Chapter 10

Introduction to Memristive HTM Circuits

Alex James, Timur Ibrayev, Olga Krestinskaya and Irina Dolzhikova

Additional information is available at the end of the chapter

http://dx.doi.org/10.5772/intechopen.70123

Abstract

Hierarchical temporal memory (HTM) is a cognitive learning algorithm intended to mimic the working principles of neocortex, part of the human brain said to be responsible for data classification, learning, and making predictions. Based on the combination of various concepts of neuroscience, it has already been shown that the software realization of HTM is effective on different recognition, detection, and prediction making tasks. However, its distinctive features, expressed in terms of hierarchy, modularity, and sparsity, suggest that hardware realization of HTM can be attractive in terms of providing faster processing speed as well as small memory requirements, on-chip area, and total power consumption. Despite there are few works done on hardware realization for HTM, there are promising results which illustrate effectiveness of incorporating an emerging memristor device technology to solve this open-research problem. Hence, this chapter reviews hardware designs for HTM with specific focus on memristive HTM circuits.

Keywords: hierarchical temporal memory, spatial pooler, temporal memory, memristor, non-volatile memory, memristive crossbars

1. Introduction

The ideas that created a basis for development of hierarchical temporal memory (HTM), a type of machine learning algorithm that emerged from the consideration of the Bayesian neural network (BNN) and spatial-temporal algorithm, was first introduced by Jeff Hawkins in 2004 in his book *On Intelligence* [1] written in collaboration with Sandra Blakeslee. One year later, in 2005, Hawkins launched Numenta company that worked on the implementation of HTM technology. The first version of the HTM algorithm implementation was developed in

2006 and is now known as HTM-Zeta1 [2]. Three more years of work on the improvement of the HTM-Zeta1 produced a new version of HTM algorithm that is based on the cortical learning algorithm (CLA) and is now presented in the updated white paper of Numenta [3].

The aim of this chapter is to familiarize the reader with one of the latest versions of the machine learning algorithms in the face of hierarchical temporal memory (HTM). The focus of the chapter is utilization of the nanoscale memristive devices for hardware implementations of the HTM. However, due to the complex background of the novel learning algorithm the authors purposefully start from the general descriptions and useful explanations of the major concepts. A variety of concrete examples of HTM realizations in both software and hardware is presented in this chapter. This is done to give the reader comprehensive understanding of the current situation in the HTM research area.

The chapter is organized as follows. We discuss the structure and main concepts of the HTM algorithm in Section 2. The description of the concepts closely follows the white paper documents of Numenta company. In addition to that, a high level overview of the mathematical formulation of HTM phases is given based on the framework that has been demonstrated by Mnatzaganian et al. [4]. Section 3 summarizes the existing implementations of the HTM in both software and hardware. This is followed by the short Section 5 that is dedicated to the memristive device, which is currently widely used for different application. Most importantly, memristor demonstrated a great potential to be especially useful for neuromorphic applications. One of such application is realization of the synaptic behavior in the circuits. This is what Section 5 is about. Finally, Section 6 summarizes the works that demonstrated the application of memristive devices for implementation of hardware designs of HTM.

2. Structure and basic concepts of HTM

For the timely varying set of data to be analyzed the regular horizontal organization that is being utilized in the most of the conventional computer memories will not work. There should be a hierarchy notion that needs to be implemented. This is what the major idea of HTM is based on. As the name suggests the HTM memory utilizes the hierarchical structure with the cells being the primary elements [3]. **Figure 1** depicts an example of hierarchical structure of HTM consisting of three levels. The primary element of the HTM is a cell. These cells perform the function of artificial neurons that resembles the functionalities of the biological neuron. Several cells are grouped to form a node, which is also known as column. The set of columns are combined to form a region, and several regions are interconnected in a hierarchical manner, where the higher level in the hierarchy utilizes the patterns that were learned from the lower levels. The connectivity of the neurons in biology is established by synapses. Being inspired from this concept, HTM cells are interconnected using synapses. In the framework of HTM, the strength of the synapse is known as permanence [3]. The permanence value determines which of the synapses are strong enough to establish the connection for further communication that enables the learning from other cells and lower levels.

http://dx.doi.org/10.5772/intechopen.70123

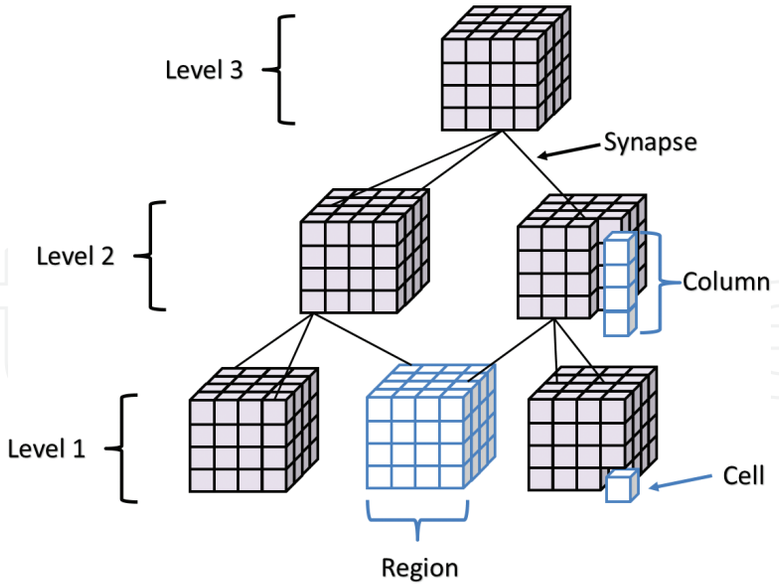

Figure 1. Detailed demonstration of the HTM structure consisting of 3 levels, each composed of certain number of regions with 16 columns each having 4 cells.

Figure 2 demonstrates how the concept of hierarchy could be used in the image recognition application. Here the single level is represented by the single region that composed of several nodes, where the number of nodes is reduced as we go from the bottom level to the top level. Reduced number of the nodes in the higher levels could be explained by the fact that as we go up, we combine certain elements to form the features. As an example let consider an input image that illustrates the face of the human. By looking at the image and using the top-down approach, we realize that the given image represents the human face that in turn has several apparent features, such as nose, eyes, and lips. We then can go deeper and consider the visible features of the eye that incorporate the iris and pupil. Finally, we might zoom in the image and realize that features of the images are formed by certain pixels. Thus, it could be seen that at different HTM levels we have access to different levels of details of the input information.

Two mechanisms that allow the HTM to operate by searching for the common patterns in the spatial domain and determining the common temporal patterns are represented by the spatial pooler (SP) and temporal memory (TM), respectively [3]. Here the term spatial pattern means that there is the combination of the input bits, which is represented by the activation of certain cells in the column, often appearing together. Temporal patterns in turn consider how the sequence of the spatial patterns changes over time [3].

For the realization of HTM, first of all, the SP is required to convert the binary form of the input data into sparse distributed representation (SDR). This type of representation is formed by activating only very small number of bits to illustrate a particular input. SDR in contrast to the dense representation reduces the requirements for the storage capacity because the input data is presented by highly distributed cells. In addition to that, the sparsity of the active bits leads to the improved robustness of the system to the noise.

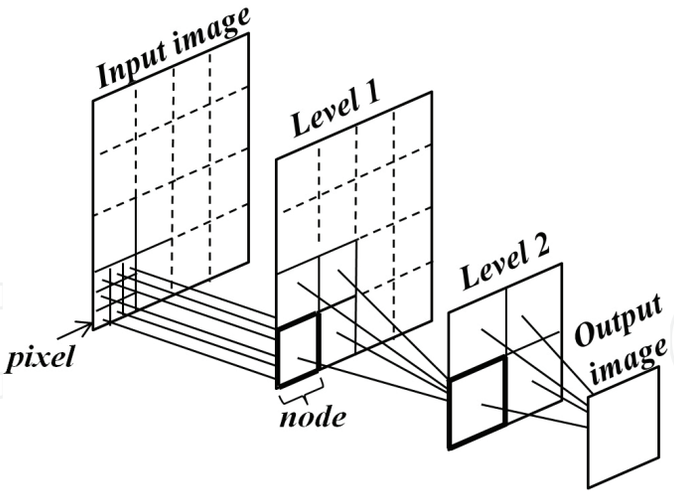

Figure 2. Illustration of the HTM hierarchy based on the image recognition example.

Before talking about the advantage of SDR based systems in terms of storage capacity itself, let first clearly understand the difference between the dense representations and SDRs, the two methods that are utilized for information storage and processing by the traditional computers and brain, respectively.

An example of storing the information in the traditional computer is as follows. The computer uses 8, 32 or 64, bit word with all different combinations of 1's and 0's. By taking any individual bit from the word, one will not get any useful information. The combination of bits itself and position of the active and inactive bits (1's and 0's) with respect to each other, this is what plays an important role. As an example 11010101 is the dense representation of the letter "j" in the ASCII code. The problem might occur even if only single bit of the word is corrupted, such as the third bit from the left swept to the 1 and as a result the dense representation 1101**1**10 will demonstrate a totally different letter in ASCII code, which is "n."

Contrary to this, the functionality of the brain is based on the SDR. This means that for the input consisting of the thousands of bits, at every instant only small portion of these bits is active (represented by 1), while others being 0. In contrast to dense representation, in SDR every bit has its meaning that defines some attribute that describes the information (for example in case of the letter the attribute could define upper-case or lower-case letter, whether it is vowel or consonant). **Figure 3a** demonstrates SDR of the 1000 bits with very small number of bits being 1. These representation can be stored by considering the positions of the 1's in the stream of bits, i.e. the indexes of active bits. Thus, the SDR on **Figure 3a** will be stored as [1, 11, 998] which requires very little space. The comprehensive description of the SDR in the HTM could be found in the technical report of Hawkins and George [2].

This idea of SDR is used in the HTM algorithm. **Figure 3b** shows an example how the input information representing the set of geometrical figures could be stored in the HTM hierarchy in the distributed way with very few of the cells in the region being activated. The realization of the SP requires consideration of the following three phases: overlap phase, inhibition phase and learning phase.

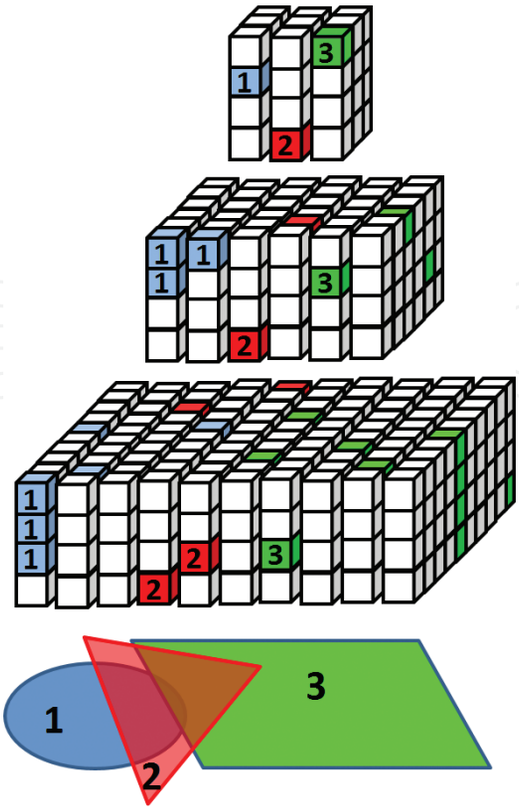

Figure 3. Example of (a) how SDR can be used to represent a bit stream and (b) how SDR is formed within HTM hierarchy to represent different geometrical figures.

A purely mathematical formalization of HTM SP was presented by Mnatzaganian et al. [4]. This mathematical framework was established to optimize the development of the HTM designs on hardware. Bellow we discuss the three phases of SP based on pseudocode that was presented in the white paper of Numenta and the work of the Mnatzaganian.

Before the first phase of SP occurs, the initialization should be performed. During the initialization, the initial synapses are established for each column and their permanence value, which is analogous to the synapse weight concept in the neuroscience, is randomly chosen. The first phase of overlap is responsible for determination of the number of active connected synapses. The activity of the synapse is determined by the input to which it is connected, i.e. the high input would make a synapse active, while low input would result in inactive synapse. The connectivity of the synapse in turn is based on the weight of the synapse: only the weights higher than the threshold would make the synapse connected. The optimum threshold depends on the distribution of the data and application of HTM. The optimized threshold can be selected empirically. The example of the empirical selection of the threshold value for pattern recognition application, where the threshold depends on the intensity of the data features, is shown in Ref. [15]. The threshold is selected by testing the HTM method with various threshold values for different databases. The overlap of the individual synapse α is represented by Eq. (1). The value of α would be 1 only if two conditions are satisfied, namely input bit u_i is 1 and weight of the weight of the synapse is higher than the threshold.

$$\alpha_i = \begin{cases} 1 & \text{if } u_i \times w_i \geq \text{threshold} \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

The total overlap of the column that consists of N synapses is given by Eq. (2).

$$\text{Overlap} = \sum_1^N \alpha_i \quad (2)$$

Based on the calculated overlap values the second stage determines the columns that will still be winners after the inhibition. The number of desired winning columns could be controlled with help of parameter k such that the winner is considered that column whose overlap value is greater than the k-th highest overlap value within the inhibition region that is represented in Eq. (3) by γ.

$$\gamma = \text{kmax}(\alpha, k) \quad (3)$$

In the phase of learning the weight values that were higher than γ value are increased by the specified amount, while the other weight decreases its value [3]. Such an update of the weights allows the consideration of the previous state when treating a new input.

TM deals with the time variable of the learning process and could be used to predict what patterns will be followed the given pattern based on the previous observations. Ideally, the learning process occurs in both SP and TM. However, in the SP the learning is based on the connections between the input bits and columns, while in TM learning considers the establishment of the synapses (connections) between the cells of the same region. The TM learns the sequences by considering an active cell and the connection that it formed to the cells that were active in the previous time instant.

The input to the TM are the winning columns from the SP. Like SP, TM consists of three phases. In the first phase, the state of each cell of the columns is checked and the certain cells of the columns are activated. The second phase is responsible for the setting the cells of the columns in predictive state, while the last phase updates the permanence of the synapses [3].

3. Software and hardware realizations of HTM

Despite HTM is classified as a type of cognitive learning algorithm, its fundamental concepts related to neuroscience and, particularly, to the working principles of neocortex make it like deep neural networks, which, in turn, have already shown significant results on a variety of real world problems. Combined with the distinctive features of the algorithm, expressed by its sparsity, hierarchy, and modularity, HTM algorithm attracted interest of various research groups. There is now several research works, which illustrate effectiveness of the algorithm on various recognition tasks. For example, work in Ref. [5] verifies HTM capability on abnormality detection, Refs. [6, 7] present results for pattern and object recognitions, in Ref. [8] HTM was used for object categorizations. Similarly, works in Refs. [9, 10] illustrate suitability of the algorithm for robotics and movement detection, respectively. However, these works were focused on the software realizations of HTM.

Despite it was originally developed as the software algorithm, promising results reported in the works listed above resulted in the rise of interest in a hardware realization of HTM. Limited number of works proposes various designs for the hardware realization of HTM within digital, analog, and mixed-signal domains. The work in Ref. [11] proposed conceptual application-specific integrated circuit (ASIC) design for HTM. The work in Ref. [12] proposed reconfigurable field-programmable gate array (FPGA) implementation. The work in Ref. [13] proposed non-volatile HTM spatial pooler design using flash memories. The work in Ref. [14] proposed mixed-signal design for HTM based on the combination of spin-neuron devices and memristor crossbars. The work in Ref. [15] proposed analog design for HTM utilizing memristor crossbar circuits.

The works in Refs. [13–15] are based on the implementation of HTM using non-volatile memories. This is driven by the fact that, in order to compute large amount of data, HTM should have dense neuronal structures with considerations on area and power efficiency. Based on these parameters, the work in Ref. [16] provides data illustrating superior potential of memristor devices over flash memories conventionally used to implement brain-inspired architectures.

4. Memristor for HTM

Memristor (memory resistor) is a non-linear device firstly proposed by Leon Chua in 1972 and firstly fabricated only in 2003 by HP Labs [17]. The main distinctive feature of the device is its ability to change its state not only according to the current input value, but also according to the history of the inputs. Moreover, additional advantages of memristors include low on-chip area, resulting from fabrication methods differing from methods used for silicon devices, and low power consumption, resulting from absence of leakage currents associated with high power dissipation in CMOS technology [17]. These properties of the device resulted in its applications in brain-inspired architectures. Single memristor may be used to represent synaptic connection (or, simply synapse) with its memristance value representing strength of the connection between pre- and post-synaptic neurons. Possibility to combine synaptic processing as well as memory storage within single device made memristors invaluable in the design of synapses, critical building blocks in architectures based on neuroscientific concepts. Nanoscale devices allow compact storage of many synapses as well as enable parallel processing within architecture.

5. Single synapse circuit realizations using memristor devices

Single synapse is said to be a connection between two communicating neurons. In biology, in simple terms communication occurs when transmitting neuron, i.e. pre-synaptic neuron, sends information via signals to receive neuron, i.e. post-synaptic neuron, through synaptic connection. Because there are a huge number of neuronal interconnections, the receiving signal importance is said to be dependent on the strength (also called as the weight) of synaptic connection. It means that neuronal cell, receiving various signals from the number of pre-synaptic neurons, selects and processes important signal according to the weight of synapse.

It is usually assumed that learning appears when the weight of the synapse increases or decreases, according to the importance of the information received through that particular synaptic connection. Hence, it is also crucial to consider effect of time-variance of the incoming data stream on the strength of synapse.

Based on the neuroscientific concepts, synapse, hence, is required not only to be able to process input value but also to store the weight value, determining its importance. Because the latter attribute is dependent on the time variance of the input data, memristor is attractive device, which can combine neuronal functions, such as memorization and data processing, within single unit. Depending on the input signal type, i.e. either current or voltage mode design, there are various possibilities of establishing synapse circuits using memristor devices and CMOS transistors. In particular, work in Ref. [16] presents basic synapse circuits.

The work in Ref. [18] proposed single synapse circuit with specific intentions to precisely model synapse as it is explained and used within HTM framework. **Figure 4** illustrates the circuit design of the single synapse, which can be divided into four parts: input current mirror, memristor, buffer, and voltage-to-current converting NMOS. First part is responsible for establishment of the pre-synaptic signal. The second part is responsible for storing the weight of the synapse, expressed by the memristance value of the memristor. These two parts are responsible for implementation of overlap phase of HTM, such that the voltage across memristor represents the product of the input signal (expressed by current value) and the weight of the synapse (expressed by the memristance value). The third part of the circuit is buffer and is responsible for activation function establishment. It means that buffer outputs voltage value of 1 V if the overlap value is higher than the threshold and outputs 0 V if it is not. Hence, if the input to that synapse is active connected, then the buffer will output high value, and if it is

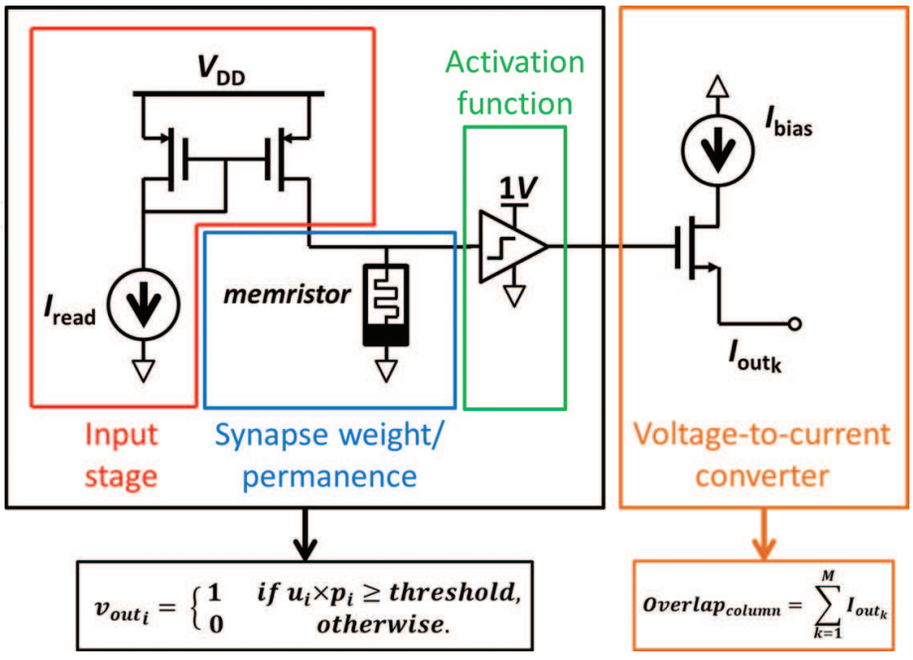

Figure 4. Single synapse circuit design proposed in Ref. [18] designed specifically for HTM architecture.

either not active (input bit value is low) or not connected (weight value is low), then the buffer will output low value. Finally, NMOS transistor is required to convert voltage signal into current signal to sum up all overlap values of individual synapses within single HTM column.

6. HTM circuit realizations using memristor devices

As HTM is a nascent area in brain inspired machine learning algorithms and architectures, a very small number of the HTM circuit configurations have been proposed. There are two major architectures proposed for the HTM hardware implementation with memristive circuits. These are memristor crossbar array-based HTM implementation integrated with CMOS circuits [15] and the HTM implementation using memristive arrays and spin-neuron devices [14]. Both hardware configurations were tested for pattern recognition application.

The memristor-CMOS circuit implementation of HTM has been proposed in Ref. [15]. In this work, the hardware implementation of the HTM SP, applied for face and speech recognition tasks, is presented. The main role of the HTM SP in the proposed system is to extract the most relevant spatial features from the images and remove irrelevant ones for further application for face and speech recognition using template matching method. The feature extraction relies on SDR of the features. The selection of the most relevant features from the sparse data depends on the threshold value shown in Eq. (1). The sparse features greater than

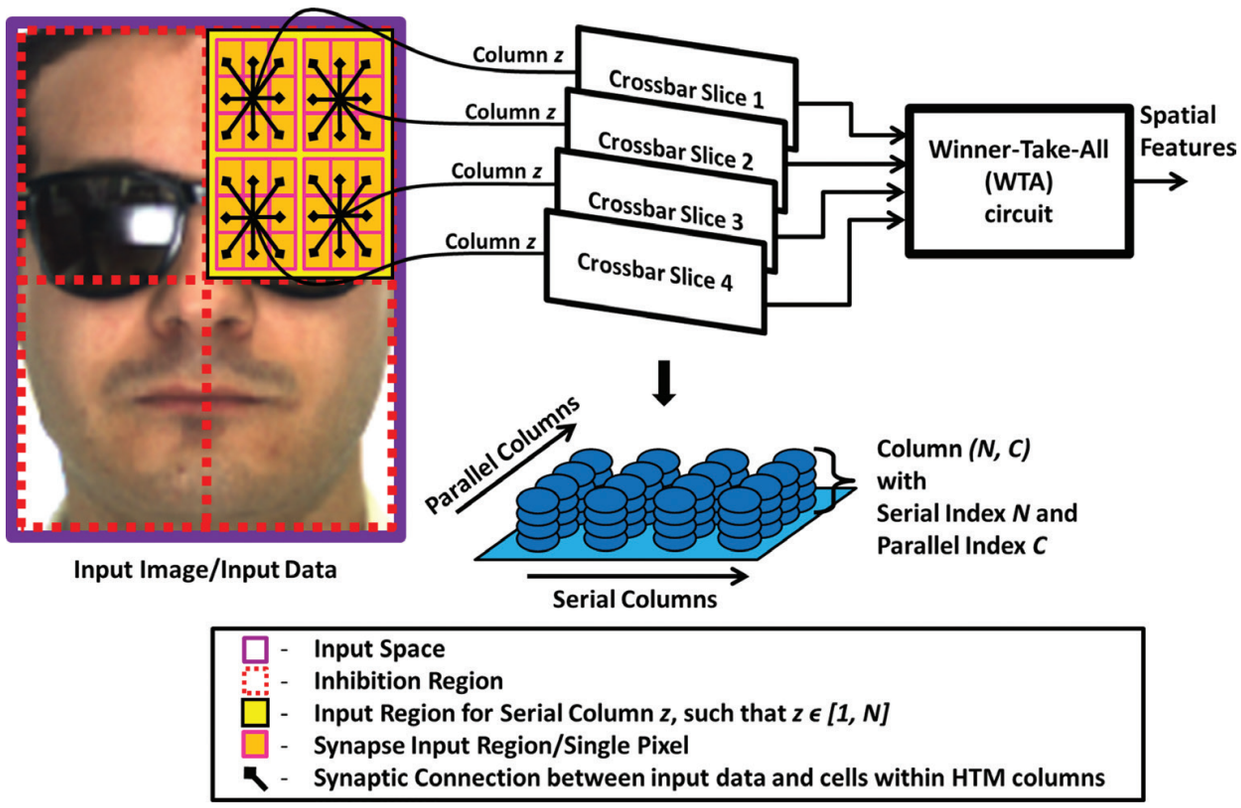

Figure 5. HTM image processing method proposed in Ref. [15].

the threshold are stored in the memory. The example of the input image is shown in **Figure 5**. The image is divided into blocks with a certain number of pixels, which are the inputs to the crossbar slice circuits.

The number of crossbar slices corresponds to the number of the image blocks. Each crossbar slice refers to a single column in the set of serial columns in HTM. The number of synapses in the column corresponds to the number of synapses or pixels in the image blocks. Several image blocks form a single inhibition block. The number of inhibition blocks equals to the number of parallel columns in the HTM. Each inhibition region consists of several serial columns. The set of pixels in the serial column correspond to a single crossbar slice. The sum of the pixels within each crossbar slice is calculated. The outputs from the crossbar slices are fetched to the winner take all (WTA) circuit to produce the final set of the sparse image features. The highest value of crossbar slice outputs is selected by the WTA. Then, the inhibition region is binarized. The inhibition region component corresponding to the highest value of the crossbar slices is represented as 1, and the other components are set to 0. This binarized inhibition regions represent the final set of the sparse image features.

The overall architecture of the HTM crossbar slice circuit consists of the memristor crossbar, read and write circuits, synapse overlap calculation circuit and column overlap calculation circuit, shown in **Figure 6**. The memristors in the memristive crossbar array are used to represent the synapse weight and strength of the synaptic connection. The memristors in read/write part of the circuit implement the expected ideal permanence and determine the final connection of the column, which is either active or inactive. The memristors in the crossbar array are preprogramed

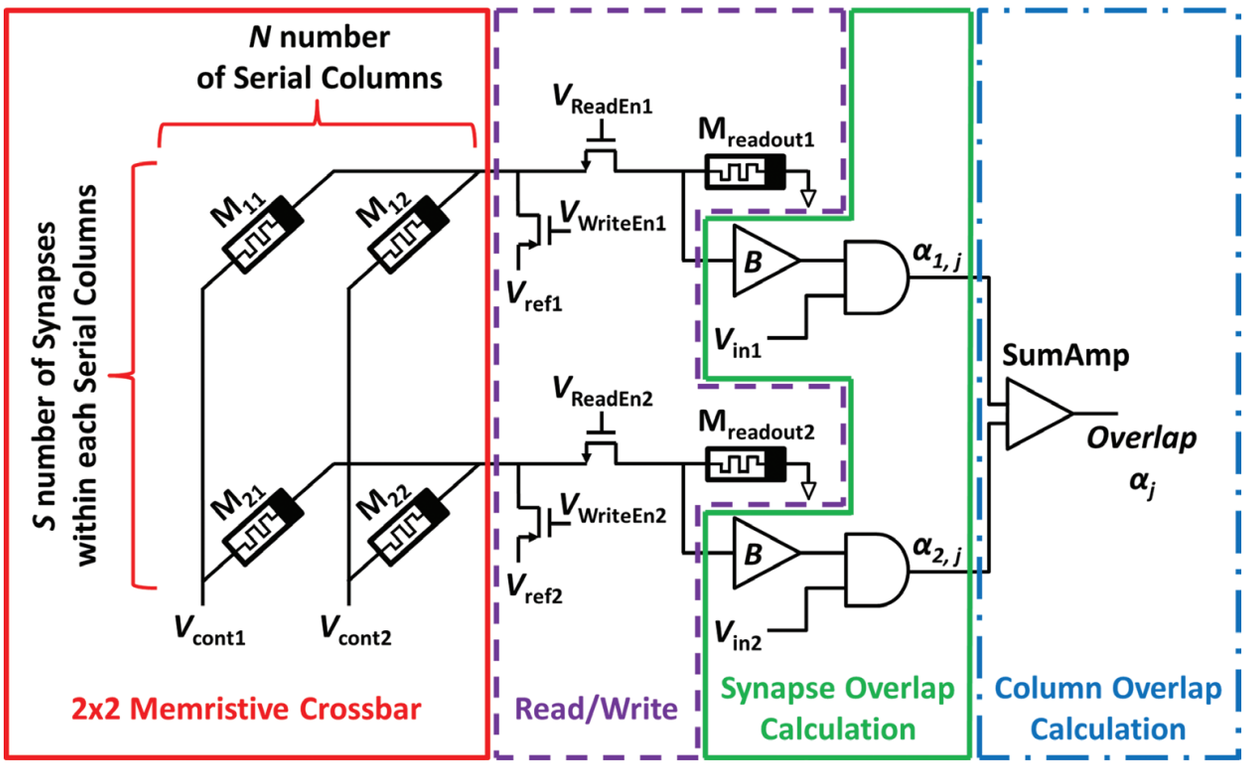

Figure 6. HTM circuit implementation [15].

in the initial stage and later are updated only during the learning phase. When the synaptic connection is determined, the connection value (output from the read/write circuit) is fetched to the AND gate through the buffer, which performs thresholding operation, shown in synapse overlap calculation stage in **Figure 6**. The buffer converts the signal from read/write stage to the binary connectivity value, which is either 0 or 1. The single synapse overlap calculation is made by the AND gate, where one input is the binary synapse connectivity value and the other is the input signal from the processed pattern. The column overlap calculation is made by the summing amplifier that is used for the summation of all single synapses.

After the overlap phase, the inhibition phase is executed, where the overlaps of the parallel columns are compared using WTA circuit, illustrated in **Figure 7**. The overlap values obtained from the summing amplifier are fetched to the WTA circuit and the highest value is selected and set to 1 with the help of bias current source and the comparators. The other values from the inhibition region are set to 0. The output of the WTA circuit is index of the winning columns, which are used for classification process. The circuit implementation results show the power consumption of 31.567 uW and on-chip area of 2.735 um^2 to process 2 column block with 2 synapses. The area and power required to process an individual synapse overlap are 1.37 um^2 and 31.56 uW, respectively. For the column overlap calculation the area of 4.325 um^2 and the power of 160 uW are required.

After the HTM SP processing, the features are compared with the stored training patterns, and the classification of the input pattern is executed using a memristive pattern matcher shown in **Figure 8**. Memristive pattern matcher is based on memristive the XOR gate, which, in turn, consists of the memristive NOR gate and the CMOS inverter. The output of the memristive pattern matcher is either 1 for the detected match or 0 for the mismatch.

The other research work representing memristive circuits-based HTM implementation is presented in Ref. [14]. In this work, memristive crossbar arrays are combined with spin-neuron devices. The system is tested on object and handwritten text recognition. It is assumed that

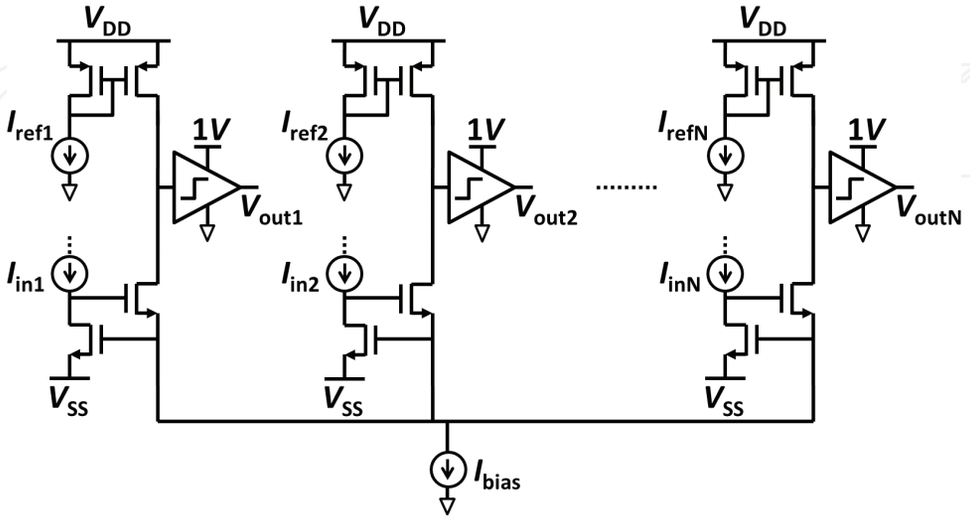

Figure 7. Winner take all circuit as illustrated in Ref. [15].

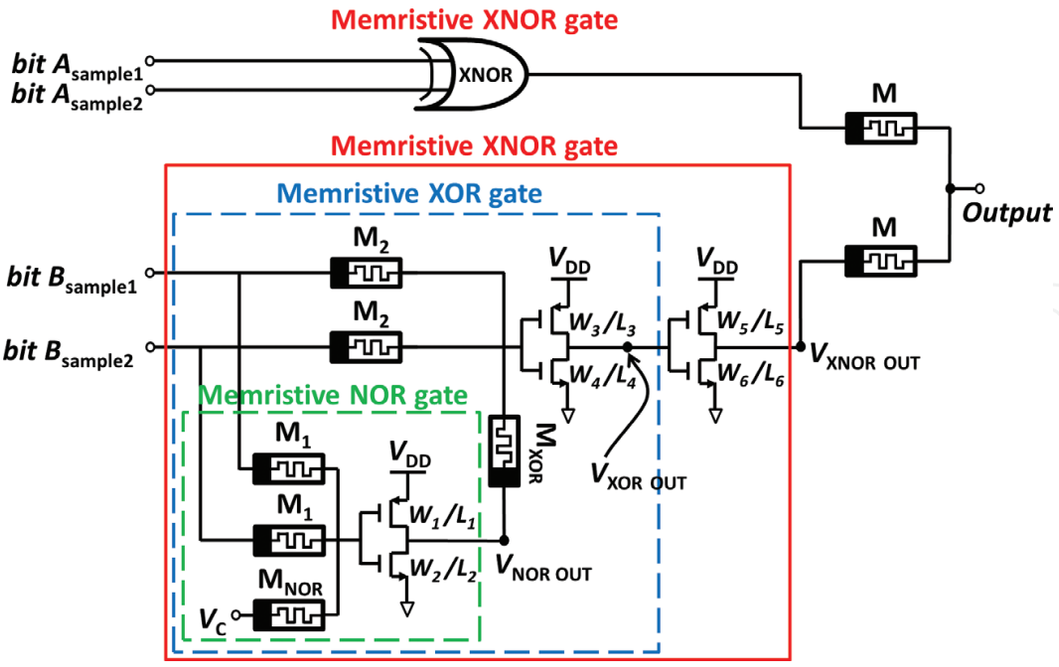

Figure 8. Memristive pattern matcher presented in Ref. [15].

the training of the system is executed by the software and the hardware part of HTM is used for inference (testing) stage, where pattern classification is performed.

The overall HTM hardware configuration is illustrated in **Figure 9**. Image processing is performed in a hierarchical manner. The processed image is divided into 16 patches of the same size, which are fetched into separate HTM nodes in Level 1. Therefore, there are 16 HTM nodes in Level 1. The number of inputs to each Level 1 HTM node equals to the number of pixels in the patch. Level 1 nodes produce 16 outputs, which are divided into 4 groups of 4 Level 1 outputs. Each group of these outputs is fetched into particular Level 2 HTM node. Level 2 contains 4 HTM nodes producing 4 separate outputs. Level 2 outputs are input to Level 3 HTM node, which calculates the final output value.

The overall architecture of the implemented HTM node consists of resistive crossbar network (RCN), successive approximation register analog to digital converter (SAR ADC) and winner take all (WTA) circuit. RCN is used to calculate dot products (DP) in the HTM SP part and the HTM TM part. RCN enables a parallel processing of all image pixels from a single image patch. The digital input pattern from the image patch is fetched into the HTM SP, where it is converted to analog form for the RCN processing. The RCN circuit performs the DP calculation producing an analog output vector. This vector is detected and converted to the digital form with the spin-neuron-based SAR ADC, which forms the output vector of the HTM SP. This output vector is sent to the HTM TM circuit, where it is converted into the analog form again followed by the DP calculation and SAR ADC processing, which forms the output vector of the HTM TM. The HTM TM output vector is fetched to WTA circuit, which identifies the index of the winner from temporal group.

The circuit example of RCN with a single digital input and three spin-neurons is shown in **Figure 10**. To convert digital RCN inputs to the analog form, a deep triode current source

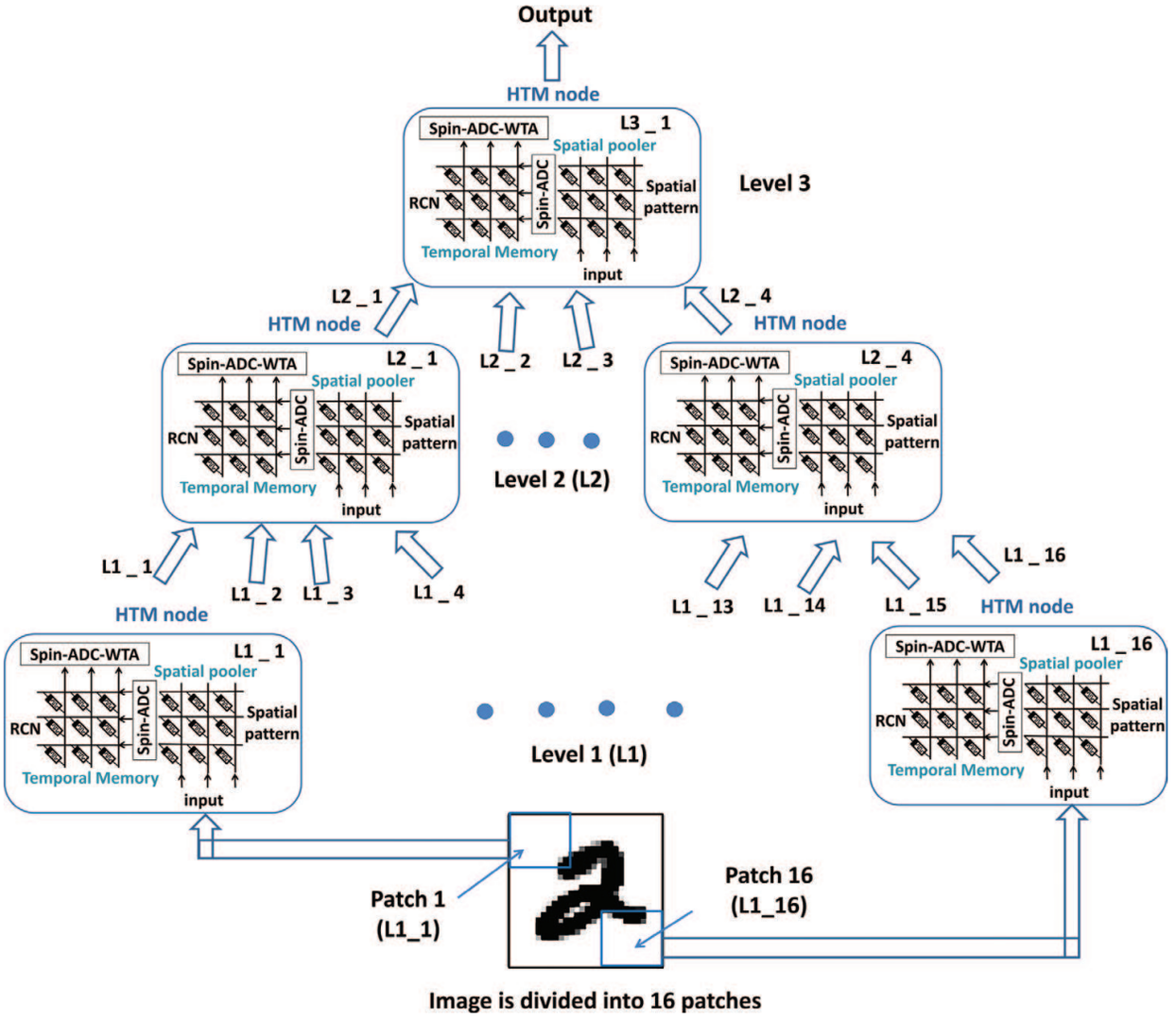

Figure 9. HTM hardware configuration proposed in Ref. [14].

digital to analog converter (DTCS DAC) based on the switching circuit is used. Next, the analog DTCS ADC outputs are fetched to RCN circuit.

The example of 3 × 3 RCN circuit is shown in **Figure 11**. The RCN has a configuration of the memristive array. Such configuration enables calculation of the DP required for the SP as well as for the TM. Therefore, the pixels of the input image matrix are processed in parallel. The input voltage V_i in a horizontal upper row is multiplied by each preprogramed memristor conductance value within the raw g_{ij}. Then, the correlation between the stored patterns and the input signals is calculated as an output current using Eq. (4).

$$I_j = \sum_i V_i * g_{ij} \tag{4}$$

Next, the output current of each RCN column is detected and converted to the digital value using SAR ADC, which is based on spin neuron devices (shown in **Figure 12**). The performance of the spin neuron device is akin to the current mode comparator functionality. The

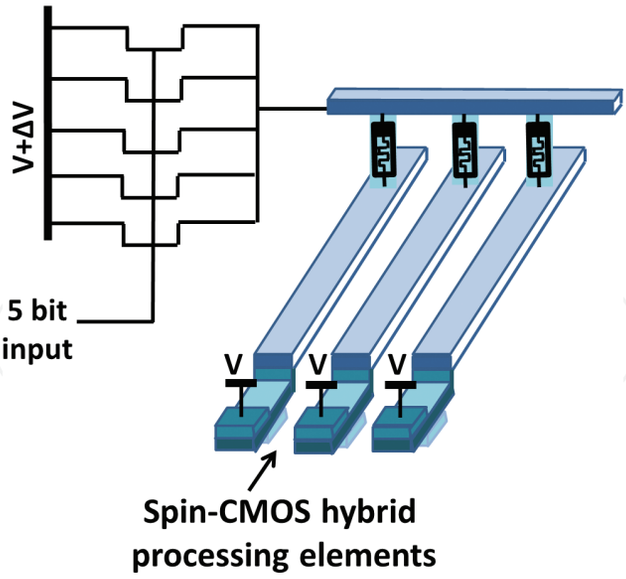

Figure 10. RCN with a single digital input and three spin-neurons [14].

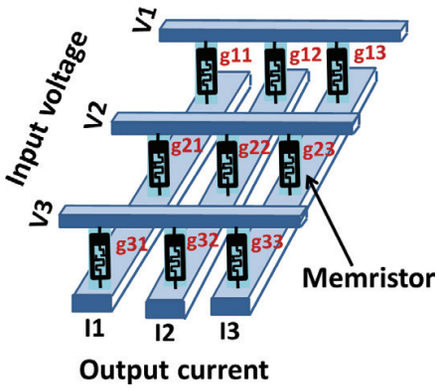

Figure 11. RCN array [14].

spin-neuron may be switched with the particular current flowing through it; therefore it is suitable for the SAR ADC design. In the spin neuron SAR ADC, the DTCS DAC converts the digital value stored in the approximation register to the form of the analog current. Then, this current is compared with the RCN output current produced by the spin-neuron. Finally, a special latch is used to detect output stage. Having a advantage of low power consumption and high resolution, the spin neuron-based SAR ADC enables the conversion of the analog currents from RCN to the digital HTM SP and HTM TM outputs denoted as the input densities over the spatial patterns (HTM SP outputs) and the input densities over the temporal groups (HTM TM outputs) [14]. Finally, the WTA circuit determines the winner of each HTM block. If the HTM block is at the highest hierarchy level (output node), the WTA identifies the class index of an input pattern. Otherwise, if it is located at the non-output node, the WTA determines the winning index of a particular temporal group.

Table 1 shows the comparison of two main implementations of HTM discussed in this Chapter.

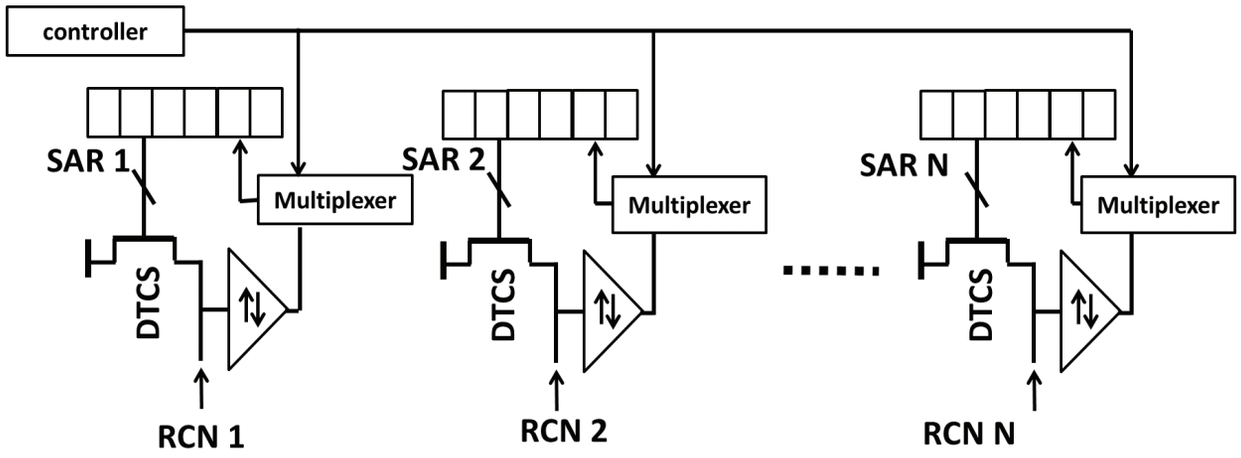

Figure 12. Successive approximation register analog to digital converter (SAR ADC) [14].

	HTM implementation presented in Ref. [14]	**HTM implementation presented in Ref. [15]**
Algorithm	SP+TM	SP
Hardware implementation	Analog/digital	Purely analog
Technology	Memristors (RCN), 20 × 2 nm spin neuron device (SAR-ADC), spin-CMOS hybrid processing elements based on domain wall neuron (DWN)	50 × 50 nm memristive devices, 180 nm CMOS technology
Architecture	Crossbar	Neuron units
Application	Hand-written digit recognition	Face and speech recognition

Table 1. The comparison of the existing hardware implementations of HTM.

7. Summary

HTM is a type of cortical learning algorithm that aims to resemble the functionalities of the neocortex. The great potential of the HTM for applications related to classification and prediction making pushed for consideration of hardware realizations of the algorithm. The objective of this chapter was to overview the major concepts of HTM and discuss the most recent implementations of HTM in hardware. Currently, there are different hardware implementations of HTM, including the designs based on FPGA and ASIC, that could be found in the literature. However, the discussion in this chapter is focused around the memristor based realizations of HTM. Despite this, for comprehensiveness of the chapter, the authors summarize the general state of the HTM in terms of other hardware implementations [11–13] and provide an interested reader with the references for further reading.

The properties of memristors connected with the small feature size and negligible leakage current of the devices make them very attractive for large-scale circuit designs that reduces the overall on-chip area and power consumption. In addition to that, ability of the memristor

to remember its state and change it based on history of applied voltages makes it suitable for neuromorphic designs. Thus, the nanoscale memristive devices could be used to mimic the synaptic connection of the neuron.

After giving an introduction to the HTM concepts and the memritive devices the chapter presents designs of two recent HTM hardware implementations. One of the designs is based on the memristive arrays and spin-neuron devices [14], while the second one integrates the memristor crossbar array and memristor-CMOS circuits [15].

Author details

Alex James*, Timur Ibrayev, Olga Krestinskaya and Irina Dolzhikova

*Address all correspondence to: apj@ieee.org

School of Engineering, Nazarbayev University, Astana,Kazakhstan

References

[1] Hawkins J, Blakeslee S. On Intelligence. New York St. Martin's Griffin: Macmillan; 2004

[2] Hawkins J, George D. Hierarchical Temporal Memory: Concepts, Theory and Terminology. Technical report. Numenta; 2006

[3] Hawkins J., Ahmad S., Dubinsky D. Hierarchical Temporal Memory Including HTM Cortical Learning Algorithms. Technical report, Numenta, Inc, Palto Alto. 2010

[4] Mnatzaganian J, Fokoué E, Kudithipudi D. A Mathematical Formalization of Hierarchical Temporal Memory's Spatial Pooler. Frontiers in Robotics and AI. 2016; 3:81. DOI: 10.3389/frobt.2016.00081

[5] Wu J, Zeng W, Chen Z, Tang XF. Hierarchical temporal memory method for time-series-based anomaly detection. In: 2016 IEEE 16th International Conference on Data Mining Workshops (ICDMW); Barcelona, Spain; 2016. pp. 1167-1172. DOI: 10.1109/ICDMW.2016.0168

[6] Ramli I, Ortega-Sanchez C. Pattern recognition using hierarchical concatenation. In: 2015 International Conference on Computer, Control, Informatics and its Applications (IC3INA); Bandung; 2015. pp. 109-113. DOI: 10.1109/IC3INA.2015.7377756

[7] Farahmand N, Dezfoulian MH, GhiasiRad H, Mokhtari A, Nouri A. Online temporal pattern learning. In: 2009 International Joint Conference on Neural Networks; Atlanta, GA; 2009. pp. 797-802. DOI: 10.1109/IJCNN.2009.5178844

[8] Csapo AB, Baranyi P, Tikk D. Object categorization using VFA-generated nodemaps and hierarchical temporal memories. In: 2007 IEEE International Conference on Computational Cybernetics; Gammarth; 2007. pp. 257-262. DOI: 10.1109/ICCCYB.2007.4402045

[9] Seok K.H., Kim Y.S. A new robot motion authoring method using HTM. In: Control, Automation and Systems, 2008. ICCAS 2008. International Conference on; IEEE; 2008. p. 2058-2061

[10] Zhang S, Ang MH, Xiao W, Tham CK. Detection of activities for daily life surveillance: Eating and drinking. In: HealthCom 2008 - 10th International Conference on e-health Networking, Applications and Services; Singapore; 2008. pp. 171-176. DOI: 10.1109/HEALTH.2008.4600131

[11] Melis WJC, Chizuwa S, Kameyama M. Evaluation of the hierarchical temporal memory as soft computing platform and its VLSI architecture. In: 2009 39th International Symposium on Multiple-Valued Logic; Naha, Okinawa; 2009. pp. 233-238. DOI: 10.1109/ISMVL.2009.11

[12] Zyarah AM. Design and analysis of a reconfigurable hierarchical temporal memory architecture [thesis]. 1 Lomb Memorial Dr, Rochester, NY 14623: Rochester Institute of Technology; 2015

[13] Streat L, Kudithipudi D, Gomez K. Non-volatile Hierarchical Temporal Memory: Hardware for Spatial Pooling. arXiv preprint arXiv:1611.02792; 2016

[14] Fan D, Sharad M, Sengupta A, Roy K. Hierarchical temporal memory based on spin-neurons and resistive memory for energy-efficient brain-inspired computing. IEEE Transactions on Neural Networks and Learning Systems. Sept. 2016;**27**(9):1907-1919. DOI: 10.1109/TNNLS.2015.2462731

[15] James AP, Fedorova I, Ibrayev T, Kudithipudi D. HTM spatial pooler with memristor crossbar circuits for sparse biometric recognition. IEEE Transactions on Biomedical Circuits and Systems. 2017;**PP**(99):1-12. DOI: 10.1109/TBCAS.2016.2641983

[16] Merkel C, and Kudithipudi D. Neuromemristive systems: A circuit design perspective. In: Advances in Neuromorphic Hardware Exploiting Emerging Nanoscale Devices. Springer India; 2017. pp. 45-64

[17] Strukov DB, Snider GS, Stewart DR, Williams RS. The missing memristor found. Nature. 2008;**453**(7191):80-83

[18] Ibrayev T, James AP, Merkel C, Kudithipudi D. A design of HTM spatial pooler for face recognition using memristor-CMOS hybrid circuits. In: Circuits and Systems (ISCAS), 2016 IEEE International Symposium on; IEEE; 2016. pp. 1254-1257

IntechOpen