8T SRAM Cell as a Multi-bit Dot Product Engine for Beyond von-Neumann Computing

Article *in* IEEE Transactions on Very Large Scale Integration (VLSI) Systems · February 2018

DOI: 10.1109/TVLSI.2019.2929245

CITATIONS
43

READS
646

4 authors, including:

Indranil Chakraborty
Purdue University
48 PUBLICATIONS **518** CITATIONS
SEE PROFILE

Amogh Agrawal
Purdue University
32 PUBLICATIONS **237** CITATIONS
SEE PROFILE

Kaushik Roy
Purdue University
1,241 PUBLICATIONS **37,836** CITATIONS
SEE PROFILE

8T SRAM Cell as a Multi-bit Dot Product Engine for Beyond von-Neumann Computing

Akhilesh Jaiswal*, Indranil Chakraborty*, Amogh Agrawal, Kaushik Roy, *Fellow, IEEE*
*(Equal Contributor)

***Abstract*—Large scale digital computing almost exclusively relies on the von-Neumann architecture which comprises of separate units for storage and computations. The energy expensive transfer of data from the memory units to the computing cores results in the well-known von-Neumann bottleneck. Various approaches aimed towards bypassing the von-Neumann bottleneck are being extensively explored in the literature. These include *in-memory* computing based on CMOS and beyond CMOS technologies, wherein by making modifications to the memory array, vector computations can be carried out as close to the memory units as possible. Interestingly, in-memory techniques based on CMOS technology are of special importance due to the ubiquitous presence of field-effect transistors and the resultant ease of large scale manufacturing and commercialization. On the other hand, perhaps the most important computation required for applications like machine-learning *etc.* comprises of the dot product operation. Emerging non-volatile memristive technologies have been shown to be very efficient in computing analog dot products in an *in-situ* fashion. The memristive analog computation of the dot product results in much faster operation as opposed to digital vector in-memory bit-wise Boolean computations. However, challenges with respect to large scale manufacturing coupled with the limited endurance of memristors have hindered rapid commercialization of memristive based computing solutions. In this work, we show that the standard 8 transistor (8T) digital SRAM array can be configured as an *analog-like in-memory multi-bit dot product engine*. By applying appropriate analog voltages to the read-ports of the 8T SRAM array, and sensing the output current, an approximate analog-digital dot-product engine can be implemented. We present two different configurations for enabling multi-bit dot product computations in the 8T SRAM cell array, without modifying the standard bit-cell structure. Since our proposal preserves the standard 8T-SRAM array structure, it can be used as a storage element with standard read-write instructions, and also as an *on-demand* analog-like dot product accelerator.**

***Index Terms*—In-memory computing, SRAMs, von Neumann bottleneck, convolution, dot product.**

I. Introduction

State-of-the-art computing platforms are widely based on the von-Neumann architecture [1]. The von-Nuemann architecture is characterized by distinct spatial units for *computing* and *storage*. Such physically separated memory and compute units result in huge energy consumption due to frequent data transfer between the two entities. Moreover, the transfer of data through a dedicated limited-bandwidth bus limits the overall compute throughput. The resulting memory bottleneck is *the major throughput concern* for hardware implementations of data intensive applications like machine learning, artificial intelligence *etc*.

A possible approach geared towards high throughput beyond von-Neumann machines is to enable distributed computing characterized by tightly intertwined storage and compute capabilities. If computing can be performed inside the memory array, rather than in a spatially separated computing core, the compute throughput can be considerably increased . As such, one could think of *ubiquitous* computing on the silicon chip, wherein both the logic cores and the memory unit partake in compute operations. Various proposals for *'in-memory'* computing with respect to emerging non-volatile technologies have been presented for both dot product computations [2], [3] as well as vector Boolean operations [4]. Prototypes based on emerging technologies can be found in [3], [5] .

With respect to the CMOS technology, Boolean in-memory operations have been presented in [6] and [7]. In [6] authors have presented vector Boolean operations using 6T SRAM cells. Additionally, authors in [7] have demonstrated that the 8 transistor (8T) SRAM cells lend themselves easily as vector compute primitives due to their decoupled read and write ports. Both the works [6] and [7] are based on vector Boolean operations. However, perhaps the most frequent and compute intensive function required for numerous applications like machine learning is the *dot product* operation. Memristors based on resistive-RAMs (Re-RAMs) have been reported in many works as an analog dot product compute engine [4], [8]. Few works based on analog computations in SRAM cells can be found in [9], [10]. Both these works use 6T SRAM cells and rely on the resultant accumulated voltage on the bit-lines (BLs). Not only 6T SRAMs are prone to read-disturb failures, the failures are also a function of the voltage on the BLs. This leads to a tightly constrained design space for the proposed 6T SRAM based analog computing. In this paper, we employ 8T cells that are much more robust as compared to the 6T cells due to isolated read port. We show that without modifying the basic bit-cell for the 8T SRAM cell, it is possible to configure the 8T cell for in-memory dot product computations. Note, in sharp contrast to the previous works on in-memory computing with the CMOS technology, we enable *current based, analog-like* dot product computations using robust digital 8T bit-cells.

The key highlights of the present work are as follows:

1) We show that the conventional 8T SRAM cell can be used as a primitive for analog-like dot product computations, without modifying the bit-cell circuitry. In addition, we present two different configurations for enabling dot product computation using the 8T cell.
2) Apart for the sizing of the individual transistors consisting the read port of the 8T cell, the basic bit-cell

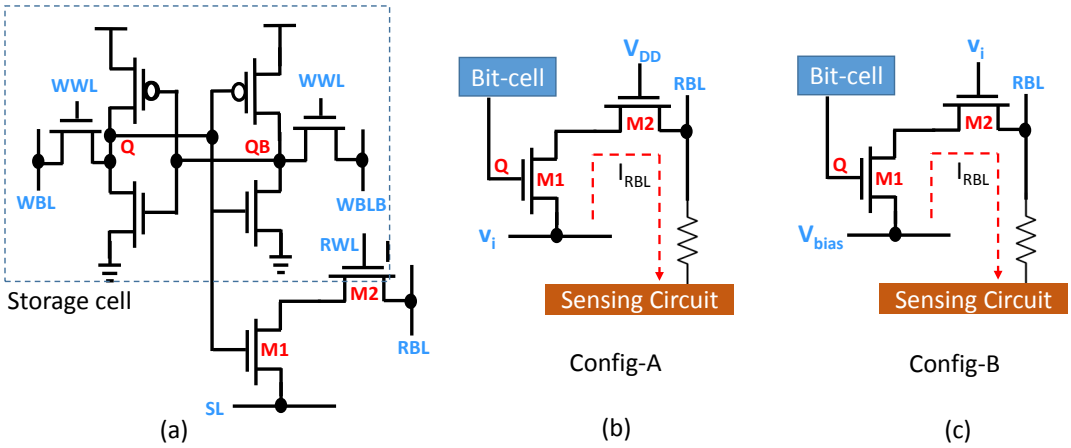

Fig. 1: (a) Schematic of a standard 8T-SRAM bit-cell. It consists of two decoupled ports for reading and writing respectively. (b) First proposed configuration (Config-A) for implementing the dot product engine using the 8T-SRAM bit-cell. The SL is connected to the input analog voltage v_i, and the RWL is turned ON. The current I_{RBL} through the RBL is sensed and is proportional to the dot product $v_i \cdot g_i$, where g_i is the ON/OFF conductance of the transistors $M1$ and $M2$. (c) Second proposed configuration (Config-B). The input analog voltages are applied to the RWL, while the SL is supplied with a constant voltage V_{bias}. The current through the RBL is sensed in the same way as in Config-A.

structure remains unaltered. Thereby, the 8T SRAM array can also be used for usual digital memory read and write operations. As such, the presented 8T cell array can act as a dedicated dot product engine or as an *on-demand* dot product accelerator.

3) A detailed simulation analysis using 45nm predictive technology models including layout analysis has been reported highlighting the various trade-offs presented by each of the two proposed configurations.

II. 8T-SRAM as a Dot Product Engine

A conventional 8T bit-cell is schematically shown in Fig. 1(a). It consists of the well-known 6T-SRAM bit-cell with two additional transistors that constitute a decoupled read port. To write into the cell, the write word-line (WWL) is enabled, and write bit-lines (WBL/WBLB) are driven to V_{DD} or ground depending on the bit to be stored. To read a value from the cell, the read bit-line (RBL) is pre-charged to V_{DD} and the read word-line (RWL) is enabled. Note, that the source-line (SL) is connected to the ground. Depending on whether the bit-cell stores a logic '1' or '0', the RBL discharges to 0V or stays at V_{DD}, respectively. The resulting voltage at the RBL is read out by the sense amplifiers. Although 8T-cells incur a $\sim$30% increase in bit-cell area compared to the 6T design, they are read-disturb free and more robust due to separate read and write path optimizations [11].

We now show how such 8T-SRAMs, with no modification to the basic bit-cell circuit (except for the sizing of the read transistors), can behave as a dot product engine, without affecting the stability of the bits stored in the SRAM cells. We propose two configurations - *Config-A* and *Config-B*, for enabling dot-product operations in the 8T-SRAMs. Config-A is shown in Fig. 1(b). The inputs v_i (encoded as analog voltages) are applied to the SLs of the SRAM array, and the RWL is also enabled. The RBL is connected to a sensing circuitry, which we will describe later. Thus, there is a static current flow from the SL to the RBL, which is proportional to the input v_i and the conductance of the two transistors $M1$ and $M2$. For simplicity, assume that the weights (stored in the SRAM) have a single-bit precision. If the bit-cell stores '0', the transistor $M1$ is OFF, and the output current through the RBL is close to 0. Whereas if the bit-cell stores a '1', the current is proportional to $v_i \cdot g_{ON}$, where g_{ON} is the series 'ON' conductance of the transistors. Assume similar inputs v_i are applied on the SLs for each row of the memory array. Since the RBL is common throughout the column, the currents from all the inputs v_i are summed into the RBL. Moreover, since the SL is common throughout each row, the same inputs v_i are supplied to multiple columns. Thus, the final output current through RBL of each column is proportional to $I^j_{RBL} = \Sigma(v_i \cdot g^j_i)$, where g^j_i is the 'ON' or 'OFF' conductance of the transistors, depending on whether the bit-cell in the i-th row and j-th column stores a '1' or '0', respectively. The output current vector thus resembles the vector-matrix dot product, where the vector is v_i in the form of input analog voltages, and the matrix is g^j_i stored as digital data in the SRAM.

Let us now consider a 4-bit precision for the weights. If the weight $W^j_i = w_3 w_2 w_1 w_0$, where w_i are the bits corresponding to the 4-bit weight, the vector matrix dot product becomes:

$$\Sigma(v_i \cdot W^j_i) = \Sigma[v_i \cdot (2^3 w_3 + 2^2 w_2 + 2^1 w_1 + w_0)]$$
$$= \Sigma(v_i \cdot 2^3 w_3) + \Sigma(v_i \cdot 2^2 w_2) + \Sigma(v_i \cdot 2^1 w_1) + \Sigma(v_i \cdot w_0)$$

Now, if we size the read transistors $M1$ and $M2$ of the SRAM bit-cells in column 1 through 4 in the ratio $2^3 : 2^2 : 2^1 : 1$, as shown in Fig. 2, the transistor conductances in the 'ON' state would also be in the ratio $2^3 : 2^2 : 2^1 : 1$. Thus, summing the currents through the RBLs of the four columns yields the required dot product in accordance to the

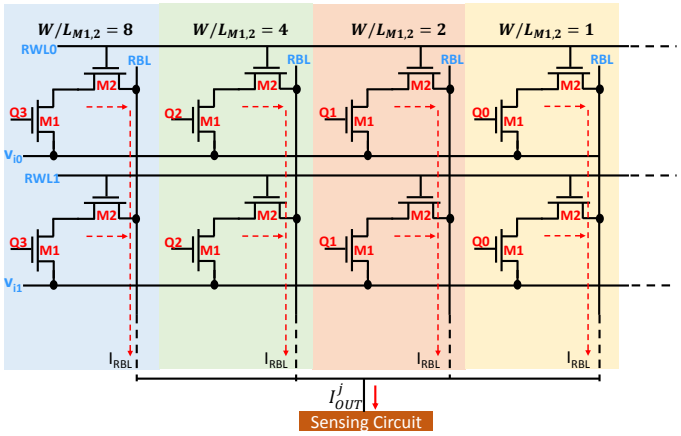

Fig. 2: 8T-SRAM memory array for computing dot-products with 4-bit weight precision. Only the read port is shown, the 6T storage cell and the write port are not shown. The array columns are grouped in four, and the transistors $M1$ and $M2$ are sized in the ratio $8:4:2:1$ for the four columns. The output current I^j_{OUT} represents the weighted sum of the I_{RBL} of the four columns, which is approximately equal to the desired dot-product.

equation shown above. This sizing pattern can be repeated throughout the array. In addition, one could also use transistors having different threshold voltages to mimic the required ratio of conductances as $2^3:2^2:2^1:1$. Note that, the currents through the RBLs of the four consecutive columns are summed together, thus we obtain one analog output current value for every group of four columns. In other words, the digital 4-bit word stored in the SRAM array is multiplied by the input voltage v_i and summed up by analog addition of the currents on the RBLs. This *one-go* computation of vector multiplication and summation in a digital memory array would result in high throughput computations of the dot products.

It is worth mentioning, that the way input v_i are multiplied by the stored weights and summed up is reminiscent of memristive dot product computations [8]. However, a concern with the presented SRAM based computation is the fact that the ON resistance of the transistors (few kilo ohms) are much lower as compared to a typical memristor ON resistance which is in the range of few tens of kilo ohms [12]. As such the static current flowing through the ON transistors $M1$ and $M2$ would typically be much higher in the presented proposal. In order to reduce the static current flow, we propose scaling down the supply voltage of the SRAM cell. Note, interestingly, 8T cells are known to retain their robust operation even at highly scaled supply voltages [13]. In the next section we have used a V_{DD} lower than the nominal V_{DD} of 1V. We would now describe another way of reducing the current, although with trade-offs, as detailed below.

Config-B is shown in Fig. 1(c). Here, the SLs are connected to a constant voltage V_{bias}. The input vector v_i is connected to RWLs, i.e., the gate of $M2$. Similar to Config-A, the output current I_{RBL} is proportional to v_i. We will later show from our simulations that for a certain range of input voltage values, we get a linear relationship between I_{RBL} and v_i, which can be exploited to calculate the approximate dot product. To implement multi-bit precision, the transistor sizing is done in the same way as Config-A as represented in Fig. 2, so that the I_{RBL} is directly proportional to the transistor conductances. Key features of the proposed Config-B are as follows. V_{bias} can be adjusted to reduce the current flowing through the RBLs. The input voltages v_i have a capacitive load, as opposed to a resistive load in Config-A. This relaxes the constraints on the input voltage generator circuitry, and is useful while cascading two or more stages of the dot product engine. However, as presented in the next section, Config-B has a small non-zero current corresponding to zero input as opposed to Config. A that has zero current for zero input.

In order to sense the output current at the RBLs, we use a current to voltage converter. This can most simply be a resistor, as shown in Fig. 1. However, there are a few constraints. As the output current increases, the voltage drop across the output resistor increases, which in turn changes the desired current output. A change in the voltage on the RBL would also change the voltage across the transistors $M1$ and $M2$, thereby making their conductance a function of the voltage on the RBL. Thus, at higher currents corresponding to multiple rows of the memory array, the I_{RBL} does not approximate the vector-matrix dot product, but deviates from the ideal output. This dependence of the RBL voltage on the current I_{RBL} will be discussed in detail in the next section with possible solutions.

III. Results and Discussions

The operation of the proposed configurations (Config-A and Config-B) for implementing a multi-bit dot product engine was simulated using HSPICE on the 45nm PTM technology [14]. For the entire analysis, we have used a scaled down V_{DD} of 0.65V for the SRAM cells. The main components of the dot-product engine implementation are the input voltages and conductances of the transistors for different states of the cells. A summary of the analysis for the two configurations is presented in Fig. 3. In Fig. 3, we have assumed a sensing resistance of 50-ohms connected to the RBL. Note, a small sense resistance is required to ensure that the voltage across the sensing resistance is not high enough to drastically alter the conductances of the connected transistors $M1$ and $M2$.

In Fig. 3(a)-(b) we plot the output current in RBL (I_{RBL}) as a function of the input voltage for three 4-bit weight combinations '1111', '1010' and '0100' for the two different configurations described in the previous section. The results presented are for a single 4-bit cell. To preserve the accuracy of a dot-product operation, it is necessary to operate the cell in the voltage ranges such that the current is a linear function of the applied voltage v_i. These voltage ranges are marked as linear region in Fig. 3(a)-(b). The slope of the linear section I_{RBL} versus V_{in} plot varies with weight, thus signifying a dot product operation. Further, at the left voltage extremity of the linear region, I_{RBL} tends to zero irrespective of the weight, thus satisfying the constraint that the output current is zero for zero V_{in}. It is to be noted that the two configurations show significantly different characteristics due to the different point-of-application of input voltages.

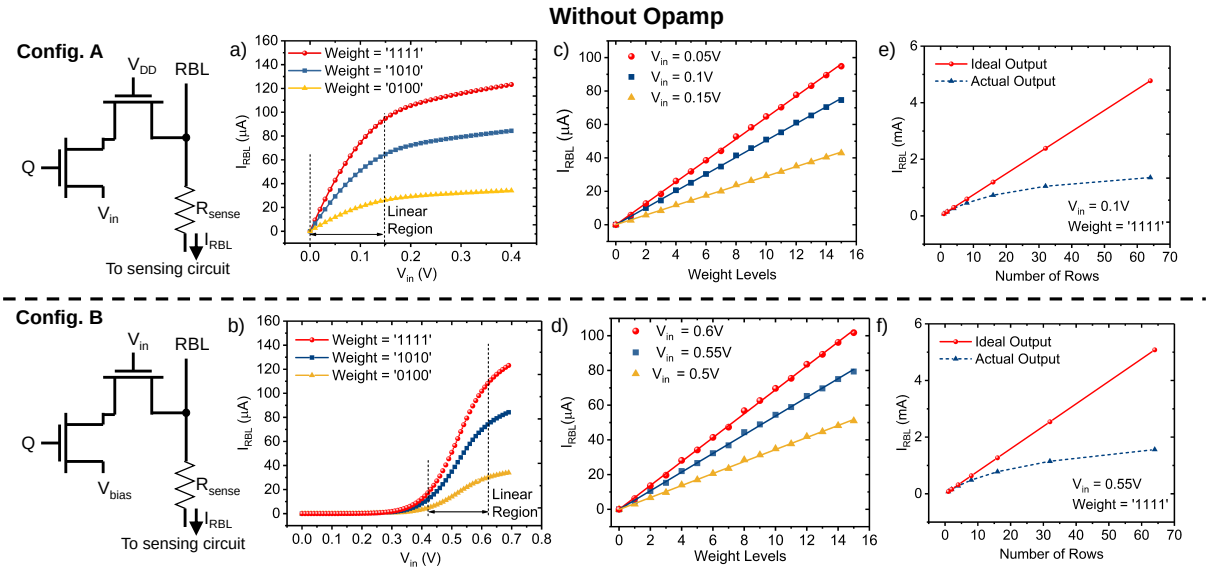

Fig. 3: I_{RBL} versus V_{in} characteristics for (a) Config. A and (b) Config. B shows the linear region of operation for different weights. I_{RBL} versus Weight levels for (c) Config. A and (d) Config. B shows desirable linear relationship at various voltages V_{in}. I_{RBL} shows significant deviation from ideal output ($I_N = N \times I_1$ with increasing number of rows for both (e) Config. A and (f) Config. B, where I_1 is the current corresponding to one row and N is the number of rows. The analyses were done for $V_{DD} = 0.65V$

Fig. 3(c)-(d) presents the dependence of the current I_{RBL} on the 4-bit weight levels for Config-A at constant voltages V_{in} = 0.05V, 0.1V, 0.15V and configuration B at V_{in} = 0.5V, 0.55V, 0.6V, respectively. Different voltages were chosen so as to ensure the circuit operates in the linear region as depicted by Fig. 3(a)-(b). Desirably, I_{RBL} shows a linear dependence on weight levels and tends to zero for weight = '0000'. The choice of any voltage in the linear regions of Fig 3(a)-(b) does not alter the linear dependence of the I_{RBL} on weight levels.

To expand the dot-product functionality to multiple rows, we performed an analysis for upto 64 rows in the SRAM array, driven by 64 input voltages. In the worst case condition, when the 4-bit weight stores '1111', maximum current flows through the RBLs, thereby increasing the voltage drop across the output resistance. Fig. 3(e)-(f) indicates that the total current I_{RBL} deviates from its ideal value with increasing number of rows, in the worst case condition. The deviation in Fig. 3(e)-(f) is because we sense the output current with an equivalent sensing resistance (R_{sense}) and hence the final voltage on the bit-line (V_{BL}) is dependent on the current I_{RBL}. At the same time, I_{RBL} is also dependent on V_{BL} and as a result the effective conductance of the cell varies as V_{BL} changes as a function of the number of rows. It was also observed that the deviation reduces with decreasing sensing resistance as expected. Another concern with respect to Fig. 3 is the fact that the total summed up current reaches almost 6mA for 64 rows for the worst case condition (all the weights are '1111').

There are several ways to circumvent the deviation from ideal behavior with increasing number of simultaneous row accesses and also reduce the maximum current flowing through the RBLs. One possibility is to use an operational amplifier (Opamp) at the end of each 4-bit column, where the negative differential input of the Opamp is fed by the bit-line corresponding to a particular column. Whereas, the positive input is supplemented by a combination of the Opamp offset voltage and any desired voltage required for suitable operation of the dot-product as shown in left hand side of Fig. 4. Opamp provides a means of sensing the summed up current at the RBL while maintaining a constant voltage at the RBL. Opams in the configuration as shown in Fig. 4 have been traditionally used for sensing in memristive crossbars as in [3].

We performed the same analysis as previously described in Fig. 3 for the two proposed configurations with the bit-line terminated by an Opamp. For our analysis, we have set $V_{pos} = 0.1V$ for the positive input of the Opamp and thus analysis is limited to input voltages above V_{pos} to maintain the unidirectional current. Note, we have used an ideal Opamp for our simulations, where the voltage V_{pos} can be accounted for both the non-ideal offset voltage of the Opamp and a combination of an externally supplied voltage. Fig. 4(a)-(b) shows the plot of I_{RBL} versus input voltage V_{in} for the two configurations. Similar behavior as in the case of Fig. 3(a)-(b) is observed even in the presence of the Opamp. However, note that the current ranges have decreased since RBL is now clamped at V_{pos}. Further, the dot-product operation is only valid for $V_{in} > V_{pos}$ and thus the acceptable input range is shifted in the presence of an Opamp. Fig. 4(c)-(d) shows the behavior of I_{RBL} versus weight levels for the two configurations and desirably, linearity is preserved.

Fig. 4(e)-(f) presents the current through the RBL as a function of the number of rows. As expected, due to the high input impedance of the Opamp, and the clamping of V_{BL} at a voltage V_{pos} the deviation of the summed up current from the ideal value have been mitigated to a huge extent. Although, the current levels have reduced significantly as compared to the

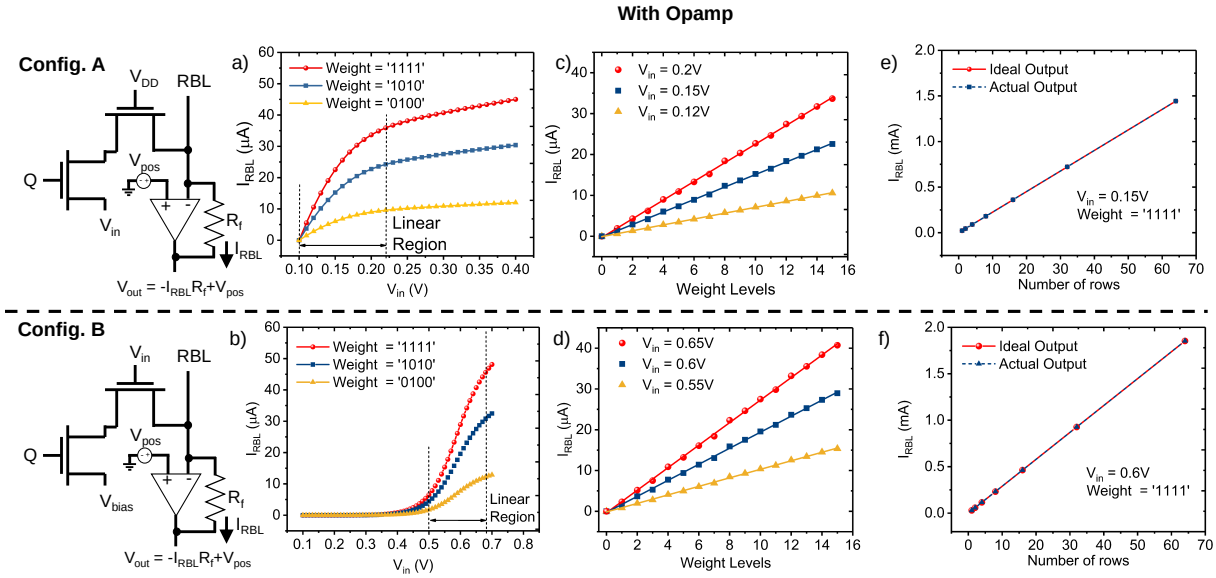

Fig. 4: I_{RBL} versus V_{in} characteristics for (a) Config. A and (b) Config. B shows the linear region of operation for different weights. I_{RBL} versus weight levels for (c) Config. A and (d) Config. B shows desirable linear relationship at various voltages V_{in}. I_{RBL} shows almost zero deviation from ideal output ($I_N = N \times I_1$ with increasing number of rows for both (e) Config. A and (f) Config. B, where I_1 is the current corresponding to one row and N is the number of rows. These analyses were done for $V_{DD} = 0.65V$

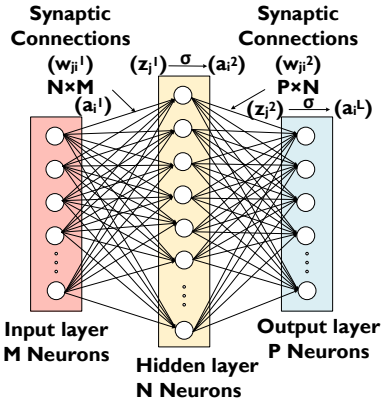

Fig. 5: Fully connected network topology consisting of 3 layers, the input layer, the hidden layer and the output layer [12]. We have used M=784, N = 500 and P = 10.

Fig. 3, the resultant current for 64 rows would still be higher than the electro-migration limit for the metal lines constituting the RBL [15]. One possible solution is to sequentially access a smaller section of the crossbar (say 16 rows at a time), convert the analog current into its digital counterpart each time and finally add all accumulated digital results. In addition use of high threshold transistors for the read port of the SRAM would also help to reduce the maximum current values. Further, the maximum current is obtained only when all the weights are '1111', which is usually not true due to the sparsity of matrices involved in various applications as in [16], [17].

We also performed functional simulations using the proposed dot-product engine based on Config. A in a fully connected artificial neural network consisting of 3 layers as shown in Fig. 5. The main motivation behind this analysis

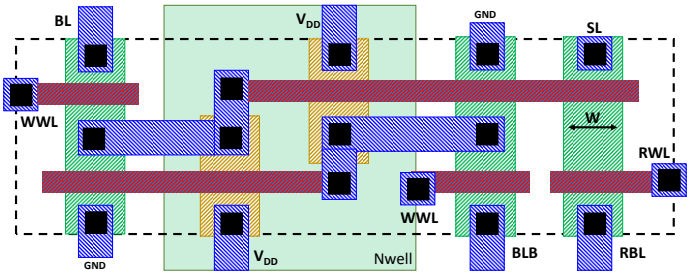

Fig. 6: Thin-cell layout for a standard 8T-SRAM bit-cell [11].

is to evaluate the impact of the non-linearity in the I-V characteristics on the inference accuracy of the neural network. We chose an input voltage range of 0.1-0.22V. As can be observed in Fig. 4(a), the I-V characteristics are not exactly linear within this range, as such a network level functional simulation is required to ascertain the impact of the non-linearity on classification accuracy. The network details are as follows. The hidden layer consisted of 500 neurons. The network was trained using the Backpropagation algorithm [18] on the MNIST digit recognition dataset under ideal conditions using MATLAB ®Deep Learning Toolbox [19].

During inferencing, we incorporated the proposed 8T-SRAM based dot-product engine in the evaluation framework by discretizing and mapping the trained weights proportionally to the conductances of the 4-bit synaptic cell. The linear range of the voltage was chosen to be [0.1-0.22V] and normalized to a range of [0 1]. The dot-product operation was ensured by normalizing the I-V characteristics for all the weight levels such that current corresponding to the highest input voltage and highest weight level is $I_{max} = V_{max} \times G_{max}$. The activa-

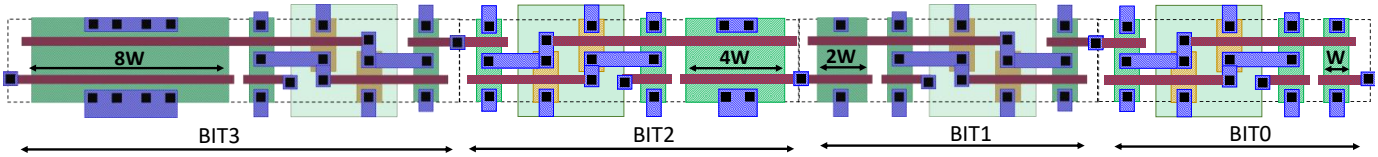

Fig. 7: Thin-cell layout for the proposed 8T-SRAM array with 4-bit precision weights. The width of read transistors of different bit positions are sized in the ratio 8:4:2:1. This incurs an area overhead of ∼15.6% compared to the standard 8T-SRAM bit-cell.

tion function of the neuron was considered to be a behavioral $satlin$ function scaled according to the scaling factor of the weights to preserve the mathematical integrity of the network. To be noted, the normalization of current and input voltage simplifies the scaling of the neuron activation function. The accuracy of digit recognition task was calculated to be merely 0.11% lower than the ideal case (98.27%) thus indicating that the proposed dot-product engine can be seamlessly integrated into the neural network framework without significant loss in performance.

Further, it is to be noted that in many cases the inherent resilience of the applications that require dot product computations can be leveraged to circumvent some of the circuit level non-idealities. Additionally, the proposed technique can either be used as a dedicated CMOS based dot product compute engine or as an on-demand dot product accelerator, wherein the 8T array acts as usual digital storage and can also be configured as a compute engine as and when required. It is also worth mentioning that the 8T cell has also been demonstrated in [7] as a primitive for vector Boolean operations. This work, significantly augments the possible use cases for the 8T cells by adding analog-like dot product acceleration.

Due to different sizing of the read transistors, there is an area penalty of using the proposed configurations, compared to the standard 8T-SRAM bit-cell used for storage. Fig. 6 shows the thin-cell layout for a standard 8T-SRAM bit-cell [11]. Note that the rightmost diffusion with width (W) constitute the read transistors ($M1$ and $M2$). To implement the 4-bit precision dot-product, we size the width of read transistors in the ratio $8:4:2:1$, as described earlier. Thus, the width of the rightmost diffusion is increased to 8W, 4W, and 2W, resulting in an area overhead of ∼39.6%, 17.1%, and 5.7% for bits 3, 2 and 1, respectively, compared to the standard minimum sized 8T bit-cell with diffusion width W. The resulting layout of first four columns for a particular row in the proposed array is shown in Fig. 7. The overall area overhead for the whole SRAM array with 4-bit weight precision, amounts to ∼15.6% compared to the standard 8T SRAM array. Note, this low area overhead results from the fact that both the read transistors $M1$ and $M2$ share a common diffusion layer and hence an increase in transistor width can be easily accomplished by having a longer diffusion, whithout worrying about spacing beween metal or poly layers.

Finally, we discuss the power consumed by the proposed dot product engine. The worst case power consumption will occur when all the inputs are at its highest operating voltage and all the weights are set to '1111'. Considering a column of 16 input nodes (16 rows), for a highest voltage of 0.22 V for Config. A with Opamp, the worst case power consumption was 128μW. The average power consumption across different voltages and weight levels was determined to be around 33.5μW for the same configuration. Similarly, for the highest voltage of 0.65 V and $V_{bias} = 0.3V$ for Config. B with Opamp, the worst case and average power consumption was 196μW and 68.1μW respectively. It is to be noted that the power in case of Config. B can be reduced by using a lower V_{bias}.

IV. Conclusion

In the quest for novel in-memory techniques for beyond von-Neumann computing, we have presented the 8T-SRAM as a vector-matrix dot-product compute engine. Specifically, we have shown two different configurations with respect to 8T SRAM cell for enabling analog-like multi-bit dot product computations. We also highlight the trade-offs presented by each of the proposed configurations. The usual 8T SRAM bit cell circuit remains unaltered and as such the 8T cell can still be used for the normal digital memory read and write operations. The proposed scheme can either be used as a dedicated dot product compute engine or as an on-demand compute accelerator. The presented work augments the applicability of 8T cells as a compute accelerator in the view that dot products find wide applicability in multiple data intensive application and algorithms including efficient hardware implementations for machine learning and artificial intelligence.

References

[1] J. Von Neumann, *The computer and the brain*. Yale University Press, 2012.

[2] J. J. Yang, D. B. Strukov, and D. R. Stewart, "Memristive devices for computing," *Nature nanotechnology*, vol. 8, no. 1, p. 13, 2013.

[3] C. Li, M. Hu, Y. Li, H. Jiang, N. Ge, E. Montgomery, J. Zhang, W. Song, N. Dávila, C. E. Graves, Z. Li, J. P. Strachan, P. Lin, Z. Wang, M. Barnell, Q. Wu, R. S. Williams, J. J. Yang, and Q. Xia, "Analogue signal and image processing with large memristor crossbars," *Nature Electronics*, vol. 1, no. 1, pp. 52–59, dec 2017. [Online]. Available: https://doi.org/10.1038%2Fs41928-017-0002-z

[4] J. Borghetti, G. S. Snider, P. J. Kuekes, J. J. Yang, D. R. Stewart, and R. S. Williams, "memristiveswitches enable statefullogic operations via material implication," *Nature*, vol. 464, no. 7290, p. 873, 2010.

[5] S. H. Jo, T. Chang, I. Ebong, B. B. Bhadviya, P. Mazumder, and W. Lu, "Nanoscale memristor device as synapse in neuromorphic systems," *Nano letters*, vol. 10, no. 4, pp. 1297–1301, 2010.

[6] Q. Dong, S. Jeloka, M. Saligane, Y. Kim, M. Kawaminami, A. Harada, S. Miyoshi, D. Blaauw, and D. Sylvester, "A 0.3 v vddmin 4+ 2t sram for searching and in-memory computing using 55nm ddc technology," in *VLSI Circuits, 2017 Symposium on*. IEEE, 2017, pp. C160–C161.

[7] A. Agrawal, A. Jaiswal, and K. Roy, "X-sram: Enabling in-memory boolean computations in cmos static random access memories," *arXiv preprint arXiv:1712.05096*, 2017.

[8] C. Li, M. Hu, Y. Li, H. Jiang, N. Ge, E. Montgomery, J. Zhang, W. Song, N. Dávila, C. E. Graves *et al.*, "Analogue signal and image processing with large memristor crossbars," *Nature Electronics*, vol. 1, no. 1, p. 52, 2018.

[9] M. Kang, S. K. Gonugondla, A. Patil, and N. R. Shanbhag, "A multi-functional in-memory inference processor using a standard 6t sram array," *IEEE Journal of Solid-State Circuits*, vol. 53, no. 2, pp. 642–655, Feb 2018.

[10] J. Lee, D. Shin, Y. Kim, and H. J. Yoo, "A 17.5-fj/bit energy-efficient analog sram for mixed-signal processing," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 25, no. 10, pp. 2714–2723, Oct 2017.

[11] K. Takeda, Y. Hagihara, Y. Aimoto, M. Nomura, Y. Nakazawa, T. Ishii, and H. Kobatake, "A read-static-noise-margin-free sram cell for low-vdd and high-speed applications," *IEEE journal of solid-state circuits*, vol. 41, no. 1, pp. 113–121, 2006.

[12] I. Chakraborty, D. Roy, and K. Roy, "Technology aware training in memristive neuromorphic systems based on non-ideal synaptic crossbars," *arXiv preprint arXiv:1711.08889*, 2017.

[13] L. Chang, Y. Nakamura, R. K. Montoye, J. Sawada, A. K. Martin, K. Kinoshita, F. H. Gebara, K. B. Agarwal, D. J. Acharyya, W. Haensch *et al.*, "A 5.3 ghz 8t-sram with operation down to 0.41 v in 65nm cmos," in *VLSI Circuits, 2007 IEEE Symposium on*. IEEE, 2007, pp. 252–253.

[14] W. Zhao and Y. Cao, "New generation of predictive technology model for sub-45 nm early design exploration," *IEEE Transactions on Electron Devices*, vol. 53, no. 11, pp. 2816–2823, 2006.

[15] G. Posser, V. Mishra, R. Reis, and S. S. Sapatnekar, "Analyzing the electromigration effects on different metal layers and different wire lengths," in *Electronics, Circuits and Systems (ICECS), 2014 21st IEEE International Conference on*. IEEE, 2014, pp. 682–685.

[16] S. Changpinyo, M. Sandler, and A. Zhmoginov, "The power of sparsity in convolutional neural networks," *arXiv preprint arXiv:1702.06257*, 2017.

[17] S. Han, J. Pool, J. Tran, and W. Dally, "Learning both weights and connections for efficient neural network," in *Advances in neural information processing systems*, 2015, pp. 1135–1143.

[18] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning internal representations by error propagation," Tech. Rep., sep 1985. [Online]. Available: https://doi.org/10.21236%2Fada164453

[19] R. B. Palm, "Prediction as a candidate for learning deep hierarchical models of data," *Technical University of Denmark*, vol. 5, 2012.