RRAM-Based Non-Von Neumann Computing Literature Review

Trenton Rochelle
Memory-Centric Computing for Processing Neural Networks

I. Introduction

Von Nuemann (VM) based computing has improved considerably throughout the years, but certain computations are limited by their frequent memory accesses which introduce large latency and energy consumption. Many of these tasks are within the artificial intelligence domain where parallel processing can improve the performance significantly, especially as the trend of larger and deeper networks continue. This trend demands increased compute power, and thus energy consumption. One solution to this problem is to merge the gap between memory and processor through hardware based vector-matrix multiplication in memory to aid the development of artificial neural networks (ANN). The most common metrics to evaluate the performance of these circuits is area, energy consumption, speed, bit precision, and accuracy (compared to a software-based equivalent).

Memristors have garnered attention within this domain due to their resistive switching qualities that are based on the voltage/current applied to them, which can then be used to represent a neural network (NN) weight. Scientists have achieved memristors that can hold a high resistance state and a low resistance state which can be representative of a single bit. These memristors can be paired with other basic circuitry elements such as transistors and capacitors and formed into a grid to achieve a dynamic memory is called Resistive RAM (RRAM) where multiply-and-accumulate (MAC) operations can be performed without the need of a central processor. This RRAM non-VM computation in memory (NVCIM) has the potential to achieve high energy and area efficiency but will only become competitive after significant neural network performance increase and area reduction compared to CMOS alternatives.

A. Memristors

This resistive switching hardware is achieved with memristors which Serrano-Gotarredona et al. [1] define as a "two-terminal electronic device which is similar to a resistor, but whose resistance changes dynamically as the device is being used."1 The most common memristors cited are those that have two states, high resistance state (HRS) and a low resistance state (LRS), both of which are constant between a specified voltage range until resistive switching. This paper will assume bipolar switching where the LRS can be triggered by increasing the voltage potential across the device until it reaches a threshold voltage termed V_{set}, while the HRS can be toggled once the voltage reaches a negative threshold termed V_{reset}. This can be shown in Fig. 1 below.

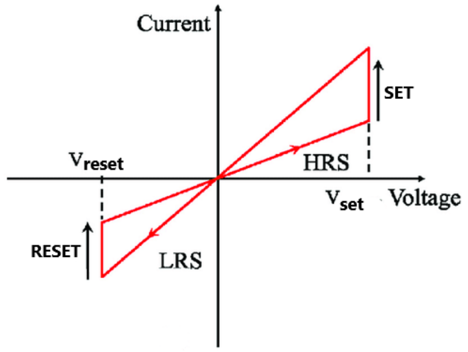

Fig. 1: Bipolar switching of single level memristor

B. RRAM

As each of these memristors effectively has two states, they can be represented as a single bit. Multiple memristors can then be used in conjunction to increase the number of bits to achieve a specific weight level. These memristors can then be arranged into cells along with transistors in a crossbar manner to create RRAM where individual cells can be interacted with through wordlines (WL) and bitlines (BL). In general, the RRAM crossbar structure is composed of one or more horizontal WLs and one or more vertical BLs. The BLs are used to "select" a specific column for which the column multiplication is accumulated, while the WLs are used to specify the rows and multiply the input pulse with the cell weight.

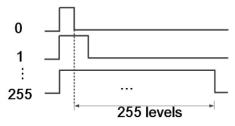

Fig. 2: Pulse width based input

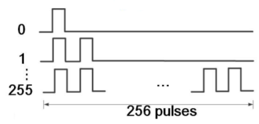

Fig. 3: Multi-pulse based input

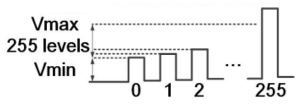

Fig. 4: Amplitude dependent input pulses

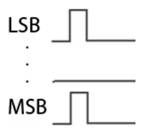

Fig. 5: Parallel input pulses

Input coding follows the same general guideline of a voltage pulse being applied, but it can vary in its structure. There are two main patterns, timing-dependent and amplitude dependent. These can either be fed in on a single line or in parallel depending on the architecture of the neuron and synapse. Timing-dependent inputs are either pulse width based, where the pulse width increases with the input value, or multi-pulse based, where the number of pulses increases with the input value. Amplitude dependent pulses increase voltage with the input value. These variations are shown in Figures 1-4 and have been adapted from Xi et al. [3].

II. Architectures

A. 1T1R Cell

One of the easiest to understand RRAM structures consists of one transistor and one memristor (1T1R) where the weights consist of a binary value. An example of the synaptic network mapping to the 1T1R RRAM structure from Yao et al. [4] is shown below. This structure maps a grayscale image of a face into the input using a multi-pulse scheme. Each source line (SL below) is a current accumulation of each input (V below) multiplied by its respective synapse/weight (W below) into the post-layer neuron. The figure also shows how each grayscale face is converted into a value from 0-255 which determines the number of input pulses.

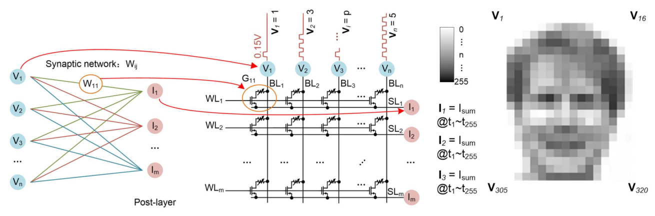

Fig. 6: 1T1R RRAM MAC structure [4]

B. Differential Lines

Another problem with memristors is their strictly positive conductance values. Bankman et al. [5] solve this by using a differential system where each weight has a negative copy of it (a multilevel memristor is simulated with where weight ϵ [-2,2]). These weight values feed into separate accumulation lines which are subtracted in the form of $G_m - G_p$, where G_m and G_p are the conductance of the differential weights, and later divided by the normalizing constant $2 * G_u$ where G_u is the step size. A figure of this is shown below where the inputs are fed in parallel and each memristor per cell contains the same weight.

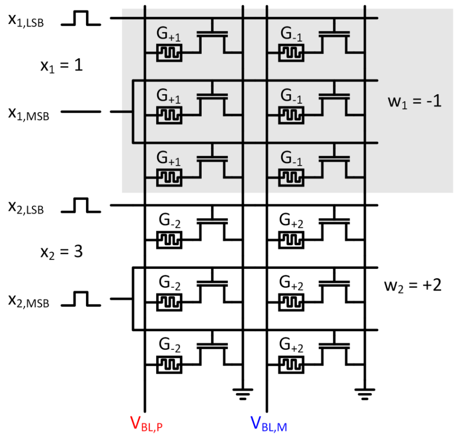

Fig. 7: Weight-activation dot product worldline group [5]

C. 4R Cell

The differential duplicated columns can be removed by introducing a map that converts purely positive conductances to a range of positive, negative, and zero weight. By introducing a cell that has 4 (or more) parallel memristors, we achieve a conductance that is determined by the amount of "on" and "off" memristors.

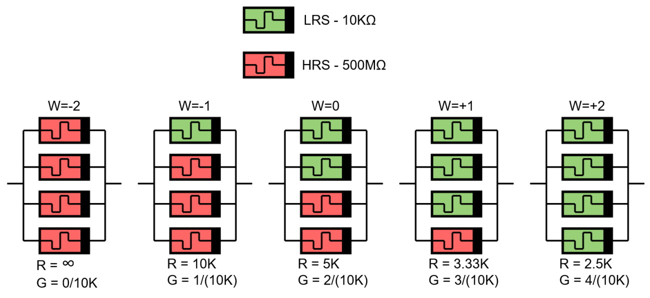

Fig. 8: Weight-conductance mapping of parallel memristors

The total resistance of each cell can be calculated as the parallel resistance of the four memristors with n memristors being turned on:

$$R_{cell} = 1/(n/LRS + (4-n)/HRS)$$

Where LRS and HRS are the resistance values of the low and high resistance states, respectively. Because the HRS is orders of magnitude larger than the LRS and contributes negligibly to the overall resistance, the resistance and conductance of the cell can be simplified to the following:

$$R_{cell} = Infinite, n = 0$$

$$R_{cell} = LRS/n, n > 0$$

$$G_{cell} = n/LRS$$

The cells are placed in a crossbar structure where the input along the wordline is an amplitude-based single pulse. The mapping of X to the input follows the following:

$$X \epsilon [0 - X_{max}]$$

$$memristor\, operating\, positive\, voltage\, range \,\epsilon [0, 1.5V]$$

$$X_{normalized}\, \epsilon [0 - 1] = X[:]/X_{max}$$

$$applied\, wordline\, voltage\, \epsilon [0, 1.5V] = V = 1.5 * X_{normalized}$$

With the bitlines grounded, the following currents can be calculated:

$$I_{cell} = V * G_{cell} = V * (w + 2)/LRS$$

$$I_{bitline} = \sum_{i=1}^{n} I_{cell,i}$$

Due to the shifting of negative weights to purely positive conductance values, differing input and weights values that lead to the same outputs will accumulate different current that must be adjusted. As shown below, BL0,BL1,BL2 are 120,170,160 uA respectively, while BL3,BL4,BL5 are 140,190,180 uA respectively.

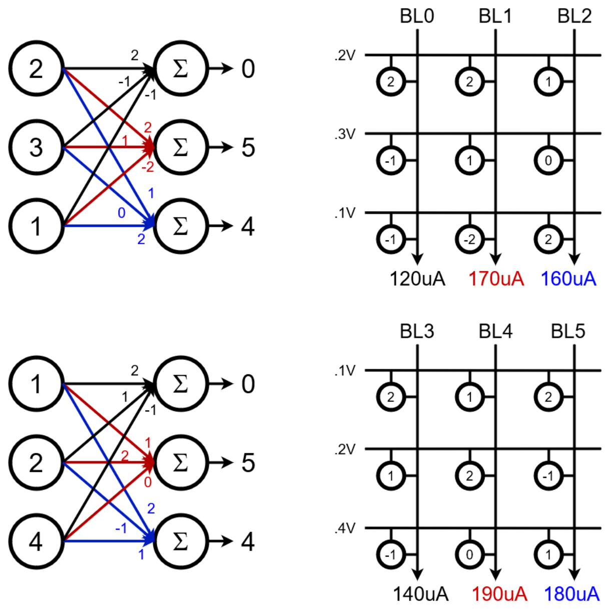

Fig. 9: Equivalent outputs with differing accumulations

While the bitlines between the two examples do not match up, the differences between the bitlines in their respective examples are the same. The current on each bitline is a result of the applied voltages so a constant reference current cannot be subtracted, but the reference current can be calculated and then subtracted from each bitline. This reference current is equal to the current accumulated for an output of 0 and can be forced by multiplying any input by the zero weight. This reference current can either be calculated outside of the crossbar MAC or inside with an extra column of preset memristor values. The adjusted bitlines for the two examples in turn switch to 0uA,50uA,40uA which calculate to a current step of 10uA per output value step.

$$I_{cell}, w = 2 = V * G_{cell}, w = 2 = V * 2/LRS$$

$$I_{reference} = \sum_{i=1}^{n} (2 * V_i/LRS)$$

$$I_{bitline,adjusted} = I_{bitline} - _{reference}$$

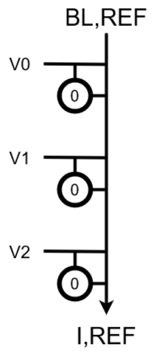

Fig. 10: Reference bitline to be subtracted

Because current subtraction is complicated, an easier method is to convert the currents into voltages and then subtract the reference voltage. To convert the current into voltages, a current sense amplifier is used with a preset or adjustable gain to amplify the voltage drop across a very small (to prevent a current calculation error) shunt resistor. Once we have the reference voltage, it can then be subtracted through the use of an operational amplifier with a gain of 1.

Like the previously mentioned reference current, the reference voltage can be calculated in analog within the crossbar structure or digitally outside the crossbar and then converted into analog through a DAC. These can two be seen at the “Current To Voltage” label in Figure 11 below, with inside (left) and outside (right) crossbar implementations. If the reference voltage is performed outside the crossbar structure and ahead of time of the MAC operation within memory, the DAC could be able to feed into the current-sense amplifier as a voltage bias, thus subtracting the voltage and skipping the second stage of operational amplifiers, but this circuit is neither realized nor shown in this paper and merely is a suggestion to possibly improve power and latency. The calculations for current to voltage and subsequent subtraction are as follows:

$$V(I_{reference}) = I_{reference}) * R_{shunt}$$

$$V_{reference} = V(I_{reference}) * Gain$$

$$V_{bitline} = V(I_{bitline}) * Gain$$

$$V_{bitline,adjusted} = V_{bitline} - V_{reference}$$

$$V_{bitline,adjusted} = V(I_{bitline,adjusted}) * Gain$$

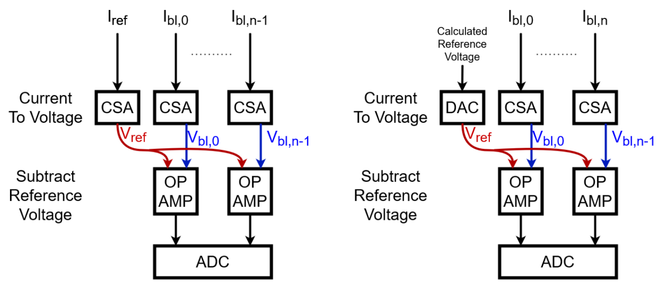

Fig. 11: Amp-to-Volt conversion and reference subtraction

The adjusted bitline voltages can then be fed into an ADC layer that encompasses both the bit-digitization and voltage comparator function that scales with a precomputed constant. We can directly calculate Y = X * W by combining the previous equations:

$$V_{bitline,adjusted} = V(\sum_i (V_i * (w_i + 2)/LRS) - \sum_i (2 * V_i/LRS)) * Gain$$

$$V_{bitline,adjusted} = V(\sum_i (V_i * w_i/LRS)) * Gain$$

$$V_i = 1.5 * X_i/X_{max}$$

$$V_{bitline,adjusted} = V(i(X_i * w_i/LRS)) * Gain * 1.5/X_{max}$$

$$V_{bitline,adjusted} = Y * ((1.5 * Gain)/(X_{max} * LRS))$$

$$Y = V_{bitline,adjusted} * constant$$

D. *4R1T Cell*

While the previous cell allows for computationally valid MAC operations, all memristor values are pre-set and constantly connected to the bit lines. With all memory cells having an effect on the bitlines, the full 2D array of the RRAM-CIM should be used to avoid sneak currents or charging unused, possibly floating wordlines that could contribute to RC delay. A starting idea was to introduce a connecting transistor between the four parallel memristors and the bitline to allow for dynamic disconnecting of cells. This idea was explored through simulations of bitline current, but ultimately the accumulated current cannot be converted to the correct Y output due to the interfering nature of the relatively large value of the transistor on resistance, R_{on}.

Through simulations of the previous weights and inputs of the 3x3 RRAM grid, once R_{on} reaches 10% of the LRS, each step in Y completely breaks down and Y cannot be recovered. This is devastating for this structure as R_{on} is generally in the range of many kΩ for 130nm.

$$R_{cell} = Infinite, n = 0$$

$$R_{cell} = Ron + LRS/n, n > 0$$

$$G_{cell} = 1/(Ron + LRS/n), n > 0$$

$$I_{cell} = V * G_{cell} = V/(R_{on} + LRS/(w + 2))$$

$$I_{bitline} = \sum_{i=0}^{n} I_{cell,i}$$

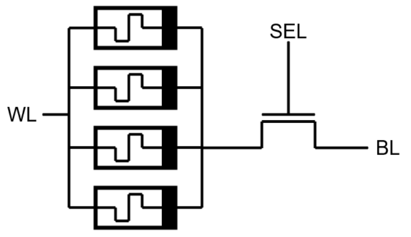

Fig. 12: 4R1T Cell with a select transistor

E. *4R4T Cell*

To adjust for a resistance that is proportional with the number of memristors turned on, one transistor is attached to each memristor as shown in Figure 13.

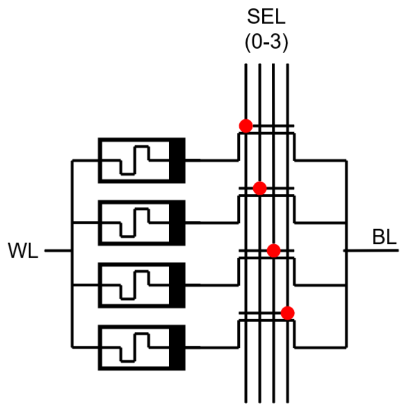

Fig. 13: 4R4T Cell with 4 independent select transistors

The select line(s) run vertically connecting all cells within a column, where the red circle attached to the line and gate on the transistor indicate a connection. This select process can be performed with a single line where all transistors within a column are connected and turned on simultaneously, or with four lines where the ith transistor of each cell is connected together. While the four line select introduces a larger area, it allows the ability for selective connection of the ith memristor and can facilitate the ability to dynamically change the value of the memristor, compared to the 4R cell which requires a preset array. The four line select allows for differing voltage drops across each parallel memristor within the cell compared to the single line select where the voltage drop across the parallel memristors is identical, which prohibits achieving weights -1,0, and +1, as all of the parallel memristors can only simultaneously set or reset at a given time, only allowing the weights -2 and +2.

An algorithm for manipulating the four line select, as well as the wordline and bitline, to achieve a dynamic training of the parallel memristors is not achieved yet as this is a somewhat novel approach and is outside the scope of this paper, but a column would most likely require 4 steps to update the weights as the ith memristor in each cell would need to be isolated from the other 3.

While the training problem is potentially solved with this structure given the correct algorithm mentioned above, only specific columns can be disconnected from the RRAM. This can be achieved by simply not charging the select lines of specific columns. To disconnect whole wordlines within the RRAM as well, a 4R8T structure would be necessary, as shown below. The drawback to the 4R8T cell is the increase in size, enable line and transistor charging, as well as the fact that if the worldline enable of row i is disabled, all i+1 rows will also be disables as the select line will become floating, thus turning off all of the transistors.

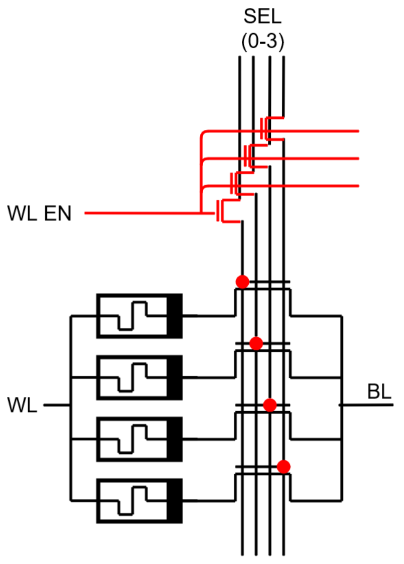

Fig. 14: Wordline enabled 4R8T cell and 4 select transistors

To validate the correctness of the 4R4T cell during the MAC operation, we will follow the same math as before. The assumed operation of the selection lines are that they are charged to a voltage well above the threshold value such that variances between the transistor on-resistances are negligible. The only metric that changes is cell current which is shown below.

$$I_{cell} = V * G_{cell} = V * (w + 2)/(LRS + Ron)$$

Once this cell current is accounted for, the same math can be followed from the 4R cell, and this leads to a slight change in the constant scalar that is multiplied by Y.

$$V_{bitline,adjusted} = V(\sum_i (X_i * w_i/(LRS + R_{on}))) \\ *Gain * 1.5/X_{max}$$

$$V_{bitline,adjusted} = Y * ((1.5 * Gain)/(X_{max} * (LRS + R_{on})))$$

$$Y = V_{bitline,adjusted} * constant$$

Because an hspice simulation was not able to be achieved in time, a comprehensive analysis of power was not performed for this paper, but a calculation of peak power draw flowing through the cells to ground can be estimated. With the previous assumption that the operating voltage range of the memristors is 0V-1.5V, testing two different LRS resistances of 10kΩ and 100kΩ for each memristor, and a transistor on resistance of 10kΩ, the worst case scenario for an RRAM of 1024x1024 rows & columns is each memristor weight having a value of 2, and the applied voltage to each row is 1.5V.

$$R_{cell} = R_{on} + LRS/4 = 12.5k\Omega and 35k\Omega$$

$$P = V2/R$$

$$P = (1.52 * 1024 * 1024)/R_{cell} = 188.74W \,\&\, 67.4W$$

While the peak power consumption is 188.74W and 67.4W for a LRS of 10kΩ and 100kΩ respectively, this current draw time through the RRAM is incredibly short as the input is provided as a pulse that is as short as possible and also provides a stable ADC read. Based on the large power consumed during the MAC operation, a timing analysis should be completed to minimize input pulse length.. Outside of active computation, the RRAM array should consume no power as the memristors are nonvolatile and no CMOS is needed within the cells. Additionally, this worst case scenario is likely to never occur as the weights and inputs will vary significantly. A potential look for an “average” MAC is half of the max input voltage, .75V, and the zero-weight memristor, LRS/2.

$$R_{cell} = R_{on} + LRS/2 = 15k\Omega \,\&\, 60k\Omega$$

$$P = (1.52 * 1024 * 1024)/R_{cell} = 157.28W \,\&\, 39.3W$$

It appears that a larger LRS for the 4R4T structure leads to lower power draw during the MAC operation, but it should be noted that the current of each bitline runs through a shunt resistor which becomes amplified through the current sense amplifier. As the voltage drop across Rshuntdecreases with low bitline current accumulation, the more sensitive the current sense amplifier becomes to noise. Rshunt should be calibrated to be as large as possible to increase this drop, but Rshunt must stay as small as possible to not affect the accumulation of the bitline current. Another option to reduce power consumption that maintains the same considerations as before is to reduce the input voltage which could have a more significant effect as power scales exponentially with voltage.

F. Layout Limitations

Layout limitations exist due to the current design of a crossbar architecture. The space between the crossbars in the RRAM domain are conventionally filled with memristors and potentially a diode thin film that prevents current "sneak paths". With the 4R4T/4R8T design, this area would need to be sufficiently large enough, as well as the manufacturing capability, to place each memristor in series with a transistor. Additionally, current accumulation lines need to be evaluated for their impact on current as thin bitlines could contribute significant resistance.

G. Conclusion

Implementing a programmable, trainable, low-energy, high speed, high density RRAM includes many research challenges that span many domains such as material science, layout design, architecture, and algorithms. This paper shows how RRAM can store positive and negative weights without a differential weight storage, and that scale linearly with the number of memristors by using the 4R cell, with the ability to enable/disable entire bitlines (4R4T) and wordslines (4R8T), and the potential to isolate individual weights in a cell (4R8T), at the expense of increased size. Much of the workarounds present in this paper could be ameliorated with a true multi-level memristor, but none were found at the time of writing.

REFERENCES

[1] T. Serrano-Gotarredona, T. Masquelier, T. Prodromakis, G. Indiveri, and B. Linares-Barranco, "STDP and STDP variations with memristors for spiking neuromorphic learning systems," Front. Neurosci 7, 2 (2013).

[2] C. Sung, H. Hwang, and I. K. Yoo, "Perspective: A Review on Memristive Hardware for Neuromorphic Computation," J. Applied Physics, Vol. 124, No. 15, 2018, p. 151903.

[3] Xi, Yue, et al. "In-Memory Learning with Analog Resistive Switching Memory: A Review and Perspective." Proceedings of the IEEE, vol. 109, no. 1, 2021, pp. 14–42.

[4] P. Yao et al., "Face classification using electronic synapses," Nature Commun., vol. 8, no. 1, p. 15199, Aug. 2017.

[5] Bankman, D., et al. "RRAM-Based In-Memory Computing for Embedded Deep Neural Networks." 2019 53rd Asilomar Conference on Signals, Systems, and Computers, 2019.

[6] Ganesh, Pooja & Krishna, s and V, Ravi. (2019). Multi-level memristor memory: Design and performance analysis. International Journal of Innovative Technology and Exploring Engineering. 8. 723-729.

[7] Lee, Jaeheum Eshraghian, Jason Cho, Kyoung-Rok Eshraghian, Kamran. (2019). Adaptive Precision CNN Accelerator Using Radix-X Parallel Connected Memristor Crossbars.