Electronics Engineering
A.Y. 2024/2025
Neuro-inspired Electronics

RRAM-Based Analog Circuit for Matrix Inversion: Simulation and Analysis

Marco Mura

Contents

1 Introduction **1**

2 RRAM-Based Matrix Inversion: Theoretical Background **2**

2.1 RRAM-Based Matrix Inversion Circuit 2

2.2 Encoding Strategies and Non-Idealities 4

3 Simulations and Results **5**

3.1 Evaluation of Encoding Strategies under Variability and RTN Effects 5

3.2 Effects of Line Parasitic Resistances 12

3.3 Handling Matrices with Positive and Negative Entries 13

3.4 Power Consumption Estimation 14

4 Final Remarks **15**

Bibliography **16**

Chapter 1

Introduction

This report presents the work carried out to simulate, using MATLAB and SIMULINK, an RRAM-based analog accelerator designed to solve algebraic linear inverse problems. The simulations were performed under both ideal and non-ideal conditions, employing different encoding strategies to evaluate the circuit performance and robustness.

The remainder of this report is organized as follows:

- **Chapter 2** provides the theoretical background of the RRAM-based analog circuits employed in this project for solving the linear inverse problem. A brief overview of the encoding strategies and the main sources of non-idealities is also presented.
- **Chapter 3** describes in detail the most significant simulations and the corresponding results, followed by a discussion of their implications and relevance to the overall system performance.
- **Chapter 4** presents the concluding remarks, summarizing the main contributions and key findings of this work, as well as its limitations and possible directions for future improvements.

Chapter 2

RRAM-Based Matrix Inversion: Theoretical Background

This chapter provides a concise overview of the fundamental theoretical concepts underlying the work presented in this project. Section 2.1 introduces the RRAM-based AMC circuit for solving linear systems as proposed in [1]. Section 2.2 briefly discusses possible encoding strategies for RRAM devices and the main sources of non-idealities.

2.1 RRAM-Based Matrix Inversion Circuit

The linear inverse problem consists of solving the following matrix equation:

$$\boldsymbol{A}\boldsymbol{x} = \boldsymbol{y}, \tag{2.1}$$

where $\boldsymbol{y}$ is known and $\boldsymbol{x}$ is the unknown vector to be determined. Equation 2.1 represents a system of linear equations, the solution of which is computationally more demanding than a simple matrix–vector multiplication (MVM) in digital computers.

In [1], among others, an AMC circuit based on crosspoint RRAM arrays is proposed to tackle this problem. This circuit combines conventional CMOS technology with emerging memory devices to efficiently solve such systems. This approach leverages the benefits of in-memory computing while maintaining high area efficiency.

Fig. 2.1 illustrates an AMC circuit in which feedback loops are established between the rows and columns of a crosspoint RRAM array through operational amplifiers (OAs) to perform matrix inversion. Each row of the array is connected to the inverting input of an OA, while the columns are linked to the OA outputs. During operation, negative feedback forces the row potentials to virtual ground. The input vector $\boldsymbol{y}$ is applied as external voltages across load resistors, whereas the output voltages correspond to the vector $\boldsymbol{x}$, generating currents flowing toward the row lines held at virtual ground.

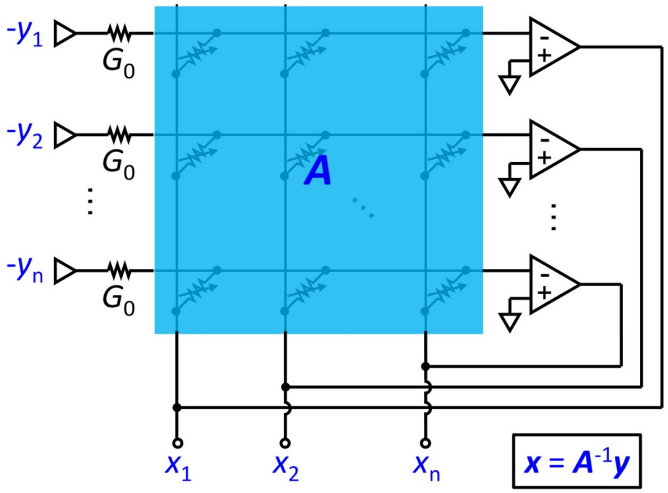

Figure 2.1: AMC circuit for matrix inversion.

According to Kirchhoff's current law, the sum of currents at each array row is zero, leading to the relationship $\boldsymbol{A}\boldsymbol{x} - \boldsymbol{y} = 0$. Given $\boldsymbol{A}$ and $\boldsymbol{y}$, the solution for $\boldsymbol{x}$ is obtained as:

$$\boldsymbol{x} = \boldsymbol{A}^{-1}\boldsymbol{y}, \tag{2.2}$$

where $\boldsymbol{A}^{-1}$ denotes the inverse of matrix $\boldsymbol{A}$. For the inversion to be valid, $\boldsymbol{A}$ must be a square $(n \times n)$ and non-singular matrix. The circuit in Fig. 2.1 operates when $\boldsymbol{A}$ contains only positive values. To invert a matrix that also includes negative entries, the original matrix $\boldsymbol{A}$ must be decomposed as $\boldsymbol{A}_+ = \frac{|\boldsymbol{A}|+\boldsymbol{A}}{2}$ and $\boldsymbol{A}_- = \frac{|\boldsymbol{A}|-\boldsymbol{A}}{2}$ (both containing only positive elements), and the circuit must be modified as shown in Fig. 2.2.

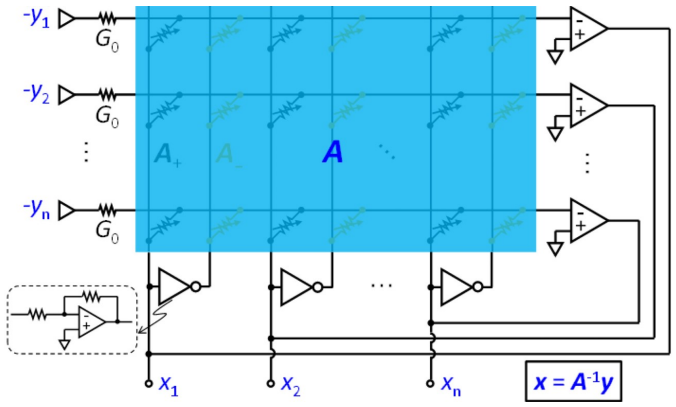

Figure 2.2: Matrix inversion circuit for matrices containing negative entries.

The stability of the circuit is not addressed here. Information about it can be found in [1].

2.2 Encoding Strategies and Non-Idealities

RRAM Encoding Strategies

To enable RRAM devices to operate as analog elements, a set of stable and reliable resistance levels is required. Several methods can be used to achieve this condition; the following encoding techniques were tested in this project:

- $\boldsymbol{I_{comp}}$ **modulation:** by limiting the current during the SET operation (current compliance), the device can reach different low-resistance states (LRS).
- $\boldsymbol{V_{reset}}$ **modulation:** adjusting the RESET voltage results in different high-resistance states (HRS).
- **Multiple devices:** multiple devices can be combined to store a single value when higher precision is required than what can be achieved with one device. In this case, the resistance levels of each device were still obtained through current compliance modulation.

Non-Idealities in Resistive Crossbars

Resistive crossbar arrays, while promising for in-memory analog computing, inherently suffer from several device and circuit level non-idealities [2] that degrade computational accuracy. These imperfections contribute differently to the overall deviation from the ideal behavior.

In this work, two major classes of non-idealities are considered:

- **Device variability and Random Telegraph Noise (RTN):** These belong to the class of stochastic non-idealities, arising from intrinsic device-to-device and cycle-to-cycle variations in RRAM conductance, as well as temporal fluctuations caused by RTN. Their effect is a random perturbation of the stored conductance values, leading to noisy outputs and an increased Root Mean Square Error (RMSE) in the computed vector.
- **Line parasitic resistances:** These are linear circuit non-idealities associated with resistive drops along word and bit lines. As the array size or wire resistance increases, the applied voltages across the cells become non-uniform, resulting in systematic current attenuation and linear distortion of the computed output. This effect becomes particularly relevant when comparing low-parasitic and high-parasitic conditions.

Overall, variability and RTN introduce random errors, while line parasitics cause deterministic distortions. Both mechanisms contribute to a degradation of accuracy, reflected in an increased RMSE with respect to the ideal case.

Chapter 3

Simulations and Results

This chapter presents the most significant experiments conducted in this project and discusses the main results, highlighting the key observations derived from them. All simulations were performed using MATLAB and SIMULINK. The MATLAB scripts and SIMULINK models are available in the attached `Simulations` folder. Multiple Monte Carlo trials were executed to ensure statistical reliability. The behavior of different RRAM technologies was explored for 5x5 and 7x7 RRAM crossbar circuit (5x5 case simulation results only are shown for reducing the simulation time). The Normalized Root Mean Square Error (NRMSE) is used as the main figure of merit to evaluate how accurately the analog circuit performs the linear inversion task. The final section of the chapter is devoted to estimating the power consumption of the circuits employed in the simulations.

3.1 Evaluation of Encoding Strategies under Variability and RTN Effects

This section investigates the different encoding strategies introduced in Chapter 2. The focus is on how the NRMSE varies with the number of levels used to store the weights in the devices (from 2 to 7 bits of resolution). The comparison between the ideal case and the non-ideal case, where variability and Random Telegraph Noise (RTN) are included, is carried out for each of the four technologies used in the simulations.

I_{comp} Encoding

This subsection reports the simulation results obtained for the four technologies when programming the devices using current compliance (I_{comp}) modulation (Fig. 3.2–3.5). Fig. 3.1 shows the Simscape implementation used in the simulations—a 5×5 passive RRAM crossbar circuit including the operational amplifiers.

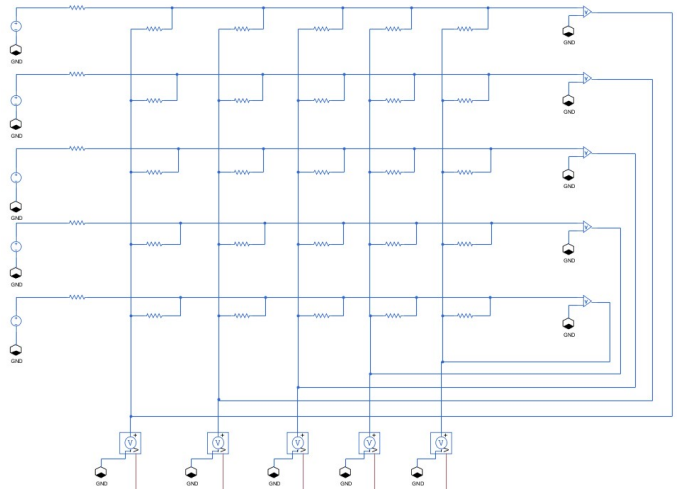

Figure 3.1: 5×5 passive RRAM crossbar Simscape circuit.

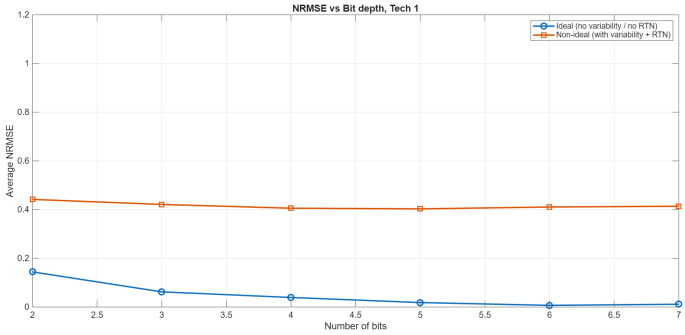

Figure 3.2: Average NRMSE vs. number of bits used to store the weights, ideal and non-ideal cases, technology 1.

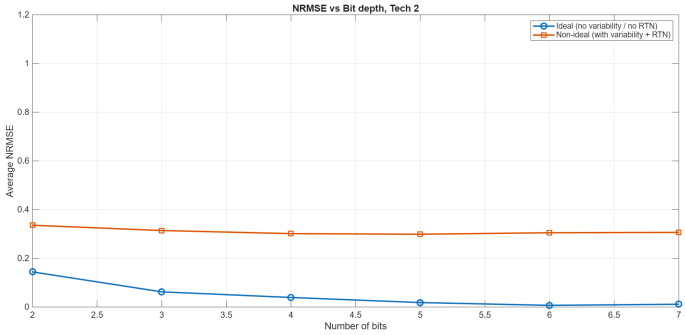

Figure 3.3: Average NRMSE vs. number of bits used to store the weights, ideal and non-ideal cases, technology 2.

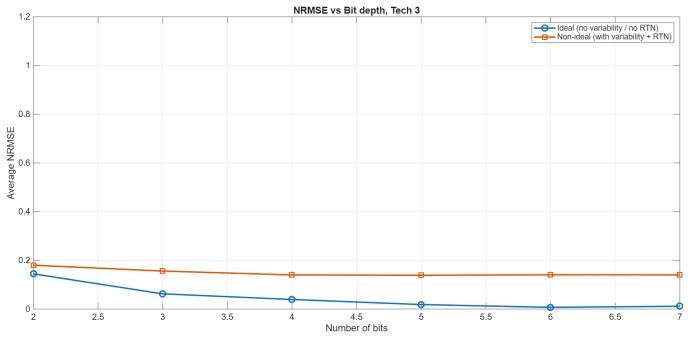

Figure 3.4: Average NRMSE vs. number of bits used to store the weights, ideal and non-ideal cases, technology 3.

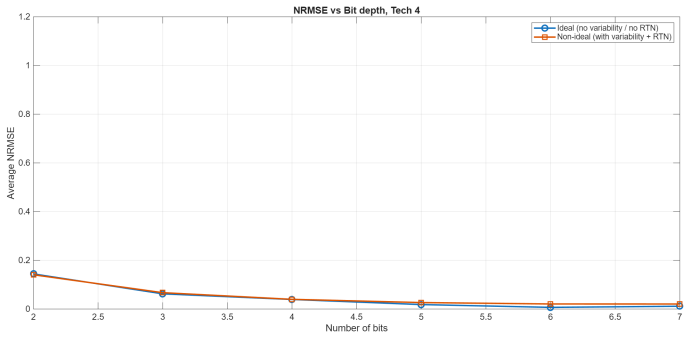

Figure 3.5: Average NRMSE vs. number of bits used to store the weights, ideal and non-ideal cases, technology 4.

As expected, in the ideal case, the NRMSE decreases as the bit resolution increases, reaching very low values (below 0.05, depending on the technology) when $N_{bits} > 4$. In contrast, in the non-ideal case, the NRMSE is generally higher and tends to saturate for $N_{bits} > 4$, indicating no benefit in using more than four bits (i.e., 16 levels) to represent the weights associated with matrix $\boldsymbol{A}$. This behavior is due to non-idealities that make some resistive states indistinguishable when too many levels are used.

Moreover, non-idealities introduce variability in the overall circuit behavior, the extent of which depends on the specific technology. As an example, Figures 3.6 and 3.7 show the dispersion of NRMSE across simulation trials for technologies 1 and 3, respectively. Technology 4 exhibits excellent performance even under non-ideal conditions, with NRMSE values almost indistinguishable from the ideal case, indicating that variability and RTN have a negligible effect on this technology.

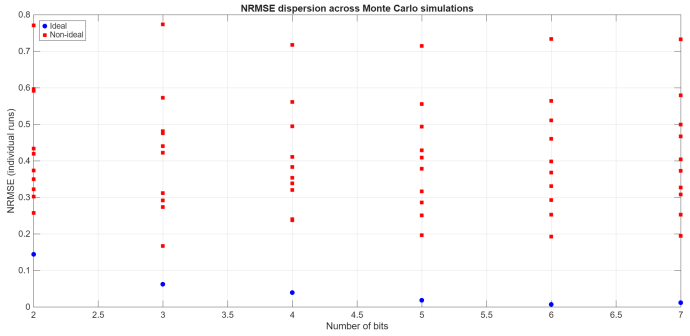

Figure 3.6: NRMSE dispersion across simulation trials vs. number of bits, ideal and non-ideal cases, technology 1.

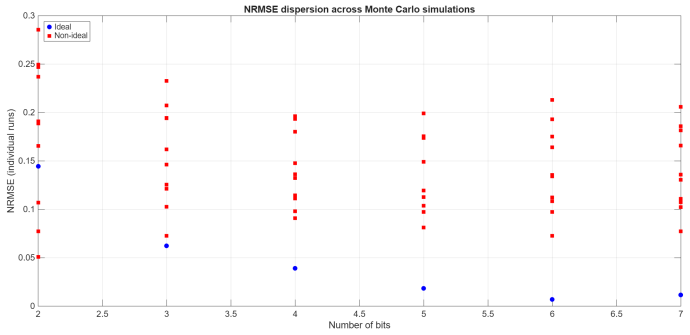

Figure 3.7: NRMSE dispersion across simulation trials vs. number of bits, ideal and non-ideal cases, technology 3.

V_{reset} Encoding

The simulations presented above are repeated using a different encoding strategy, namely modulation of the RESET voltage. Figures 3.8–3.11 show the obtained results. The same circuit used in the previous subsection (Fig. 3.1) is employed here as well.

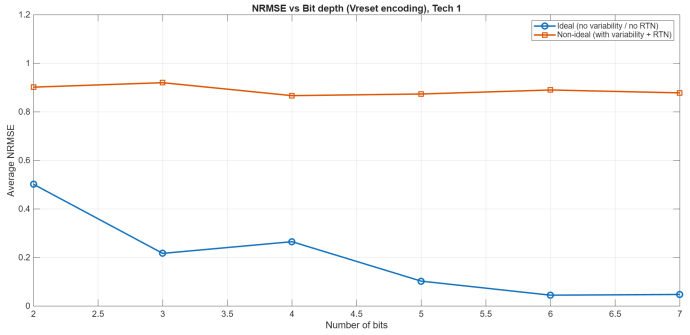

Figure 3.8: Average NRMSE vs. number of bits used to store the weights, ideal and non-ideal cases, technology 1.

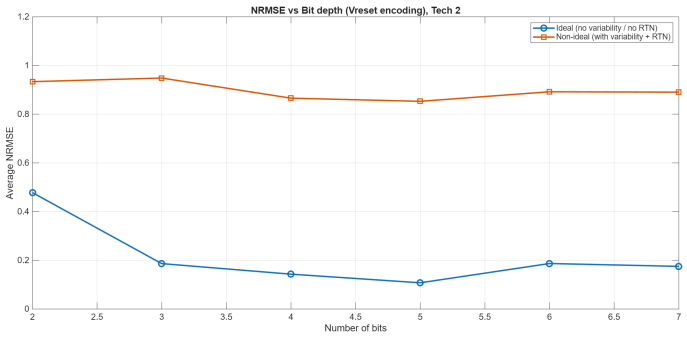

Figure 3.9: Average NRMSE vs. number of bits used to store the weights, ideal and non-ideal cases, technology 2.

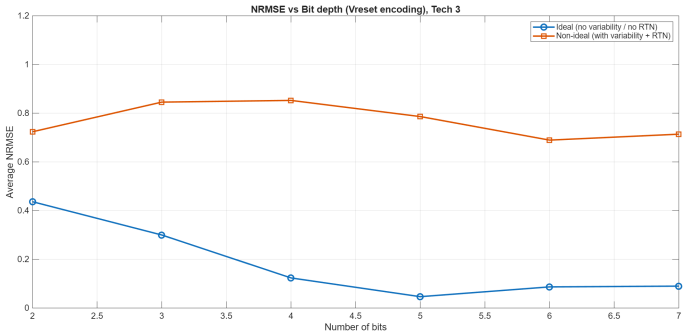

Figure 3.10: Average NRMSE vs. number of bits used to store the weights, ideal and non-ideal cases, technology 3.

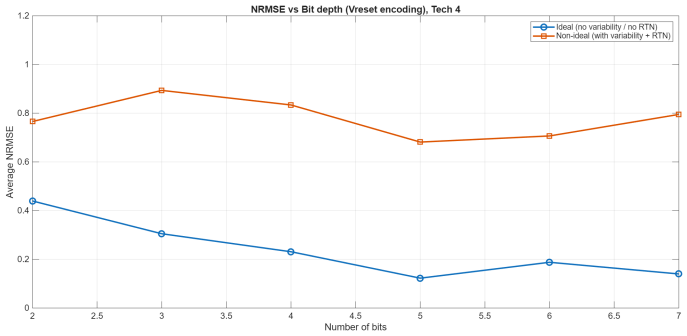

Figure 3.11: Average NRMSE vs. number of bits used to store the weights, ideal and non-ideal cases, technology 4.

The general trends observed here are similar to those discussed in the previous subsection. However, both the ideal and, more notably, the non-ideal cases exhibit higher NRMSE values. In the non-ideal scenario, the NRMSE remains high for almost all bit resolutions and across all technologies, making it impractical to obtain an accurate solution for the linear inversion task. This degradation in performance can be attributed to the nature of the mapping between the matrix $\boldsymbol{A}$ and the resistive states when the RESET voltage is used as the programming parameter. Specifically, the relationship between the device resistance and the RESET voltage is highly nonlinear, which necessitates a logarithmic mapping. Such nonlinearity amplifies the effect of device variability and RTN, resulting in poorer accuracy compared to the I_{comp}

encoding approach.

Multiple Devices Encoding

Finally, this subsection presents the results obtained using the third investigated encoding strategy. The circuit configuration was modified as shown in Fig. 3.12. Instead of using an $N \times N$ RRAM crossbar to store the weights, an $N \times 2N$ configuration was adopted—meaning that two columns are used to represent each column of matrix $\boldsymbol{A}$. Each pair of devices stores a single value, by splitting it into its most significant (MS) and least significant (LS) parts. The currents flowing through the LS branches are scaled down by the number of layers used for each device, ensuring that their contribution is proportionally smaller. Figures 3.13–3.16 show the obtained results.

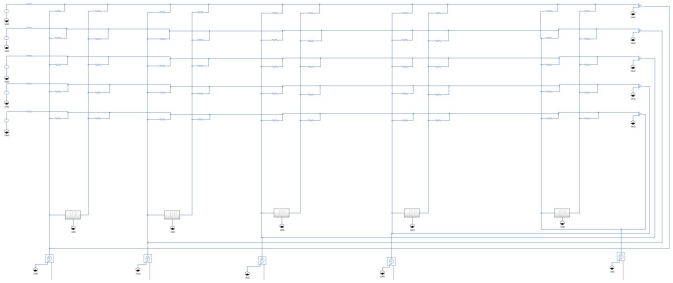

Figure 3.12: 5×5 passive RRAM crossbar Simscape circuit for the multiple devices encoding strategy.

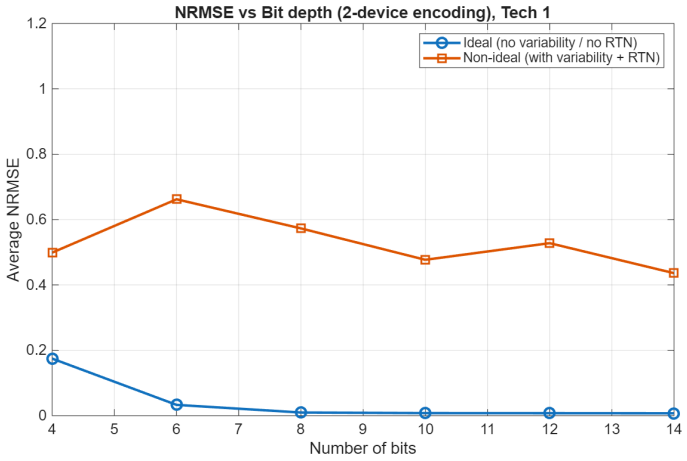

Figure 3.13: Average NRMSE vs. number of bits used to store the weights, ideal and non-ideal cases, technology 1.

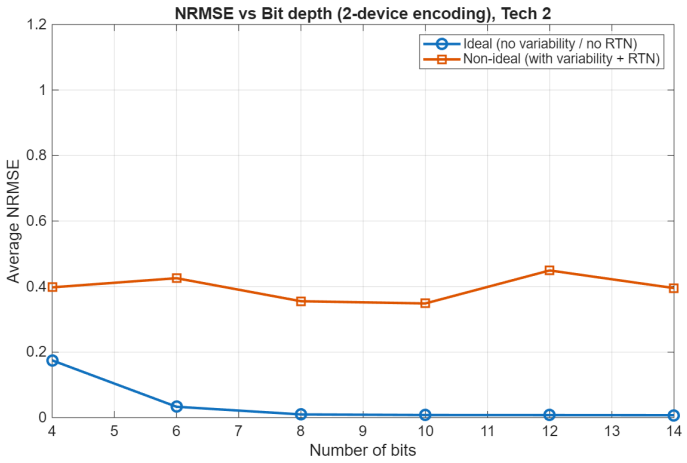

Figure 3.14: Average NRMSE vs. number of bits used to store the weights, ideal and non-ideal cases, technology 2.

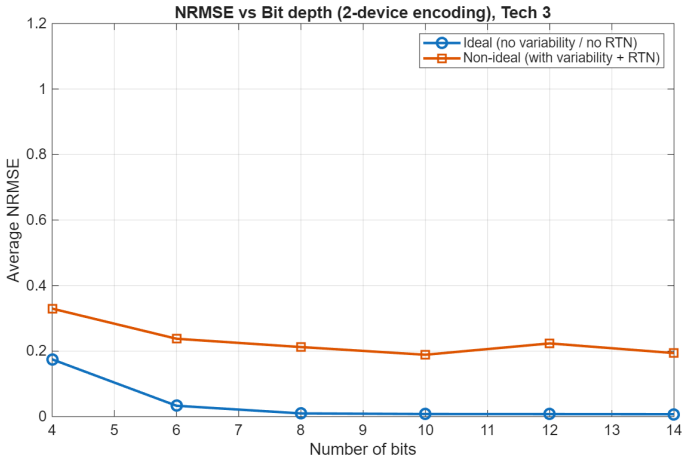

Figure 3.15: Average NRMSE vs. number of bits used to store the weights, ideal and non-ideal cases, technology 3.

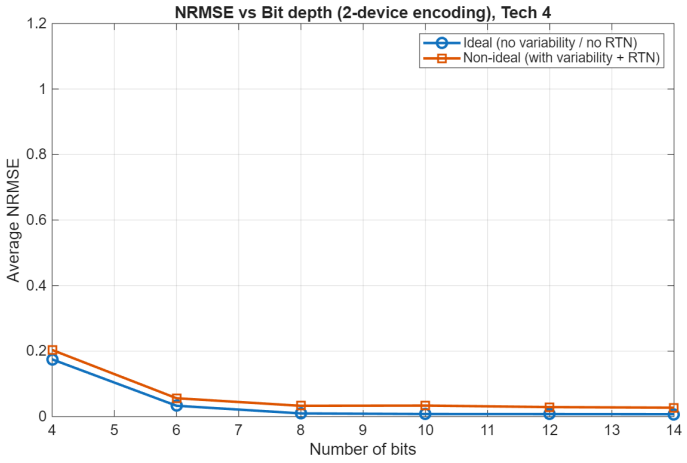

Figure 3.16: Average NRMSE vs. number of bits used to store the weights, ideal and non-ideal cases, technology 4.

As shown by the results, using two devices to store a single weight ideally allows for better performance compared to using one device with the same number of bits per device. This improvement can be clearly observed by comparing Fig. 3.2 and Fig. 3.13. However, in practice, the situation is more complex. When examining the non-ideal curves, it becomes evident that there is no significant improvement in NRMSE under non-ideal conditions when comparing the case where each weight of matrix $\boldsymbol{A}$ is mapped to a single device with the case where two devices are used. This indicates that the potential theoretical benefits of the multiple-device encoding approach are largely mitigated by the effects of variability and RTN in realistic scenarios. On the contrary, using two devices to store a single weight has the drawback of requiring a larger area and additional components, including operational amplifiers, which are among the main sources of power consumption in this type of circuit.

3.2 Effects of Line Parasitic Resistances

In this section, line parasitic resistors are introduced in the 5×5 crossbar circuit to evaluate the impact of this non-ideality. The current compliance encoding technique is used with a fixed bit resolution of 4 bits. Simulations are performed by increasing the parasitic resistance value from 3 Ω to 103 Ω in 20 Ω steps. Figure 3.17 shows the modified circuit, based on the one in Fig. 3.1, including the line parasitic resistors. Figure 3.18 reports the simulation results. Only the outcome for technology 3 is presented, as the other technologies exhibit the same qualitative behavior.

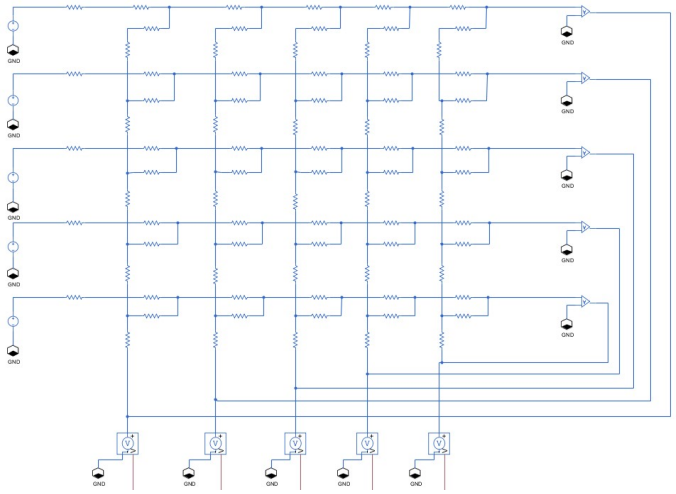

Figure 3.17: 5×5 passive RRAM crossbar Simscape circuit including line parasitic resistors.

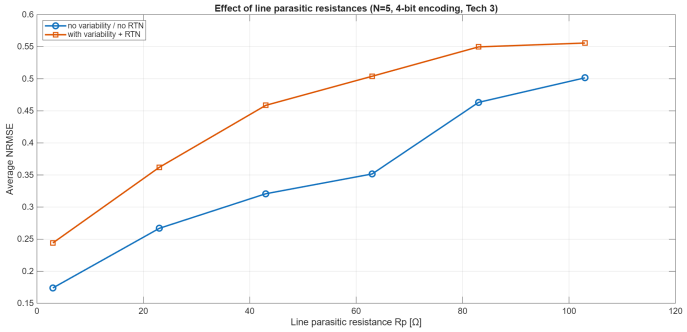

Figure 3.18: Average NRMSE vs. line parasitic resistance, ideal and non-ideal cases, technology 3.

As expected, the performance degrades as the value of the line parasitic resistance (assumed constant for all lines) increases. A resistance as low as 43 Ohm results in an NRMSE of approximately 0.32 in the ideal case and about 0.46 when variability and RTN are included. Achieving acceptable accuracy becomes unrealistic for higher parasitic resistance values. Moreover, as the size of the RRAM crossbar increases, the overall circuit performance is expected to worsen further, thus limiting the feasible size of the linear inverse problem.

3.3 Handling Matrices with Positive and Negative Entries

So far, only matrices with positive entries have been considered. In this section, matrices containing both positive and negative elements are analyzed. The procedure to handle such matrices was described in Chapter 2. The modified circuit is shown in Fig. 3.19, where line parasitic resistors are not included. As in the previous section, only one representative result is shown, since the others exhibit similar trends.

Figure 3.20, when compared with Fig. 3.2, demonstrates a noticeable degradation in performance when negative entries are included in the matrix. This degradation is likely due to the additional uncertainty introduced by the more complex mapping between the matrix elements and the conductance values. Another drawback of this approach is the need for a more complex circuit architecture, requiring exactly twice the number of components, which results in a larger area and higher power consumption.

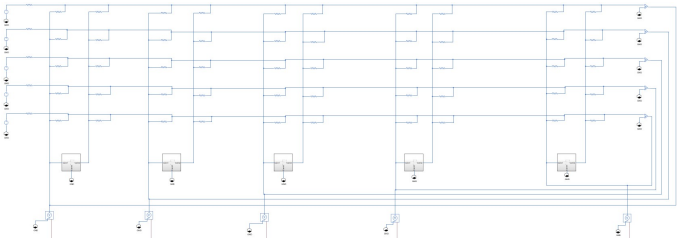

Figure 3.19: 5×5 passive RRAM crossbar Simscape circuit for matrices with both positive and negative entries.

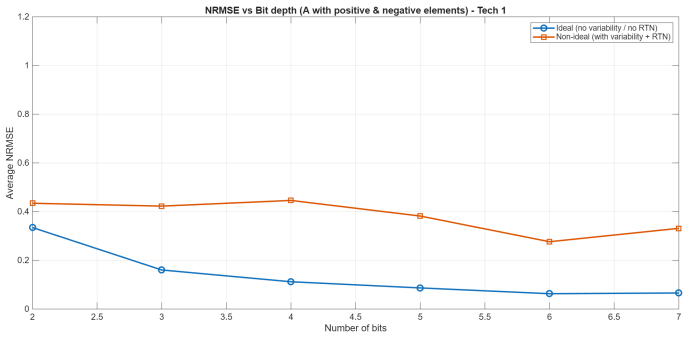

Figure 3.20: Average NRMSE vs. number of bits used to store the weights for a matrix with both positive and negative entries, ideal and non-ideal cases, technology 1.

3.4 Power Consumption Estimation

This section presents a methodology to estimate the power consumption of the proposed crossbar circuits. The analysis focuses on the read operation only, neglecting the write phase in which the RRAM devices are programmed to store the weights of matrix A. Let us start from the simplest configuration shown in Fig. 3.1. Two main contributions must be considered: the effective analog RRAM crossbar power consumption $P_{crossbar}$ and the power dissipated by the operational amplifiers P_{op-amp}. For the crossbar contribution, the power dissipated on each row i is approximately $G_0 \bar{V}_{read}^2$. Assuming an average $\bar{V}_{read} \sim 0.05$ V, $G_0 = 10^{-4}$ S, and $N = 5$ rows, we obtain $P_{crossbar,row} \sim 1$ μW. The power on each column depends on whether the devices are programmed in the low-resistance state (LRS) or high-resistance state (HRS). In the LRS case, with $G_{i,j} \sim 10^{-3}$ S and $V_{x,j} \sim 10^{-2}$ V (from empirical simulations), a 5×5 crossbar yields $P_{crossbar,column} \sim 2.5$ μW. In the HRS case, with $V_{x,j} \approx 10^{-3}$ V and $G_{i,j} \sim 10^{-4}$ S, the resulting $P_{crossbar,column} \approx 1$ nW is negligible compared to $P_{crossbar,row}$. Therefore, we can estimate $P_{crossbar} \sim 1$ μW overall. For the operational amplifier contribution, assuming ultra–low-power op-amps with a quiescent current $I_q \sim 1$ μA and supply voltage $V_{DD} \sim 1.2$ V, the total dissipation is around $P_{op-amp} \sim 10$ μW. Hence, the peripheral (op-amp) power consumption dominates over the crossbar's intrinsic analog power. Line parasitic resistances therefore have a negligible impact on total power consumption. For more complex circuits, such as those in Fig. 3.12 and Fig. 3.19, which employ multiple devices per weight or handle matrices A with negative entries, the overall power increase is mainly due to the additional peripheral circuitry (i.e., more op-amps) rather than to the larger number of RRAM devices. In summary, the estimated power consumption of a 5×5 crossbar circuit ranges from a few tens of μW for simpler configurations to about one hundred μW for more sophisticated architectures.

Chapter 4

Final Remarks

We now draw some conclusions regarding the work carried out and the results obtained. This work investigated the use of RRAM-based analog circuits to solve linear inverse problems through simulation in MATLAB and SIMULINK.

The results presented in Chapter 3 demonstrate that RRAM-based analog circuits represent a promising alternative to traditional CMOS technology for performing such computations, enabling low-latency and energy-efficient in-memory processing. These findings confirm the potential of analog computing for matrix-based operations and highlight the advantages of resistive memory devices in this context.

However, non-idealities such as device variability, Random Telegraph Noise (RTN), and line parasitic resistances can significantly degrade computational accuracy, depending also on the specific encoding strategy adopted. This suggests that careful design trade-offs between precision, circuit complexity, and robustness are necessary for practical implementations.

Future work could include exploring additional encoding strategies, modeling other non-idealities, and extending the analysis to larger matrices to further assess scalability and performance. Furthermore, experimental validation on fabricated RRAM arrays would provide a valuable benchmark for the simulation-based results obtained in this project.

Bibliography

[1] Zhong Sun and Daniele Ielmini. "Invited Tutorial: Analog Matrix Computing With Crosspoint Resistive Memory Arrays". In: *IEEE Transactions on Circuits and Systems II: Express Briefs* 69.7 (2022).

[2] Indranil Chakraborty et al. "Resistive Crossbars as Approximate Hardware Building Blocks for Machine Learning: Opportunities and Challenges". In: *Proceedings of the IEEE* 108.12 (2020).