2022 IEEE 28th International Symposium on On-Line Testing and Robust System Design (IOLTS) | 978-1-6654-7355-2/22/$31.00 ©2022 IEEE | DOI: 10.1109/IOLTS56730.2022.9897183

Yield Evaluation of Faulty Memristive Crossbar Array-based Neural Networks with Repairability

Anu Bala*, Saurabh Khandelwal*, Abusaleh Jabir* and Marco Ottavi†
*School of ECM, Oxford Brookes University, UK
Email: {15057719, skhandelwal, ajabir}@brookes.ac.uk
†University of Rome Tor Vergata and Italy, University of Twente, The Netherlands
Email: m.ottavi@utwente.nl

***Abstract*—This paper evaluates the yield of a memristor-based crossbar array of artificial neural networks in the presence of stuck-at-faults (SAFs). A technique based on Markov chains is used to estimate the yield in the presence of stuck-at-faults. This method provides a high degree of accuracy. Another method that is used for analysis and comparison is the Poisson distribution, which uses the sum of all repairable fault patterns. A fault repair mechanism is also considered when evaluating the yield of the memristor crossbar array. The results demonstrate that the yield could be improved with redundancies and a higher repairable stuck-at-fault ratio.**

***Keywords*-Memristor, Memristive Crossbar, Stuck-at-Faults, Neural Network**

I. Introduction

A memristive crossbar architecture provides fast, low power circuits for high precision matrix-vector multiplication [1], [2]. The computational accuracy of memristor-based neural circuits is considerably affected by stuck-at-faults in memristor devices, as it is difficult to avoid stuck-at-faults during the manufacturing processes. Memristors can be stuck in a low or high resistive state. Therefore, these faults can limit the recognition accuracy of memristor-based neural circuits [3], [4].

In most cases, neural networks can accept a limited number of faulty synaptic weights; however, a high defect rate dramatically reduces the accuracy of the matrix-vector calculations. The retraining and weight mapping processes are used to train the neural network with faulty memristors [5]–[7].

This paper aims to calculate the yield of a memristor crossbar array used for neural networks using a Markov chain, which provides ease of use without sacrificing accuracy and representativeness. The benefit of the repair process is also considered while using the Markov chain model. The Poisson distribution approach proposed in [8] is also used for analysis and comparison as it is a faster industry-based approach for embedded SRAMs. Different aspects of these two evaluation techniques are also discussed.

The rest of the paper is organized as follows: Section II describes the yield models used for yield evaluation in this paper. Section III explains the simulation of a memristor crossbar array with stuck-at-faults. It also explains the state diagram and transition rates of the Markov chain that are used to evaluate the yield. The yield evaluation results are demonstrated in Section IV. Section V compares the results of both the methods used for yield calculations. The paper is concluded in Section VI.

II. Background

The yield is described as the probability of chip acceptability during the manufacturing process.

A. The Markov chain modeling

The Markov chains [9], [10] are a stochastic model that represents a series of probable events in which the next state's probabilities are entirely based on the events in the current state, not the previous states.

A labeled directed graph $G = (V, E)$ can be used to describe a Markov chain with state-space V and transition matrix P, where the edges are defined by nonzero probability transitions.

$$E = (u, v)|P_{u,v} > 0 \tag{1}$$

Here, an edge from u to v is labeled by the probability $P_{u,v}$. The matrix will be $N \times N$ if the Markov chain has N potential states. This matrix's rows must add up to one. An $N \times 1$ Initial State Vector is also included in a Markov chain.

In Markov chains, a higher-order transition matrix is used to determine the probability of that transition occurring over a number of steps.

B. Poisson Distribution

Let λ_0 be the mean number of faults of each type in a memristor crossbar array. The probability that a crossbar array has k faults is determined by the Poisson distribution function as shown in Eq. 2

$$P(k) = \frac{e^{-\lambda_0}\lambda_0^k}{k!}, \quad for \;\; k = 0, 1, 2 \cdots \tag{2}$$

The probability of a chip having no faults is known as the yield, which is determined for k = 0 by Eq. 3 i.e., if there is no redundancy on the chip.

$$Y = P(0) = e^{-\lambda_0} \tag{3}$$

978-1-6654-7355-2/22/$31.00 ©2022 IEEE

The additivity of faults is a particularly valuable characteristic of the Poisson distribution model. After repairing the memristor crossbar array with the stated redundancy, the yield is calculated as the sum of the probabilities of different types of faults by using Eq. 4.

$$Y = \sum P_{SA0}(i) P_{SA1}(j) \tag{4}$$

Here, the repair process is defined as a method that can correct the majority of the faults and Y is the yield after repairing various types of faults e.g. (i+j).

III. Stuck-at-Faults Simulation

The probability of stuck-at-faults (SAFs) in the memristor crossbar is higher than the other faults like operational faults and variation in conductance values, and these faults have an adverse effect on the performance of memristor crossbar-based neural networks [3], [11]. While a neural network can usually accept a limited number of faulty weights, a higher rate of SAFs, especially SAFs at low resistance states, drastically reduces computational accuracy.

Research shows that some re-training techniques are proposed for memristive crossbar arrays that tolerate SAFs by utilizing the inherent fault tolerance of neural networks [6], [11]. These techniques are useful if a fault-tolerant network can be retrained with similar recognition accuracy. However, when the percentage of defects is higher than the neural network's natural fault tolerance, performance suffers.

To estimate the yield, the Markov model introduced in [12] is used in the memristor crossbar array and this model is based on the approach presented in [13] as it provides accurate results. In this paper, two types of stuck-at-faults are considered termed as $SA0$ (High Resistive State) and $SA1$ (Low Resistive State). $SA0$ is considered a repairable fault as it is assumed to be re-trained in the neural network. The $SA1$ fault is considered an unrepairable fault as it is hard to re-train in the network [11], [14]. Two and four spare columns are used to replace the faulty memristors in the crossbar array.

First, a 4x4 memristor crossbar is considered for yield evaluation and its state transition diagram is shown in Fig. 1. Here, circles denote the states and the edges denote the transition from one state to another. It starts from state G, which represents the 'GOOD' state and ends at state F which represents the 'FAIL' state. 'GOOD' state implies no faults in the crossbar and 'FAIL' implies that there is no further state to consider and the system fails. One single fault is processed at a time. Each state is represented by (i, j) where i represents the spare column and j represents the repairable faults. The next state depends on the present state in this algorithm.

The transition rate, denoted by λ is represented by the edge between two transition states. It is the weighted sum of all possible faults that can produce that transition. The weight represents the number of elements that are affected by that fault. The transition rates with the Markov model are shown in Table I. As shown in this table, the transition rate $\lambda_{G,(i^1,j^0)}$ is from a state with no spare column to a state with one spare column, where i represents the spare column and j represents the number of repairable faults. The transition rate is calculated by considering the faults that are responsible for generating that transition. For example, the transition rate $\lambda_{SA0(n_c n_r)}$ is the probability of having a fault in any row and column in the memristor crossbar, where the number of rows is indicated by n_r and the number of columns is indicated by n_c in Table I. The transition rates for a 4x4 crossbar state diagram calculated using a Markov chain are illustrated in Table I. The values of the fault rates must be divided by the number of columns or rows, whichever is greater (or their product). This is because, when describing a transition rate, all defects are considered to be independent and their weighted probabilities are assumed [12].

The yield is calculated by solving the following equation with a continuous-time Markov chain (CTMC):

$$P = P(0).A^k \tag{5}$$

$P(0)$ is a vector whose elements are the probability of being in a given state $(G, 1, 2, ..., F)$, and A is the generating matrix whose elements are the transition rates; k is the average defect rate. A matrix multiplication at each step is required to solve Eq. 5. After repair, the yield is calculated as the probability of not being in the fail state.

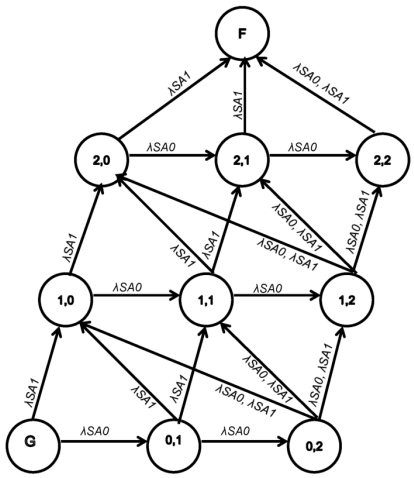

Fig. 1: State diagram for 4×4 memristor crossbar array.

IV. Results and Discussion

The yield analysis of the memristor crossbar is done using MATLAB [15]. The simulation results for a 4×4 crossbar array are shown in Fig.2. The average defect rate for this varies from 0 to 40 with a step size of $\Delta\lambda$=10^{-3}. The results show that the yield of a 4×4 crossbar with no spare column is less than with two spare columns. Additionally, different fault probabilities are considered to compare the yield of the crossbar as shown in Fig. 2. Three probability cases are considered as follows:

TABLE I: Transition Table

Transition Rates	Weighted sum of probable faults
$\lambda_{G,(i^0,j^1)}$	$\lambda_{SA0}(n_c n_r)$
$\lambda_{G,(i^1,j^0)}$	$\lambda_{SA1}(n_c n_r)$
$\lambda_{(i^0,j^n),(i^0,j^{n+1})}$	$\lambda_{SA0}(n_c n_r - n)$
$\lambda_{(i^0,j^n),(i^1,j^0)}$	$\lambda_{SA1}(n_c - (n_c - 1))(n_r - n)$
$\lambda_{(i^0,j^{n=m}),(i^1,j^0)}$	$\lambda_{(SA1+SA0)}(n_c - (n_c - 1))(n_r - n)$
$\lambda_{(i^0,j^n),(i^1,j^n)}$	$\lambda_{SA1}(n_c - n)n_r$
$\lambda_{(i^0,j^{n=m}),(i^1,j^n)}$	$\lambda_{(SA1+SA0)}(n_c - n)n_r$
$\lambda_{(i^1,j^{n-1}),(i^1,j^n)}$	$\lambda_{SA0}(n_c - 1)n_r - (n-1)$
$\lambda_{(i^1,j^{n-1}),(i^2,j^0)}$	$\lambda_{SA1}(n_c - n)n_r$
$\lambda_{(i^1,j^{n=m}),(i^2,j^0)}$	$\lambda_{(SA1+SA0)}(n_c - n)n_r$
$\lambda_{(i^1,j^n),(i^2,j^{n-1})}$	$\lambda_{SA1}(n_c - (n_c - 1))(n_r - n)$
$\lambda_{(i^1,j^{n=m}),(i^2,j^{n-1})}$	$\lambda_{(SA1+SA0)}(n_c - (n_c - 1))(n_r - n)$
$\lambda_{(i^2,j^{n-1}),(i^2,j^n)}$	$\lambda_{SA0}(n_c - 2)n_r - (n-1)$
$\lambda_{(i^2,j^{n-1}),F}$	$\lambda_{SA1}(n_c - 2)n_r - (n-1)$

- $SA0$>$SA1$. In this case, we considered the ratio of the $SA0$ faults to be higher than the $SA1$ faults.
- $SA0$<$SA1$. In this case, the ratio of $SA1$ faults is considered higher than $SA0$ faults.
- $SA0$=$SA1$. In this case, the ratio of both faults is considered equal.

With the increasing percentage of $SA0$ faults the yield increases. The yield is higher in the case where the ratio of $SA0$ faults is higher.

As we discussed in the previous section, if the percentage of SAFs increases beyond the internal tolerance of the neural network, then the computational accuracy decreases. Additionally, SAF probabilities increase with the larger array sizes. The yield is calculated for different-sized memristor crossbar arrays. The chosen array sizes are 4×4, 128×128, and 256×256. For all these array sizes, the yield has been calculated with zero and two spare columns and different fault ratios have been considered. The plot reported in Fig. 3, Fig. 4 shows the yield for 128×128, 256×256 memristor crossbar array with and without redundancies. The value of λ_0 varies from 0 to 40 in the plots reported in Fig. 3 and Fig. 4. All the resulted plots show that the yield is higher with spare columns as compared to no spare columns. In terms of fault ratio, results show that with the increase of repairable fault ratio, the yield increases. Fig. 5 and Fig. 6 illustrate the yield with two, four and no spare columns for the array sizes 256×256 and 512×512, respectively. Both SAFs are considered equal in these cases and the value of λ_0 varies from 0 to 30. The results demonstrate that adding more redundancies could boost the yield. As a result, crossbar array yield can be improved by using effective re-training methods for memristive neural networks with SAFs and redundancies.

V. Comparison

The yield calculated by using two methods is compared in this section. For comparison, the Poisson distribution [8] and the Markov chain model [12] are used to calculate yield. Many characteristics of these two methods are different. The Markov chain uses a repair procedure that evolves in real-time, whereas the Poisson distribution uses a static probabilistic analysis.

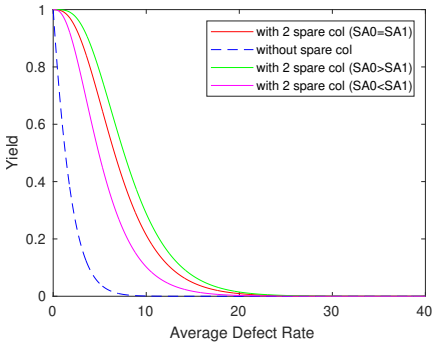

Fig. 2: Yield evaluation varying fault ratio for 4×4 memristor crossbar array.

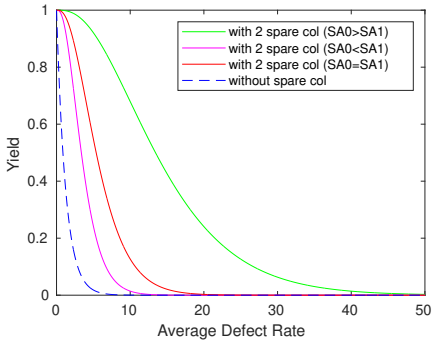

Fig. 3: Yield evaluation varying fault ratio for 128×128 memristor crossbar array.

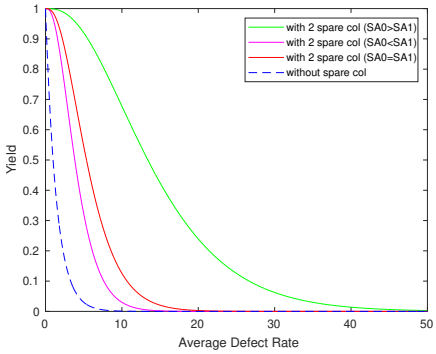

Fig. 4: Yield evaluation varying fault ratio for 256×256 memristor crossbar array.

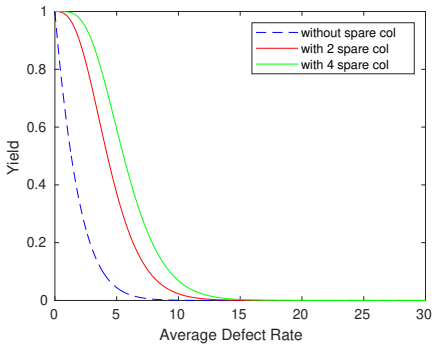

Fig. 5: Yield evaluation for 256×256 memristor crossbar array.

TABLE II: Comparison Table for Array size 4×4

	λ_0=0.2			λ_0=0.4			λ_0=2			λ_0=4		
	PD	MC	Difference	PD	MC	Difference	PD	MC	Difference	PD	MC	Difference
no spare col	0.9047	0.9042	-0.0005	0.8178	0.8162	-0.0016	0.3383	0.3323	-0.006	0.0916	0.0898	-0.0018
two spare col	0.9998	0.9999	1e-04	0.9988	0.9996	0.0008	0.891	0.9606	0.0696	0.5312	0.7945	0.2633
four spare col	1	1	0	1	1	0	0.9930	1	0.007	0.8730	1	0.127

TABLE III: Comparison Table for Array size 32×32

	λ_0=0.2			λ_0=0.4			λ_0=2			λ_0=4		
	PD	MC	Difference	PD	MC	Difference	PD	MC	Difference	PD	MC	Difference
no spare col	0.8178	0.8177	-1e-04	0.8178	0.8177	-1e-04	0.3383	0.3371	-0.0012	0.0916	0.0903	-0.0013
two spare col	0.9988	0.9989	1e-04	0.9988	0.9989	1e-04	0.891	0.9054	0.0144	0.5312	0.5711	0.0399
four spare col	1	1	0	1	1	0	0.9930	0.9951	0.0021	0.8730	0.9040	0.031

TABLE IV: Comparison Table for Array size 128×128

	λ_0=0.2			λ_0=0.4			λ_0=2			λ_0=4		
	PD	MC	Difference	PD	MC	Difference	PD	MC	Difference	PD	MC	Difference
no spare col	0.9047	0.9047	0	0.8178	0.8177	-1e-04	0.3383	0.3379	-0.0004	0.0916	0.0911	-0.0005
two spare col	0.9998	0.9998	0	9988	9989	1e-04	0.891	0.8988	0.0078	0.5312	0.5501	0.0189
four spare col	1	1	0	1	1	0	0.9930	0.9936	0.0006	0.8730	0.8812	0.0082

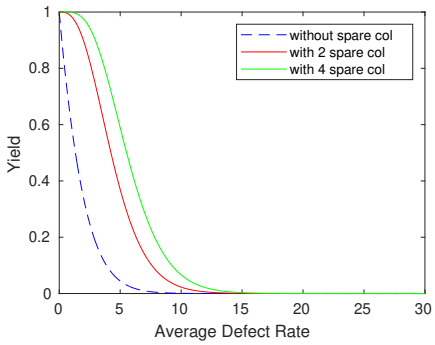

Fig. 6: Yield evaluation for 512×512 memristor crossbar array.

Yield is evaluated using these methods by considering different average defect rates and array sizes. The results are calculated and compared using two, four and no spare columns as shown in Tables II, III and IV. The same fault ratio is used in both methods. The execution processes of these two methods are also different in terms of memory size and aspect ratio. The Poisson distribution method does not contain these. In terms of complexity, the Markov chain approach is more complex than the Poisson distribution as it uses matrix multiplication as shown in Eq. 5. However, the number of matrix multiplications in this equation is proportional to the desired accuracy, which is higher for lower values of execution steps $\Delta\lambda$ and the highest value of the average fault number λ_0. On the other side, for available spare columns and a given number of faults, the Poisson distribution method determines if there is a repair configuration authorised by the available spare columns for each possible defect. As a result, it is required to compute fewer operations than the Markov chain. However, the Poisson distribution ignores the fault overlaps; it does not consider two horizontal pairs overlapping or two stuck-at-faults on the same column. As the position of the faults also affects the vector-matrix multiplication in memristive neural networks. We considered a lower to higher average defect rate for comparison and these values are considered based on the data used in [12]. The evaluated yield results for comparison are shown in Tables II, III and IV. The results show that both methods produce close or similar results for smaller values of λ_0 and the yield calculated by using the Poisson distribution is underestimated for higher values of average defect rate. As a result, higher defect rates should be considered in order to obtain more accurate results, and the Markov chain method is more effective for memristor crossbar based neural networks as it considers fault positions and array sizes while evaluating the yield and provides more accurate results for larger array sizes.

VI. Conclusion

In this paper, a Markov chain-based approach is used to calculate the yield of a memristor crossbar array. Yield is estimated for different sized memristor crossbar arrays with varying fault ratios. A Poisson distribution approach is also considered to evaluate the yield and compare it. The yield is calculated by considering zero, two, and four redundancies for chosen array sizes. The Markov chain is more complex than the Poisson distribution in terms of complexity. However, in terms of flexibility and accuracy, the Markov chain provides more accuracy with a higher average defect rate, as the results are shown. It is more flexible because the crossbar array can consider defect positions, which is also helpful in re-training methods used for memristive crossbar neural networks with stuck-at-faults. Thus, to obtain more accuracy, higher values of the average defect rate should be chosen. Hence, the Markov chain can be more effective for calculating the yield for memristive crossbar arrays used for deep neural networks because it provides more accurate results with a higher average defect rate.

REFERENCES

[1] C. Liu, Q. Yang, B. Yan, J. Yang, X. Du, W. Zhu, H. Jiang, Q. Wu, M. Barnell, and H. Li, "A memristor crossbar based computing engine optimized for high speed and accuracy," in *2016 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*. IEEE, 2016, pp. 110–115.

[2] M. Hu, J. P. Strachan, Z. Li, E. M. Grafals, N. Davila, C. Graves, S. Lam, N. Ge, J. J. Yang, and R. S. Williams, "Dot-product engine for neuromorphic computing: Programming 1t1m crossbar to accelerate matrix-vector multiplication," in *2016 53nd acm/edac/ieee design automation conference (dac)*. IEEE, 2016, pp. 1–6.

[3] L. Xia, W. Huangfu, T. Tang, X. Yin, K. Chakrabarty, Y. Xie, Y. Wang, and H. Yang, "Stuck-at fault tolerance in rram computing systems," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 8, no. 1, pp. 102–115, 2017.

[4] C.-Y. Chen, H.-C. Shih, C.-W. Wu, C.-H. Lin, P.-F. Chiu, S.-S. Sheu, and F. T. Chen, "Rram defect modeling and failure analysis based on march test and a novel squeeze-search scheme," *IEEE Transactions on Computers*, vol. 64, no. 1, pp. 180–190, 2014.

[5] O. Tunali and M. Altun, "A survey of fault-tolerance algorithms for reconfigurable nano-crossbar arrays," *ACM Computing Surveys (CSUR)*, vol. 50, no. 6, pp. 1–35, 2017.

[6] L. Chen, J. Li, Y. Chen, Q. Deng, J. Shen, X. Liang, and L. Jiang, "Accelerator-friendly neural-network training: Learning variations and defects in rram crossbar," in *Design, Automation Test in Europe Conference Exhibition (DATE), 2017*, 2017, pp. 19–24.

[7] B. Zhang, N. Uysal, D. Fan, and R. Ewetz, "Handling stuck-at-faults in memristor crossbar arrays using matrix transformations," in *Proceedings of the 24th Asia and South Pacific Design Automation Conference*, 2019, pp. 438–443.

[8] X. Wang, M. Ottavi, and F. Lombardi, "Yield analysis of compiler-based arrays of embedded srams," in *Proceedings 18th IEEE Symposium on Defect and Fault Tolerance in VLSI Systems*. IEEE, 2003, pp. 3–10.

[9] J. R. Norris and J. R. Norris, *Markov chains*. Cambridge university press, 1998, no. 2.

[10] G. O. Roberts, "Markov chain concepts related to sampling algorithms," *Markov chain Monte Carlo in practice*, vol. 57, pp. 45–58, 1996.

[11] C. Liu, M. Hu, J. P. Strachan, and H. Li, "Rescuing memristor-based neuromorphic design with high defects," in *2017 54th ACM/EDAC/IEEE Design Automation Conference (DAC)*, 2017, pp. 1–6.

[12] M. Ottavi, "Evaluating the yield of repairable srams for ate: Ieee transactions on instrumentation and measurement," *IEEE transactions on instrumentation and measurement*, 2006.

[13] B. Ciciani and G. Iazeolla, "A markov chain-based yield formula for vlsi fault-tolerant chips," *IEEE transactions on computer-aided design of integrated circuits and systems*, vol. 10, no. 2, pp. 252–259, 1991.

[14] C. Yakopcic, R. Hasan, and T. M. Taha, "Tolerance to defective memristors in a neuromorphic learning circuit," in *NAECON 2014-IEEE National Aerospace and Electronics Conference*. IEEE, 2014, pp. 243–249.

[15] C. B. Moler, *Numerical computing with MATLAB*. SIAM, 2004.