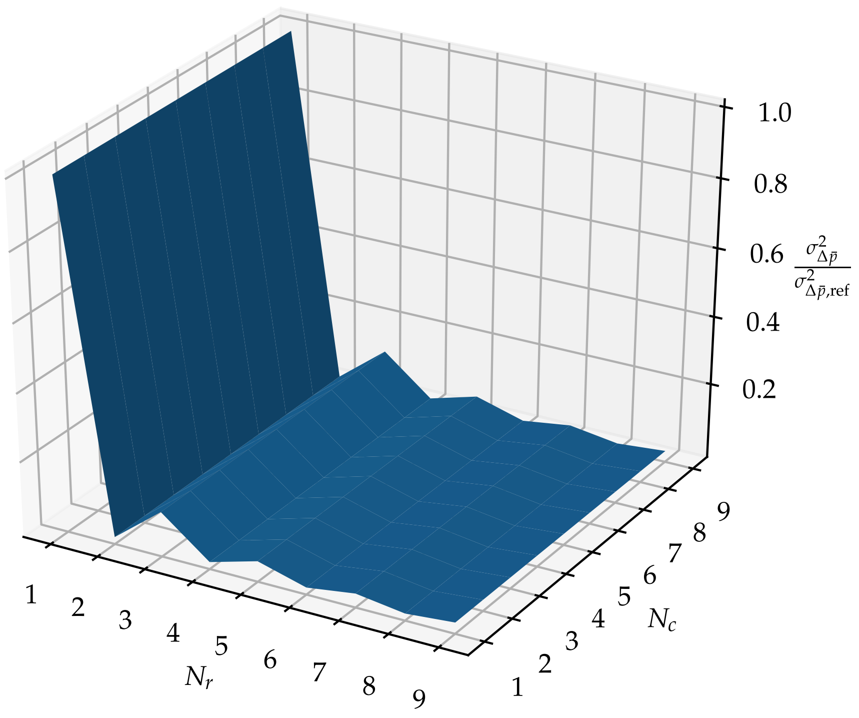


Fig. 4.7 Mismatch variance computed through (4.17) and normalized to the value obtained for $N_r = N_c = 1$. ($w = 0.02$, $l = 0.001$)

expression for $\mathbf{1}_{\mathrm{odd}}(n)$ that is also suitable for symbolic manipulation is given by

$$\mathbf{1}_{\mathrm{odd}}(n) = \frac{1-(-1)^n}{2}.$$

Combining expressions (4.16) and (4.15), and using Taylor's expansion of (4.11) around 0 (because of the large correlation lengths), mismatch variance can be expressed as:

$$\begin{aligned}\sigma^2_{\Delta\bar{p}} &\simeq \frac{2\alpha}{\pi\Lambda_x\Lambda_y} p^2_{xu}\left(\frac{1}{N_r^2}\mathbf{1}_{\mathrm{odd}}(N_r) + \frac{p^2_{yu}}{2}\mathbf{1}_{\mathrm{even}}(N_r)\right) \\ &= \begin{cases} \dfrac{2\alpha}{\pi\Lambda_x\Lambda_y}\dfrac{p^2_{xu}}{N_r^2} & \text{if } N_r \text{ odd} \\ \dfrac{2\alpha}{\pi\Lambda_x\Lambda_y}\dfrac{p^2_{xu}p^2_{yu}}{2} & \text{if } N_r \text{ even .}\end{cases}\end{aligned} \tag{4.17}$$

The dependence on geometry is explicit in the term N_r^{-2} and is also embedded in the value of p_{yu}. If the device segments are tightly spaced, i.e. $s_x = s_y \approx 0$, (4.17) reduces to

$$\sigma^2_{\Delta\bar{p}} \simeq \begin{cases} \dfrac{2\alpha}{\pi\Lambda_x\Lambda_y}\dfrac{l^2}{N_r^2} & \text{if } N_r \text{ odd} \\ \dfrac{2\alpha}{\pi\Lambda_x\Lambda_y}\dfrac{l^2 w^2}{2N_r^2 N_c^2} & \text{if } N_r \text{ even} \end{cases} \tag{4.18}$$

which is depicted in Figure 4.7 with respect to $\sigma^2_{\Delta\bar{p},\mathrm{ref}}$, evaluated for ($N_r$, N_c) = (1, 1). The plot highlights the constant-valued peaks whenever the number of rows is odd, notwithstanding the fact that as the number of columns grows, the increasingly smaller device segments are more uniformly spread in the grid area. The valleys in the plot actually vary with N_c, but it is not apparent in linear scale.

Intuitively, increasing N_c changes the absolute positions of the centroids of A and B with respect to the origin of the grid, as shown in Figure 4.6c, although their relative distance is unaffected. Since N_r odd implies a non-common-centroid geometry, the constant centroid distance results in a constant contribution to the variance. Increasing the number of rows to a larger odd number, such a distance decreases, but is still invariant with respect to N_c. Conversely, whenever N_r is even the centroids of A and B coincide and higher order effects result in N_c actually having an effect on the variance.

Variance in a Mirrored Interdigitated Layout

As already pointed out, the simple interdigitated layout in Figure 4.6b is not common centroid whenever the number of rows is odd. If, at the same time, the number of columns is even, it is actually possible to obtain a common centroid geometry simply by mirroring the right half of the structure, as shown in Figure 4.6d. The expressions of segment distances describing such a layout require the introduction of the sets $L = \{1,\ldots,N_c/2\}$ and $R = \{N_c/2+1,\ldots,N_c\}$, which characterize the columns as belonging either to the left (straight) or the right (mirrored) half of the array. Thus

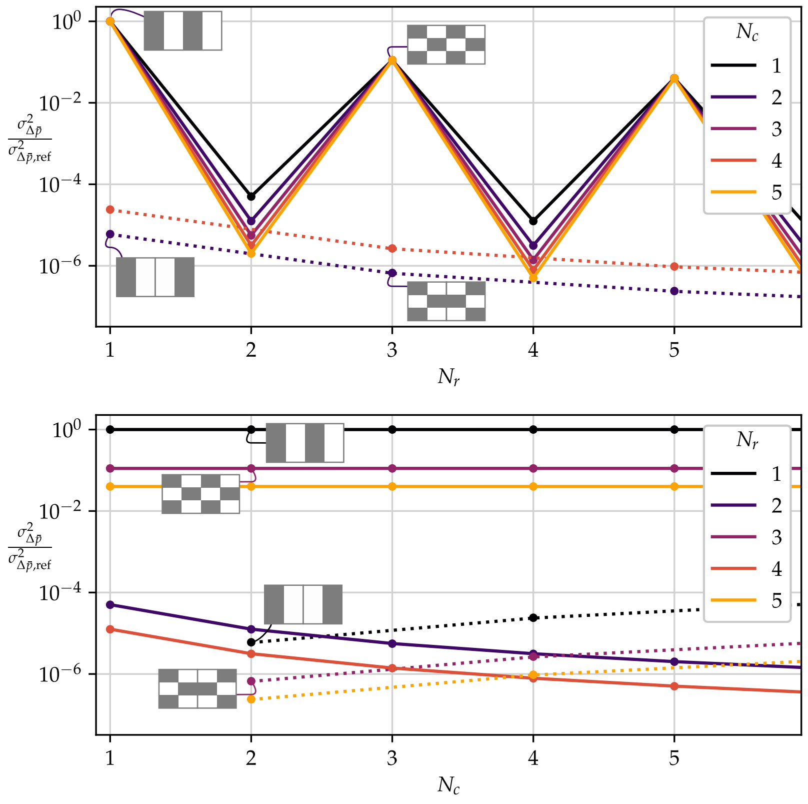


Fig. 4.8 Mismatch variance normalized with respect to the value for $N_r = N_c = 1$. Lines are obtained through a Monte Carlo simulation using the model in [1], markers using the asymptotic expressions derived in this work. Solid lines refer to the simple interdigitated layout, dotted lines to the mirrored geometry. ($w = 0.02$, $l = 0.001$, $s_x = s_y = 0$, $\Lambda_x = \Lambda_y = 10^3$, $\alpha = 1$)

we can write:

$$\begin{aligned}
p_x^{AA} = p_{xu}\, \big| 2\,(c_i - c_j) &\\
&+ \mathbf{1}_{\mathrm{odd}}(r_i + r_j)\,\big[\mathbf{1}_{\mathrm{L}}(c_i)\mathbf{1}_{\mathrm{L}}(c_j) - \mathbf{1}_{\mathrm{R}}(c_i)\mathbf{1}_{\mathrm{R}}(c_j)\big]\\
&+ \mathbf{1}_{\mathrm{even}}(r_i)\mathbf{1}_{\mathrm{even}}(r_j)\mathbf{1}_{\mathrm{L}}(c_i)\mathbf{1}_{\mathrm{R}}(c_j)\\
&+ \mathbf{1}_{\mathrm{odd}}(r_i)\mathbf{1}_{\mathrm{odd}}(r_j)\mathbf{1}_{\mathrm{R}}(c_i)\mathbf{1}_{\mathrm{L}}(c_j)\big|\\
p_x^{AB} = p_{xu}\, \big| 2\,(c_i - c_j) &\\
&+ \mathbf{1}_{\mathrm{even}}(r_i + r_j)\,\big[\mathbf{1}_{\mathrm{L}}(c_i)\mathbf{1}_{\mathrm{L}}(c_j) - \mathbf{1}_{\mathrm{R}}(c_i)\mathbf{1}_{\mathrm{R}}(c_j)\big]\\
&+ \mathbf{1}_{\mathrm{even}}(r_i)\mathbf{1}_{\mathrm{odd}}(r_j)\mathbf{1}_{\mathrm{L}}(c_i)\mathbf{1}_{\mathrm{R}}(c_j)\\
&+ \mathbf{1}_{\mathrm{odd}}(r_i)\mathbf{1}_{\mathrm{even}}(r_j)\mathbf{1}_{\mathrm{R}}(c_i)\mathbf{1}_{\mathrm{L}}(c_j)\big|\,.
\end{aligned}$$

Distances along y are unchanged with respect to (4.16).

Following the same procedure as in the previous section, the variance can be expressed as:

$$\sigma^2_{\Delta\bar{p}} \simeq \frac{2\alpha}{\pi\Lambda_x\Lambda_y}\frac{3}{2}\left(\frac{N_c}{N_r}\right)^2 p_{xu}^4 \qquad \begin{matrix}\text{if } N_r \text{ odd}\\ \text{and } N_c \text{ even}\end{matrix} \tag{4.19}$$

Being p_{xu} a constant, the dependence on the geometry is determined only by the ratio $(N_c/N_r)^2$. Intuitively, as N_r is increased, the centroids in each half of the array get closer, increasing the correlation between the devices and thus reducing the variance. Conversely, as N_c increases, the distance between the centers of the two halves of the array grows, resulting in a larger variance.

4.3.4 Results

Figure 4.8 shows the normalized behaviour of variance for the layouts analyzed in this work. The results from the approximate models (4.17) and (4.19) are compared to the variance computed on 10000 Monte Carlo trials of the CAD-friendly model developed by Lu in [1] and analyzed also in [66]. The maximum relative error between datapoints in Figure 4.8 is less than 2%, validating the approximate expressions. For the sake of clarity, the geometries corresponding to some of the data points have been superimposed to the plot. In particular they represent the structures forced to be common centroid, highligting the reduction in variance obtained by such a modification.

In Fig. 4.9 we have computed the asymptotic expressions derived in this work for two-fingers geometries, normalized with respect to the variance $\sigma^2_{\Delta\bar{p},\mathrm{ref}}$ of the reference, single-finger layout ($N_r = N_c = 1$). With negligible device spacing, the cross-coupled layout performs better than the mirrored-linear one whenever $(w/l)^2 < 12$.

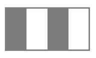
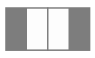
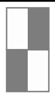
(1,2)  linear	(1,2)  half-mirrored linear	 (2,1) cross-coupled
$\sigma^2_{\Delta\bar{p},\mathrm{ref}}$	$\frac{3}{2}\left(l+s_x\right)^2\sigma^2_{\Delta\bar{p},\mathrm{ref}}$	$\frac{1}{2}\left(\frac{w}{2}+s_y\right)^2\sigma^2_{\Delta\bar{p},\mathrm{ref}}$

Fig. 4.9 Mismatch variance for two-finger alyouts, with respect to the single-finger reference case $(N_r, N_c) = (1, 1)$.

4.4 Stability analysis of a leakage compensator

In its most trivial form, a sample-and-hold circuit consists of a capacitor, as in Fig. 4.10a, tracking the input voltage until its access transistor M_1 is in the ON state. As soon as the transistor turns OFF, the charge in the capacitor is fixed and the voltage information is available for other circuits to operate on. A real MOS transistor in its OFF state, however, does not completely isolate the capacitor, causing a discharge over time because of leakage currents through the channel and the diffusions.

In general, once a signal sample is stored, the input voltage will still vary. Since this makes the analysis more complex, let us limit ourselves to the "T" shaped switch topology of Fig. 4.10b [69]. Among its advantages, this solution removes the input voltage dependency by shorting the inner node to the reference voltage. Moreover, transistors M_2 and M_3 will be replaced by a Norton equivalent model as depicted in Fig. 4.10c.

The leakage current cancellation circuit is shown in Fig. 4.10d. It was originally introduced in [70] and experimentally validated, observing up to $20\times$ reduction of the voltage decay with respect to an uncompensated hold cell. The circuit requires a replica of the original hold cell where the capacitance is scaled down by a factor $k < 1$. Having unequal values, their discharge rates will differ, resulting over time in an increasing voltage $v_{\text{diff}}(t) = v_h(t) - v_{\text{rep}}(t)$. A compensation current $i_{\text{inj}}(t)$, proportional to $v_{\text{diff}}(t)$, is injected back into both cells, reducing the net current flow through the capacitors.

Assuming $R \to \infty$, a steady state condition is reached with the leakage current I_L compensated exactly and the hold voltage, after an initial decay, preserved indefinitely. Conversely, finite resistance values prevent a steady state condition from being achieved and require appropriate sizing of the loop elements in order to satisfy admissible voltage drop specifications.

In the following, for a comprehensive analysis of the properties of the system, we assume that the replica circuit is not identical to the original one due, for example, to circuital mismatch. In details, assuming $t = 0$ is the sampling instant, the main inputs applied to the circuit are the sampled voltages $v_h(0^-) = V_0$ and $v_{\text{rep}}(0^-) = V_0 + \Delta V_0$ and the constant leakage current $I_L u(t)$, where $u(t)$ is the unitary step function, valued 1 for $t \geq 0$ and zero otherwise. One of the main sources for the sampling voltage error ΔV_0 is the charge injected by the access transistors upon their transition

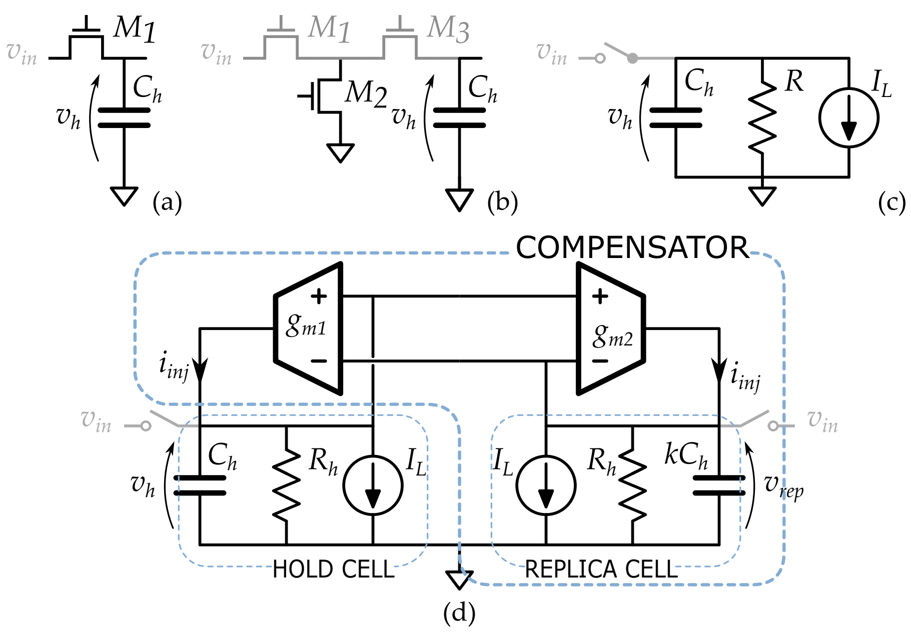


Fig. 4.10 (a) Basic implementation of a sample-and-hold circuit. (b) Implementation by means of a T-shaped switch. (c) Equivalent circuit for (b) in the hold mode. (d) Schematic of the leakage compensator under analysis.

to the off state, an effect that depends on the capacitance values, asymmetric by design. As such, it will affect even the nominal circuit.

The Laplace-domain block-diagram description of the compensator is depicted in Fig. 4.11. Notice how the initial conditions applied to the capacitors act as currents in the transformed domain. This is obtainable by transforming the constitutive equation of a linear capacitor taking into account its initial condition [71]:

$$\begin{aligned}
I_c(s) = \mathscr{L}\{i_C(t)\} &= \mathscr{L}\left\{C\frac{dv_C(t)}{dt}\right\} \\
&= C\left(s\mathscr{L}\{v_C(t)\} - v_C(0^-)\right) \\
&= sCV_c(s) - Cv_C(0^-) \\
I_c(s) + Cv_C(0^-) &= sCV_c(s)\,.
\end{aligned}$$

Two transformed currents flow through the capacitor, the second of which represents the initial conditions and corresponds to a Dirac-delta distribution in time domain and a constant in Laplace domain.

The equivalent load impedances in each cell are:

$$Z_h = \frac{R_h}{1+sR_hC_h} \qquad Z_{\text{rep}} = \frac{R_{\text{rep}}}{1+skR_{\text{rep}}C_h}.$$

4.4.1 Analysis of the nominal circuit

We will initially neglect any mismatch, i.e., $R_{\text{rep}} = R_h$. The transconductances will be considered identical as well, and without any reactive effect, i.e. $g_{m1}(s) = g_{m2}(s) = g_{m0}$.

Being the circuit linear, we apply the superposition principle and derive the response $V_h(s)$ with respect to each input:

$$V_h(s) = H|_{V_0}V_0 + H|_{\Delta V_0}\Delta V_0 + H|_{I_L}\frac{I_L}{s}.$$

Elementary algebraic manipulations lead to the desired transfer functions, approximated under the assumption of $g_{m0}R_h \gg 1$:

$$\begin{aligned}
H|_{V_0}(s) &\simeq g_{m0}R_h^2(1-k)C_h\frac{1}{(1+\tau_1 s)}\\
H|_{\Delta V_0}(s) &\simeq -g_{m0}R_h^2kC_h\frac{1}{(1+\tau_1 s)(1+\tau_2 s)}\\
H|_{I_L}(s) &\simeq -R_h\frac{1+T_1 s}{(1+\tau_1 s)(1+\tau_2 s)},
\end{aligned}$$

with:

$$\tau_1 = g_{m0}R_h^2(1-k)C_h, \quad \tau_2 = \frac{k}{1-k}\frac{C_h}{g_{m0}}, \quad T_1 = kR_hC_h.$$

Being both τ_1 and τ_2 positive, the two poles lie in the left half-plane, hence the system is stable for all parameter values. The dominant (slowest) time constant, τ_1, is given by the time constant of the original hold cell (without compensation), R_hC_h, increased by a scaling factor $g_{m0}R_h(1-k) \gg 1$. The voltage decay can therefore be slowed down as much as needed by increasing the transconductance. At the same time, τ_2 becomes increasingly smaller. Notice also how $H|_{V_0}$ is a single-pole transfer function.

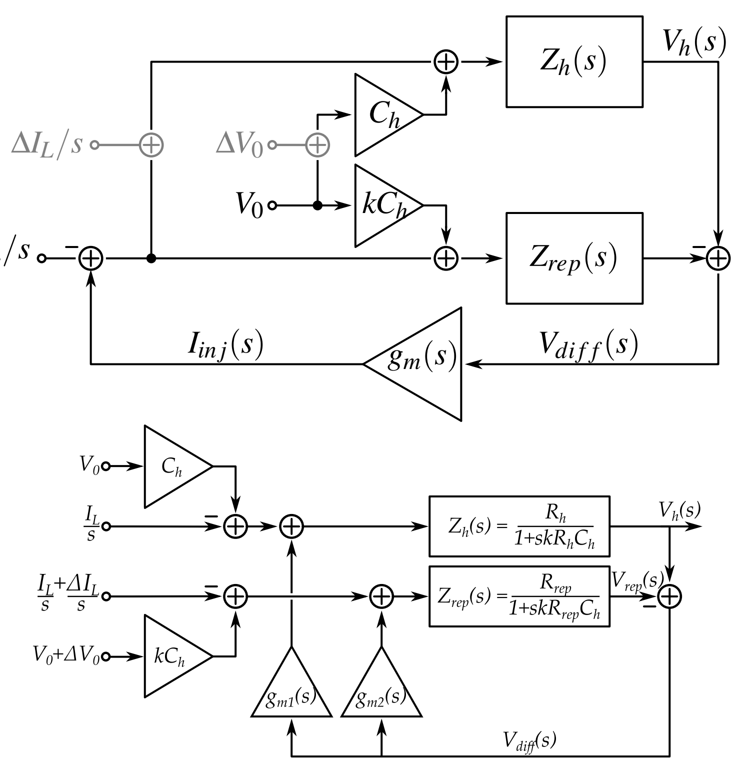


Fig. 4.11 Complete block diagram of the compensator.

The time-domain response $v_h(t)$ is given by the sum of corresponding terms:

$$
\begin{aligned}
v_h|_{V_0}(t) &\simeq V_0 \exp\left(-\frac{t}{\tau_1}\right) \\
v_h|_{\Delta V_0}(t) &\simeq -\Delta V_0 \frac{k}{1-k}\left[\exp\left(-\frac{t}{\tau_1}\right) - \exp\left(-\frac{t}{\tau_2}\right)\right] \\
v_h|_{I_L}(t) &\simeq -R_h I_L \left[1 - \exp\left(-\frac{t}{\tau_1}\right)\right. \\
&\qquad \left. + \frac{k}{1-k}\frac{1}{g_{m0}R_h}\exp\left(-\frac{t}{\tau_2}\right)\right]
\end{aligned}
\tag{4.20}
$$

The sampled voltage decays according to the nominal time constant. The voltage asymmetry ΔV_0, applied to the replica cell, is propagated to the original cell and attenuated at least by a factor $\frac{k}{1-k}$. The effect of the leakage current is also slowed down by the increased time constant.

According to the expressions in (4.20), the circuit can significantly reduce the leakage-induced discharge, though the assumptions of identical values for some of the parameter are not realistic and the effect of their asymmetries must be evaluated.

4.4.2 Effects of device mismatches

The condition of exactly equal resistances, transconductances and leakage currents will be nearly impossible to achieve in practice. Although mismatches in leakage currents will not change the stability properties of the system, being only a different input signal for the same feedback loop, their effect has to also be evaluated and, if needed, mitigated.

A (constant) asymmetry in the leakage current can be modeled as a $\Delta I_L u(t)$ flowing through the replica cell. The additional term in the expression of $V_h(s)$ is given by $H|_{\Delta I_L}(s) \cdot \Delta I_L/s$, with:

$$H|_{\Delta I_L}(s) \simeq g_{m0} R_h^2 \frac{1}{(1+\tau_1 s)(1+\tau_2 s)} .$$

The corresponding time-domain contribution is:

$$v_h|_{\Delta I_L}(t) \simeq g_{m0} R_h^2 \Delta I_L \left[1 - \exp\left(-\frac{t}{\tau_1}\right) + \frac{k}{g_{m0}^2 R_h^2 (1-k)^2} \exp\left(-\frac{t}{\tau_2}\right) \right] . \tag{4.21}$$

This time, the $g_{m0} R_h$ factor that slows the decay down also acts as a scaling coefficient for the entire expression. Under the reasonable assumption of hold times much smaller than the dominant time constant τ_1, equation (4.21) becomes:

$$v_h(t)|_{\Delta I_L}(t) \simeq \frac{1}{C_h(1-k)} \Delta I_L t .$$

The circuit behaves almost as an ideal integrator of the leakage current mismatch term, with an effective capacitance close to the hold capacitance C_h. The only terms that can limit the error due to such an asymmetry, once the hold time is selected, are the value of C_h, which should be enlarged if needed, and the amount of current ΔI_L, which has to be minimized by ensuring that the transistors in both cells are as matched as possible. Parameter k is already assumed to be small, hence its effect is not significant in this expression.

The presence of mismatch terms affecting $R_{rep} = R_h(1+\Delta R_h/R_h)$ and $g_{m2} = g_{m0}(1+\Delta g/g_{m0})$, modifies the circuit open-loop gain, which becomes:

$$\begin{aligned} T' = & g_{m0}\left(1+\frac{\Delta g_{m0}}{g_{m0}}\right)\frac{R_h(1+\Delta R_h/R_h)}{1+skR_hC_h(1+\Delta R_h/R_h)} \\ & -g_{m0}\frac{R_h}{1+sR_hC_h}. \end{aligned}$$

The zeroes of the corresponding characteristic equation $1+T'=0$ represent the time constants of the systems. While τ_2 is mostly unaffected, the dominant time constant indeed becomes:

$$\begin{aligned} \tau_1' & \simeq R_hC_h(1-k)\frac{g_{m0}R_h}{1+g_{m0}R_h\left(\frac{\Delta R_h}{R_h}+\frac{\Delta g_{m0}}{g_{m0}}+\frac{\Delta R_h}{R_h}\frac{\Delta g_{m0}}{g_{m0}}\right)} \\ & \simeq \frac{R_hC_h(1-k)}{\frac{\Delta R_h}{R_h}+\frac{\Delta g_{m0}}{g_{m0}}+\frac{\Delta R_h}{R_h}\frac{\Delta g_{m0}}{g_{m0}}}. \end{aligned} \tag{4.22}$$

Whenever the 1 at the denominator becomes negligible, i.e. $g_{m0}R \gg 1$ and the mismatch terms are significant, the sign of the time constant is determined by those of the $\Delta R_h/R_h$ and $\Delta g_{m0}/g_{m0}$ terms, which are unknown a priori. The time constant can therefore become negative and the exponential decay may turn into an exponential growth of the output voltage $v_h(t)$, i.e. a right half-plane pole arises. Hence the system is *conditionally stable*, depending on the statistical properties of the mismatch terms.

Note however that, to first order, the absolute variations $|v_h(t)-V_0|$ are independent of the sign of τ_1' for $t \ll \tau_1'$,.

As a final note, since the dominant time constant is limited in magnitude by the $\Delta R_h/R_h$ and $\Delta g/g_{m0}$ terms, a further increase of the transconductance is ineffective in reducing the observed voltage decay and requires either a reduction of the mismatch terms or an increase in C_h.

4.4.3 Stability with a dynamic transconductance

Up to this point, the transconductances have been considered as constants, resulting in compensator dynamics which are at most those of a second order system, with two real poles. Here we will consider identical transconductances with a dominant pole, i.e. $g_{m1}(s) = g_{m2}(s) = g_m(s) = g_{m0}\frac{1}{1+s/p_g}$. The order of the system is thus increased, and complex conjugate poles may arise, leading to damped oscillations in the system response.

Two cases will be presented, the first in which the pole frequency is independent of g_{m0}, the second where the frequency is given by $\frac{g_{m0}}{C_g}$, as it happens in a two-stage transconductance amplifier with g_{m0} determined by the input stage only.

Constant pole frequency

The open-loop gain with a dynamic conductance is expressed by:

$$T'' = g_{m0}\frac{1}{1+s/p_g}\left(Z_{rep}(s) - Z_H(s)\right).$$

The characteristic equation $1 + T'' = 0$ is linear in g_{m0} and can be studied with the standard root locus technique [72].

With $n = 3$ poles and $m = 1$ zero in the expression T'', control systems theory guarantees that the complex conjugate poles arising for any value of g_{m0} will not give rise to instability, since the root locus will follow vertical asymptotes, whose angles with respect to the positive real axis are:

$$\theta_{a,\nu} = \frac{2\nu - 1}{n-m}\pi = \left\{\frac{\pi}{2}, \frac{3}{2}\pi\right\},$$

with $\nu \in \{0, 1\}$ the index of the pole. The asymptote crosses the real axis at a point:

$$\begin{aligned}\sigma_a &= \frac{1}{n-m}\left(\sum_{i=1}^{n} p_i - \sum_{i=1}^{m} z_i\right)\\ &= -\frac{1}{2}\left(\frac{1}{R_hC_h} + \frac{1}{kR_hC_h} + p_g\right),\end{aligned} \tag{4.23}$$

with p_i and z_i the poles and zeroes of T. If the pole frequency $-p_g$ is independent of the transconductance and $p_g > 0$, then the system is unconditionally stable.

Pole frequency proportional to g_{m0}

If instead $p_g = \frac{g_{m0}}{C_g}$, i.e., proportional to the transconductance, the characteristic equation can be manipulated into an expression quadratic in g_{m0}. The standard root locus technique is no more applicable and we would have to resort to the polynomial root locus method [73]. Its most peculiar feature is that each point of the root locus may be obtained for multiple values of the gain variable that parametrizes the curve (in our case, g_{m0}). Since our interest is mainly in the stability of the feedback loop, we will limit our analysis to finding the conditions of system instability.

Table 4.6 Routh-Hurwitz table for the characteristic polynomial when the dominant pole in $g_m(s)$ is proportional to g_{m0}.

3	$a_3 = kR_h^2C_h^2C_g$	$a_1 = g_{m0}^2R_h^2C_h\,(1-k)$
2	$a_2 = (1+k)R_hC_hC_g$	$a_0 = g_{m0}$
1	$b_1 = \frac{g_{m0}R_hC_h}{1+k}\left[(1-k^2)g_{m0}R_h - k\right]$	
0	$c_1 = g_{m0}$	

In order to evaluate the presence of roots with positive real part in a polynomial equation $a_ns^n + \ldots + a_1s^1 + a_0 = 0$, we can readily apply the Routh-Hurwitz criterion [72]. The method requires the construction of a table whose top two rows contain the polynomial coefficients in a specific order, as in Table 4.6, constructed under the assumptions $g_{m0}R_h \gg 1$ and $R_hC_h \gg C_g/g_{m0}$. In a third order equation, two more elements have to be evaluated, whose expressions are:

$$b_1 = \frac{a_2a_1 - a_3a_0}{a_2}, \qquad c_1 = \frac{b_1a_0}{b_1} = a_0\,.$$

The criterion states that, moving along the first column, every change of sign of the elements in consecutive rows corresponds to a root with positive real part. Conversely, sign permanence is equivalent to a root with negative real part. All the terms but b_1 are always positive, hence the stability of the closed loop system depends on the sign of b_1. If positive, all roots have negative real part, if negative, a couple of unstable roots arises. The corresponding inequality

$$\left(1-k^2\right)g_{m0}R_h > k$$

is immediately verified under the initial assumption $g_{m0}R \gg 1$, hence the closed-loop system is stable.

4.4.4 Numerical evaluations

The root loci of the closed-loop system, parameterized by the value of g_{m0}, are shown in Fig. 4.12. They have been evaluated for $R_h = 1\,\mathrm{G\Omega}$, $C_h = 100\,\mathrm{pF}$ and $k = 0.1$. Parameter values have been selected to validate the analytical models and may be far from realistic conditions. Plots 4.12a and 4.12b, correspond to the model with a constant transconductance, with no reactive effects. The system is a second-order one, with clearly separated real roots. In Fig. 4.12b the positive zero for a $\Delta g_{m0}/g_{m0} = -0.1$ determines a transition to unstable behavior as g_m0 increases. Conversely, in 4.12c a dominant pole in $g_m(s)$, at -5 krad/s, results in a third-order system, possibly with a pair of complex-conjugate poles. The vertical asymptote abscissa has a real part of -2.55 krad/s as computable also by (4.23). Finally, Fig.4.12d, 4.12e and f depict the loci for $C_g/C_h = 1/50, 1/100, 1/1000$, respectively. As the ratio becomes smaller, the region enclosed by the complex conjugate loci shrinks. That same region is characterized by extremely low values of transconductance and may not be observed in practice.

The transient behaviour of $v_h(t)$ in different operating conditions has been verified in SPICE simulations and is depicted in Figure 4.13. The `V(vh_no_comp)` curve represents the uncompensated voltage decay of a hold cell, with a leakage current $I_L = 10\,\mathrm{nA}$ and an sampled voltage $V_0 = 1\,\mathrm{V}$. Applying around the cell the compensator without any mismatch, the `V(vh_nom)` waveform is obtained, showing a fast, limited-amplitude initial transient followed by a slower decay. The addition of an asymmetry on the transconductance $\Delta g_{m0}/g_{m0} = -0.1$ determines the exponential growth of `V(vh_mism)`. Finally, a constant pole in $g_m(s)$, together with the previous mismatch, results in the damped oscillations observed in `V(vh_dyn)`.

Evidently, the theoretical analysis, validate by numerical evaluations proves the extreme sensitivity of the active compensator to all sorts of hardware nonidealities and its initially assumed robustness is not at all there in reality. Let us then go back to the simplest solution of trying to cancel the leakage current components by an entirely passive structure, made only of MOS switches.

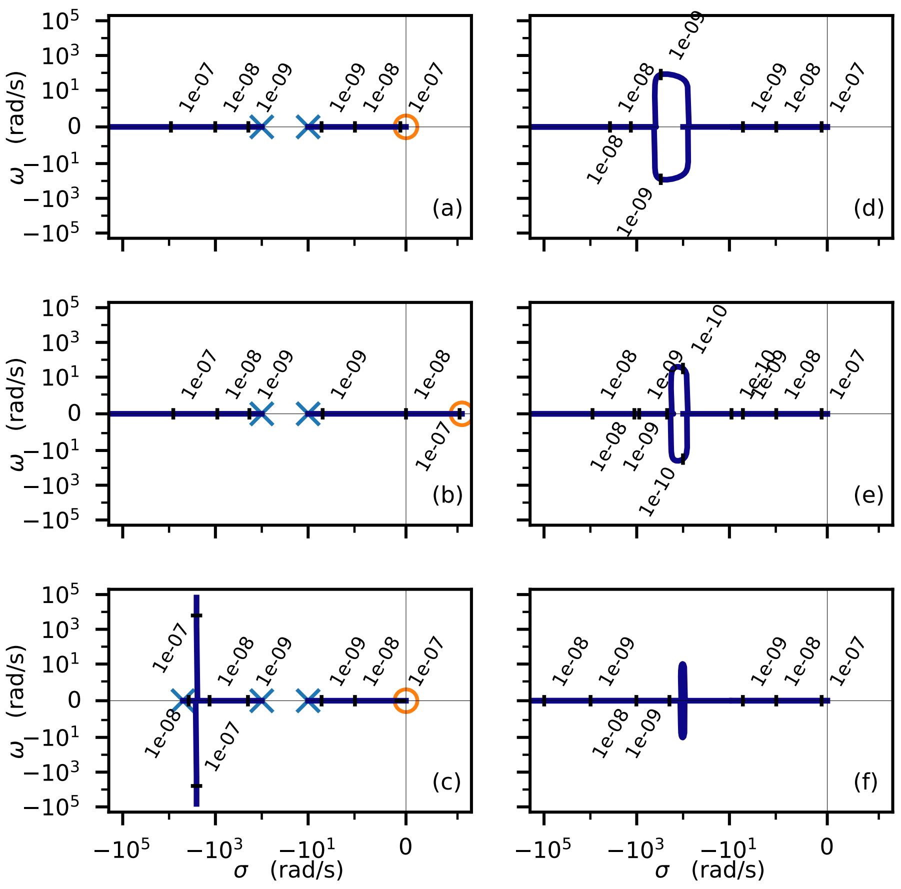


Fig. 4.12 Root loci for the closed loop poles of the leakage compensator as a function of the g_{m0} parameter, in S. (a) Nominal circuit. (b) Circuit with a $\Delta g_{m0}/g_{m0} = -0.1$ mismatch term. (c) Nominal circuit with a constant pole in $g_m(s)$. (d) Nominal circuit with a $g_m(s)$ pole proportional to g_{m0}, $p_g = g_{m0}/C_g$, with $C_g \in \{C_h/50, C_h/100, C_h/1000\}$. Both plot axis employ a symmetric log scale so that positive and negative values can be represented over a wide range. The region $x \in [-1,1]$, $y \in [-1,1]$ is in linear units.

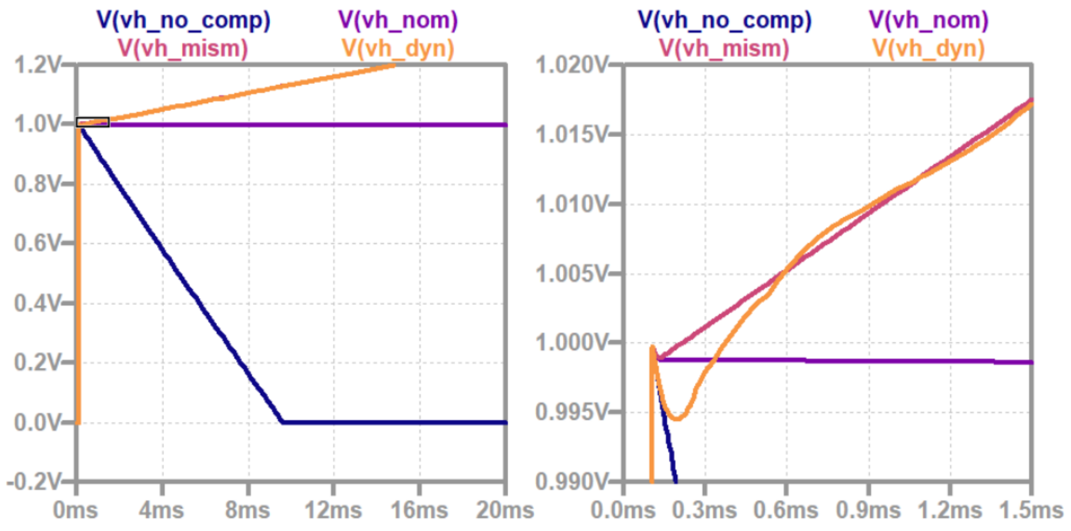


Fig. 4.13 Transient behaviour of the voltage in the hold cell for several system configurations. The subplot on the right provides the zoomed-in view of the rectangular region highlighted on the left.

Chapter 5

Phase-Change Analog In-Memory Computing

There once was a brainy baboon,
Who always breathed down a bassoon,
For he said, “It appears
That in billions of years
I shall certainly hit on a tune.”

New Pathways in Science (1935)
SIR ARTHUR S. EDDINGTON

This chapter initially deviates from the previous discussion, since we are going to evaluate the performance of a different material technology as a potential candidate for low-energy analog processing.

In the following we will focus on Phase-Change technology, whose history goes back to the seminal works by S.R. Ovshinsky in 1968 [74]. Since then, these materials have found applicability in commercial devices, e.g. as optical storage media as DVD-RAM and Blu-ray discs, or as compact digital memory elements in automotive-grade integrated circuits [75]. In the realm of digital memories interfaced with general purpose computing platforms, PCMs would be filling the performance/cost gap between DRAMs, fast, reliable but non-volatile, and NAND flash, slow and with a limited life span. Here we will use them as analog memories, trying to evaluate whether the current state of both the technology and the circuit topologies used to drive them can enable them for real-world applications.

The chapter will start with an overview of the device physics and the two hardware (HW) platform we have employed to evaluate the devices. Device programming and measurement acquisition are then described. Finally, we will detail the construction of application-based DC models, to enable virtual testing of the devices in the context of classification and regression tasks by means of Neural Networks (NNs), as well as Compressed Sensing (CS) encoding.

5.1 Phase-Change Technology

Phase-change materials are made of chalcogenide alloys containing elements from group V of the periodic table, most often tellurium [75]. Their definining feature is the ability to reversibly switch between an amorphous and crystalline state, both stable at room temperature and having wildly different optical and electrical properties. These differences allow the atomical structure of the lattice to encode and store information, acting as proper optical and electrical memories.

The phase transition underlying the operations of phase-change materials is induced by thermal processes, using either optical sources or Joule heating due to electric currents. In the following we will only consider electric interactions and we will be referring specifically to the Ge-Sb-Te (GST) alloy, being currently the most extensively researched.

The typical physical structure we will consider is a sandwich of active phase-change material between two metal electrodes, as depicted in Fig. 5.1a. A heating element, usually fabricated from doped TiN, acts as a contact from the bottom electrode to the active region and shows higher electrical resistance and lower thermal conductivity than typical metals. Current flowing through the heater raises the temperature of the small volume of material above it through Joule heating. Once the melting temperature is reached, (depending on the actual composition, but usually in the range of $400-800\,^{\circ}\mathrm{C}$) the phase-change material liquefies. Controlling the cooling profile, the final crystalline state of the molten volume can be tuned. An abrupt temperature drop freezes the atoms in a disordered, highly resistive, amorphous state, also defined RESET. If instead the current is decreased gradually,

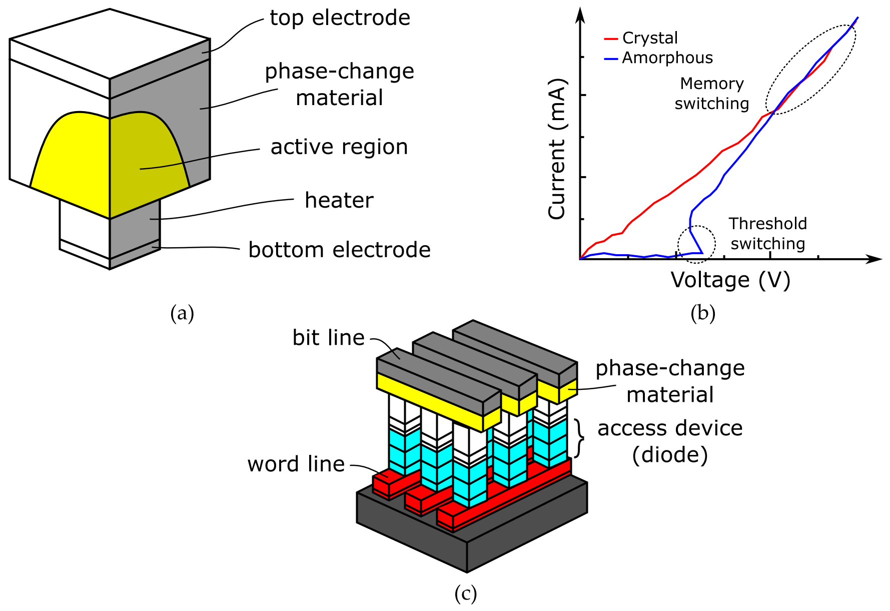


Fig. 5.1 (a) Elements of a mushroom-type PCM cell. (b) Typical electrical response of PCM cells programmed in either the RESET or SET states. (c) .

crystal nucleation and growth allow the crystallization of the volume, with a lower final resistance. The crystalline properties of this state, defined as the SET state, vary based on the parameters of the electric stimulus being applied. A trapezoidal pulse of high intensity creates a polycrystalline volume, having several grains. A low intensity pulse leads more easily to a crystalline column of material passing through the amorphous dome that forms on top of the electrode. A series of pulses of decreasing amplitude results in a polycrystalline structure, having large grains in the active area of the device. This dependence of the final crystalline state on the qualities of the programming pulses potentially allows a fine-grained control of the programmable volume, enabling multilevel programmability and is an actively researched topic.

An I/V characterization of a typical PCM cell highlights one of its peculiarities, a phenomenon called *threshold switching*, clearly visible in Fig. 5.1b. At sufficiently high bias voltages the RESET state I/V curve collapses into the SET state one, with a significantly lower resistance. This is a purely electronic phenomenon and does not induce any permanent change in the crystalline lattice. Indeed, removing the bias restores the original device state. The root causes explaining the threshold switching phenomenon are not yet understood [75]. Some hypothesis include recombination through trap levels with generation driven by electric field, or the creation and expansion of conductive filaments that act as shorts bypassing the highly resistive amorphous region. Notwithstanding the alternative explanations, threshold switching is extremely advantageous to reprogram the device, as it allows higher current levels even through cells in RESET state without requiring extremely high bias voltages.

The non-volatile, reversible state change is instead called *memory switching*. It enables the encoding of information in the crystalline state, which can be recovered through a low-voltage readout to avoid excessive perturbation of the state itself.

Phase change material can potentially enable extremely scalable arrays through architectures of the type depicted in Fig 5.1c. A thin-film chalcogenide layer is deposited continuously on top of a row of heating elements, with different active volumes insulated by the same material in the amorphous state. Using as a metric the *minimum cell half-pitch* F, i.e. half the distance between identical element in a uniform array, the minimal device area would be $4F^2$. The scaling limitation stems both from data retention (a smaller active volume more rapidly relaxes back to a crystalline state), and the necessity of having access devices capable of supplying the RESET current pulses. BJT selectors are more compact than MOSFETs, with areas of $8F^2$ as compared to the $20F^2$ of MOSFETs, though the advantage of an easy integration to CMOS system-on-chip solutions with only a few additional masks makes MOSFET technology extremely more valuable. As a reference, embedded flash memories require 10+ additional masks in the Back End Of the Line (BEOL) [75].

Typical implementations subdivide the entire PCM array into smaller tiles so that leakage, power-consumption and speed can meet specifications. Array tiles can be than grouped in partitions and planes, with several tiles sharing the same sense amplifiers. The IR voltage drop along the bit- and word-lines plays a major role in limiting the size of the array tiles, with consequences on the programming voltage, which has to compensate for the drop along the interconnection, a higher deselection voltage on the word line and, in general, unbalances observed in the cells farther from the line drivers. As the size of the tile increases, so does the capacitive coupling between adjacent lines, slowing down the readout and adding latency.

One of the major drawbacks of phase-change materials is that the amorphous volume experiences structural relaxation over time, leading to a resistance increase described by the power-law relationship [75]

$$R(t) = R_0 \left(\frac{t}{t_0} \right)^{\nu} , \tag{5.1}$$

with R_0 and t_0 constants, and the drift exponent $\nu \simeq 0.1$. These parameters depend both on temperature and device aging. The drift phenomenon is a minor issue in binary digital applications, as the two states, SET and RESET, diverge over time. For the use as multilevel cells this is, however, one of the major challenges to be addressed. Indeed even if the (full) SET state does not experience significant drift, partial SET states achieved with appropriate programming sequences still containing residual amorphous volumes of varying size experiencing significant resistance variations over time.

The only general consensus on the causes leading to the resistance drift is that it is due to some kind of relaxation. Other than that, conflicting theories have been proposed, considering the disorder, defect density or stress as the root causes of the phenomenon [75]. A correlation to the presence of grain boundaries has been identified, hence an improvement is to be expected if low-current programming currents are employed. The specific material composition also affects drift, with Ge-rich alloys enhancing the thermal stability of the RESET state.

Time also plays a role in the progressive change of the chemical composition of the active volume. Indeed, each of the chemical species composing the phase-change alloy experience different phenomena, leading to the independent migration, even in opposite directions. As devices age, Sb atoms move towards the electrode, while Ge is pushed out into the solid phase. As a consequence the molten region remains Sb-rich and Ge-depleted, with a progressive change of the maximum achievable SET conductance, and leading to reliability issues in the long term.

In general, device performance is strongly dependent on the PCM material properties. A high contrast between logical states immediately maps to the resistance difference between the amorphous and crystalline states. Low power consumption requires lower melting temperature, but a not-too-low resistivity in the crystalline state to reduce the RESET current. The crystallization speed determines the switching (reprogramming) speed. Data retention is affected both by the crystallization temperature and the drift exponent in the amorphous state. Long term endurance needs lower variations of volumetric density between amorphous and crystalline

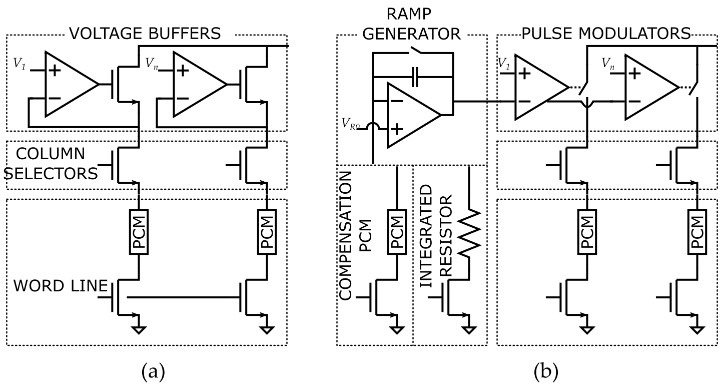


Fig. 5.2 Architecture of the (a) Mk-I and (b) Mk-II PCM chips.

states. Finally, high scalability requires stronger control on the interface effects that become significant at small volumes.

Through scaling, energy consumption can also be reduced by requiring significantly lower currents during device programming, e.g., from melting current over 500 uA at the 90 nm node (heater diameter 56 nm) to under 100 µA at the 16 nm node (heater diameter of 10 nm) [75].

5.2 Available HW Platform

To evaluate the requirements, limitations and overall feasibility of energy-efficient, PCM-based data processing schemes, we have used measurements from two different AIMC implementations [76, 77]. The core PCM devices had the same technological properties and the main differences were in terms of the surrounding circuitry, as highlighted in Fig. 5.2. Both hardware architectures were the result of a collaboration between the design team at STmicroelectronics and a research group based at the University of Bologna. They provided us with extensive measurements for the characterization of the major performance parameters of the specific implementation, with a focus on the PCM devices.

In the following we will alternate between results obtained with the first design iteration, hereafter denoted Mk-I, and the second iteration, Mk-II. Each design had its own evaluation board, usable through a custom graphical user interface to define repeatable programming and data-acquisition procedures.

The Mk-I test vehicle [76] is a 2-Mb PCM memory array, realized in a 90-nm BJT-CMOS-DMOS (BCD) technology for automotive applications. It employs a crossbar 3D Memory, with long lines of a Ge-rich GST compound. The chip includes 8 independent 256 kB macrocells, with a PCM elementary cell based on an NMOS selector and occupying $0.19\,\mu\text{m}^2$ of silicon area. The programming-pulse generator includes a 6-bit current-DAC and multi-output current mirror. Their combination allows both the shaping of the time-domain currents being used to program the devices, as well the ability to simultaneously program multiple cells. As RESET current pulses have a much higher intensity, the maximum parallelism is more limited (20) as compared to the one achievable during SET programming (80).

From our point of view, the major feature of the Mk-I chip is the possibility to drive the PCM cells through the direct application of an analog voltage, as shown in Fig. 5.2a. The consequences of this driving setup will be analyzed later on in this chapter.

The Mk-II chip takes a different approach in the use of PCM devices. They are no more driven by an analog voltage. The external inputs are now encoded as constant-amplitude, width-modulated voltage pulses [77], obtained by comparing the external inputs to a voltage ramp.

The reference ramp itself is generated by integrating a constant current flowing through a resistive element, being either an integrated resistor or a PCM cell. Selecting the latter enables an effective drift-compensation strategy, since the output of a single MAC becomes dependent on a ratio of PCM conductances. This is a well known principle in circuit design, for ratios of homogeneous quantities inherently compensate variations common to the individual elements. The effectiveness of this compensation strategy stems from the fact that, as the PCM cells within the analog array become increasingly more resistive over time, the current flowing through them decreases[1]. The reference PCM cell, becoming more resistive as well, decreases the slope of the voltage ramp, enlarging the duration required to reach the value of the externally supplied input voltage. To first order, the integration of smaller currents over a longer time compensates the drift.

Two drawbacks can be identified with this compensation scheme. The first concerns the inherent variability of the reference cell. Though the ratio of homogeneous elements is advantageous to cancel out common perturbations, the cell-to-cell pro-

[1]Remember that the PCM cells in the Mk-II chip are driven by a constant voltage.

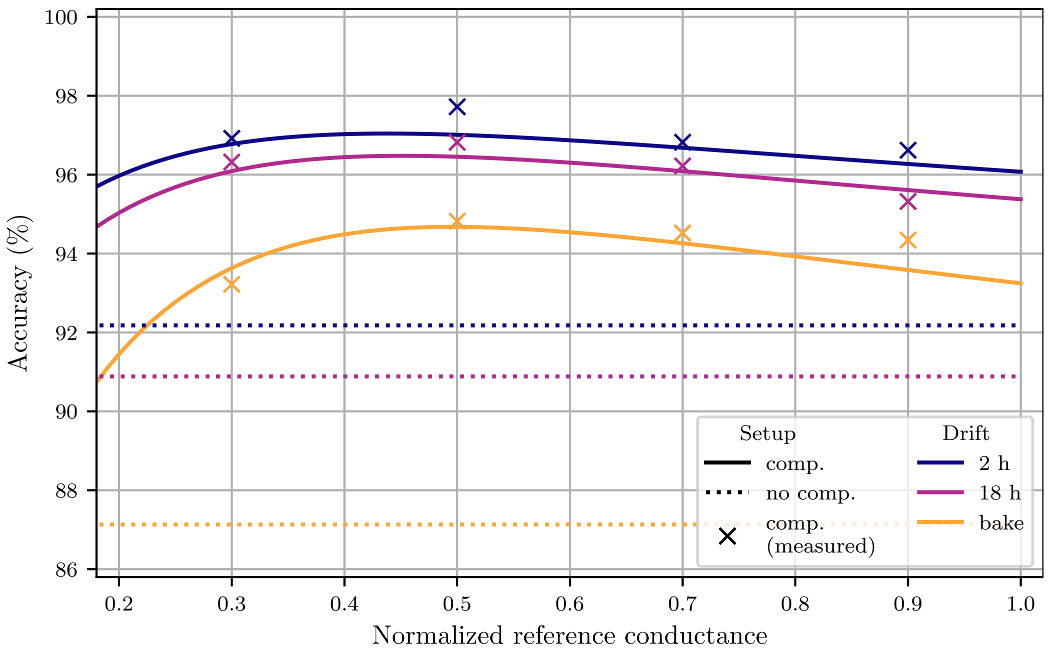


Fig. 5.3 MAC accuracy as a function of $\frac{g_{\mathrm{REF}}}{g_{\mathrm{MAX}}}$. Continuous lines refer to simulated results after 2- and 18-hours room temperature and 90°C bake drift. Dashed lines represent the MAC accuracy in the same conditions when no compensation is adopted. Crosses report the results of experimental evaluation of MAC accuracy for different g_{REF} levels.

gramming error cannot be compensated exactly, especially since for convenience we have a single reference conductance for every row of the analog array. Then, in a multilevel use case, one reference cell is shared by a row of differently programmed conductances and it has been observed how the drift exponent shows a dependency on the actual programming state of the device. Therefore the reference conductance can only be tuned so that *on average* it performs well. Fig. 5.3 indeed shows how different values of the reference conductance lead to different average performance in 10k random MAC operations, with the optimum normalized range being $[0.4-0.6]$. A direct comparison of the compensated and uncompensated accuracy under different drift setups can be seen in Fig. 5.4, with the compensated cells maintaining their performance over time.

5.2.1 PCM Programming

The unpredictability of crystal nucleation and growth currently prevents the accurate programming of arbitrary PCM conductance values. The trivial application of

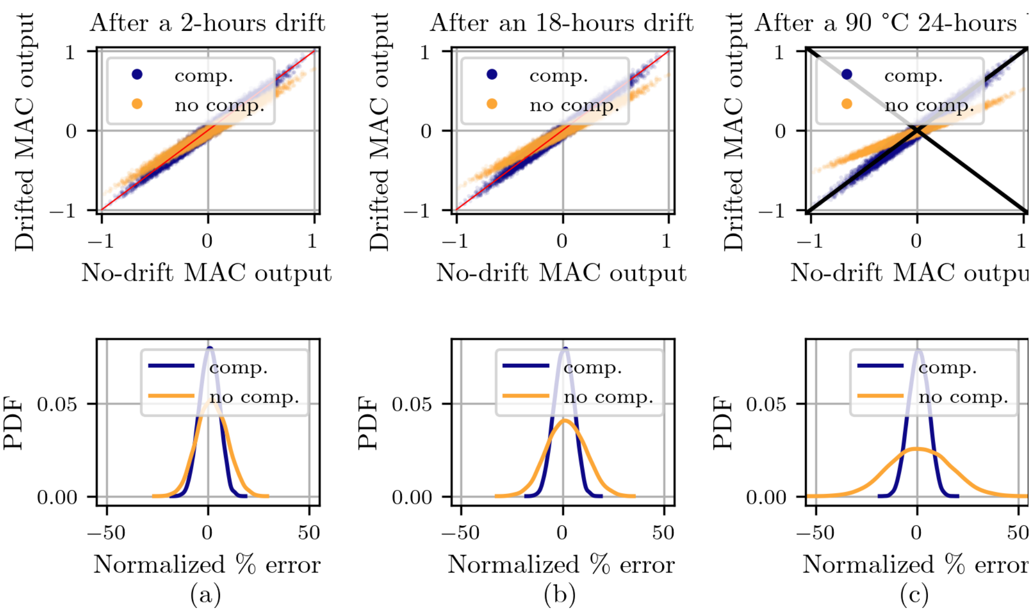


Fig. 5.4 (Top) Comparison of the MAC output for compensated and uncompensated cells (a) after 2 hours from programming, (b) after 18 hours and (c) after a 24-hours bake at 90 °C. (Bottom) Distribution of the MAC error ε.

programming pulses of different intensities, without any feedback to monitor the actual states of the programmed devices is only sufficient in binary applications, provided that some margin is guaranteed between the RESET and SET states.

Program-and-verify algorithms iteratively stimulate and characterize PCM cells to ensure their conductance falls within a given tolerance of some target value. Part of the measurements used in this work has been obtained from PCM cells programmed through a variant of the SET-staircase technique defined in [78] and whose details are depicted in Fig. 5.5. A hard RESET pulse is applied to the cell, bringing its conductance to its lowest value, and progressively increasing it by partial SET pulses, until its value is acceptable.

The feedback control currently requires digital processing, but one could easily foresee an entirely analog loop, with an elementary state machine controlling the evolution of the programming algorithm based on some external inputs, like target value and tolerance.

Through this iterative programming algorithm, up to 16 non overlapping levels have been obtained. Their cumulative distribution functions are shown in Fig. 5.6.

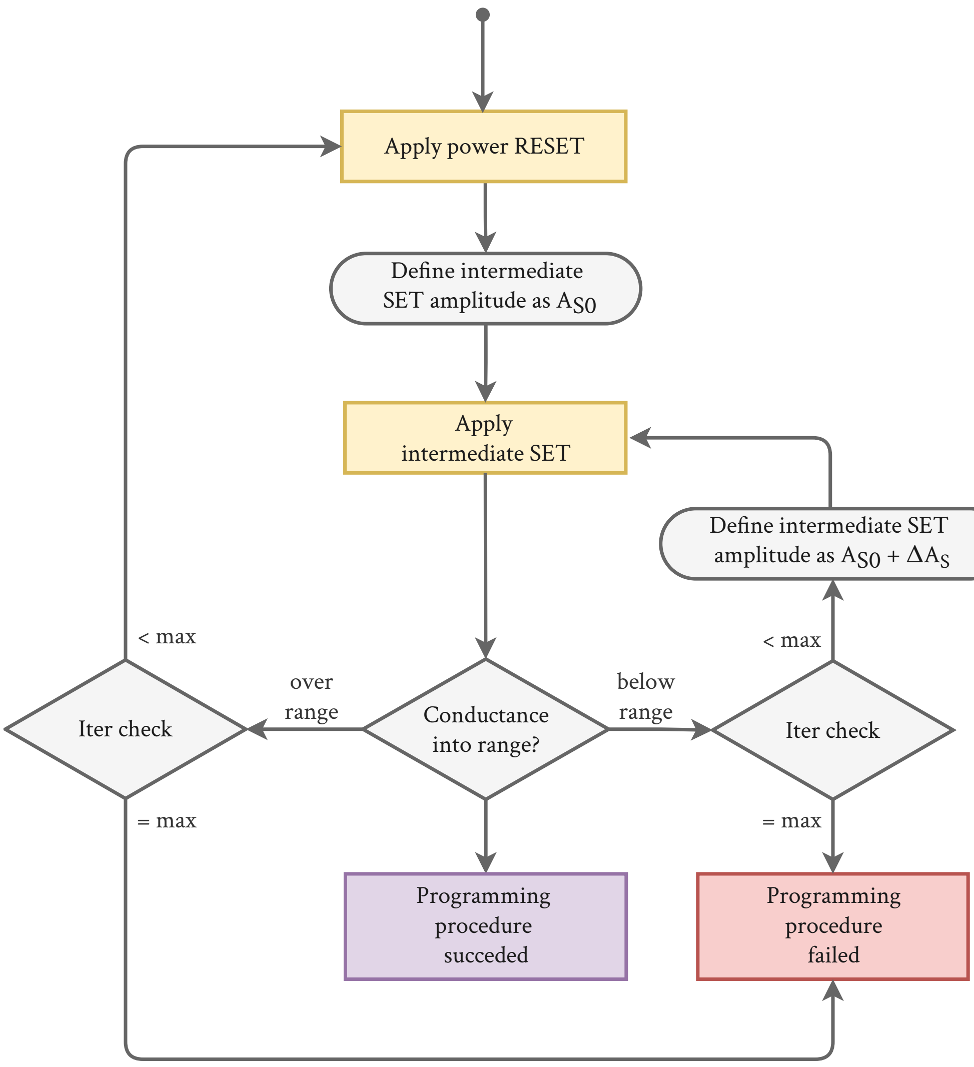

Apply power RESET
Define intermediate SET amplitude as A_{S0}
Apply intermediate SET
Define intermediate SET amplitude as $A_{S0} + \Delta A_S$
< max
over range
Conductance into range?
below range
< max
Iter check
Iter check
= max
= max
Programming procedure succeded
Programming procedure failed

Fig. 5.5 Outline of the cells programming algorithm.

5.3 Conductance Measurements and Processing

The Mk-I chip has been characterized by programming 5120 cells through one-shot SET pulses. Different data sets have been acquired for different choices of pulse amplitudes. Immediately after programming, a DC voltage has been slowly swept across the cells to obtain their I/V transfer curves, depicted in Fig. 5.7a.

As it is clearly visible, the intensity of the SET pulses is not an accurate metric to predict the programmed conductance. Therefore, to extract typical behaviors, a different approach has been selected. Since our main goal is the evaluation of

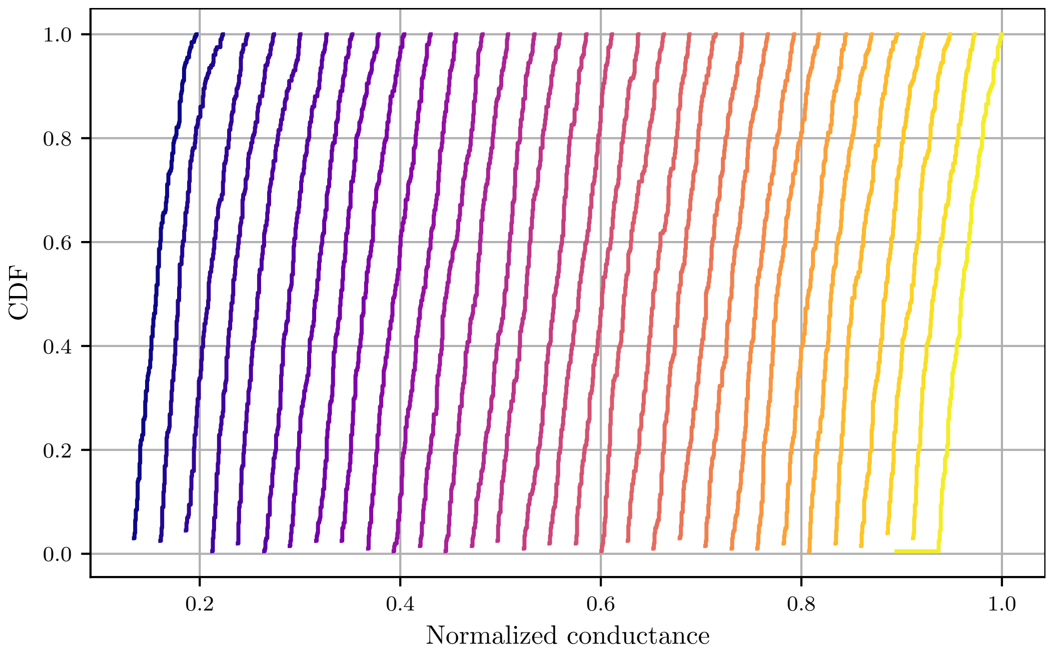


Fig. 5.6 Cumulative distribution functions of cells programmed to 32 equispaced target conductances.

whether PCM technology represents a valid candidate for the implementation of energy efficient data processing schemes, the device-level metric of interest is the conductive state of the PCM cell, independently of the procedure applied to achieve it. This is actually justified in practice by the existence of iterative programming algorithms that, as previously stated, can guarantee accurate programming within a given tolerance. As a consequence, the same dataset of I/V curves has been interpreted in a different way. The programming state has been defined as a grid of quantized currents, obtained at a reference voltage of $v_{ref} = 0.3\,\text{V}$. All the measured I/V curves falling within a $\pm 5\%$ of one of the grid points contribute to an averaged, nominal behavior of an ideal PCM device. The result is a new categorization of the same curves, as shown in Fig. 5.7b, and immediately leads to the *nominal* device curves in Fig. 5.8a. The 3-D space depicting the measurements changes from the (SET pulse intensity, applied voltage, device current) coordinates, to the (current bin, applied voltage, device current) ones.

A selection of three such average behaviors is shown in Fig. 5.8a, chosen to highlight significantly different conductance behaviors. The current profiles have been normalized with respect to the maximum value of the state #1 curve. The saturation observed at $v > 0.8$ is due to the access transistors entering the saturation

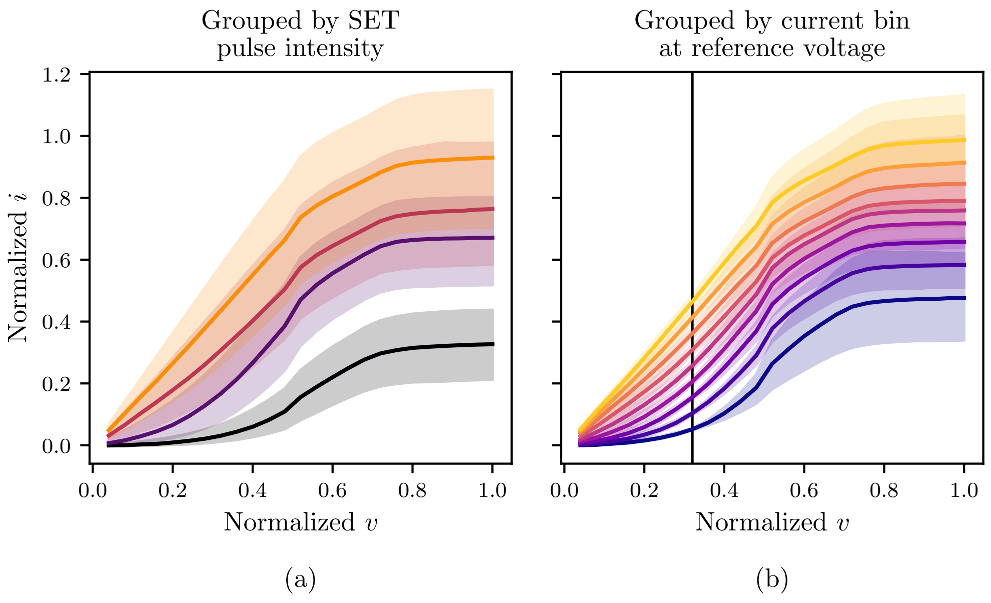


Fig. 5.7 Alternative logical interpretation of PCM DC characteristics in different programming states. (a) Grouping by programming pulse intensity. (b) Grouping by current level at reference voltage (vertical line).

region. Even though not an inherent feature of the I/V response of an isolated PCM cell. It is taken into account as an unavoidable issue in the specific hardware architecture selected for this specific chip. Consequently, conclusions will be drawn taking it into account.

The three curves depicted in Fig. 5.8a will be considered as the nominal i(v) behavior of typical cells in three different programming states, denoted as #1, #2 and #3. Fig. 5.8 also shows the large-signal, normalized conductances defined as $g = i/v$. Notice how an ideal resistor would result in an horizontal line in Fig. 5.8b, but only the curve associated to state #1 gets close to it in the lower half of the voltage domain. The voltage dependency of the cell conductance has extremely detrimental effects in applications where the exact values has to be known, as we will analyze in Section 5.4.2.

The Mk-II chip allows individual programming of the array cells, but the readout is only available through the computation of a MAC operations. The extraction of single-cell measurements therefore requires in this case performing a MAC operation with only one nonzero coefficient. The devices on this chip were all programmed through the iterative programming procedure of Fig. 5.5.

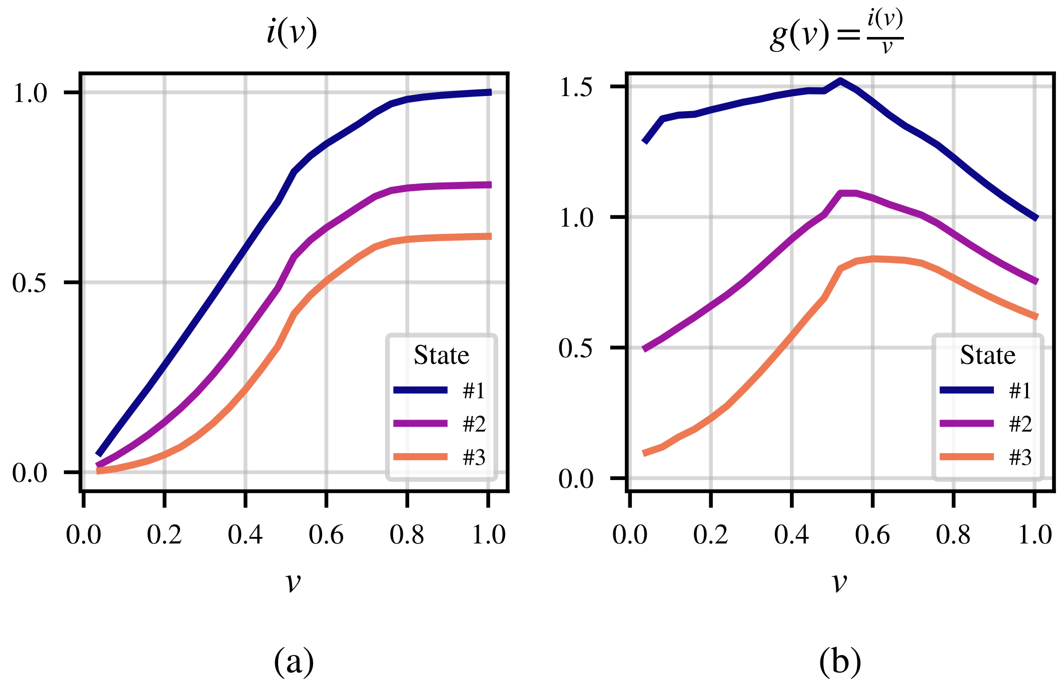


Fig. 5.8 (a) Average, normalized i/v characteristics of PCMs in three different states. The curves have all been normalized with respect to the maximum current in state #1, I_0 ($i = I/I_0$). The applied voltages have been normalized as well with respect to the maximum applied voltage V_0 ($v = V/V_0$). (b) Large-signal, normalized conductance $g = i/v$.

As this architecture encodes inputs as constant-amplitude pulses, the nonlinearity observed in the I/V curves of the previous setup is now absent. Each device operates at the same operating point for a time proportional to the acquired input.

To summarize, two hardware platforms have been used to characterize phase-change devices.

For the Mk-I chip:

- Applied voltages are continuous in nature, leading to the observation of a nonlinear I/V dependence.
- Programming is performed by one-shot current pulses of controllable intensity.
- The cell programmed state is extracted from their vicinity to an arbitrary quantization grid, neglecting the amplitude of the programming pulse.

Conversely, for the Mk-II chip:

- Inputs are encoded as the duration of constant-amplitude voltages pulses.

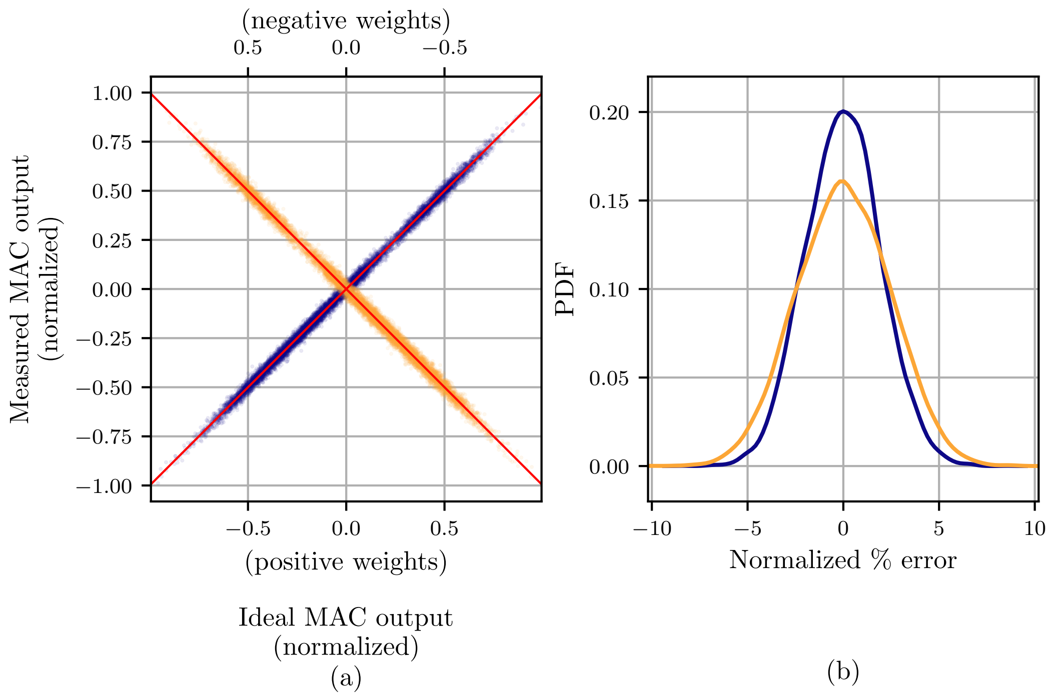


Fig. 5.9 Accuracy of the AIMC unit: (a) Measured MAC operations as a function of the ideal MACs, where MAC coefficients are implemented with programmable integrated resistances. MAC weights w_i employed for the yellow (purple) data are negative (positive). (b) MAC error distributions of the two data sets.

- A hardware drift compensation scheme is introduced, introducing the output dependency on the ratio of PCM conductances
- An iterative programming procedure is applied to have a finer control of the cell conductance.

5.4 Device Modeling

In this section we will construct numerical models describing the most relevant features of PCM cells, eventually including effects from the surrounding circuitry. The sole objective of the models is to provide a way of rapidly verifying whether real-world applications could benefit from the use of PCM analog arrays, notwithstanding their nonidealities. Consequently, the models will not be concerned with the underlying device physics, but will only describe the phenomena observed at their terminals, specifically for the technology being employed and with a description suitable for the application being evaluated. We will cover the general procedures

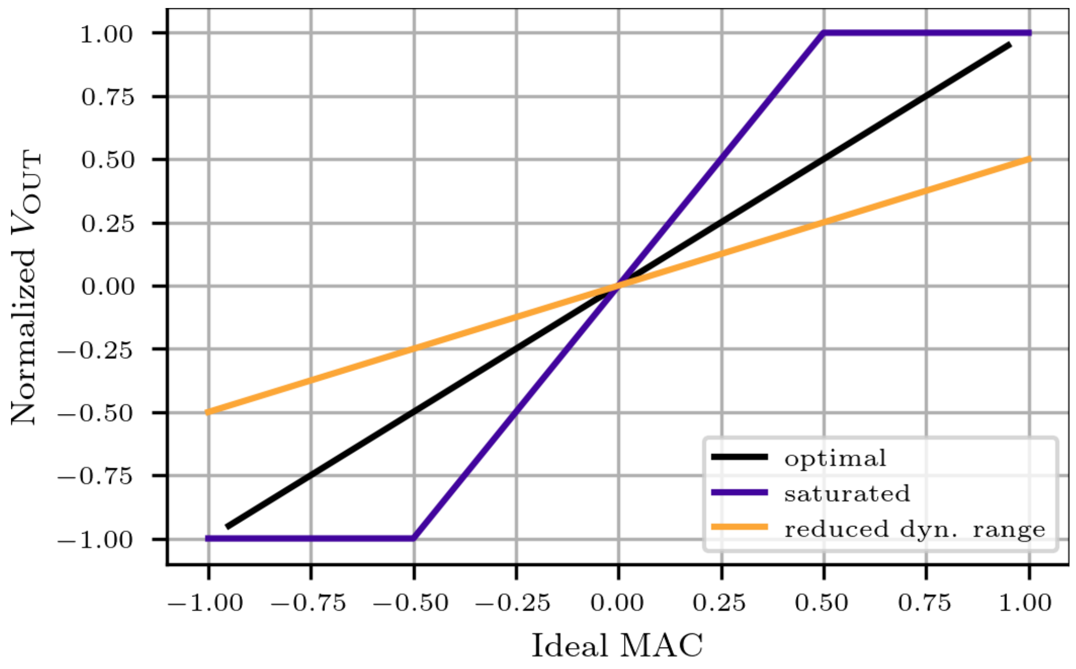


Fig. 5.10 Effects of different values for the reference conductance g_{REF} on the output swing of the MACs. *i*) optimal swing (Equation (6)); *ii*) saturation due to low g_{REF} level; *iii*) swing reduction induced by PCM cells drift with constant g_{REF}.

being used, as well as the results, but it is important to keep in mind that for us the models only represent a mean to an end. As such, we will refer to them as "Application-based models".

For the Mk-I chip, having the possibility of applying analog voltages to the elementary PCM cell, splines and polynomials of arbitrary degree will be employed to model the nonlinear I/V curves observed in Section 5.4.2. With the newer Mk-II chip, the nonlinearity disappears by virtue of the time-domain encoding of the external inputs, and the focus is on the dependencies of both programming spread and drift over time on the nominal conductance value. This data will be fitted either by a `tanh` function or a degree-3 polynomial. In the following we will go through the details and the caveats of the modeling procedures employed.

5.4.1 Application-based modeling

The work is a result of an increasingly better understanding of the possibilities offered by the technology, more complex applications being evaluated, a better underlying hardware (since the Mk-II chip has been available). The complexity of the models have followed the development.

Binary compressed sensing (CS) encoder

A PCM analog array implements the matrix-vector product between an appropriately selected weight matrix (the encoder) and a window of the input signal to be acquired. Being the matrix elements, i.e. the programmed conductances, either zero or (logically) 1, there is no need to model the dependency of PCM conductances on their programming state in a continuous manner, and two I/V curves obtained in specific programming states are sufficient (one for the logical 0, the other for the logical 1). Therefore different representative I/V curves for PCM devices in discrete programming states have been computed by averaging subsets of programmed cells behaviors as shown in Fig. 5.11a.

Deep neural networks (DNNs)

The PCM array implements the matrix-vector product at the core of a neural layer. Since the layer weights have to be trained for a specific task, the model has to be compatible with standard training procedure. The most commonly used is the *backpropagation* [79]. It requires a model differentiable in terms of the matrix weights, so that the weight updates can be computed to minimize an appropriately selected loss function, computed on the output. Additionally to the continuous $I(V)$ description required by the previous application, there is now the need for a differentiable $I(V, \mathrm{state})$ description. The definition of a state dimension through the binning of measured I/V curves described in Section 5.4.2 allows the construction of a three-dimensional model, through the interpolation of the available datapoints by two-dimensional splines and polynomials as depicted in Fig. 5.11b.

Binary CS and DNNs with time-encoded inputs

Overall, the applications remain the aforementioned ones, though wider and deeper DNNs are now used to solve more complex tasks. The conductance nonlinearity is avoided by operating the PCM devices at constant voltage and encoding the information as pulse durations. The focus is now on the accuracy of the programming procedure and the conductance drift over time. Models for the programming spread, mean drift and drift spread are developed. Due to their visual appearance, standard deviations versus target conductance are modeled using a scaled and shifted `tanh` function, while the mean drift uses a degree-3 polynomial description. All the models are only function of the nominal conductance, i.e. the target used by the programming algorithm. The three modeled terms are shown in Fig. 5.11c.

A summary of the applications being evaluated, their requirements and the models being used is presented in Table 5.1.

5.4.2 Numerical models for voltage-domain inputs

The starting point for the construction of a three-dimensional model is the averaged I/V curves obtained in Section 5.3 and depicted for convenience in Fig. 5.12a. Their intersection with the vertical line at $V = 0.4\,\text{V}$ represents the nominal state of the programmed conductance and allows us to extend the view into a 3-D plot, as shown in Fig. 5.12b. Once a conductance state has been identified, this description would already be useful for the initial use-case of a binary Compressed Sensing encoder, by interpolating the I/V datapoints of a single curve. Nevertheless we will immediately proceed to the derivation of the complete 3-D numerical model and use that to extract the required curves even in the case of the binary application.

The search for a simple and efficient numerical interpolation of the available data readily points towards spline representation [80]. They define a piecewise sequence of low-degree polynomials, constructed so that the transition between curve segments is smooth. Their main advantage is that, through a low degree approximation of a limited region, they prevent the formation of artifacts observed with high order models. We will select splines as the reference model of our nominal PCM conductances in the Neural Network test-setups to be analyzed later on.

However, we will not be able to use them during training, since at the time of this work no spline primitives were available in the commonly used software libraries for Neural Network applications, namely Keras and Pytorch. Hence the need to find a different description, more suitable for that specific task. Our selection has been polynomials, both for their computational simplicity and their universality.

A limitation immediately stands out. The peculiarities of the surface to be interpolated require especially high polynomial degrees, which results in the insurgence of

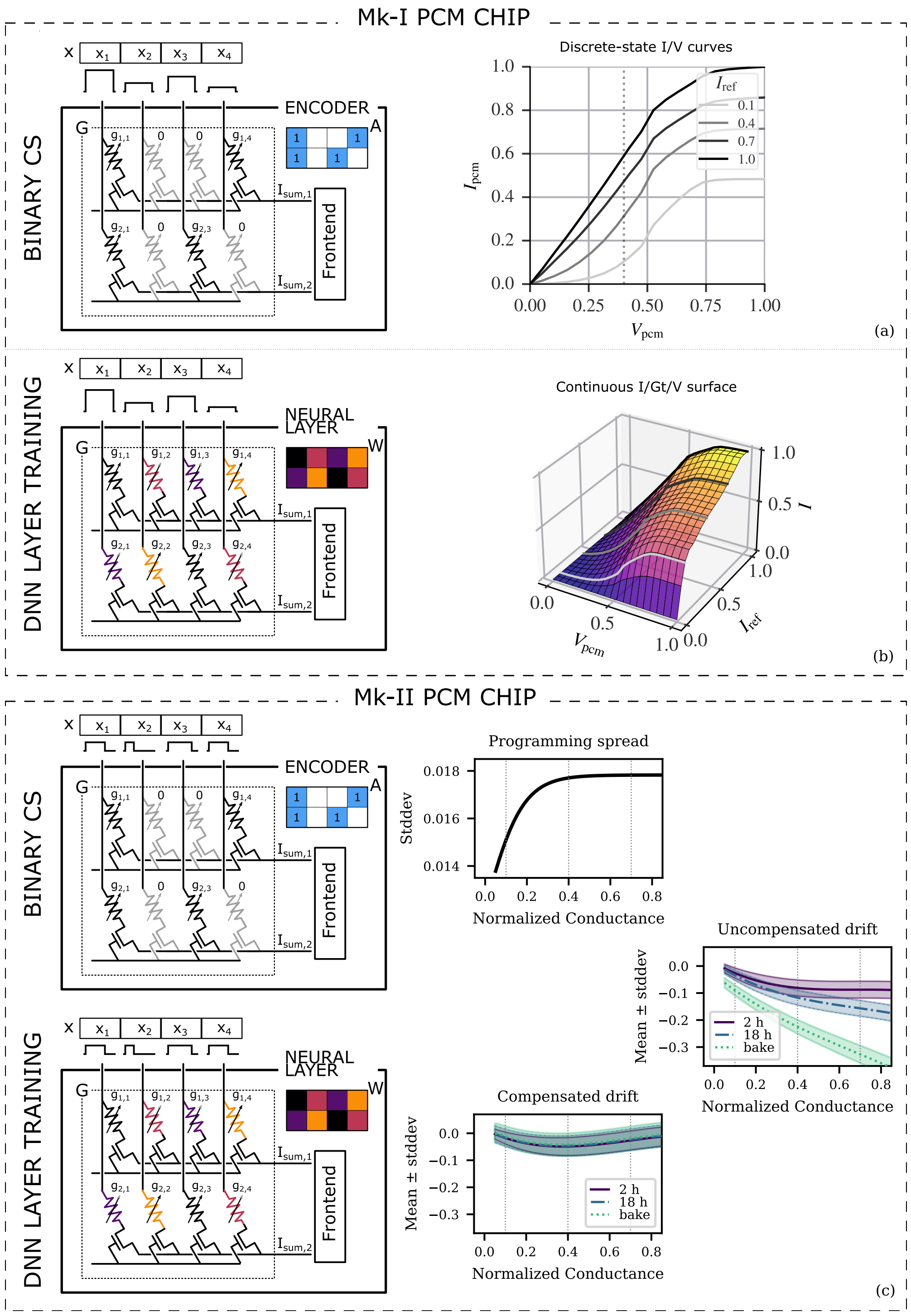


Fig. 5.11 Practical use cases and their respective PCM models. (a)-(b) Voltage-domain PCM inputs. (a) Discrete I-V curves are used for the binary CS application. (b) A continuous surface for the training of DNNs. (c) Time-domain PCM inputs. Programming and drift-induced variabilities are used for both the CS and DNN applications being tested.

Application	Requirements	Conductance Model
Binary Compressed Sensing (CS) Encoder	Matrix-vector products with matrix entries either 0 or 1	Averaged I/V curves in a specific state
Deep Neural Networks (DNNs)	Continuous model of the synapse output as a function of synapse weight (PCM conductance) and external input (applied voltage)	2D spline and polynomial interpolation of the averaged I/V curves including a continuous dependency on the programming state
Binary CS and DNNs with time-encoded analog array inputs	Estimation of programming spread and drift over time versus the conductance state of the device	Programming error stddev: tanh Drift mean: `poly` Drift stddev: tanh

Table 5.1 Applications being evaluated in this work and corresponding requirements and properties of the models extracted from the conductance measurements.

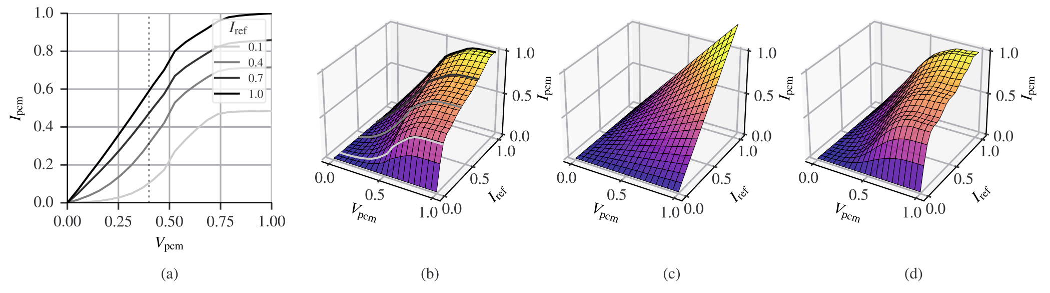


Fig. 5.12 (a) Average, normalized *I*–*V* characteristics of PCM devices in four different states. (b) Spline-based interpolation of the average, normalized PCM behaviours, highlighting the four states depicted in (a). (c) Low-order polynomial fitting of the same data. (c) High-order polynomial fit.

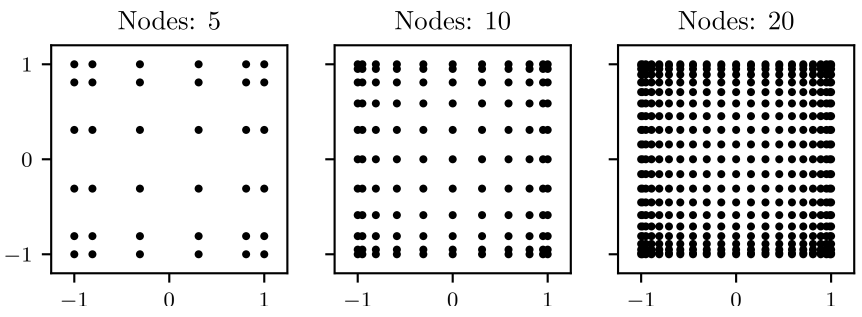


Fig. 5.13 Chebyshev grids to prevent the insurgence of Runge's phenomenon in high-order polynomial fit.

Runge's phenomenon at the boundary of the domain if the interpolation is performed on a uniformly-spaced grid [81]. At the application level, since the backpropagation algorithm used to train Neural Networks has to glide over the surface, ringing introduces nonmonotonicities which may be detrimental when finding optimal weight values. The problem has been thoroughly studied, and a robust mitigation strategy is to select the sampling point for the interpolation as the zeroes of Chebyshev polynomials, whose value can be computed as the projection on the horizontal axis of a uniformly sliced half circle:

$$x_k = \cos\left(\frac{2k-1}{2n}\pi\right), \qquad k = 1, \ldots, n\,.$$

The corresponding grid of sampling points becomes more dense at the boundary, as shown in Fig. 5.13.

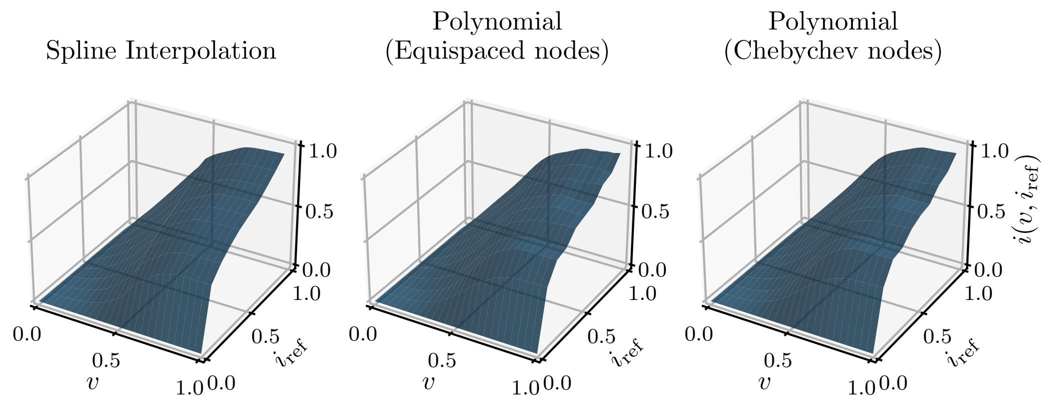


Fig. 5.14 Comparison of the numerical representations of the same datapoints, using a spline interpolation and a polynomial one both on an equispaced and a Chebychev grid.

Since we had to work with the data we already had, being acquired on a uniform grid, we started from the spline model, considered as our reference, and resampling it over the previously defined Chebyshev grid, using a grid resolution dependent on the degree of the polynomial. The resulting surfaces are shown in Fig. 5.14 During training, the degree of the polynomial can be considered as a hyperparameter, with lower degrees leading to faster trainings, though with a potential performance penalty.

As the polynomials would be called a gazillion times during a single NN training operation, the grouping and reuse of elementary terms can save an enormous amount of computational resources, with immediate consequences on training time. To this end, we have taken advantage of Horner factorisations of multivariate polynomials [82], as implemented in the `multivar_horner` Python package. [2]

Horner factorisation groups individual terms in the expanded polynomial description, to reduce the number of mathematical computations and increase the numerical stability [82]. The uniqueness of Horner factorization however, only applies to univariate polynomials, where their canonical form

$$f(x) = a_0 + a_1 x + a_2 x^2 + \ldots + a_d x^d$$

[2]https://pypi.org/project/multivar-horner/

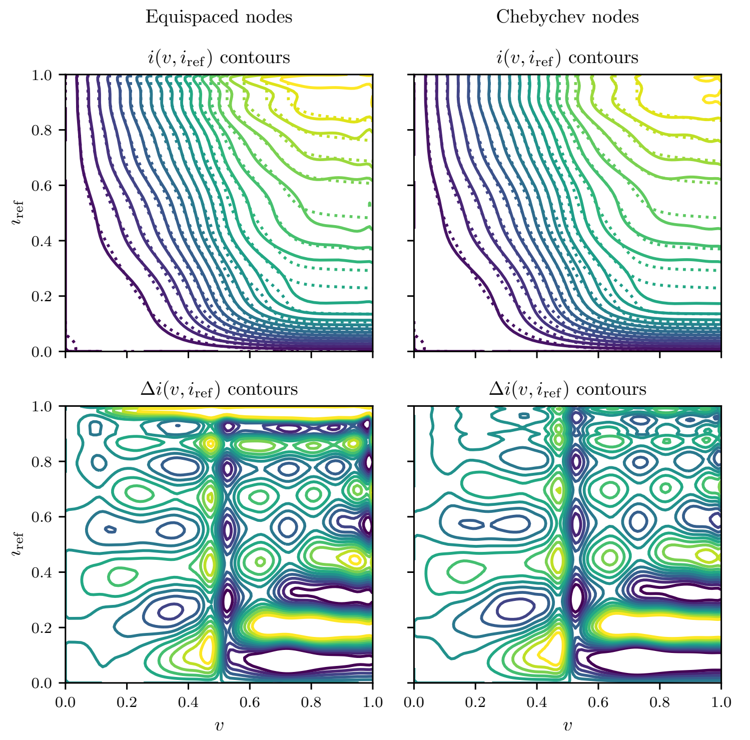


Fig. 5.15 Level curves to appreciate the advantage of a Chebychev grid. (top) Level curve of the PCM current. Dashed lines refer to the spline model. (bottom) Level curve of the difference between the polynomial and spline model. Notice the stronger oscillations observed at the boundaries of the plots.

is readily transformed into

$$f(x) = a_0 + x(a_1 + x(\dots x(a_d)\dots)).$$

The library uses a recursive procedure to obtain good factorisations in the multivariate case. At each iteration, the most common factor among all monomials is factored out. The stability advantage is not really relevant in our context, as it is truly observable only when the number of coefficients is above 100 (Fig. 3 in [82]). Conversely, the number of operations can be reduced 10-fold even for smaller polynomials.

The methodology presented here, summarized as:

1. Data acquisition and processing
2. Spline fitting for a locally-accurate low-order description
3. Chebyshev resampling to mitigate Runge's phenomenon when using high-order models

4. Polynomial fitting for use in applications where spline primitives are not readily available,

could be applied to arbitrary devices, for which specific software libraries are still lacking, or to construct models based on specific measurements.

5.4.3 Numerical models for time-domain inputs

The improvements in the Mk-II PCM chip made the previous modeling effort irrelevant, since the individual PCM devices are operated at a constant voltage. At the same time, the programming procedures being used have improved, with program-and-verify techniques now available and guaranteeing a stricter control on the actual device state.

Our focus, both to characterize the properties of the physical devices and understanding the resilience of the end applications, moved towards the quantification of the spread observed after programming, as well as the monitoring of conductance changes over time.

A set of 450 cells has been programmed repeatedly, each time to one of 32 target levels, with a tolerance of 5%. The cells have been characterized by performing a MAC operation[3] involving a single device The characterization has been repeated over time, monitoring the evolution of the individual conductances both with and without the hardware drift compensation strategy described in Section 5.2. The quantities to be modeled are: the programming variability Δg_p, describing the residual spread of conductances programmed within a given tolerance, measured after a few milliseconds from programming, and modeled as a zero-mean gaussian random variable at every target conductance point. The drift conductance variation, $\Delta g_d(t) = g(t) - g(t_0)$, modeled as a gaussian distribution whose mean and standard deviation are both function of the nominal conductance. Standard deviations, both for the programmed conductances and the drift-induced spread, are described by:

$$\sigma(g) = \sigma_0 + \sigma_1 \tanh(g/\gamma_0). \tag{5.2}$$

[3]the most elementary operation available on this AIMC chip is a MAC operation, involving multiple PCM cells, thus to characterize a single cell, all the remaining ones within the same MAC group have to be RESET

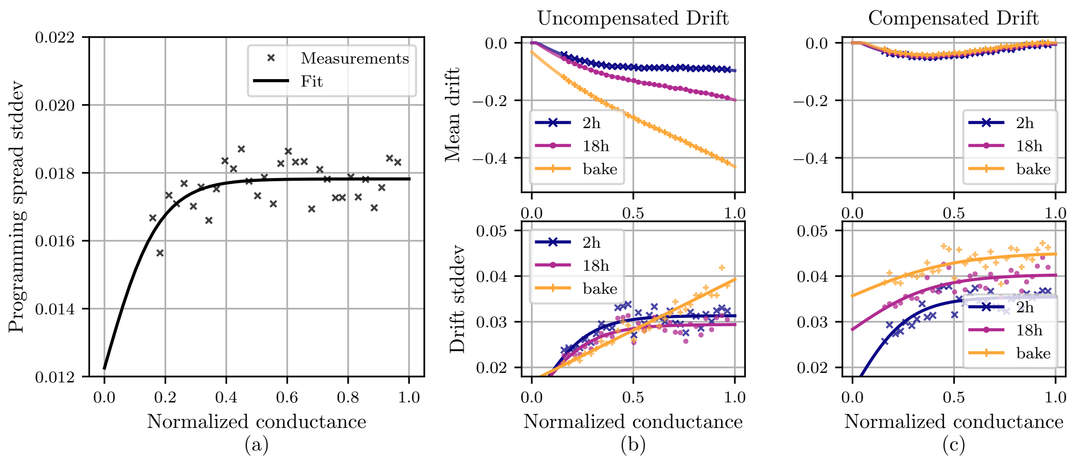


Fig. 5.16 (a) Standard deviation of the spread resulting from the iterative programming procedure, as a function of the average conductance of each programmed level. (b) Mean and standard deviation of the drift-induced conductance variation for cells without compensation and (c) with compensation.

Note that the model is not continuous over time, and different coefficients for the numerical fit are associated to different drift times/conditions. Conversely, the mean drift $\mu(g)$ trend has been described by polynomials of degree 3. A least squares fit using the Levenberg-Marquardt algorithm has been used to find the parameters σ_0, σ_1 and γ_0 and the polynomial coefficients. The models are all shown in Fig. 5.16.

5.5 PCM-based Compressed Sensing Encoding

This section explores the tradeoffs observed while implementing Compressed Sensing (CS) encoding schemes using PCM-based analog arrays. At the same time, appropriate decoding schemes are either proposed or identified in the existing literature to better cope with the nonidealities of the encoding platforms.

We will be focused on binary CS encoding as it determines a marginal performance drop when sensing matrix entries are constrained in a discrete set, e.g. binary $\{0,1\}$, antipodal $\{-1,1\}$ or ternary $\{-1,0,1\}$ matrices [33]. Moreover, since architectures where positive-only conductances can have a negative effect on the output require additional circuitry, e.g. current mirrors [76], this also leads to the most compact and efficient hardware implementation.

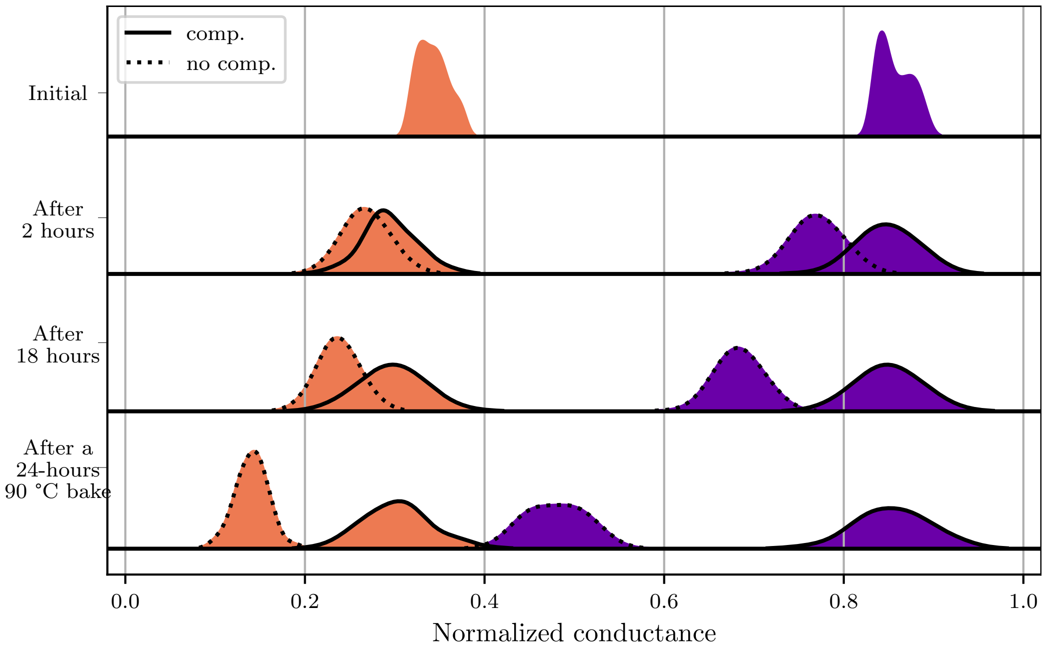


Fig. 5.17 Evolution over time of two batches of PCM devices, programmed to a target normalized conductance of 0.35 and 0.85. Both compensated and uncompensated cells are shown.

We will discuss first the results obtained with the Mk-I PCM chip, where the signals applied to the analog array are analog voltages applied as-is to the PCM cells, hence exciting the nonlinear behaviour observed in Section 5.4.2. Conventional reconstruction algorithms are not able to tolerate such a variability of the nominal conductance and a procedure to recover the original signals is proposed. Then we will present results built on top of the Mk-II platform, where the focus moves towards the reliability of well-known CS decoders to unpredictable changes (either because of an uncertain programming or for the drift over time) in the encoding stage weights.

5.5.1 Voltage-domain inputs

Let us start from the elementary encoding operation of a *linear* Compressed Sensing encoder repeated here for convenience:

$$y = A \cdot x, \tag{5.3}$$

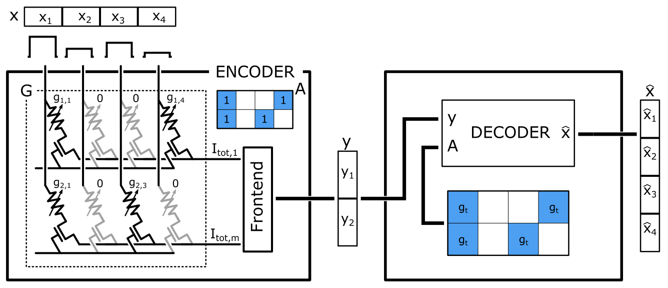


Fig. 5.18 PCM encoder followed by a one-shot decoder, using an estimate of the input-dependent PCM conductance.

with $A \in \mathbb{R}^{m \times n}$ the sensing matrix, $x \in \mathbb{R}^n$ the input and $y \in \mathbb{R}^m$ the measurements vector.

The most natural choice is to map ideal zeros to conductances in the RESET state, a highly resistive state with more than 2 orders of magnitude resistance as compared to common SET states, so that the individual current through those devices can reasonably be neglected, implementing an almost ideal zero.

The nonzero element, conversely, can be mapped to any SET state, though as shown previously in Fig. 5.8b, different states have wildly different conductance profiles versus the applied voltage, with the largest conductances being the most well-behaved, with a region in the lower half of the voltage domain being almost constant and a drop in the upper half. This effect, already discussed in Section 5.4.2, and associated to the saturation of the access transistors, can also be observed in terms of the differential conductance Fig. 5.8b to highlight how voltage variations in the saturated region are indistinguishable, being all mapped to the same current level.

If the sensing matrix A is implemented a conductance matrix G, the input x and measurements y can be thought of as voltages and currents, respectively. Furthermore, if the conductance matrix is built as a grid of PCM devices, we can immediately see how the application of different input voltages to the analog array changes the conductances of the array elements themselves. Eq. (5.3) then becomes:

$$y = A(x) \cdot x \,. \tag{5.4}$$

This is a critical issues, since at the decoder side we are assuming the nominal value for the sensing matrix and the actual changes depend on a quantity we still have to estimate. The effect of the uncertainty in the actual values of the sensing matrix weights, although deterministic in nature as it depends on a repeatable physical phenomenon, is still extremely detrimental to the reconstruction performance of known decoding algorithms, as clearly visible in Fig. 5.19a, where the reconstruction target of 50 dB is far from the actual performance obtained.

The iterative reconstruction technique has been tested with 1000 signal instances, sparse in the Discrete Cosine Transform basis, with $n = 256$, a sparsity level of 26 non-null coefficients, and a high-pass spectrum profile [33, Section 2.3]. The sensing matrices are binary, of size 128×256, with ideal zeroes, and ones being implemented by one of the conductance models in Fig. 5.8b. The minimization procedure being applied is the Orthogonal Matching Pursuit (OMP) [83].

Before the encoding step is performed, white gaussian noise is added to the signal to reach a predetermined value of Signal to Noise ratio (SNR), denoted as Intrinsic SNR (ISNR). Up to some denoising effects, which Compressed Sensing is known to possess [33], this defines a target for correct recovery of the original signal and makes the experimental setup more realistic. The reconstruction quality obtained at iteration p is measured by the Reconstruction Signal-to-Noise Ratio (RSNR), defined as:

$$\mathrm{RSNR} = \mathrm{SNR}(x, \hat{x}_{|p}) = 20 \log \frac{||x||_2}{||x - \hat{x}_{|p}||_2}$$

Looking at Fig. 5.19a, higher-valued conductances having less variation in relative terms lead to a better reconstruction quality, but still incur 20 dB of RSNR drop.

As we are used to do in Electronics, whenever some nonlinearity is the issue, we find ways to limit the signal excursions so that we locally have a more well-behaved response of the devices. Fig. 5.19b tries to analyze what happens when the dynamic range of the input signal is reduced, assuming (and it is not the only available choice, but a reasonable one) that the noise level is independent of the input signal. As the signal is reduced, the noise power stays constant to the point where the two are no more distinguishable in time-domain. Considering the input voltage dynamic ranges

(DR), we thus have:

$$\mathrm{ISNR}\,|_{\mathrm{DR}} = \mathrm{ISNR}\,|_{\mathrm{DR}_{\mathrm{ref}}} - 20\log\frac{\mathrm{DR}_{\mathrm{ref}}}{\mathrm{DR}}.$$

In general, being the effective ISNR our target reconstruction quality, maximizing it by making use of the largest possible dynamic range is desirable. However the higher the allowed voltage excursions, the higher the variation of the conductances, the lower the chances that the iterative technique will be able to recover the original signal. This trade-off suggests the presence of an optimal dynamic range, as indeed shown in the figure. The top two rows, corresponding to conductances of type #1 and #2, confirm the expectation that the dynamic range has to be maximized, while avoiding the region where the access transistors saturate, resulting in a low differential conductance. The results for the [0:1] dynamic range show indeed a long tail towards RSNR values lower than those of the [0:0.8] range, affecting around 25% of the tested signal instances. Conversely, for conductances of type #3, the ISNR value is reached only for the minimum dynamic range, [0:0.2]. Limiting the allowed dynamics to constrain the observed variability is indeed the typical design choice in this context, and under such a severe variation the proposed technique is not able to extend the range further than that.

From Fig. 5.19a, only the largest conductance state, used in the lower half of the voltage domain manages to achieve the reference performance level. At the maximum available dynamic range, performance does not increase and the answer lies in the behavior of the PCM cell in the right half of the domain. The presence of access transistors, discussed in Section 5.2, leading to the saturation of the I/V curves in Fig. 5.8a is the culprit. An alternative point of view is to consider that the differential conductance of the PCM device characteristic, i.e. the slope of the tangent lines to the curves in Fig. 5.8, which go to zero at high bias voltages, hence attenuating any information a signal may contain in that range.

Iterative decoding

The main idea for the proposed decoder starts from reanalyzing the issue at hand, i.e., the fact that the sensing matrix is modified by the input signal. Initially, the decoder has no information on the signal-dependent variability of the sensing matrix and has to assume some nominal value, e.g., that every matrix element has the nominal

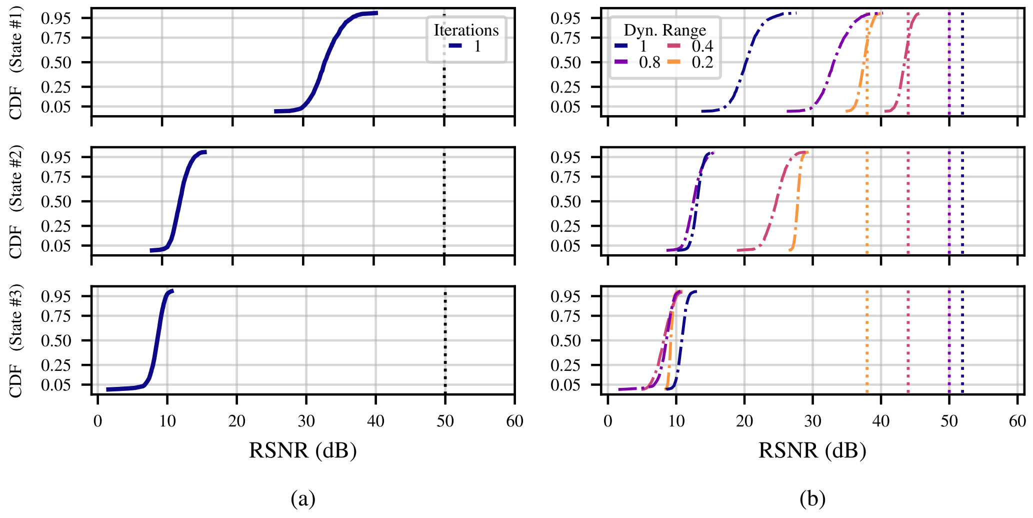


Fig. 5.19 (a) Empirical CDFs of RSNR values for the three conductance models, with input signals in the [0:0.8] dynamic range for a one-shot reconstruction. (b) Empirical CDFs of RSNR values for the three conductance models, one-shot reconstruction, at different dynamic ranges for the input signals. Dash-dotted vertical lines, both in (a) and (b), define the ISNR level, i.e. the target reconstruction quality.

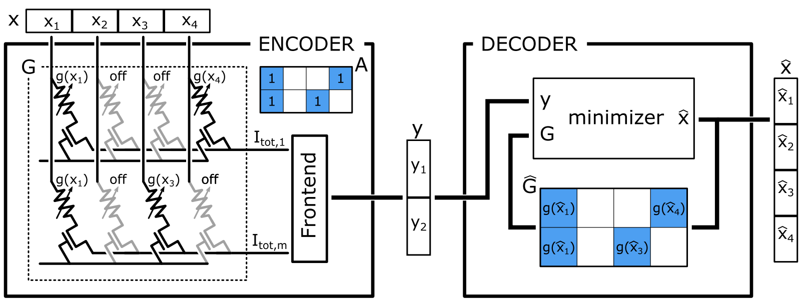


Fig. 5.20 Architecture of a PCM-based system for CS applications with the proposed decoder. The encoder implements the ideal, binary sensing matrix A by PCM conductances. The decoder reconstructs the original signal by iteratively computing the estimate of the real sensing matrix $\hat{G}$.

Iteration	Sensing Matrix	Estimate
0	$G(x)$	$\widehat{x_{\|1}}$
1	$G\left(\widehat{x_{\|1}}\right)$	$\widehat{x_{\|2}}$
2	$G\left(\widehat{x_{\|2}}\right)$	$\widehat{x_{\|3}}$
	$\dots$	
n	$G\left(\widehat{x_{\|n}}\right)$	$\widehat{x_{\|n+1}}$

Table 5.2 Sequence of iterative estimations $\widehat{x_{|n}}$ generated by the iterative decoding strategy in Fig. 5.20.

conductance, observed at a specific voltage. After an estimate of the input has been produced, and given a model of the voltage dependency of the PCM conductances (in our case, the same numerical model used within the encoder), the sensing matrix fed to the decoder can be re-evaluated so that each entry is aware of the signal estimate, and it is updated accordingly, as seen in Fig. 5.20, with the introduction of a feedback path from the reconstructed input $\hat{x}$, to the sensing matrix, so that $\hat{G} = G(\hat{x})$ is fed to a standard reconstruction algorithm. A second estimate of the original input is so obtained. Repeating this process multiple times, as shown in Table 5.2 one hopes to converge to the original signal. Empirical evidence shows that it is indeed true, in a significant number of cases.

The pseudocode of this iterative reconstruction is described in Algorithm 2.

Algorithm 2: Iterative CS decoder for signal dependent sensing matrices

```
Data: $A$, $y$, $g(x)$, $x_{\text{ref}}$
Result: $\hat{x}$
$\hat{G} \leftarrow g(x_{\text{ref}}) \cdot A$
repeat
    $\hat{x} \leftarrow \text{minimizer}(y, \hat{G} \cdot D)$
    $\hat{G}_{k,j} \leftarrow g(\hat{x}_j) A_{k,j} \quad \forall k = 1, \dots, m;\ j = 1, \dots, n$
    Compute convergence metric
until convergence or timeout
```

As the proposed decoding strategy requires repeated calls of an inner minimizer, which tries to solve an inherently complex, ill-posed mathematical problem, the use of the computationally light Orthogonal Matching Pursuit (OMP) minimizer is almost mandatory, if one only has access to limited computational resources.

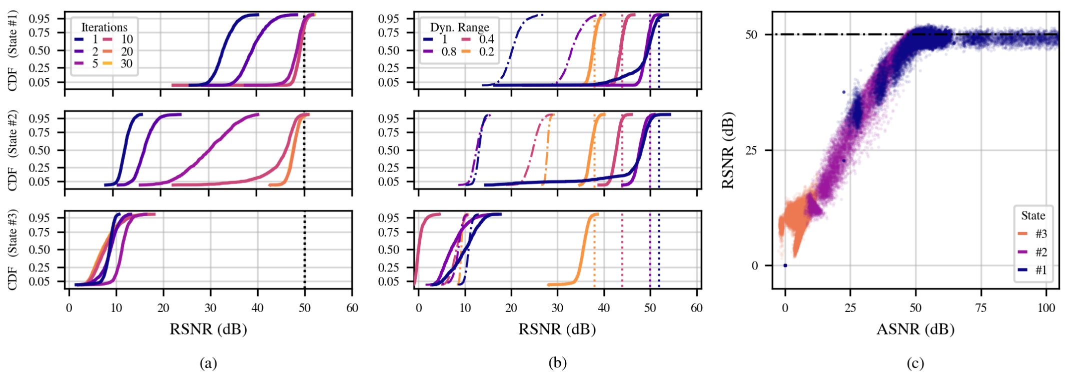


Fig. 5.21 (a) Empirical CDFs of RSNR values for the three conductance models, with input signals in the [0:0.8] dynamic range at a selected number of iterations for the decoding algorithm. (b) Empirical CDFs of RSNR values for the three conductance models, 30 iterations, at different dynamic ranges for the input signals. Dotted curves represent the results at the first iteration. Dash-dotted vertical lines, both in (a) and (b), define the ISNR level, i.e. the target reconstruction quality. (c) Mapping of RSNR values against ASNR, at a fixed dynamic range [0:0.8]. For each of the 1000 signal instances, 30 data points are represented, one for each iteration of the decoding procedure. The horizontal line represents the ISNR value. The oblique line is the plane bisector.

In Fig. 5.21a. Conductances of type #2, show the most interesting behavior, starting with a median RSNR of 12 dB, they rise to 30 dB in 5 iterations and 48 dB in 20 runs. The performance increase makes this programming state, whose performance at iteration 1 is insufficient for virtually all applications, worth considering, with the advantage of a reduction in energy consumption. Indeed, the expected power consumption (accounting for the probability distribution of the input signal samples) of the PCM array decreases by 28% if conductances of type #2 are chosen in place of type #1.

One could try to force the sensing matrix adaptation into the minimizer itself. We have not explored the issue enough to provide any guidelines, but preliminary observations show that a premature adaptation may prevent the iterative strategy from converging to the result, leaving it stuck in the wrong spot. A possibility could be to use the Generalized-OMP minimizer, which selects multiple vectors at once, thus adding some level of robustness against the overfitting of the sensing matrix to a single basis vector.

Up to this point, no convergence metric has been defined, and the proposed decoding technique has been characterized against a predefined number of iterations. Since we have observed how the obtainable reconstruction quality saturates in a number of iterations that depends heavily on the behavior of the conductance, being lower for less variable ones, the definition of an efficient halting criterion is highly desirable to minimize the number of runs.

A good candidate for a convergence metric is $\left\|\hat{x}_{|p} - \hat{x}_{|p-1}\right\|_2$, since this also implies convergence of the effective sensing matrices $\hat{G}_{|p}$. It is convenient to observe the SNR values computed from this difference, named Algorithmic SNR (ASNR), defined as

$$\mathrm{ASNR} = \mathrm{SNR}(\hat{x}_{|p} - \hat{x}_{|p-1}) = 20 \log \frac{\left\|\hat{x}_{|p}\right\|_2}{\left\|\hat{x}_{|p} - \hat{x}_{|p-1}\right\|_2},$$

and depicted in Fig. 5.21c against the corresponding values of RSNR, for all the instances used in Fig. 5.21a. Empirical evidence supports that RSNR and ASNR are strongly linked (at least until the ISNR level is reached) with the advantage that the latter does not need knowledge of the original signal x.

Observing that the clouds corresponding to different programming states reach different maximum levels of ASNR, defining convergence by setting a threshold on the observed ASNR is not efficient, since the threshold itself would have to be adapted to the conductance type. Moreover, it has been empirically observed that several signal instances are not able to achieve the threshold and run out of iterations nonetheless. A promising approach yet to be explored is to monitor the ASNR for the current signal instance until a local maximum is reached.

5.5.2 Time-domain inputs

The Mk-II PCM AIMC platform eliminates the need to address the conductance nonlinearity, as the applied inputs are now encoded with constant amplitude, width-modulated pulses. The remaining non-idealities to focus our attention on are the conductance programming spread and the drift of the programmed state over time.

This section continues in the use of binary sensing matrices, where the zeroes are considered ideal, as being implemented by cells in the RESET state. Conversely, ones are realized by cells in a partial SET state, in one of three representative levels selected to observe how different spread and drift device properties affect the quality of the reconstructed signals.

An elementary weight within the sensing matrix $G \in \mathbb{R}^{m\times n}$ is described by:

$$g_{ji} = g_T + \Delta g_p + \Delta g_d(\Delta t)\,, \tag{5.5}$$

with g_T the nominal target conductance, Δg_p the programming spread and $\Delta g_d(\Delta t)$ the conductance drift.

The Δg_p and $\Delta g_d(\Delta t)$ terms are modeled as described in Section 5.4.3. The former is a zero-mean gaussian error term whose variance is a function of the nominal value g_T. The latter is still a gaussian random variable, whose mean and variance are both function of g_T and of the *equivalent drift time*, i.e., the drift setup being considered.

A simulated performance evaluation has been run on 1000 signal instances, sparse in the Discrete Cosine Transform basis, with $n = 256$, $k = 26$ and a high-pass spectral profile, and 1000 binary sensing matrices, with 20% nonzero elements. A compression ratio equal to 2 has been selected, consequently, $m = 128$. The performance metric being considered is still the Reconstruction SNR (RSNR).

Each decoder requires knowledge of the sensing matrix being applied to the input at the encoder. However, in accordance with (5.5), the only information truly available is the target conductances g_T used to implement the nonzero elements of A. Here we consider to have reliable information also on the mean component of the drift $\mu_d(g,\Delta t)$, as a result of the application of the hardware compensation scheme presented in Section 5.2. This additional information is provided to the decoder to explore the performance obtained after some time from the original programming of the analog array. A more general characterization of the drift effect (i.e., without considering any compensation technique) would require the inclusion of additional information to be provided to the decoder, such as PCM devices technology, chip-to-chip variations and real-time working temperature, whose effects can be attenuated through the implementation of the compensation scheme. The hardware compensation is therefore expected to provide a more robust and extensive model of the drift phenomena.

Conversely, the reconstructor is unaware of the individual spread introduced by both the programming procedure and drift over time. As a consequence, each element in a given realization of the sensing matrix used at the decoder is identical, being either a true zero or a conductance g_T whose value has drifted over time by means of an estimation of $\mu_d(g_T, \Delta t)$.

All algorithms have been tested in a Python environment, using the `spgl1`[4], and `magni` [84] libraries, the latter including an implementation of the GAMP decoder. The GOMP algorithm has been constructed from an existing implementation of OMP[5]. The SPGL1 decoder requires a parameter `sigma` representing the expected measurement error. For each target conductances being tested, `sigma` has been computed by encoding a batch of inputs with the perturbed and ideal sensing matrices, computing the norm of the difference and then averaging across the batch. The GAMP decoder requires instead an estimate of the input channel properties, namely, the mean and variance of the nonzero components in the sparse representation of the inputs. Moreover an estimate of the variance of the measurement noise is needed for the output channel model.

To explore the tradeoffs involved in a PCM-based CS system, both in terms of performance and energy consumption, three normalized target conductances have been selected, 0.1, 0.4, and 0.7, to represent the nonzero values in the binary matrix. These value are highlighted in Fig. 5.16 as vertical dotted lines. At the lowest conductance, one observes the smallest programming spread, as well as small variability induced by the drift. However, in relative terms this represents the worst case scenario. Conversely, at the highest conductance, PCM devices show the best properties, in terms of relative spread, at the cost of an increased energy consumption, both in the programming phase and in the computation procedure.

Effects of the Programming Variability

Firstly, the reconstruction performance is evaluated considering the programmed PCM cells in t_0. Hence, the nonzero elements in G are implemented by

$$g_{ji} = g_T + \Delta g_p(g_T), \tag{5.6}$$

with the decoder being aware only of the nominal value g_T.

The behavior of the relative standard deviation of the programming spread immediately maps to the reconstruction performance observed in Fig. 5.22a for the

[4]https://spgl1.readthedocs.io/en/latest/
[5]https://github.com/davebiagioni/pyomp

three target conductances being tested. Increasing g_T decreases the spread in relative terms and leads to better performance, though with increasingly less benefits, as summarized in Fig. 5.22b.

From the standpoint of the reconstruction algorithms, GAMP is the most suitable decoder under any circumstance. Strikingly, the performance of GOMP, which is unaware of the properties of the input or output channels, is extremely close, with <2 dB degradation compared to GAMP, requiring however a much lower computational complexity and faster decoding times.

In relation to the encoding energy, Figure 5.22c shows that the additional energy cost incurred with larger conductances does not result in a proportional gain in performance, even when using the best performing decoder being tested, i.e., GAMP. This leads to a tradeoff between the desired average reconstruction accuracy, which is already > 30 dB for $g_T = 0.4$ and the energy expenditure at the encoder. Indeed, the energy consumption associated to a specific choice of target conductance is proportional to the whole current absorbed by the AIMC unit. Accordingly, considering the contribution of PCM cells only, the total current I_{TOT_j} required in the j-th MAC computation of can be described using the well-known Ohm's law:

$$I_{\mathrm{TOT}_j} = V_{\mathrm{REF}} \sum_{i=1}^{n} g_{ji} = V_{\mathrm{REF}} N_{\mathrm{ON}_j} g_T \quad (j = 1, \dots, m), \tag{5.7}$$

where g_{ji} has been assumed equal to g_T, according to (5.6), and N_{ON_j} is the number of cells in SET state of the j-th computation. Therefore, the total energy required to execute a single MAC operation, involved in the encoding phase of the CS algorithm, can be assumed proportional to the target conductance. Fig. 5.22c confirms that energy scales linearly with conductance, as the point clouds are equispaced along the energy axis for equidistant conductance values g_T. At the same time, the figure highlights the decreasing benefits in terms of reconstruction quality obtained by higher values of target conductance.

Effects of Drift

Let us now include the drift of the conductance for PCM devices when the hardware compensation scheme is employed. In this setup, the nonzero elements in G are

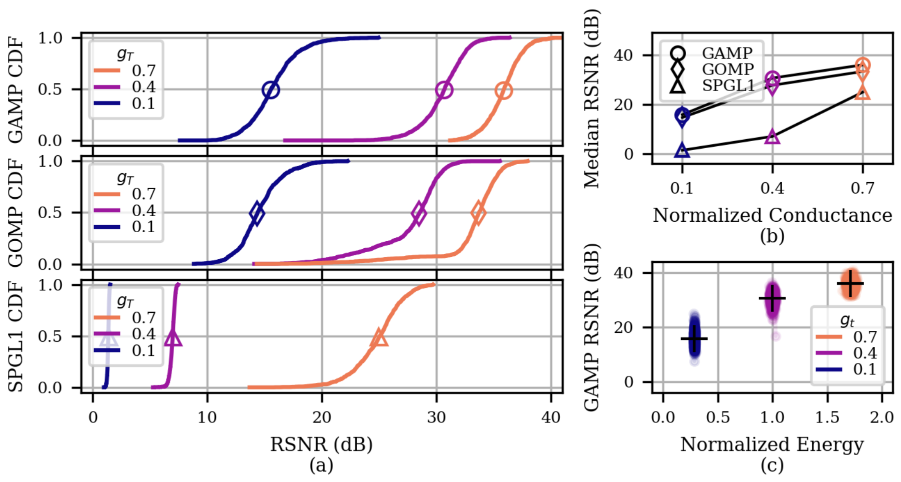


Fig. 5.22 (a) Performance of different reconstruction algorithms under the sensing matrix uncertainty introduced by different levels of target conductance. (b) Comparison of the median RSNR extracted from (a). (c) GAMP reconstruction accuracy versus encoding energy. The crosses highlight the mean energy and mean RSNR points in each cloud. The energy axis has been normalized with respect to the mean value of the $g_T = 0.4$ setup. According to (5.7), the normalized encoding energy is proportional to the total current employed in each MAC operation.

implemented by

$$g_{ji} = g_{p_{ji}} + \Delta g_d(g_{p_{ji}}, \Delta t)\,, \tag{5.8}$$

where $g_{p_{ji}} = g_T + \Delta g_p(g_T)$ represents the perturbed conductance after programming. The decoder knows g_T and an approximation of the mean component of Δg_d, i.e., its mean value $\mu_d(g_T, \Delta t)$, assumed to be well characterized and therefore predictable. Note that the drift at the decoder is evaluated at g_T instead of $g_{p_{ji}}$, leading to a residual mismatch in the representation of the sensing matrix due to the finite (but tunable) precision of the programming algorithm. In case uncompensated PCM devices were employed, whose conductance drop would be up to one order of magnitude larger, as shown in Fig. 5.16, the CS reconstruction performance would dramatically decrease, along with considerable modeling-related issues. In any case, the only significant effect being evaluated here is provided by the spread components of programming and drift.

The simulation results are observed in Fig. 5.23, where only the results for the GAMP and GOMP decoders are shown, being the most relevant. Notice how the curves associated to different drift setups have very limited degradation, notwith-

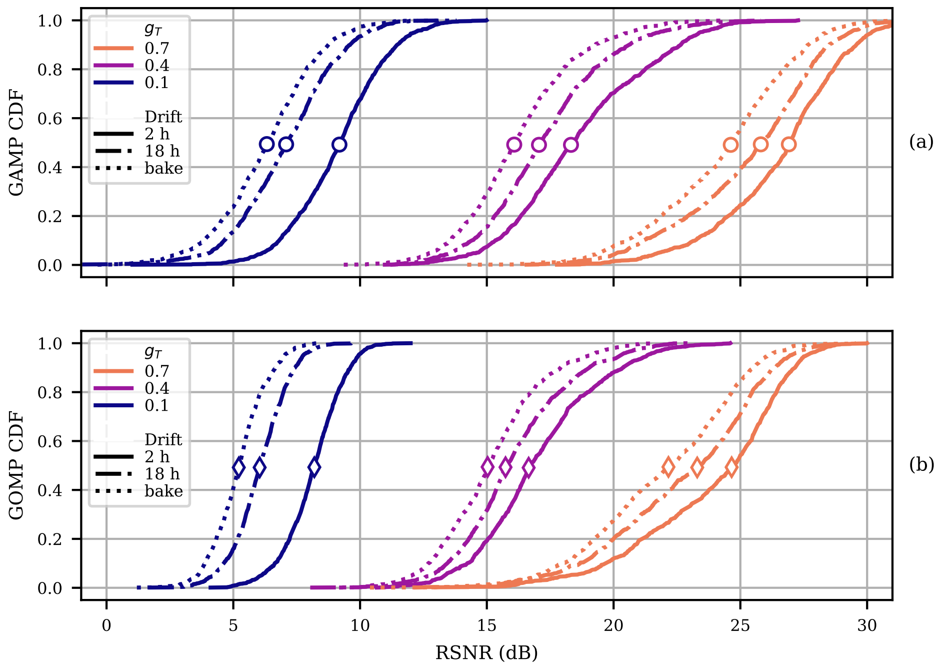


Fig. 5.23 Reconstruction performance for different drift setups: after 2 hours, after 18 hours and after a 24-hours bake at 90 °C. Results are shown for different target conductances, employing the hardware compensation scheme and the (a) GAMP and (b) GOMP decoders.

standing the extreme conditions being tested. Indeed, the loss in median performance observed with the GAMP decoder, from 2 hours to the end of the bake, is limited to 2.3 dB for both the $g_T = 0.7$ and $g_T = 0.4$ setups.

As a reference example, the application in [85], identifies a 21-dB and a 34-dB RSNR levels as reference thresholds for signal reconstruction quality. Considering the results obtained after the programming procedure of Fig. 5.22a, the lowest threshold is achieved by both the GAMP and GOMP decoders with $g_T = 0.4$ or 0.7, whereas the highest threshold is only reached by GAMP when using $g_T = 0.7$. Including then the drift effects, as depicted in Fig. 5.23, the GAMP algorithm is able to preserve a 21-dB RSNR only for $g_T = 0.7$.

5.6 PCM-based Neural Network

The past few years have seen revolutionary improvements in the efficacy of Deep Neural Networks (DNNs) in solving real world tasks. At the time of writing, Chat-GPT [6] has just been released. Its rapid adoption, with 1M+ users in less than a week is a testament to its ability at generating relevant text content. At the opposite side of the spectrum, the TinyML movement [7] is aggregating academic and industry professionals to develop techniques and frameworks for low power implementations of machine learning tools to be used with limited computational and energy resources. As NNs are spreading towards the edges of the data acquisition chains, efficient implementations are sought after. In this context, analog arrays represent a valid candidate both in terms of speed, with their $\mathscr{O}(1)$ computational complexity, and energy efficiency.

In this Section, we will analyze the results obtained when either the PCM models from the Mk-I and Mk-II platforms are employed as the elements implementing the matrix-vector multiplications at the core of the neural layers. We will first observe how the backpropagation procedure is able to select the network weights even with the nonlinearity of the PCM elements, to solve both the MNIST and Fashion-MNIST classification tasks and a custom-defined regression task. Moving on, to the PCM models from the Mk-II chip, we analyze the effects of programming variability and conductance drift on the accuracy obtained on the CIFAR-10 classification task using the Lenet-5 and VGG-8 DNNs.

5.6.1 Voltage-domain inputs

In a traditional dense layer, the core operation for the j-th neuron is $h_j = f(b_j + \sum_i w_{j,i} x_i)$, with inputs x_i, weights $w_{j,i}$, bias terms b_j, and nonlinear activation $f(\cdot)$. Aiming towards a circuital implementation where inputs are voltages, and they are weighted by conductances programmed in different states, the expression becomes $h_j = f(b_j + \sum_i I(x_i, w_{j,i}))$, where we neglect any additional term introduced by electrical noise, programming noise or even quantization of the inputs or the outputs.

[6]https://openai.com/blog/chatgpt
[7]https://www.tinyml.org/

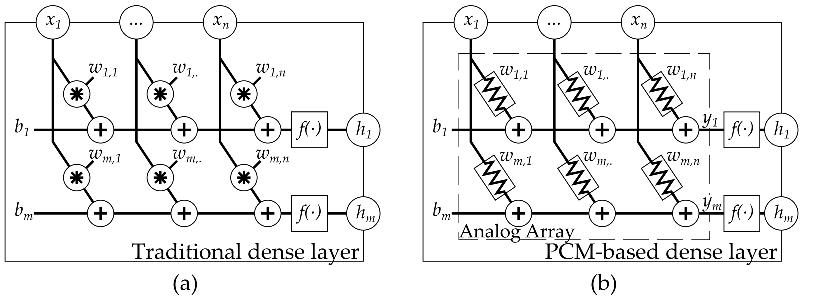


Fig. 5.24 (a) Traditional dense and (b) PCM-based dense neural layer structures. The analog array is highlighted in the latter, where we assume that the summation nodes are held at a constant voltage for it to function properly.

Training a layer defined as such does not pose any difficulty, provided that $I(x_i, w_{j,i})$ is differentiable with respect to the weights [79]. Standard software frameworks are able to automatically differentiate even complex expressions if described in terms of their library of operators [86]. To the best of our knowledge, no spline primitives are available as elementary operators, hence the need to have an alternative description of the synapses, expressed in terms of available operators. Such a description is in our case of polynomial type. Potentially, a more physically-informed model could be used as well, though as our measurement data includes significant effects from the access devices surrounding the PCM cells, we have preferred to have a unique model that could describe the behaviour of the entire circuital block over the full voltage domain.

An additional advantage is that, at training time, the network could be trained on a model of the neuron which is computationally simpler, though potentially less accurate, enabling a faster exploration of different network topologies.

Two case studies will be analyzed in the following: a classification task performed on the Fashion-MNIST dataset [87], and a regression problem, in which the network has to estimate a parameter describing the spectral content of randomly generated signal instances.

Results

In the following we will show numerical results on the training of neural networks in which one layer is PCM-based. In all setups, the performance of a neural network employing only conventional dense layers and having the same structure, is used

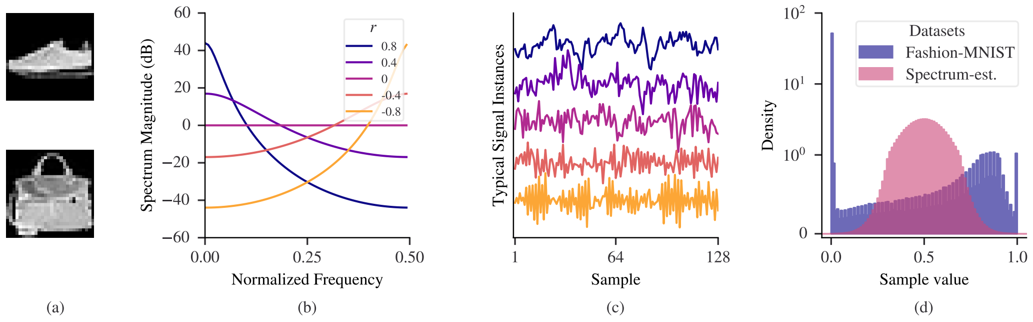


Fig. 5.25 (a) Examples of Fashion-MNIST instances, employed in Section 5.6.1 (b) Spectrums for a selection of r with corresponding representative instances of length 128 in (c). This data is used with the network in Section 5.6.1. (d) Normalized histograms of input values for the two datasets being employed. A linear scale is used in the range $[0,1]$ of the vertical axis.

as a reference. To train the PCM-based network, the PCM synapses are always described by their polynomial model, with an arbitrarily selected degree and by identifying $L = 10$ different reference currents. An initial performance metric is thus obtained, related exclusively to the use of the polynomial. The final evaluation is then performed on the same network, preserving the trained weights, but replacing in the PCM-based layer the polynomial model with the spline one, representing our reference model for nominal PCM devices.

Since in a physical implementation the state of a PCM cell cannot be programmed to arbitrary accuracy, we also test the robustness of the network towards this kind of perturbation. We model the variation of the PCM state with a white gaussian noise added to the nominal values of the weights (i.e., those suggested by training) during the final evaluation. The variance of the weight noise is normalized to the nominal value, so that their ratio is fixed. Clipping is then applied to ensure that the noisy weights are still within the validity range of the numerical models.

Two different applications are shown, trying to highlight the different features of the setups presented in this work. The optimizer in all setups is Adam [88], with parameters: learning rate equal to 10^{-4} and the exponential decay rate for the 1st and 2nd moments equal to 0.9 and 0.999. Results are condensed in Fig. 5.26.

Fashion-MNIST Classification

The dataset is made of grayscale images of clothing articles, in a 28×28 pixel format. Two examples are shown in Fig. 5.25a. The neural network topology being considered has an input-flattening layer followed by a single dense layer with 10 output nodes and sigmoid activation functions. The loss function is the sparse categorical cross-entropy. While the conventional reference network has no constraints on the weights, in the PCM-based one we have introduced a “bathtub” regularization to force them within the [0, 1] range. This implies a physical realization requiring only positively-contributing PCM synapses on each layer output.

To asses the performance we use here the accuracy defined as the correct classification rate. Analyzing the results shown in Fig. 5.26a, a monotonic trend is clear, up to order 27, with networks trained on a high-order polynomial model almost matching the performance of the reference network.

The fact that the weights obtained by training a low-order polynomial, as that depicted in Fig. 5.12c is already sufficient to solve the classification task with ~ 0.78 accuracy has been associated to the statistical distribution of pixel intensities. Being their density concentrated around the extremes of the available range, as shown in Fig. 5.25d, the inherent nonlinearity of the models is not significantly excited. The model feature that matters is that their output is different for low and high input values. Both the spline and polynomials being employed possess such a feature, resulting in a limited performance drop with respect to the reference case.

The application of noise on the trained weights only becomes significant around 10% relative standard deviation. With respect to the noiseless setup, the performance loss is still within 3.5%. State-of-the-art iterative programming techniques of the physical devices may indeed be able to achieve such a level of programming accuracy [89, 90], thus incurring a limited performance penalty on the network task.

Spectral Estimation Regression

The second task being evaluated is a regression problem artificially constructed so that the nonlinearity of the PCM *I–V* characteristic can be excited even more.

The problem is that of estimating the properties of the Fourier-spectrum of random signal instances. Given a value $-1 < r < 1$, let us define an autocorrelation

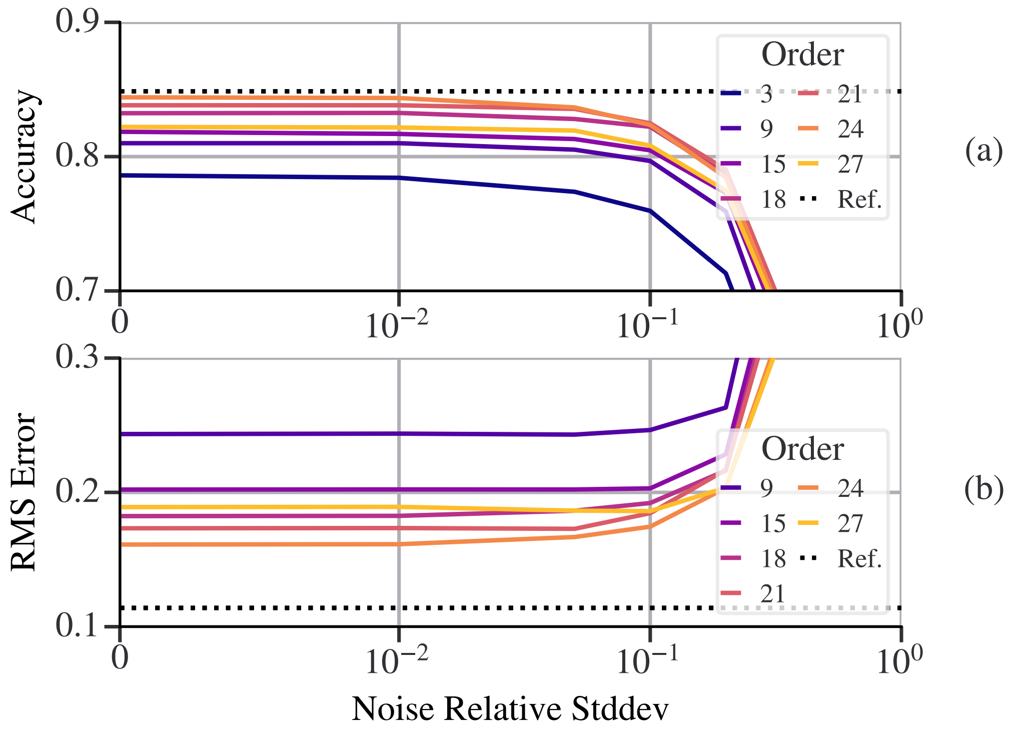


Fig. 5.26 Results for (a) Fashion-MNIST classification, and (b) spectrum estimation regression. The black, dotted line represents the performance obtained by a neural network employing standard dense layers, without noise. Solid lines refer to the performance of networks using the spline PCM model, with the weights trained on the polynomial description of the device, and additional noise included during the evaluation phase.

matrix K such that $K_{j,k} = r^{|j-k|}$, $1 \leq j,k \leq n$, with n the length of the signal instances being observed. The power spectrum of the stationary stochastic process thus defined is [33]:

$$\Psi(f) = \frac{1-r^2}{1+r^2-2r\cos(2\pi f)}\,.$$

Its profile is high-pass for $-1 < r < 0$, flat/white for $r = 0$ and low-pass for $0 < r < 1$. Examples of spectra for different values of r are shown in Fig. 5.25b, with corresponding representative signal instances depicted in Fig. 5.25c.

Given a value for r, signals can be generated by computing instances of a multivariate gaussian distribution $\mathcal{N}(0,K)$. Inverting the relationship between the power spectrum and r is not possible, and the neural network has to estimate it by looking at each signal instance and providing an answer in the $[-1,1]$ range.

The network structure being tested operates on signal instances of 32 samples and it has three dense layers of size 256, 256 and 1. The first two layers have relu activation functions, while the output layer has none. The loss function is the mean squared error, while the performance metric being observed is the root mean squared (RMS) error. A conventional network with such a structure achieves a 0.114 RMS estimation error.

The weighting coefficients of the PCM-based layer in this case have been constrained in the range [-1, 1] by a "bathtub" regularization. From an implementation point of view, this requires a way for a PCM cell, to have a negative contribution on the sum of synapses currents. The practical approach could involve, for example, having two arrays of conductances, with opposite contributions on the output [91],[92].

Results in Fig. 5.26b highlight a monotonic trend up to order 24, with a sudden worsening of performance observed at 27.

The detrimental effect of the additive noise on the weights is still under control for 10% relative standard deviation, with variations on the order of 0.014 RMS error as compared to the same setup, with noiseless weights.

5.6.2 Time-domain inputs

To evaluate the performance of the proposed variability mitigation strategies on an actual application, a classification task on the well know CIFAR-10 dataset has been selected as a testbench [93]. Two popular neural networks have been used, the Lenet-5 [94] and the VGG-8 [95], having significantly different complexities, with $\sim 8 \times 10^5$ and $\sim 4 \times 10^7$ trainable parameters, respectively. Their implementation has been suitably modified so that each synapse would emulate a PCM device, with the possibility of enabling conductance programming variability and drift at will.

With reference to a typical dense layer, the description of the j-th neuron output is $h_j = f(b_j + \sum_i w_{j,i} x_i)$, with inputs x_i, weights $w_{j,i}$, bias terms b_j and nonlinear activation $f(\cdot)$. A PCM-based layer driven by time-encoded inputs would instead be represented by [96]:

$$h_j = f\left(b_j + \sum_i k \frac{g_{j,i}}{g_{\mathrm{REF}}} V_i\right). \tag{5.9}$$

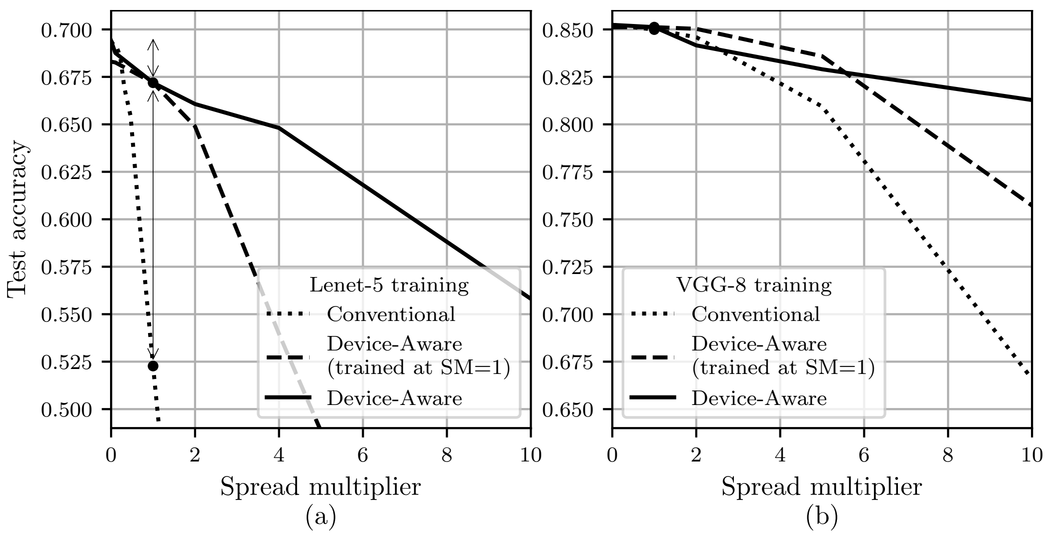


Fig. 5.27 Accuracy of the trained networks versus the programming spread scaling coefficient, both for the conventional (dotted line) and device-aware (DA) trainings: continuous line refers to results when inference is done with the same spread multiplier (SM) used for the training; in the dashed one the training is performed with SM = 1 while inference is implemented considering different values of SM. (a) Lenet-5 and (b) VGG-8 DNNs.

This same reasoning can be trivially extended to convolutional layers and allows the definition of a fully PCM-based DNN.

If programming noise and drift are being introduced, the elementary synapse conductance becomes

$$g_{j,i} = g = g_0 + \Delta g_p(g_0) + \Delta g_d(g_0, \Delta t)\,, \tag{5.10}$$

where $\Delta g_p(g_0)$ is the programming-induced variability, having a normal distribution $\mathcal{N}\left(0, \sigma_p(g_0)\right)$ and $\Delta g_d(g_0, \Delta t)$ models the drift by drawing from a

$$\mathcal{N}\left(\mu_d(g_0, \Delta t), \sigma_d(g_0, \Delta t)\right)$$

distribution, using the models depicted in Fig. 5.16.

Both neural networks have been trained with the Adam optimizer [97], using the following parameters: exponential decay rate for the 1st and 2nd moments equal to 0.9 and 0.99, and learning rate equal to 10^{-2} for the Lenet-5 network and 10^{-3} for the VGG-8 one. Whilst training, the learning rates have been halved whenever the process would reach a plateau for a predefined amount of epochs.

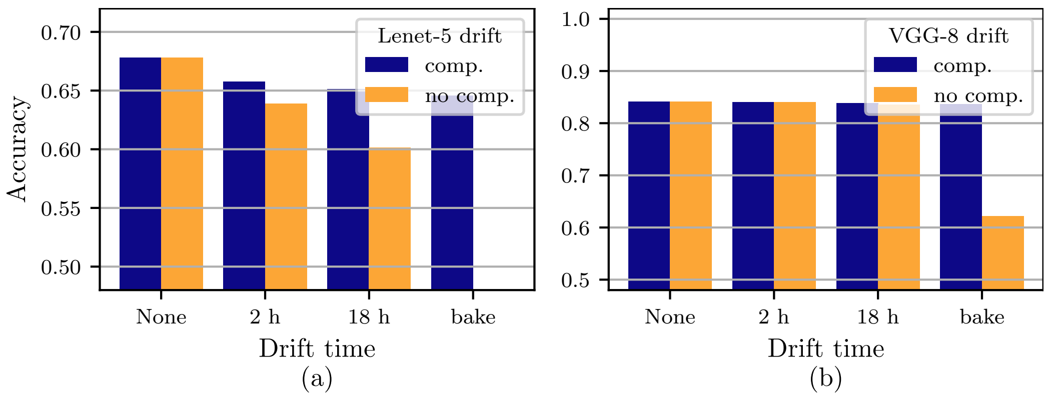


Fig. 5.28 Accuracy achieved when drift is applied to the DNN weights at inference time, both with and without compensation. (a) Lenet-5 and (b) VGG-8 DNNs.

Let us first observe how the two DNNs, trained without any weight variability, perform when the Δg_p term is introduced only at inference time. To widen the scope of the analysis, the injected perturbation is scaled by a multiplying factor. One reason to do it could be to relax the tolerance $\delta_g = 0.025g^{\mathrm{MAX}}$ of the programming algorithm described in Section 5.2.1, allowing it to converge in a lower number of iterations, speeding up the initial setup of the memory or a possible refresh of its values. The dotted curves in Fig. 5.27 highlight the subitaneous loss of performance as soon as noise is injected in the Lenet-5 DNN. The larger VGG-8 network, other than having a higher accuracy, is also more resilient towards the injected perturbation. This is thought to be the effect of the additional redundancy introduced by the larger number of weights, as in [98]. The datapoint corresponding to a spread multiplier (SM) of 1 has been highlighted, as it corresponds to the performance observed under the current programming parameters.

To make the network aware of the programming spread affecting its weights, a training methodology inspired by the *fake quantization* procedure [99] has been employed. This is a known methodology for the construction of NNs robust against synapse variability, and has been used extensively in the Literature [100, 101]. It requires, at train time, the addition of a perturbation before the weights are actually applied to the inputs. This obviously affects the network result, hence the starting point of the backpropagation algorithm [79]. The weight-update process then computes the derivative with respect to the original, nominal weights. Empirical evidence shows that this makes the network more resilient to weight variations. The original technique was devised for the purpose of making the network robust towards weight quantization. In that case, the properties of the injected variability would have been dependent on the number of allowed levels. For the PCM-based layers, instead, the injected perturbation models the programming-induced variability, i.e., the Δg_p term in (5.10). Results in Fig. 5.27 plotted as solid lines refer to DNNs trained and evaluated with an identical spread multiplier. The performance gain is much more pronounced for the smaller Lenet-5 than the larger VGG-8, so much so that the former becomes implementable also on the currently available technology. At a multiplier of 1, the Lenet-5 shows a 2.2% drop (69.4% down to 67.2%) in accuracy compared to the ideal, unperturbed, setup and a 15% increase (52.2% to 67.2%) with respect to the conventionally-trained DNN with weight perturbation injected at evaluation-time. This result, in conjuction with recent observations on the issues with the IR drop in large PCM arrays [102], highlights the value of the device-aware training technique to construct small and robust DNNs. Dashed plots of Fig. 5.27 represent the DNNs accuracy as a function of the increasing spread multiplier applied in the inference phase, while the device-aware training is performed with a constant Δg_p (i.e., with a spread multiplier being kept equal to 1 in the training phase). Results show an accuracy decrease with respect to the previous case, but still with a performance gain against the employment of a conventional training, allowing thus to tolerate a higher spread on the networks coefficients. From a power consumption point of view, the device-aware technique permits then to take advantage of less sophisticated programming procedures, with a consequent gain in speed and energy efficiency of the programming phase.

Additional nonidealities, e.g., quantization of pre- and post-activation signals, the presence of parasitic elements along the conductive paths, to mention a few, have not been included in the analysis, which focuses only on the properties of the PCM devices. However, it has been observed how device-aware training techniques do not need to accurately describe the variability of interest, because of an inherent ability of the training to lead to networks robust against effect different from the perturbations used in training [100, 101]. In any case, the same injection-perturbation principle used in this work could be used to address signal quantization and the presence of parasitic elements [101]. Concerning the former, 6 bits have proven to be sufficient to limit the performance degradation below 1% point, while 5 bits introduce, for a network trained only to address conductance variability, a loss around 5% points.

Having a network that can tolerate programming variability, the final step is to observe its robustness against weight drift. Both networks, trained with a spread

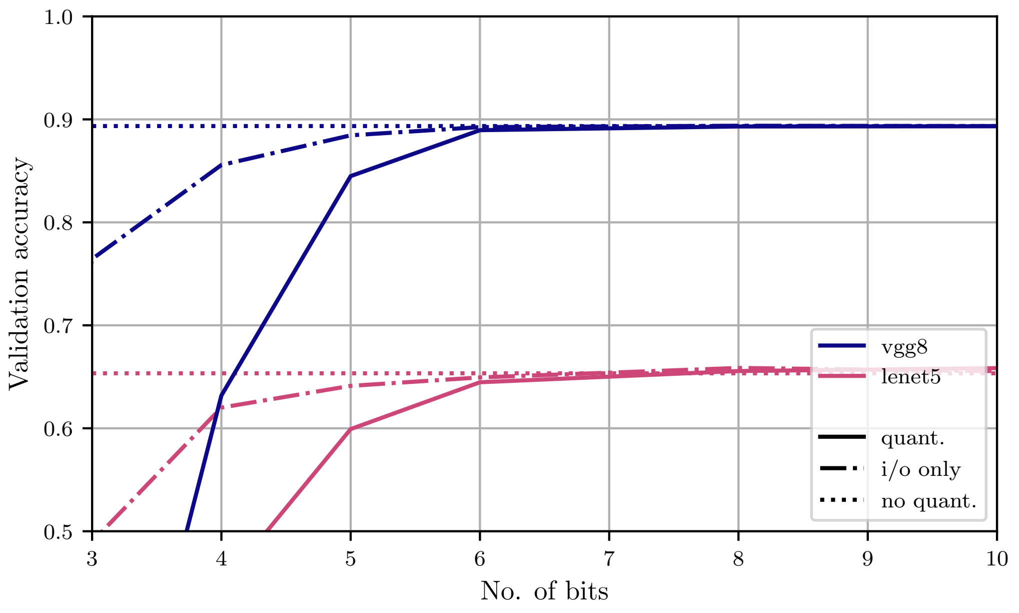


Fig. 5.29 Classification accuracy when quantizing only the global network I/O vs the I/O at every layer.

multiplier of 1 (i.e., with programming tolerace $\delta g = 0.025 g^{\mathrm{MAX}}$), have been re-evaluated by introducing the drift component of the conductance Δg_d at inference time. From Fig. 5.28 it is clear how the presence of the hardware compensation allows the accuracy to be retained over time. The accuracy gain after the 24-hours 90 °C bake is 36% for the Lenet-5 (even though the corresponding point for the uncompensated evaluation falls outside the range of the plot) and 22% for the VGG-8 DNN. While the drop with respect to the no-drift condition is 3% and 0.2%. Still, the benefit is larger for the smaller network. However, even the VGG-8 one, which would lose significant accuracy after the 24-hours bake, would be able to preserve its original performance with the introduction of the compensation technique.

Chapter 6

Conclusion

This work has focused on the analysis of low-energy data processing architectures. Two alternative directions have been undertaken. A CMOS charge-redistribution SAR converter has been modified to enable Compressed Sensing functionalities. Compared to existing solutions, it avoids additional active circuits. Details of the design, as well as tangential analysis developing analytical models of specific issues have been presented. The extra results are the development of an approximate model of the integral and differential nonlinearities in C-2C D/A topologies, when parasitic capacitance loads the inner isolated nodes. The effect of interdigitated geometries on the matching properties of nominally identical elements is evaluated and models are presented to compute the expected mismatch. The search for robust leakage compensation architectures has lead us to the extensive analysis of stability and mismatch robustness of a known circuit topology, highlighting its practical unfeasibility because of its extreme sensitivity to the slightest of nonidealities.

A different approach has been undertaken in the second half of the work, which is based on Phase-Change Memories (PCM). The DC response of PCM devices has been evaluated, as well as programming and drift variability, for two physical chips using different driving methodologies, namely amplitude-modulated voltage pulses and constant-voltage width-modulated pulses. The performance of the technology have been tested in two different applications, compressed sensing encoding, and neural network classification and regression tasks. Measurements-based numerical models have been constructed, though not physically based, they are built to accurately map the observed behavior. This new promising technology is still far from unleashing its true potential, but the work shows that ad-hoc robust architectural choices, as seen for the hardware drift compensator, can greatly enhance their practicality. Surely the following years will see tremendous progress in the device composition, geometry, driving and readout circuit, as well as nonideality mitigation strategies. The combined optimizations at several abstraction levels will definitely pave the way for a widespread application of PCM technology in computational nodes.

References

[1] N. Lu, "Modeling of Distance-Dependent Mismatch and Across-Chip Variations in Semiconductor Devices," *IEEE Transactions on Electron Devices*, vol. 61, no. 2, pp. 342–350, Feb. 2014. doi: 10.1109/TED.2013.2283076

[2] C. Paolino *et al.*, "A Practical Architecture for SAR-based ADCs with Embedded Compressed Sensing Capabilities," in *2019 15th Conference on Ph.D Research in Microelectronics and Electronics (PRIME)*, 2019, pp. 133–136. doi: 10.1109/PRIME.2019.8787816

[3] C. Paolino *et al.*, "A passive and low-complexity Compressed Sensing architecture based on a charge-redistribution SAR ADC," vol. 75, pp. 40–51, 2020. doi: 10.1016/j.vlsi.2020.05.007

[4] C. Paolino *et al.*, "An architecture for ultra-low-voltage ultra-low-power compressed sensing-based acquisition systems," in *2021 IEEE Nordic Circuits and Systems Conference (NorCAS)*, 2021, pp. 1–7. doi: 10.1109/NorCAS53631.2021.9599652

[5] C. Paolino *et al.*, "Asymptotic Expressions of Mismatch Variance in Interdigitated Geometries," in *2020 IEEE International Symposium on Circuits and Systems (ISCAS)*, 2020, pp. 1–5. doi: 10.1109/ISCAS45731.2020.9181159

[6] C. Paolino *et al.*, "Stability and Mismatch Robustness of a Leakage Current Cancellation Technique," in *2021 IEEE International Symposium on Circuits and Systems (ISCAS)*, 2021, pp. 1–5. doi: 10.1109/ISCAS51556.2021.9401430

[7] C. Paolino *et al.*, "Compressed Sensing by Phase Change Memories: Coping with Encoder non-Linearities," in *2021 IEEE International Symposium on Circuits and Systems (ISCAS)*, 2021, pp. 1–5. doi: 10.1109/ISCAS51556.2021.9401176

[8] C. Paolino *et al.*, "Decoding Algorithms and HW Strategies to Mitigate Uncertainties in a PCM-Based Analog Encoder for Compressed Sensing," vol. 13, no. 1, p. 17, 2023. doi: 10.3390/jlpea13010017

[9] C. Paolino *et al.*, "Phase-Change Memory in Neural Network Layers with Measurements-based Device Models," in *2022 IEEE International Symposium*

on Circuits and Systems (ISCAS), 2022, pp. 1536–1540. doi: 10.1109/ISCAS48785.2022.9937856

[10] A. Antolini *et al.*, “Combined HW/SW Drift and Variability Mitigation for PCM-based Analog In-memory Computing for Neural Network Applications,” pp. 1–1, 2023. doi: 10.1109/JETCAS.2023.3241750

[11] M. Unser, “Sampling-50 years after Shannon,” vol. 88, no. 4, pp. 569–587. doi: 10.1109/5.843002

[12] S. Greco and P. Valabrega, *Algebra lineare*. Libreria editrice universitaria Levrotto & Bella, 2009.

[13] S. Theodoridis, *Machine Learning*. Elsevier, 2015.

[14] Y. C. Eldar and G. Kutyniok, *Compressed Sensing: Theory and Applications*. Cambridge University Press, 2012.

[15] M. Mangia *et al.*, *Adapted Compressed Sensing for Effective Hardware Implementations*. Springer International Publishing, 2018.

[16] E. van den Berg and M. P. Friedlander, “Probing the pareto frontier for basis pursuit solutions,” *SIAM Journal on Scientific Computing*, vol. 31, no. 2, pp. 890–912, Jan. 2009. doi: 10.1137/080714488

[17] J. Wang *et al.*, “Recovery of Sparse Signals via Generalized Orthogonal Matching Pursuit: A New Analysis,” *IEEE Transactions on Signal Processing*, vol. 64, no. 4, pp. 1076–1089, Feb. 2016. doi: 10.1109/TSP.2015.2498132

[18] J. T. Parker *et al.*, “Compressive sensing under matrix uncertainties: An Approximate Message Passing approach,” in *2011 Conference Record of the Forty Fifth Asilomar Conference on Signals, Systems and Computers (ASILOMAR)*, Nov. 2011, pp. 804–808. doi: 10.1109/ACSSC.2011.6190118

[19] M. Shoaran *et al.*, “Compact low-power cortical recording architecture for compressive multichannel data acquisition,” *IEEE Trans. Biomed. Circuits Syst.*, vol. 8, no. 6, pp. 857–870, Dec. 2014. doi: 10.1109/TBCAS.2014.2304582

[20] J. Zhang *et al.*, “An efficient and compact compressed sensing microsystem for implantable neural recordings,” *IEEE Trans. Biomed. Circuits Syst.*, vol. 8, no. 4, pp. 485–496, Aug. 2014. doi: 10.1109/TBCAS.2013.2284254

[21] F. Pareschi *et al.*, “Hardware-Algorithms Co-Design and Implementation of an Analog-to-Information Converter for Biosignals Based on Compressed Sensing,” *IEEE Trans. Biomed. Circuits Syst.*, vol. 10, no. 1, pp. 149–162, Feb. 2016. doi: 10.1109/TBCAS.2015.2444276

[22] X. Chen *et al.*, “A sub-nyquist rate compressive sensing data acquisition front-end,” *IEEE J. Emerg. Sel. Topics Circuits Syst.*, vol. 2, no. 3, pp. 542–551, Sep. 2012. doi: 10.1109/JETCAS.2012.2221531

[23] J. Yoo *et al.*, “A 100MHz-2GHz 12.5x sub-Nyquist rate receiver in 90nm CMOS,” in *2012 IEEE Radio Frequency Integrated Circuits Symposium*, Jun. 2012, pp. 31–34. doi: 10.1109/RFIC.2012.6242225

[24] J. Haboba *et al.*, “A Pragmatic Look at Some Compressive Sensing Architectures With Saturation and Quantization,” *IEEE J. Emerg. Sel. Topics Circuits Syst.*, vol. 2, no. 3, pp. 443–459, Sep. 2012. doi: 10.1109/JETCAS.2012.2220392

[25] J. Romberg, “Compressive sensing by random convolution,” *SIAM J. Imaging Sci.*, vol. 2, no. 4, pp. 1098–1128, 2009. doi: 10.1137/08072975X

[26] A. Harms *et al.*, “A constrained random demodulator for sub-nyquist sampling,” *IEEE Trans. Signal Process.*, vol. 61, no. 3, pp. 707–723, 2013. doi: 10.1109/TSP.2012.2231077

[27] A. Ravelomanantsoa *et al.*, “Simple and efficient compressed sensing encoder for wireless body area network,” *IEEE Trans. Instrum. Meas.*, vol. 63, no. 12, pp. 2973–2982, Dec. 2014. doi: 10.1109/TIM.2014.2320393

[28] M. Mangia *et al.*, “Zeroing for HW-efficient compressed sensing architectures targeting data compression in wireless sensor networks,” *Microprocessors and Microsystems*, vol. 48, pp. 69–79, Feb. 2017. doi: 10.1016/j.micpro.2016.09.007

[29] M. Mangia *et al.*, “Rakeness-Based Design of Low-Complexity Compressed Sensing,” *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 64, no. 5, May 2017. doi: 10.1109/TCSI.2017.2649572

[30] M. Mangia *et al.*, “Deep Neural Oracles for Short-window Optimized Compressed Sensing of Biosignals,” *IEEE Trans. Biomed. Circuits Syst.*, vol. 14, no. 3, pp. 545–557, Jun. 2020. doi: 10.1109/TBCAS.2020.2982824

[31] F. Chen *et al.*, “Design and Analysis of a Hardware-Efficient Compressed Sensing Architecture for Data Compression in Wireless Sensors,” *IEEE J. Solid-State Circuits*, vol. 47, no. 3, pp. 744–756, Mar. 2012. doi: 10.1109/JSSC.2011.2179451

[32] D. Bellasi *et al.*, “A Broadband Multi-Mode Compressive Sensing Current Sensor SoC in 0.16 um CMOS,” *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 66, no. 1, pp. 105–118, Jan. 2019. doi: 10.1109/TCSI.2018.2846573

[33] M. Mangia *et al.*, *Adapted compressed sensing for effective hardware implementations: A design flow for signal-level optimization of compressed sensing stages*. Springer International Publishing, 2018.

[34] N. Cleju, “Optimized projections for compressed sensing via rank-constrained nearest correlation matrix,” *Applied and Computational Harmonic Analysis*, vol. 36, no. 3, pp. 495–507, May 2014. doi: 10.1016/j.acha.2013.08.005

[35] J. Xu *et al.*, "Optimized projection matrix for compressive sensing," *EURASIP Journal on Advances in Signal Processing*, p. 560349, 2010. doi: 10.1155/2010/560349

[36] M. Mangia *et al.*, "Rakeness in the design of analog-to-information conversion of sparse and localized signals," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 59, no. 5, pp. 1001–1014, May 2012. doi: 10.1109/TCSI.2012.2191312

[37] A. Marchioni *et al.*, "Resource redistribution in internet of things applications by compressed sensing: A survey," in *IEEE international symposium on circuits and systems (ISCAS)*, May 2018, pp. 1–5. doi: 10.1109/ISCAS.2018.8351891

[38] A. Mirrashid and A. A. Beheshti, "Compressed remote sensing by using deep learning," in *9th international symposium on telecommunications (IST)*, Dec. 2018, pp. 549–552. doi: 10.1109/ISTEL.2018.8661112

[39] K. Kulkarni *et al.*, "ReconNet: non-iterative reconstruction of images from compressively sensed measurements," in *IEEE conference on computer vision and pattern recognition (CVPR)*, Jun. 2016, pp. 449–458. doi: 10.1109/CVPR.2016.55

[40] A. Mousavi and R. G. Baraniuk, "Learning to invert: signal recovery via deep convolutional networks," in *IEEE international conference on acoustics, speech and signal processing (ICASSP)*, Mar. 2017, pp. 2272–2276. doi: 10.1109/ICASSP.2017.7952561

[41] W. Shi *et al.*, "Deep networks for compressed image sensing," in *IEEE international conference on multimedia and expo (ICME)*, Jul. 2017, pp. 877–882. doi: 10.1109/ICME.2017.8019428

[42] A. Mousavi *et al.*, "A deep learning approach to structured signal recovery," in *3rd annual allerton conference on communication, control, and computing (allerton)*, Sep. 2015, pp. 1336–1343. doi: 10.1109/ALLERTON.2015.7447163

[43] M. Iliadis *et al.*, "Deep fully-connected networks for video compressive sensing," *Digital Signal Processing*, vol. 72, pp. 9 – 18, 2018. doi: 10.1016/j.dsp.2017.09.010

[44] J. Zhang and B. Ghanem, "ISTA-Net: interpretable optimization-inspired deep network for image compressive sensing," in *IEEE/CVF conference on computer vision and pattern recognition*, Jun. 2018, pp. 1828–1837. doi: 10.1109/CVPR.2018.00196

[45] B. Sun *et al.*, "A deep learning framework of quantized compressed sensing for wireless neural recording," *IEEE Access*, vol. 4, pp. 5169–5178, 2016. doi: 10.1109/ACCESS.2016.2604397

[46] A. Beck and M. Teboulle, “A fast iterative shrinkage-thresholding algorithm for linear inverse problems,” *SIAM Journal on Imaging Sciences*, vol. 2, no. 1, pp. 183–202, 2009. doi: 10.1137/080716542

[47] P. E. McSharry *et al.*, “A dynamical model for generating synthetic electrocardiogram signals,” *IEEE Trans. on Biom. Eng.*, vol. 50, no. 3, pp. 289–294, Mar. 2003. doi: 10.1109/TBME.2003.808805

[48] S. Mallat, *A wavelet tour of signal processing: the sparse way*. Elsevier, 2008.

[49] E. van den Berg and M. P. Friedlander, “SPGL1: A solver for large-scale sparse reconstruction,” Apr. 2015.

[50] D. Gangopadhyay *et al.*, “Compressed sensing analog front-end for bio-sensor applications,” *IEEE J. Solid-State Circuits*, vol. 49, no. 2, pp. 426–438, Feb. 2014. doi: 10.1109/JSSC.2013.2284673

[51] Y. Zhu *et al.*, “A 10-bit 100-MS/s Reference-Free SAR ADC in 90 nm CMOS,” *IEEE J. Solid-State Circuits*, vol. 45, no. 6, pp. 1111–1121, Jun. 2010. doi: 10.1109/JSSC.2010.2048498

[52] F. Maloberti, *Data Converters*. Springer Electronics & Electrical Engineering, 2007.

[53] M. Pelgrom, “Nyquist Analog-to-Digital Conversion,” in *Analog-to-Digital Conversion*, M. Pelgrom, Ed. Cham: Springer International Publishing, 2017, pp. 285–403.

[54] D. Osipov and S. Paul, “Two-step reset method for energy-efficient SAR ADC switching schemes,” *Electronics Letters*, vol. 52, no. 10, pp. 816–817, May 2016. doi: 10.1049/el.2016.0890

[55] J.-Y. Lin and C.-C. Hsieh, “A 0.3 V 10-bit 1.17 f SAR ADC With Merge and Split Switching in 90 nm CMOS,” *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 62, no. 1, pp. 70–79, Jan. 2015. doi: 10.1109/TCSI.2014.2349571

[56] T. Iizuka *et al.*, “Comprehensive Analysis of Distortion in the Passive FET Sample-and-Hold Circuit,” *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 65, no. 4, pp. 1157–1173, Apr. 2018. doi: 10.1109/TCSI.2018.2797987

[57] Y. Zhang *et al.*, “A 10-b 200-kS/s 250-nA Self-Clocked Coarse–Fine SAR ADC,” *IEEE Trans. Circuits Syst. II Express Briefs*, vol. 63, no. 10, pp. 924–928, Oct. 2016. doi: 10.1109/TCSII.2016.2538139

[58] A. Abidi and H. Xu, “Understanding the regenerative comparator circuit,” in *Proceedings of the IEEE 2014 Integrated Circuits Conference*. IEEE, Sep. 2014, pp. 1–8. doi: 10.1109/CICC.2014.6946003

[59] F. Trinca, “Test and characterization of an integrated circuit implementing a low-energy compressed-sensing based acquisition system,” Master’s thesis, Politecnico di Torino, 2022.

[60] M. Pelgrom *et al.*, “Matching properties of mos transistors,” vol. 24, no. 5, pp. 1433–1439, 1989.

[61] R. D. Middlebrook, *Differential Amplifiers*. New York: John Wiley & Sons Inc, 1963.

[62] D. M. Binkley, *Tradeoffs and optimization in analog CMOS design*. Chichester, England; Hoboken, NJ: John Wiley & Sons, 2008.

[63] J. A. Croon *et al.*, *Matching properties of deep sub-micron MOS transistors*, ser. The Kluwer international series in engineering and computer science. New York: Springer, 2005, no. 851.

[64] D. Reid *et al.*, “Understanding LER-Induced MOSFET vt Variability—Part I: Three-Dimensional Simulation of Large Statistical Samples,” *IEEE Transactions on Electron Devices*, vol. 57, no. 11, pp. 2801–2807, Nov. 2010. doi: 10.1109/TED.2010.2067731

[65] M. Conti *et al.*, “Parametric yield formulation of MOS IC’s affected by mismatch effect,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 18, no. 5, pp. 582–596, May 1999. doi: 10.1109/43.759074

[66] T. Poiroux *et al.*, “Multiscale statistically correlated variability: A unified model for computer-aided design,” *IEEE Transactions on Electron Devices*, vol. 62, no. 11, pp. 3605–3612, Nov. 2015. doi: 10.1109/TED.2015.2478912

[67] M. Conti *et al.*, “Layout-based statistical modeling for the prediction of the matching properties of MOS transistors,” *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*, vol. 49, no. 5, pp. 680–685, May 2002. doi: 10.1109/TCSI.2002.1001958

[68] A. Hastings, *The art of analog layout*, 2nd ed. Upper Saddle River, NJ: Pearson/Prentice Hall, 2006.

[69] K. Ishida *et al.*, “Managing subthreshold leakage in charge-based analog circuits with low-VTH/ transistors by analog T- switch (AT-switch) and super cut-off CMOS (SCCMOS),” *IEEE J. Solid-State Circuits*, vol. 41, no. 4, pp. 859–867, Apr. 2006. doi: 10.1109/JSSC.2006.870761

[70] L. S. Y. Wong *et al.*, “Leakage current cancellation technique for low power switched-capacitor circuits,” in *2001 Int. Symp. on Low Power Electronics and Design*, 2001, pp. 310–315. doi: 10.1145/383082.383175

[71] A. H. Zemanian, *Distribution theory and transform analysis: an introduction to generalized functions, with applications*. New York, USA: McGraw-Hill, 1965.

[72] K. Ogata, *Modern control engineering*, 5th ed., ser. Prentice-Hall electrical engineering series. Instrumentation and controls series. Boston: Prentice-Hall, 2010.

[73] B. J. Wellman and J. B. Hoagg, "Root locus for a controller class that yields quadratic gain parameterization," in *2013 American Control Conference*, Jun. 2013, pp. 6625–6630. doi: 10.1109/ACC.2013.6580879

[74] S. R. Ovshinsky, "Reversible Electrical Switching Phenomena in Disordered Structures," *Physical Review Letters*, vol. 21, no. 20, pp. 1450–1453, Nov. 1968. doi: 10.1103/PhysRevLett.21.1450

[75] A. Redaelli, Ed., *Phase Change Memory*. Cham: Springer International Publishing, 2018.

[76] M. Carissimi *et al.*, "2-Mb Embedded Phase Change Memory With 16-ns Read Access Time and 5-Mb/s Write Throughput in 90-nm BCD Technology for Automotive Applications," in *ESSCIRC 2019 - IEEE 45th European Solid State Circuits Conference (ESSCIRC)*, Sep. 2019, pp. 135–138. doi: 10.1109/ESSCIRC.2019.8902656

[77] A. Antolini *et al.*, "An embedded PCM Peripheral Unit adding Analog MAC In-Memory Computing Feature addressing Non-linearity and Time Drift Compensation," in *ESSCIRC 2022 - IEEE 48th European Solid State Circuits Conference (ESSCIRC)*, Sep. 2022, pp. 109–112. doi: 10.1109/ESSCIRC55480.2022.9911447

[78] N. Papandreou *et al.*, "Programming algorithms for multilevel phase-change memory," in *2011 IEEE International Symposium of Circuits and Systems (ISCAS)*, May 2011, pp. 329–332. doi: 10.1109/ISCAS.2011.5937569

[79] D. E. Rumelhart *et al.*, "Learning representations by back-propagating errors," *Nature*, vol. 323, no. 6088, pp. 533–536, Oct. 1986. doi: 10.1038/323533a0

[80] "Wikipedia - Spline (mathematics)."

[81] "Wikipedia - Runge's phenomenon."

[82] J. Michelfeit, "multivar_horner: a python package for computing Horner factorisations of multivariate polynomials," 2020. doi: 10.48550/ARXIV.2007.13152

[83] J. A. Tropp and A. C. Gilbert, "Signal recovery from random measurements via orthogonal matching pursuit," *IEEE Trans. on Inf. Theory*, vol. 53, no. 12, pp. 4655–4666, Dec. 2007. doi: 10.1109/TIT.2007.909108

[84] C. S. Oxvig *et al.*, "Magni: A Python Package for Compressive Sampling and Reconstruction of Atomic Force Microscopy Images," *Journal of Open Research Software*, vol. 2, Oct. 2014. doi: 10.5334/jors.bk

[85] Y. Zigei *et al.*, “The weighted diagnostic distortion (WDD) measure for ECG signal compression,” *IEEE Trans. on Biom. Eng.*, vol. 47, no. 11, pp. 1422–1430, Nov. 2000. doi: 10.1109/TBME.2000.880093

[86] M. Abadi *et al.*, “TensorFlow: large-scale machine learning on heterogeneous systems,” 2015, software available from tensorflow.org.

[87] H. Xiao *et al.*, “Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms,” *arXiv:1708.07747 [cs, stat]*, Sep. 2017.

[88] D. P. Kingma and J. Ba, “Adam: A Method for Stochastic Optimization,” Jan. 2017. doi: 10.48550/arXiv.1412.6980

[89] G. F. Close *et al.*, “A 256-Mcell Phase-Change Memory Chip Operating at 2+ Bit/Cell,” *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 60, no. 6, pp. 1521–1533, Jun. 2013. doi: 10.1109/TCSI.2012.2220459

[90] A. Antolini *et al.*, “Characterization and Programming Algorithm of Phase Change Memory Cells for Analog In-Memory Computing,” *Materials*, vol. 14, no. 7, p. 1624, Mar. 2021. doi: 10.3390/ma14071624

[91] W. Haensch *et al.*, “The Next Generation of Deep Learning Hardware: Analog Computing,” *Proceedings of the IEEE*, vol. 107, no. 1, pp. 108–122, Jan. 2019. doi: 10.1109/JPROC.2018.2871057

[92] M. Le Gallo *et al.*, “Compressed Sensing With Approximate Message Passing Using In-Memory Computing,” *IEEE Transactions on Electron Devices*, vol. 65, no. 10, pp. 4304–4312, Oct. 2018. doi: 10.1109/TED.2018.2865352

[93] A. Krizhevsky, “Learning multiple layers of features from tiny images,” Tech. Rep., 2009.

[94] Y. Lecun *et al.*, “Gradient-based learning applied to document recognition,” *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998. doi: 10.1109/5.726791

[95] X. Sun *et al.*, “Pcm-based analog compute-in-memory: Impact of device non-idealities on inference accuracy,” *IEEE Transactions on Electron Devices*, vol. 68, no. 11, pp. 5585–5591, 2021. doi: 10.1109/TED.2021.3113300

[96] A. Antolini *et al.*, “Combined HW/SW Drift and Variability Mitigation for PCM-based Analog In-memory Computing,” *Journal of Emerging and Selected Topics in Circuits and Systems (Under revision)*, 2023.

[97] D. P. Kingma and J. Ba, “Adam: A Method for Stochastic Optimization,” *arXiv:1412.6980 [cs]*, Jan. 2017.

[98] S. Ambrogio *et al.*, “Reducing the Impact of Phase-Change Memory Conductance Drift on the Inference of large-scale Hardware Neural Networks,” in *2019 IEEE International Electron Devices Meeting (IEDM)*, Dec. 2019, pp. 6.1.1–6.1.4. doi: 10.1109/IEDM19573.2019.8993482

[99] V. Peluso and A. Calimera, “Energy-driven precision scaling for fixed-point convnets,” in *2018 IFIP/IEEE International Conference on Very Large Scale Integration (VLSI-SoC)*, 2018, pp. 113–118. doi: 10.1109/VLSI-SoC.2018.8644902

[100] V. Joshi *et al.*, “Accurate deep neural network inference using computational phase-change memory,” *Nat Commun 11, 2473*, 2020. doi: 10.1038/s41467-020-16108-9

[101] Z. He *et al.*, “Noise Injection Adaption: End-to-End ReRAM Crossbar Non-ideal Effect Adaption for Neural Network Mapping,” in *Proceedings of the 56th Annual Design Automation Conference 2019*, ser. DAC ’19. New York, NY, USA: Association for Computing Machinery, Jun. 2019, pp. 1–6. doi: 10.1145/3316781.3317870

[102] D. Ielmini *et al.*, “Status and challenges of in-memory computing for neural accelerators,” in *2022 International Symposium on VLSI Technology, Systems and Applications (VLSI-TSA)*. Hsinchu, Taiwan: IEEE, Apr. 2022, pp. 1–2. doi: 10.1109/VLSI-TSA54299.2022.9770972