Analogue Auto-Associative Memory using a Multi-Valued Memristive Memory Cell

Mohammad Mahmoud A. Taha & Wim J.C. Melis
Department of Engineering Science
Faculty of Engineering and Science
University of Greenwich @ Medway
Chatham Maritime, United Kingdom
mohammad.m.a.taha@gmail.com & Wim.J.C.Melis@gre.ac.uk

Abstract—Most brain-like computing systems build up from neural networks. While there are some essential problems with this approach, it is well-known that the brain functionally operates as an associative memory. Building associative memories using conventional CMOS technology has already been performed, but this approach suffers from a lack of scalability and information density. Additionally, for a long time, one of the differences between analogue and digital electronics was the fact that digital electronics allowed for easier data storage through a variety of different memory cell architectures. These memory designs make extensive use of transistors and generally trade area, performance and power. However, memristors can be used as high density, analogue, passive storage elements and this paper presents 2 memory cell designs that allow for such multi-valued storage. The noise resistance of these cells is tested and indicates a very good tolerance to external influences, while overall they provide for a very accurate storage of data with high information density. Following on from the description of the storage cells, the paper then continues to build them into an associative memory.

Keywords—Memristive memory, Analogue memory, Associative memory, Memristors circuits, Memristive memory cell, Reliable memory, Multi-value memory, Non Volatile Memory

I. INTRODUCTION

Transistor based memories have been around for a very long time and can generally be classified to be of the SRAM or DRAM type. In either approach, there is a trade-off between power, area and performance [1]. Quite often a reduction of power requires an increasing number of transistors because it requires the ability to switch on/off certain parts of the cell in order to save power, eventually having a negative impact on area [2]. Normally, memory cells are developed to store one bit per cell, and multiple transistors are required for this single bit storage. Consequently, storing multiple bits requires a multiplication of the required number of cells and also the required number of transistors. Furthermore, due to decreasing feature sizes, transistors have become less reliable, which not only affects the overall memory performance, but has also increased leakage currents while increasing overall power consumption. Therefore, the ability to identify a passive, low power, multi-valued memory storage device would certainly be welcomed by many [3].

An associative memory is a storage circuit that mimics the low power parallel capabilities of the human brain in storing and retrieving data [4]. Associative memories depend on associating incoming data with already stored data in order to retrieve the data that matches the input. This stands in contrast to the conventional Random Access Memory (RAM) where the location of the stored data must be known in order to retrieve the desired data. In this context, there are two types of possible matching: 1) exact match, better known as *Content Addressable Memory* (CAM); and 2) best match. While in a CAM the inputs have to exactly match the stored data for the data to be retrieved [5], in the best match situation, the input does not have to exactly match the stored data. Yet, the memory chooses/retrieves the stored data that best matches the input depending on some metric. This metric could be based on either Euclidean or Hamming distances [6] for non-binary/analogue and binary/digital vectors respectively. Furthermore, associative memories can be classified according to the type of association, being 1) auto-association or 2) hetero-association, which refers to the retrieval of a pattern that is similar in format to the input pattern or of a different format from the input pattern [7].

While the passive elements: resistor, capacitor and inductor have been known for a very long time, in 1971, a fourth circuit element was added to this list by L. Chua [8]. This fourth element was called: the memristor, as short for memoryresistor [9]. In 2008, this memristor was physically fabricated at nano-scale, by a group in HP labs [10]. As a passive element, the memristor describes a non-linear relationship between charge and flux, which is normally depicted as a pinched hysteresis loop when plotting voltage versus current, as shown in Figure 1. The *Memristance*, or the actual weight of the memristor, can be altered by supplying the memristor with a particular voltage and/or current for a specific period of time. When these programming voltages/currents are removed, then the memristor remembers the last memristance [9]. The latter feature is the main reason for memristors to be used as non-volatile memories.

So, while memristors can be used for passive energy storage, when one uses them as analogue storage devices, then multi-valued storage can be achieved. Essentially, digital use is only an abstraction that would effectuate these values back to a 0 or 1, which can still be performed, although at a loss of information density.

II. Related Work

Memory cell design using memristors is certainly not a new field, and several designs have already been presented.

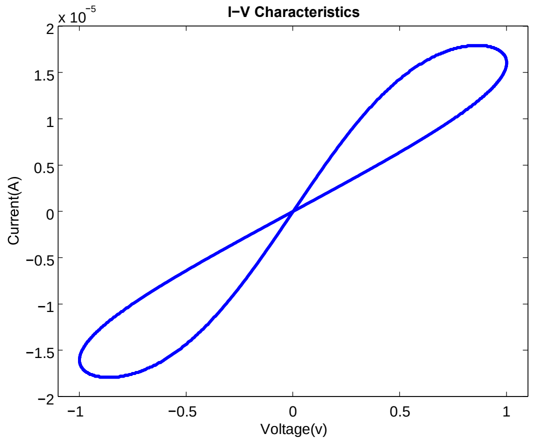


Fig. 1. Memristor Pinched Hysteresis Indicating Non-Linear Behaviour (Frequency of operation: 1 Hz)

For example, the memory cell designed by Sah in 2012 [11] consists of five memristors. Four of these memristors are operating as digital switches while the fifth one is used as an analogue memristor storing a certain value. However, while the memory cell can store positive, negative and zero weight values, there are certain shortcomings. Firstly, the weight of the memory cell can only be increased, not decreased. Secondly, the amount of logic used for the full cell versus the elements required for actual storage is of a proportion 4 to 1, which is far from optimal. Therefore, it would be beneficial if more memristors could be used towards the actual storage, as presented in [12], where a bridge of memristors allows the stored value to be increased and decreased while using 4 memristors towards the actual storage.

Over the years various researchers tried to implement associative memories using e.g. FPGAs, analogue circuits, CMOL and spintronics. In 2011 and 2012 Ang *et al.* [13], [14] proposed the design of an auto-associative memory based on a Spiking Neural Network (SNN) model implemented on an FPGA. While this design showed promising results in terms of speed, its major problem lied in the use of an FPGA, which meant that it was restricted in size/capacity while increasing the FPGA size would result in larger power consumption. Other approaches towards the implementation of associative memories using analogue electronics by [15], and digital hardware by [16] were also made, but these methodologies tend to suffer from several limitations, including: lack of scalability, susceptibility to noise and large power consumption. To reduce the power consumption Lehtonen [17] constructed an associative memory from memristors, but then using a crossbar structure. While these memristive associative memories largely overcame the scalability and power problems, there was yet a new power problem related to sneak path currents. These sneak path currents are due to the flowing of current in unwanted paths of the crossbar structure [18], resulting in extra power consumption. Another crossbar structure, namely a Resistive Crossbar Memory (RCM) based on analogue associative memory modules is proposed by Sharad *et al.* [19]. The Spin Neurons in this design use a crossbar and represent the analogue values using 5-bits. This RCM shows promising results, like being a thousand times more energy-efficient than CMOS. However, this design still suffers from the sneak path current problem, while overall not being very noise tolerant.

III. Memory Cell Designs

Memristors are essentially not that different from resistors with the only difference being that they have a polarity and depending on how the current flows through, their value will either increase or decrease. Other from that, they behave exactly the same when it comes to serial and parallel combinations, as long as one takes into account the increase/decrease in their value, which will consequently affect the overall value.

Building on this principle, one can build different types of memory cells. The first of which is characterised by having the memristors in a respective branch of the bridge, either top or bottom, to be aligned with the same polarity, as shown in Figure 2. The memristor value will change depending on the applied voltage/current and its initial R_{INIT} value, since the latter determines the starting point on the hysteresis curve. In this design, the R_{INIT} values for U2 and U5 are set to ($60k\Omega$), while those for U1 ($35k\Omega$), U3 ($15k\Omega$), U4 ($40k\Omega$) and U6 ($10k\Omega$) are all set differently. Based on the actual differences between these R_{INIT} values, the voltages at the left-most (green) pair of probes will be different from those at the right-most (red) pair of probes, and these values can differ not only in magnitude but also in sign.

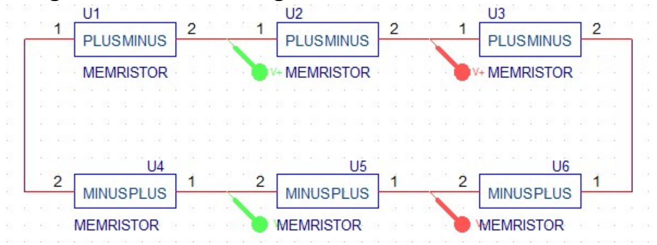


Fig. 2. Memory Cell Design that stores two different voltages with same/different sign

To set each memristor to a specific value, other from R_{INIT}, a specific voltage (5V) will be provided for a certain time-frame (0.2 sec), which changes the memristor values and consequently also the voltages detected at both sets of probes. The results of which can be seen in Figure 4(a), which shows that within the programming time-frame there is only a few points where these voltages are identical, while they normally differ continuously.

From the aspect of fault-tolerance, it can be particularly beneficial if the memory cell provides a value and its inverse, which is of particular importance in an analogue context, because signals can easily get inverted e.g. during amplification. Therefore, as a second memory cell, presented in Figure 3 was developed. Here, the middle memristors have a different polarity from the outer ones, which results in the left (green) and right-most (red) probes providing for the same magnitude but with an opposite sign (see Figure 4(b)). In this setting, the R_{INIT} values are the same for U11, U13, U14 and U16 ($80k\Omega$), while U12 and U15 have a different value ($50k\Omega$). While the similarity in values helps, the used memristors need to have a symmetrical pinched hysteresis loop, which applies, for example, for the Biolek [20] or Berdan models [21].

It is worthy noting that while this paper has restricted itself to changing R_{INIT}, one could additionally modify the values

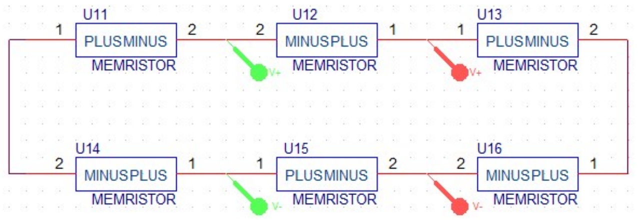


Fig. 3. Second Proposed Memory Cell Design

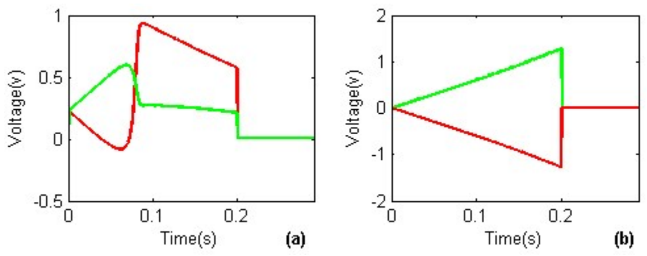


Fig. 4. Results obtained from Memory Cell #1 (a) and Memory Cell #2 (b)

for R_{ON}, R_{OFF}, p and μV to change the behaviour of the individual memristors and consequently the range of values that can be stored within these cells. However, most of these parameters are fixed at the stage of manufacturing, and so it is then only the actual programming that can change the stored value.

IV. Memory Cell Stability

Considering that the memristive memory cell retains a value based on previously applied voltage/current, that also means that as soon as a voltage/current is applied to aid with the reading out, the stored value will be affected. The probe points where the measurements take place measure the difference between two voltages, and these voltages are proportional to the resistive values of the memristors from input (left) to output/ground (right), this voltage will be proportional to the applied input voltage. Therefore, it is best to use the supply voltage to perform the measurements, since this provides for a constant reference value. In case there would be any variations to this supply voltage, then the measured value will remain proportional to the input value, which would be acceptable as long as the circuit continues to work in the same analogue/digital context and does not convert from one to the other. Therefore, pulses are used during the reading process. These pulses have a value of 5V, with an initial delay of 0.3 sec and then a frequency of every 0.2 sec, while in itself only lasting 0.001 sec. The output of applying this pulse pattern for the second memory cell is shown in Figure 5(a), while part (b) indicates that over time the actual measured values still increase, due to a change in memristance. In actual values this means that while at 0.2 sec the value is 1.29V, but at 1.9 sec, this has already become 1.40V.

To prevent the gradual change of memristance, a counter measure has to be introduced, which is achieved through the use of a second pulse that counters the effect of the first. This counter-pulse falls in between the reading pulses and is of opposite polarity, but same amplitude. When applying this principle, the measured results for the second cell then become the ones shown in Figure 6, which indicates that there is no longer a drift from the originally stored values.

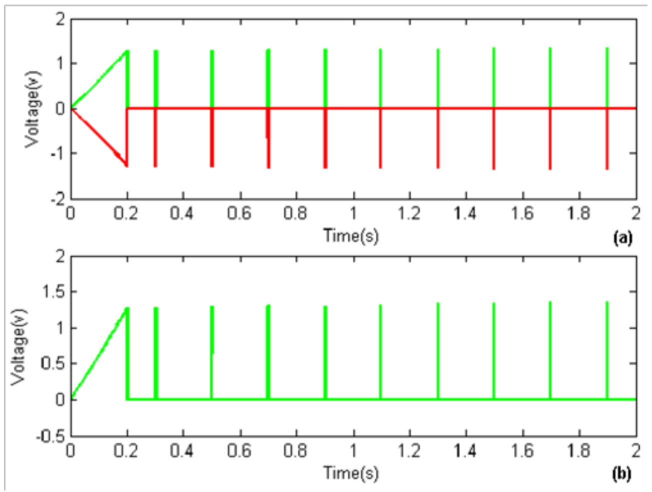


Fig. 5. Readings from both probes (a), or left most probes (b) of second memory cell.

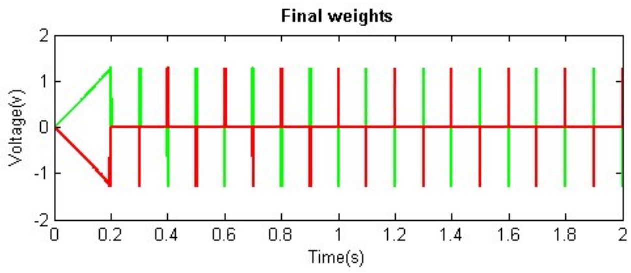


Fig. 6. Responses from reading and correction signals for second memory cell.

However, the pulses used to rectify the gradual drift are actually unwanted in terms of the actual read operation, which means they will need to be filtered out again. Depending on the polarity of the stored signal such filtering can be achieved through the use of a positive/negative high precision rectifier. These rectifiers are also used to filter out the programming part, which are essentially the first 0.2 seconds. For the second memory cell the different sign actually provides a benefit during the rectification, since one can select the pulses with opposite polarity of the normal reading operation and rectify this pulse, rather than the ”original” one. This outcome is shown in Figure 7, where one can note the pulses to be starting at 0.4 sec rather than 0.3 sec.

While most of the above discussion and results mainly focuses on the second memory cell, the same principles also apply to the first type of memory cell. Additionally, these memory cells can also be used for digital storage, which is after all only an abstraction of voltage ranges towards either high/1 or low/0. In such a digital context, the drift of the lower value could actually have a significant impact, since a low value could suddenly become a high value, while the drift of a high value would obviously remain high. However, the true potential of these memory cells lies in their use as analogue memory, of which there are few around, making comparison close to impossible, however they will only provide true value if they show stable operation under none ideal circumstances.

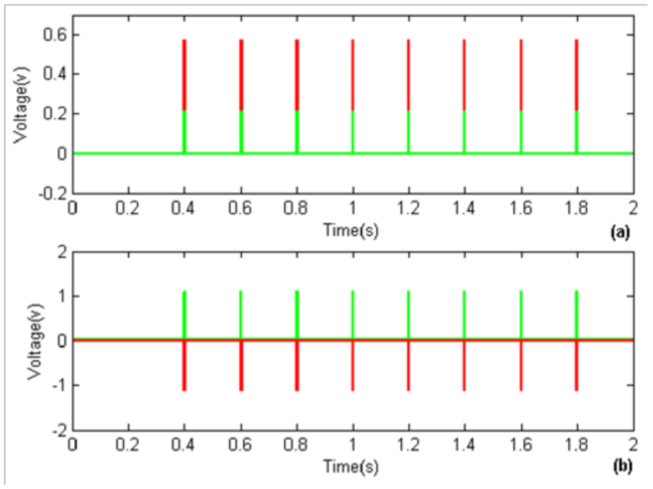


Fig. 7. Weight/Read out value from memory cell type 1 (a) and type 2 (b)

V. Memory Cell Reliability in Noisy Conditions

To test the reliability of the memristor cells in none ideal conditions, noise was introduced to the system. This noise was introduced in two separate steps. The first was during the actual programming of the memory, while the second step introduces it during the actual reading of the memory content. It goes without saying that the impact of the former is likely to be more significant. The first set of noise used was a Gaussian white noise signal, as shown in Figure 8, which was introduced during the programming. Then the values were read back, leading to the results shown in Figure 9 for the second memory cell. The output no longer shows the same height for all outputs, which is obviously due to the introduced noise. This is further confirmed by the fact that in between the reading pulses there is noise shown in the output. However, when comparing the output, with the introduced noise signal, then one can notice a significant suppression of the noise signal, which is largely due to the capacitive effect of the memristors [20]. When comparing the magnitudes of the pulses obtained under the influence of noise with the original ones in more detail, then one can notice that the differences are not large. It therefore seems that the noise affects all memory cells through shifting the circuit’s behaviour in line with the applied noise, and so the overall effect of the Gaussian noise can be considered as quite minimal. In case of using these memory cells for digital storage, the effect can even be ignored completely.

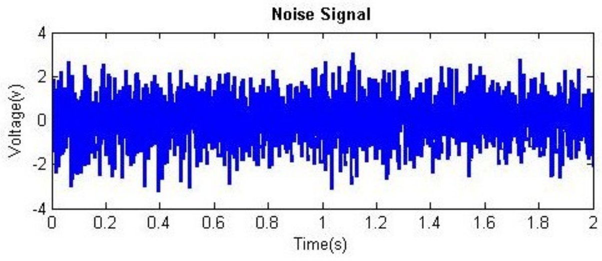


Fig. 8. Gaussian white noise used for reliability tests

Testing the first memory cell under the influence of noise provides the signals shown in Figure 10. Considering that the first memory cell is programmed with much smaller weights,

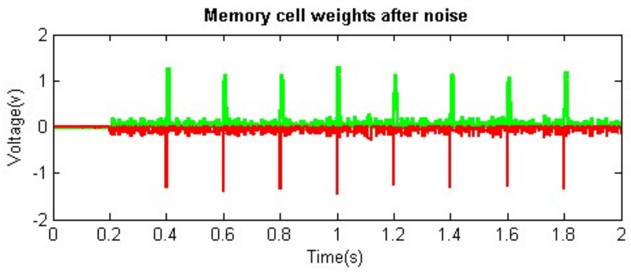


Fig. 9. Stored value weights for second memory cell when adding Gaussian noise during the programming

the results of this test provide for particularly interesting results due to the storage of small values. After all, the original weights were only 200mV and 580mV respectively for the two probe points. As can be seen from the results, the original values are still very closely achieved, even under noise conditions, especially when one takes into account that the noise amplitude goes up to about 2V. Considering that the memristor only shows its memristive characteristics in a certain frequency range one would assume that noise outside this range does not affect the device as such, which was confirmed during separate tests. This is additionally helped by the fact that memristors show their memristive effect at low frequencies, and therefore the high frequencies, which generally form the majority of the noise signals, become completely insignificant to the circuits operation.

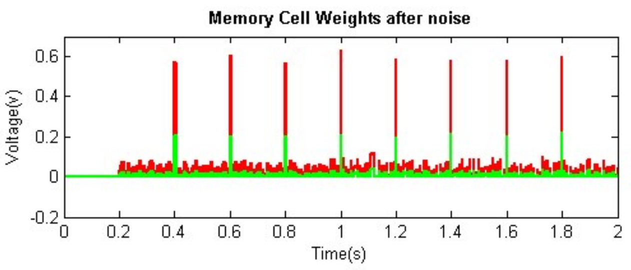


Fig. 10. Stored value weights for first memory cell when adding Gaussian noise during the programming

VI. Associative Memory Design

After introducing the memory cells, the memory cell design shown in Figure 3 was used to construct the associative memory. The designed associative memory has the ability to store several vectors, where the naming convention used is: A*B, with *A* being the number of vectors in the memory and *B* the number of cells in each vector. It is assumed that all input vectors are positive and therefore this paper only deals with the positive part of the memory cells. However, the same designs can be used when the negative part of the memory cell is used.

In order for these memristive cells to become part of an associative memory, two main tasks need to be added: 1) comparing the inputs with the stored values and calculating the Euclidean distance between them; and 2) identifying the smallest Euclidean distance among all values, as in order to identify which stored vector matches most closely to the input. Eventually, when the closest match is determined, then the corresponding vector should be provided as output of the circuit.

A. Determining Euclidean Distance

In order to determine the Euclidean distance the input and stored values are provided to a unity gain differential amplifier. The output from this differential amplifier then forms the input to an absolute value circuit shown in Figure 11. Since the differences could be positive as well as negative, their absolute value has to be determined before the differences of each element within the vector is summed to create values that represent the total difference between each stored vector and the input vector.

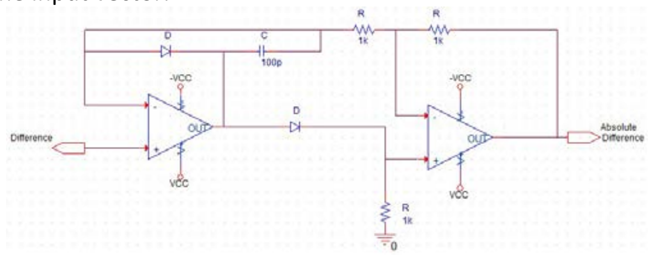


Fig. 11. Absolute Value Circuit

B. Finding the Smallest Difference

Once the differences have been identified, the smallest outcome has to be identified. This depends on the number of vectors against which comparisons have been made. For example, in the case of a 2*2 associative memory, the first phase will only output two differences, therefore the second phase will use one comparator to check which is the smallest of both, and this smallest difference will then be used in the retrieving phase to drive a 2*1*Multiplexer* (MUX) and pass the corresponding vector to the output.

If the auto-associatve memory has a size of 4*2 then the output from the first phase will be 4 differences. In order to then find the smallest difference efficiently, a multi-stage hierarchy comparator has to be used. In this specific case, this multi-stage comparator tree consists of two comparator stages. Obviously for larger vectors, a larger comparator tree would be required.

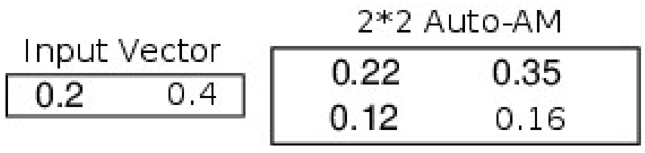
The output of the comparators is digital, namely 0 for the smallest value and 1 for the largest, which results in them being easy to directly drive a MUX. However, when a multistage comparator is needed, then this also means that the actual analogue values are lost after the first stage and so they need to be passed through a MUX that is driven by the output of this comparator to feed into the next stage.

VII. Results

A. Noise-free Associative Memories

Two different circuits, namely a 2*2 and a 4*2 AutoAssociate Memory (AM) where tested with a 2 element input vector. Figure 12 shows the stored values of the 2*2 AutoAM and the input vector, while Figure 13 shows the results obtained, which clearly indicates that the closest match is the first vector (0.22V, 0.35V).

In relation to the 4*2 Auto-AM, the values are shown in Figure 14, and the retrieved results in Figure 15, which indicates that the closest matching stored vector is (0.12V, 0.16V),



Input Vector

0.2	0.4

2*2 Auto-AM

0.22	0.35
0.12	0.16

Fig. 12. 2*2 Auto-AM Stored Values and Input Vector

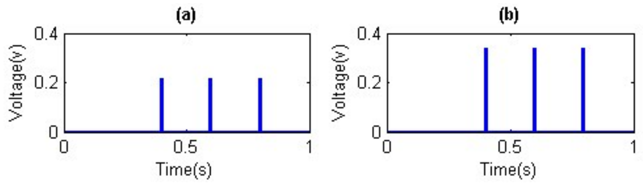
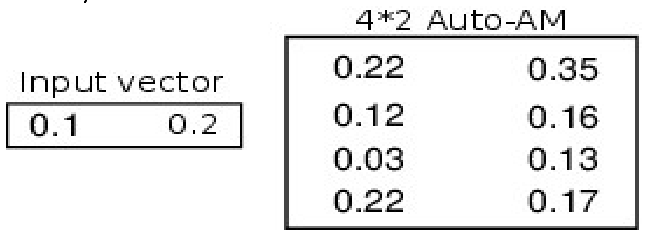


Fig. 13. Retrieved Vector Elements from the 2*2 Auto-AM (a) First Vector Element, (b) Second Vector Element

and so while the hierarchical comparator would increase the comparison time with the number of levels in the hierarchy, it in no way affects the actual identification of the closest match.



Input vector

0.1	0.2

4*2 Auto-AM

0.22	0.35
0.12	0.16
0.03	0.13
0.22	0.17

Fig. 14. 4*2 Auto-AM Stored Values and Input Vector

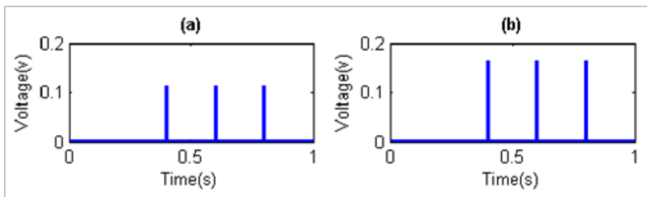
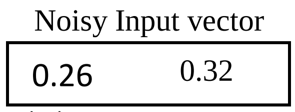


Fig. 15. Retrieved Vector Elements from the 4*2 Auto-AM (a) First Vector Element, (b) Second Vector Element

B. Noisy Associative Memory

While it is relatively straight forward to identify similarity in perfect conditions, this changes quite substantially when noise is being introduced into the system, especially considering that this is an analogue system and therefore does not benefit from the digital abstract that often tends to filter out the effect of noise. To test the robustness against noise, the noise signal shown in Figure 8 was introduced to the 2*2 Auto-AM in two ways: firstly during the actual memory cell programming and secondly using a noisy input vector.

The memory cells themselves are quite noise tolerant, and so the introduction of noise during the actual memory cell programming does not impact the stored values much because of the fact that most of the noise gets filtered out before the data is stored. This has to do with the capacitive effect of the memristor, which inherently leads to a filter being built into the system itself. Consequently, the stored vectors after introducing the noise signal to the programming are equal to those of Figure 12. Secondly, when using a noise input, as shown in Figure 16, then the results are those shown in Figure 17. In this case the results are not consistent in terms of height, which is related to the changes in noise at the input affecting the actual comparison, but on average one can still see that the they will allow the identification of the closest match. These results also indicate that the noise level in the output was suppressed, which is again due to the capacitance within the system that serve as noise filters.



Noisy Input vector

0.26	0.32

Fig. 16. Average of the noisy input vector

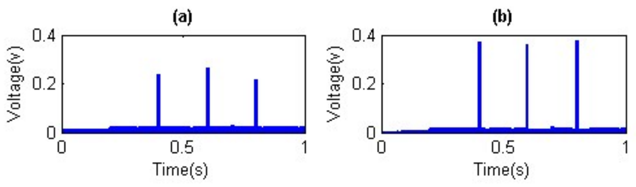


Fig. 17. Retrieved elements after introducing noise (a) First element, (b) Second element

VIII. Conclusions

The two memory cell designs presented, are constructed from passive elements and will therefore require minimal power. They allow for the storage of two different values, with either equal or different signs and values. The accuracy and reliability of the stored values is high, which allows their use for analogue storage, while they could also be used for digital storage. The principle used for multi value storage could easily be expanded towards storing even more values, while this may come at a trade-off in terms of storage flexibility.

The results show that these memory cells are highly reliable and the stored values scale with supply voltage, therefore allowing for a scaled design, as well as circuitry to operate on a concept of value comparison, rather than exact values.

These results helped in the design of the analogue associative memories. The two analogue memristive auto-associative memories presented in this paper are only a small example of the abilities of these type of memories, which largely eliminate static power consumption due to the use of passive components, while also being scalable and noise tolerant and at the same time allowing for high information density due to their analogue information storage.

Using an analogue associative memory provides further advantages due to the fact that it no longer needs ADCs and DACs and can therefore directly interact with the world around us without any loss of accuracy. Up to now, analogue storage was a challenge, but using memristors allows for reasonably convenient analogue storage, which is also noise tolerant due to the built in capacitive filter.

In the current designs, most comparisons are still performed using high precision active circuits, which will influence the required power. It is our aim to replace these circuits by less high-specification, preferably passive circuits without loosing out on the detection accuracy.

References

[1] H. Silva and S. Tiwari, "A nanoscale memory and transistor using backside trapping," *IEEE Transactions on Nanotechnology*, vol. 3, no. 2, pp. 264–269, 2004.

[2] E. N. Shauly, "CMOS Leakage and Power Reduction in Transistors and Circuits: Process and Layout Considerations," *Journal of Low Power Electronics and Applications*, vol. 2, no. 1, pp. 1–29, 2012.

[3] A. Ghofrani, M. A. Lastras-Montano, and K. T. Cheng, "Towards data reliable crossbar-based memristive memories," in *Proceedings International Test Conference*, 2013.

[4] B. Leiner, V. Lorena, T. Cesar, and M. Lorenzo, “Hardware Architecture for FPGA Implementation of a Neural Network and Its Application in Images Processing,” *2008 Electronics, Robotics and Automotive Mechanics Conference (CERMA ’08)*, pp. 405–410, 2008.

[5] S. M. Jalaleddine, “Associative Memories and Processors: The Exact Match Paradigm,” *Journal of King Saud University - Computer and Information Sciences*, vol. 11, no. All 141911999, pp. 45–67, 1999. [Online]. Available: http://linkinghub.elsevier.com/retrieve/pii/S1319157899800032

[6] M. A. Abedin, Y. Tanaka, A. Ahmadi, T. Koide, and H. J. Mattausch, “Fully parallel associative memory architecture with mixed digitalanalog match circuit for nearest Euclidean distance search,” *IEEE AsiaPacific Conference on Circuits and Systems, Proceedings, APCCAS*, vol. 00, pp. 1309–1312, 2006.

[7] R. Chakraborty, “Associative Memory Soft Computing Soft Computing,” 2010.

[8] L. Chua, “Memristor- The Missing Circuit Element,” *IEEE Transactions on Circuit Theory*, vol. CT-18, no. 5, 1971.

[9] D. B. Strukov, G. S. Snider, D. R. Stewart, and R. S. Williams, “The missing memristor found.” *Nature*, vol. 453, no. 7191, pp. 80–83, 2008.

[10] S. Gazabare, R. J. Pieper, and W. Wondmagegn, “Observations on frequency sensitivity of memristors,” *Proceedings of the Annual Southeastern Symposium on System Theory*, pp. 45–50, 2012.

[11] M. Sah, C. Yang, H. Kim, and L. Chua, “Memristor circuit for artificial synaptic weighting of pulse inputs,” in *Circuits and Systems (ISCAS), 2012 IEEE International Symposium on*, 2012, pp. 1604–1607.

[12] S. P. Adhikari, C. Yang, H. Kim, and L. Chua, “Memristor bridge synapse-based neural network and its learning,” *IEEE Transactions on Neural Networks and Learning Systems*, vol. 23, no. 9, pp. 1426–1435, 2012.

[13] C. H. Ang, C. Jin, P. H. Leong, and A. Van Schaik, “Spiking neural network-based auto-associative memory using FPGA interconnect delays,” in *2011 International Conference on Field-Programmable Technology, FPT 2011*, 2011.

[14] C. Ang, A. McEwan, A. van Schaik, C. Jin, and P. H. Leong, “FPGA implementation of biologically-inspired auto-associative memory,” *Electronics Letters*, vol. 48, p. 148, 2012.

[15] N. Lewis and S. Renaud, “Spiking neural networks ’in silico’: from single neurons to large scale networks,” in *SSD’07: Fourth Conference on Systems, Signals and Devices, Hammamet (Tunisia)*, 2007.

[16] R. J., J. B. Elizabeth, and J. J. Abbas, “Real-Time Interaction Between a Neuromorphic Electronic Circuit and the Spinal Cord,” *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 9, no. 3, pp. 0–2, 2001.

[17] E. Lehtonen, J. H. Poikonen, M. Laiho, and P. Kanerva, “Large-Scale Memristive Associative Memories,” *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 2013.

[18] M. A. Zidan, H. A. Fahmy, M. M. Hussain, and K. N. Salama, “Memristor-based memory: The sneak paths problem and solutions,” *Microelectronics Journal*, vol. 44, no. 2, pp. 176–183, 2012.

[19] S. Mrigank, F. Deliang, and R. Kaushik, “Ultra Low Power Associative Computing with Spin Neurons and Resistive Crossbar Memory,” in *In Proceedings of the 50th Annual Design Automation Conference, ACM*, vol. 1, 2013.

[20] Z. Biolek, D. Biolek, and V. Biolkova, “SPICE model of memristor with nonlinear dopant drift,” *Radioengineering*, vol. 18, no. 2, pp. 210–214, 2009.

[21] R. Berdan, C. Lim, A. Khiat, C. Papavassiliou, and T. Prodromakis, “A memristor SPICE model accounting for volatile characteristics of practical ReRAM,” *IEEE Electron Device Letters*, vol. 35, no. 1, pp. 135–137, 2014.