Neuromorphic System for Spatial and Temporal Information Processing

Abdullah M. Zyarah, Kevin Gomez, and Dhireesha Kudithipudi, *Senior Member, IEEE*

Abstract—Neuromorphic systems that learn and predict from streaming inputs hold significant promise in pervasive edge computing and its applications. In this article, a neuromorphic system that processes spatio-temporal information on the edge is proposed. Algorithmically, the system is based on hierarchical temporal memory that inherently offers online learning, resiliency, and fault tolerance. Architecturally, it is a full custom mixed-signal design with an underlying digital communication scheme and analog computational modules. Therefore, the proposed system features reconfigurability, real-time processing, low power consumption, and low-latency processing. The proposed architecture is benchmarked to predict on real-world streaming data. The network's mean absolute percentage error on the mixed-signal system is 1.129 X lower compared to its baseline algorithm model. This reduction can be attributed to device non-idealities and probabilistic formation of synaptic connections. We demonstrate that the combined effect of Hebbian learning and network sparsity also plays a major role in extending the overall network lifespan. We also illustrate that the system offers 3.46 X reduction in latency and 77.02 X reduction in power consumption when compared to a custom CMOS digital design implemented at the same technology node. By employing specific low power techniques, such as clock gating, we observe 161.37 X reduction in power consumption.

Index Terms—Neuromorphic computing, hierarchical temporal memory, synthetic synapses representation, plasticity, neocortex

1 Introduction

Over the course of the last decade, there has been a profound shift in artificial intelligence (AI) research, where biologically inspired computing systems are being actively studied to address the demand for energy-efficient intelligent devices. Biologically inspired systems, such as hierarchical temporal memory (HTM) [1], [2], have demonstrated strong capability in processing spatial and temporal information with a high degree of plasticity while learning models of the world. HTM also exhibits natural compatibility for continuous online learning [3], noise and fault tolerance [4], and low power consumption achieved through sparse neuronal activity [5], [6]. These properties make the algorithm attractive for a wide range of applications such as visual object recognition and classification [7], [8], prediction of data streams [9], natural language processing and anomaly detection [10]. Despite the fact that HTM is an attractive algorithm, it demands high computational power that cannot be fulfilled by conventional von Neumann architectures. This is because the innate HTM architecture, which is composed of thousands of neuronal circuits, requires a high-level parallelism in information processing. One may map the HTM algorithm onto a GPU. A GPU can provide the necessary parallelism, but it fails to provide satisfactory performance and demands a large power budget [11]. To this end, several research groups have attempted to develop specialized custom hardware designs to run the HTM algorithm efficiently and affordably [12]. While some of the previous designs focused only on the spatial aspect of the HTM [13], [14], [15], other endeavors incorporated both the spatial and temporal models in the same design. For instance, in 2015, Zyarah *et al.* implemented the HTM algorithm including the spatial and temporal aspects [16]. The implemented network incorporates 100 mini-columns with 3 cells each, and is verified for image classification and sequence prediction. Furthermore, it supports synthetic synapses, which are realized with distributed memory blocks, to enable synaptic pathway dynamics. The authors also optimized their design further in [17]. Weifu Li *et al.* [18], proposed a full architecture of the HTM algorithm in 2016. The proposed architecture is composed of 400 mini-columns (2 cells in each mini-column) connected in point-to-point format to the HTM input space, which eventually causes the mini-column to be in an active mode even when there is insignificant activity (noise) in the input space. When it comes to HTM memristor-based analog and mixed-signal designs, in 2016, Fan *et al.* implemented the first generation of HTM, HTM-Zeta. The authors proposed RCN (resistive-crossbar networks) pattern matching modules with core processing unit named, spin-neurons [19]. The network operation is verified for image classification in an offline fashion as the proposed design does not support online learning. In 2018, Krestinskaya *et al.* presented the full analog design of the HTM, but the temporal aspect of the implementation does not match that in the HTM sequence memory as it depends on the class map concept which matches the stored patterns with the test ones (unseen input samples) [20]. To the best of our knowledge, there is no full custom mixed-signal design of the HTM algorithm in literature with underlying

- *Abdullah M. Zyarah is with the Neuromorphic AI Lab, Department of Computer Engineering, Rochester Institute of Technology, Rochester, NY 14623. E-mail: amz6011@rit.edu.*
- *Kevin Gomez is with the Seagate Research Group, Seagate Technology, MN 55379. E-mail: kevin.gomez@seagate.com.*
- *Dhireesha Kudithipudi is with the Neuromorphic AI Lab, Department of Electrical and Computer Engineering, University of Texas at San Antonio, San Antonio, TX 78249. E-mail: dk@utsa.edu.*

Manuscript received 18 Dec. 2019; revised 22 May 2020; accepted 1 June 2020. Date of publication 5 June 2020; date of current version 9 July 2020.
(Corresponding author: Abdullah M. Zyarah.)
Recommended for acceptance by Karthik V Swaminathan.
Digital Object Identifier no. 10.1109/TC.2020.3000183

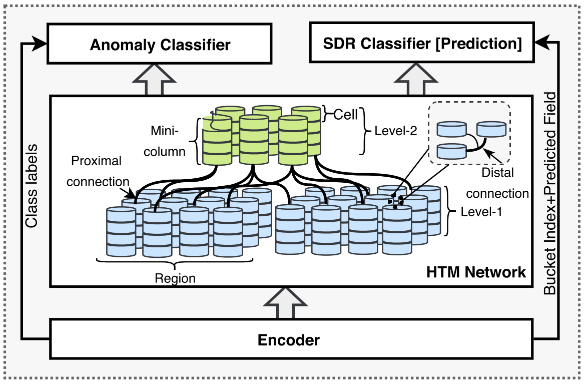

Fig. 1. High-level architecture of the HTM system with three core units: data encoder, HTM network, and classifiers. The encoder transforms the input data into binary representations. The HTM algorithm learns spatial information and captures temporal transitions, while the classifiers map the HTM output to the corresponding class labels and identify anomalies.

digital communication scheme and analog computational modules. Such a design should include the necessary reconfigurability, low energy-delay product, and a robust communication scheme, in one platform. It is important to mention here that such architectures have been explored in the context of spiking neural networks (SNNs) [21], where the communication scheme is realized with address event representation (AER), developed by Mahowald in 1992 [22]. AER takes advantage of sparse neuronal activity and high-bandwidth VLSI to enable time-multiplexed communication. Hence, it reduces the number of connections between sending and receiving neuronal arrays from n to $\log_2 n$ [23]. It turns out that AER is considered as an effective approach for point-to-point connections, but not for complex networks with sparse connections. The complex network connectivity is solved through the enhanced AER proposed by Goldberg *et al.* [24]. The enhanced AER uses look-up tables (LUTs) to describe the connectivity network between two sets of neuronal arrays. The LUT contains the sender address, destination address, and the probability of connectivity. Thus, complex networks even sparse ones can easily be implemented. However, the enhanced AER demands a large amount of memory and this makes it unsuitable for power and area constrained devices. Therefore, this paper also proposes a synthetic synapses representation (SSA) communication scheme, which leverages the linear feedback shift registers (LFSR)s to describe the sparse connections among neurons. Using the LFSRs eliminates the need for memory-based address description as the addresses between neurons are generated rather than stored. This results in a considerable reduction in the network area and power consumption.

Specific key contributions of this paper are as follows:

- Developing a memristor-based mixed-signal neuromorphic system of the HTM network including both the spatial pooler and temporal memory.
- Synthetic synapses representation (SSR) communication scheme is proposed to virtually formulate and prune the physical synaptic connections in the HTM network.
- System-level analysis of the performance, lifespan, area, and power consumption with respect to a CMOS only implementation is performed.

2 Hierarchical Temporal Memory

HTM is a biomimetic algorithm that aims to develop a computational framework capturing the structure and the algorithmic properties of the human neocortex. Structurally, the algorithm is composed of hierarchical ascending layers of cellular regions that enable the network to capture spatial and temporal information, shown in Fig. 1. Each region in the HTM is composed of building blocks, namely cells, which are arranged in columns to model biological mini-columns. The cell in HTM is just an abstract model of the excitatory pyramidal neurons. As pyramidal neurons, each cell has hundreds of synaptic connections grouped into three integration zones (or segments): proximal, distal, and apical[1] [4], [25]. The proximal segment is dedicated to receive feed-forward input i.e., observe the cellular activities in the lower layers in the hierarchy, or sensory input. Typically, activities detected on proximal segments leads to generation of neuronal action potential. The distal and apical segments, on the other hand, are dedicated to observe the cellular activities of the neighboring cells in the same region (contextual input) and higher levels in the hierarchy (feedback input), respectively. Unlike the proximal segment, the cellular activities detected by distal and apical segments lead to NMDA spikes [26]. The NMDA spikes slightly depolarize the cell without generating an action potential, giving the cell a competitive advantage in responding to future input representations [27].

Fig. 1 shows a high-level diagram of the HTM network equipped with a data encoder and multiple classifiers. The encoder transforms sensory information into binary representations, while the classifiers map the HTM output to the corresponding class labels (SDR classifier) and identify anomalies (anomaly classifier). The mixed-signal design of the SDR classifier has been developed in our previous work [6]. Thus, this work will emphasize the design and implementation of a single HTM region,[2] which is equivalent to realizing the primary sensory region in the supragranular layers of the neocortex. Given an HTM region, there are two core operations which capture the spatial and temporal information of a given input, namely the spatial pooler and temporal memory, which are discussed in the following subsections:

2.1 Spatial Pooler

The spatial pooler in the HTM is responsible for extracting and learning the spatial patterns of the sequential data. Typically, the spatial pooler models an encoded sensory input, generated by the encoder, using a population of active and inactive mini-columns chosen through a combination of competitive Hebbian learning rules and homeostasis [27]. Typically, the number of active mini-columns is limited to (2-4) percent of the total mini-columns in a given HTM region, resulting in so-called sparse distributed representation (SDR). The SDR in HTM defines the underlying data structure and enables the crucial features of the algorithm such as distinguishing the common features between inputs [28], learning

1. A cell in HTM typically has one proximal segment (shared with other cells of the same mini-column) and multiple distal and apical segments.

2. The hierarchical structure of the HTM network has not been thoroughly studied yet.

sequences, and making simultaneous predictions [29]. However, selecting the active and inactive mini-columns is determined according to the spatial response of the individual mini-columns to an input. Recall, each mini-column in the HTM observes the pattern activities in the input space using a set of proximal connections. Having a reasonable number of active proximal synapses connected to active bits (namely overlap score) initiates an action potential that enables a mini-column to compete with its neighbour for input representation. By using the k-winner-take-all (k-WTA) computation principle, the top (2-4) percent mini-columns with the highest overlap scores are activated (become winners) and inhibit their neighbors. The output of the spatial pooler is a binary vector, which represents the joint activity of all mini-columns in the HTM region in response to the current input. The spatial pooler operation can be divided into three distinct phases: initialization, overlap and inhibition, and learning, discussed in our previous work [6] and briefly described below.

In the initialization phase (Algorithm 1, lines 2-5), which occurs only once, the mini-columns' connections to the input space, synapses' permanences, and boosting factors, are initialized. Let S_p be an $n_c \times n_x$ array holding the proximal synaptic connections between n_c mini-columns and n_x-dimensional input space. Similarly, let ρ_p be an $n_c \times n_x$ array that defines the permanence of corresponding potential synapses in S_p. Given the jth mini-column, its maximum number of potential synapses (n_{sp}) is defined by the non-zero elements in $\vec{s_p}$ ($\vec{s_p}$ is a row vector in S_p) whose indices are generated by a pseudo-random number generator, and their permanence values are uniformly initialized at random between '0' and '1'. Initializing the synapses is followed by setting the boosting factor of the individual mini-columns to '1'. After the initialization, the overlap and inhibition phase (lines 8-11) starts in which the feed-forward input is collectively represented by a set of active mini-columns (winning mini-columns). Selecting the active mini-columns is done after counting mini-columns' active synapses that are associated with active bits in the input space, i.e., overlap scores (α). Mathematically, this is achieved by performing a dot product operation between the feed-forward input vector ($\vec{x}^t$) at time t and the active synapses array, as in line 9, where the active synapses array is the result of an element-wise multiplication (denoted as $\odot$) between S_p and $\bar{\rho}_p$. $\vec{b}^t$, here, denotes the boosting factor that regulates mini-column activities. $\bar{\rho}_p$ is a permanence binary array to indicate the status of each potential synapse, where '1' indicates a connected synapse and '0' an unconnected synapse. Upon the completion of computing the overlap scores, each mini-column overlap score gets evaluated by comparing it to a threshold, α_{th} (line 10). The resulting vector ($e\vec{\alpha}^t$) is an indicator vector representing the nominated mini-columns with high overlap scores. Given an inhibition radius defined by ξ, based on the mini-column overlap scores and desired level of sparsity (η), n_w number of mini-columns will be selected to represent the input, as shown in line 11. After determining the winning mini-columns in $\vec{\Lambda}^t$, the learning phase (lines 13-16) starts to update the permanence values of the winning mini-columns' synapses. The synapses' permanences are updated according to Hebbian rule [30]. The rule implies that the synapses connected to active bits must be strengthened, increasing their permanence by P_p^+, while those connected to inactive bits will be weakened, decreasing their permanence by P_p^-, as in line 14, where $\Delta\rho_p$ is the change in the permanence array for all mini-columns given an input $\vec{x}^t$, and λ denotes the sum of P_p^+ and P_p^-. After adjusting the synapse's permanence, the boosting factor of each mini-column is updated according to the mini-column's time-averaged activity level ($\bar{a}^t$) and its activity level with respect to its neighbor ($<\bar{a}^t>$) [27].

Algorithm 1. HTM-Spatial Pooling

```
   Input: $\vec{x}^t \in \mathbb{R}^{n_x}_{\{0,1\}}$, where $\vec{x}^t \subset X^t$ and $X^t \in \mathbb{R}^{n_x \times n_n}_{\{0,1\}}$;
   Output: $\vec{\Lambda}^t \in \mathbb{R}^{n_c}_{\{0,1\}}$        /* $n_c$: Number of mini-columns */
 1 // Initialization:                 /* $n_x$: Input vector length */
 2 $S_{ind} \sim$ rand.pseudo, where $S_{ind} \in \mathbb{N}^{n_c \times n_{sp}}_{\{1,n_x\}}$ ;
 3 $S_p[S_{ind}] \leftarrow 1$, where $S$ and $\rho \in \mathbb{R}^{n_c \times n_x}$ ;
 4 $\rho_p[S_{ind}] \sim$ rand.uniform[0, 1];
 5 $\vec{b}^t \in \mathbb{R}^{n_c}$, where $\forall\ b^t[j] = 1$;
 6 repeat
 7     // Overlap and Inhibition:
 8     $\bar{\rho}_p \leftarrow \mathrm{I}(\rho_p \geq P_{th})$ ;
 9     $\vec{\alpha}^t \leftarrow \vec{b}^t \odot \left[(S_p \odot \bar{\rho}_p) \cdot \vec{x}^t\right]$ ;
10     $e\vec{\alpha}^t \leftarrow \mathrm{I}(\vec{\alpha}^t \geq \alpha_{th})$ ;
11     $\vec{\Lambda}^t \leftarrow kmax(e\vec{\alpha}^t, \eta, \xi)$;        /* $kmax$: $k$-WTA function */
12     // Learning:
13     if $Learning$ == '$Enable$' then
14         $\Delta\rho_p \leftarrow \vec{\Lambda}^t.transpose \odot (S_p \odot \bar{\rho}_p) \odot (\lambda\vec{x}^t - P_p^-)$;
15         $\vec{b}^t \leftarrow e^{-\gamma(\bar{a}^t - <a^t>)}$ ;
16     end
17 until $t > n_n$
```

2.2 Temporal Memory

The temporal memory in the HTM is mainly dedicated to learn time-based sequences and to make predictions. The temporal memory operates at the cells level, specifically, the cells of the winning mini-columns. When a mini-column becomes active, at least one of its cells is selected to be active to represent the input contextually. This usually depends on whether the cells within the winning mini-columns are predicting the incoming input. If a winning mini-column has a predictive cell, that cell becomes active and inhibits other cells within the same mini-columns from being active. Otherwise, the joint activation of all cells within the mini-column represents the input and this is known as massive neurons firing or bursting. However, once a cell is activated, it forms lateral connections with the cells that were active in the previous time step. Patterns recognized by lateral connections lead to a slight depolarization of the cell soma (predictive state), subsequently predicting the upcoming events. Typically, the lateral connections are grouped into distal segments. A cell in HTM can have more than one distal segment and this grants the cells the capability to predict more unique temporal patterns. The operation of the temporal memory can be divided into three phases: mini-columns evaluation, prediction, and learning phase, described in Algorithm 2.[3]

During the mini-columns evaluation phase (Algorithm 2, Line 4-15), the active cells within the winning mini-columns are selected to represent the input within its context. Let n_m

3. Forming and pruning lateral connections are not discussed in the algorithm to avoid complexity.

be the number of cells in each mini-column, and $A^t \in R^{n_m \times n_c}_{\{0,1\}}$ is a binary array that represents the region cells' activity, where '1' indicates an active cell and '0' is inactive. Similarly, let π^t be also a binary array that has the same size of A, and the active bits in π refers to the predictive cells. An ith cell within the jth mini-column is set to be active if $\vec{\Lambda}^t_j = 1$ and the cell was in the predictive state in the previous time step i.e., $\pi^{t-1}_{ij} = 1$. Otherwise, bursting (all cells within the jth mini-column are set to be active) will take place.

Algorithm 2. HTM-Temporal Memory

```
   Input: $\vec{\Lambda}^t \in \mathbb{R}^{n_c}_{\{0,1\}}$              /* $n_c$: Number of columns */
   Output: $A^t \in \mathbb{R}^{n_m \times n_c}_{\{0,1\}}$              /* $n_m$: Number of cells */
 1 $zeros\_cnt = 0$;
 2 repeat
 3     # Phase-1: Mini-columns evaluation:
 4     for $j \leftarrow 1$ to $n_c$ do
 5         if $\vec{\Lambda}^t[j] == 1$ then
 6             for $i \leftarrow 1$ to $n_m$ do
 7                 if $\pi^{t-1}[i,j] == 1$ then
 8                     $A^t[i,j] \leftarrow 1$;
 9                 else
10                     $zeros\_cnt \leftarrow zeros\_cnt + 1$;
11             end
12             if $zeros\_cnt == n_m$ then
13                 $A^t[i,j] \leftarrow 1, \forall i$;
14             $zeros\_cnt = 0$;
15     end
16     # Phase-2: Prediction:
17     for $j \leftarrow 1$ to $n_c$ do
18         for $i \leftarrow 1$ to $n_m$ do
19             if $\vec{\Lambda}^t[j] == 0$ and $\pi^{t-1}[i,j] == 1$ then
20                 for $d \leftarrow 1$ to $n_d$ do
21                     if $D[i,j][d].MatchingSegment$ then
22                         $\Delta\rho[i,j][d] \leftarrow (A^{t-1} \odot S[i,j][d]) \times \frac{P^+}{10}$;
23                 end
24             else if $\vec{\Lambda}^t[j] == 0$ then
25                 for $d \leftarrow 1$ to $n_d$ do
26                     $\bar{\rho}[i,j][d] \leftarrow \mathrm{I}(\rho[i,j][d] \geq P_{th})$ ;
27                     $\bar{S}[i,j][d] \leftarrow A^t \odot S[i,j][d]$ ;
28                     $\alpha^t \leftarrow ||\bar{S}[i,j][d] \cdot \bar{\rho}[i,j][d]||_1$ ;
29                     if $\alpha^t \geq D_{th}$ then
30                         $D[i,j][d].ActiveSegment \leftarrow 1$;
31                         $\pi^t[i,j] \leftarrow 1$ ;
32                     else if $||A^t \cdot S^d_{ij}||_1 > 0$ then
33                         $D[i,j][d].MatchingSegment \leftarrow 1$ ;
34                 end
35         end
36     end
37     # Phase-3: Learning:
38     for $j \leftarrow 1$ to $n_c$ do
39         if $\vec{\Lambda}^t[j] == 1$ then
40             for $i \leftarrow 1$ to $n_m$ do
41                 if $A^t[i,j] == 1$ then
42                     for $d \leftarrow 1$ to $n_d$ do
43                         if $D[i,j][d].ActiveSegment == 1$ then
44                             $\Delta\rho[i,j][d] \leftarrow \lambda(A^{t-1} \odot S[i,j][d]) - P^-$;
45                     end
46             end
47     end
48 until $t > n_n$
```

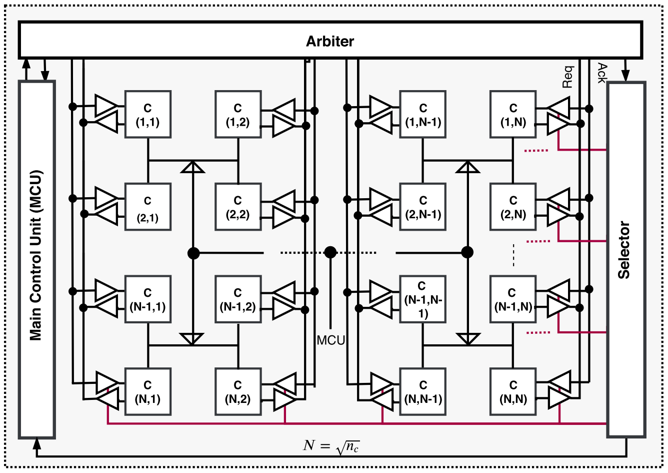

Fig. 2. High-level architecture of the HTM network, including HTM region with $\sqrt{n_c} \times \sqrt{n_c}$ mini-columns with n_m cells each, a main control unit (MCU), and an arbiter and selector.

In the second phase of temporal memory, prediction (line 17-35), the status of the cells for the next time step is evaluated. This is done via observing the distal segments activation level (α). Let D_{ij} represent a group of distal segments that belong to the ith cell within the jth mini-column, where a segment in D_{ij} indexed by d. If $\bar{\rho}^d_{ij}$ is the active distal synaptic connection within the dth segment, and $\bar{S}^d_{ij}$ holds its distal connections that are connected to active bits in A^t, the dth distal segment is set to be active segment if its $||\bar{\rho}^d_{ij} \cdot \bar{S}^d_{ij}||_1$ is greater than the segments activation threshold, Dth. Otherwise, the segment is set to a matching state if it has at least one synapse connected to an active cell in A^t. Once the status of the distal segments are determined, the cells with active distal segments are set to be in the predictive state. It is important to mention here that occasionally cells in HTM may incorrectly predict patterns. In such scenario, these cells need to have their synaptic strength reduced to lower the likelihood of incorrect prediction (as in lines 19-23). After evaluating the cells' segments, their synaptic connections are updated, which occurs during the learning phase (lines 38-47).

As aforementioned, the learning in HTM follows Hebbian's rule and it is applied solely to active cells. Given $a^t_{ij} \in A^t$, where $a^t_{ij} = 1$ and has an active segment, then all the synaptic connections that are laterally connected to previous active cells are potentiated, while those that are connected to inactive cells are depressed. This implies that the permanences of the distal synaptic connections, ρ^d_{ij}, are increased by P^+ when they are connected to active cells, otherwise, they are decreased by P^-.

3 System Design and Methodology

Fig. 2 demonstrates the high-level architecture of the developed HTM network[4] including the core units of the SSR communication scheme. Essentially, there are $\sqrt{n_c} \times \sqrt{n_c}$[5] mini-columns with n_m cells each to constitute the HTM

4. The feasibility of the HTM network scaling (beyond 1024 mini-columns) can be made possible by adopting the slicing approach proposed in our previous work [17].

5. The number of mini-columns assumed in this work is always a power of two, 2^k, where k is an integer number.

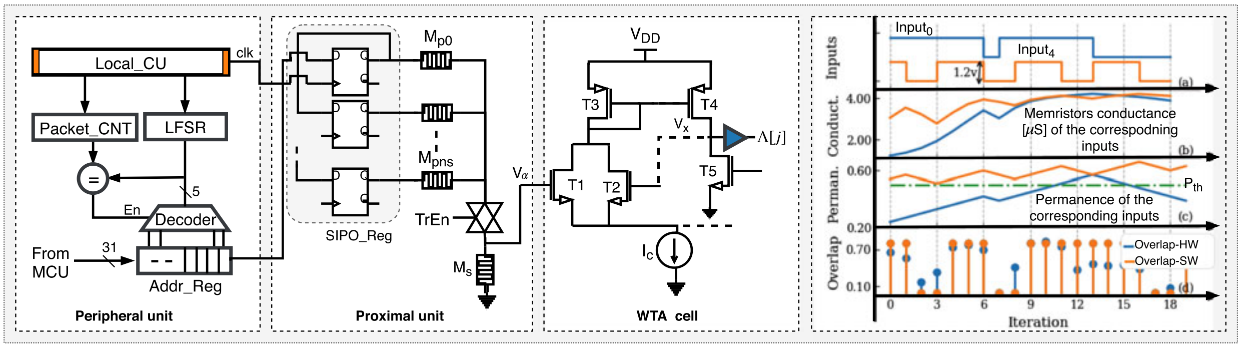

Fig. 3. (left) The circuit diagram of the HTM mini-column [6]. The circuit is composed of a peripheral unit to generate the proximal synaptic connections, a proximal unit to hold the permanences of the synaptic connections, and a WTA cell to enable mini-columns to compete with each other for input representation. (right) The impact of the synaptic permanence (denoted as Perman. for HTM-SW) modulation on the mini-column overlap score as the proximal synapses receive feed-forward input.

region. Unlike the mathematical description of the HTM which assumed 2D representation of the region for simplicity, in the hardware design, we consider a 3D architecture of the region to cut down the resources and to simplify the communication scheme considerably. The HTM region is integrated to a main control unit (MCU), and an arbiter and selector. The MCU is dedicated to control data flow, to generate the necessary control signals, and to bridge the region to the input data encoder or other regions in the hierarchy while the arbiter and selector are responsible for regulating data sharing among cells within the region. Here, the interaction among cells is based on the SSR as the cells' activity is sparse in nature, approximately 4.2 percent. At a high level, the system works as follows: when the MCU establishes a connection with the data encoder which is done through a hand-shake protocol, it commences receiving the encoded packets. The received packets are routed through the H-Tree to all the region' mini-columns. Here, we used the H-Tree structure to reduce the parasitic capacitance and to minimize the power consumption [31] of the developed system. However, there are two H-Trees, one is a digital bus (34-bit width, $1 + \log_2 n$ lines are used by the cells, where $n = n_c \times n_m$) driven by the MCU and the cells to share data. The other one (not shown in Fig. 2) is an analog line to enable mini-columns to compete against each other for input representation. When the winning mini-columns and then the cells are selected, the arbiter and selector are used to broadcast information about the current/previous active cells and their locations in the region so that lateral connections are formed and future predictions are made. In the following subsections, more details about each core unit of the HTM network are provided, while the communication scheme is discussed in a separate section.

3.1 HTM Mini-Column

The mini-columns in HTM are responsible for capturing spatial patterns of the feedforward inputs. The HTM mini-column circuit, developed in our previous work [6], is depicted in Fig. 3-(left). Succinctly, the circuit comprises a peripheral unit, a proximal unit, and WTA cell. In the peripheral unit, the proximal connections are generated and connected to the input space. The proximal unit and the WTA cell hold the proximal connections' permanences and a contesting unit that enables each mini-column to compete with its neighbors for the input representation, respectively.

In this work, the input to the mini-column is generated by the HTM random scalar encoder [32], which encodes every scalar value of the time-series data into a high-dimensional binary vector sorted into small 31-bit packets. This is to minimize data movement and the required storage units. Sequentially, each packet is fetched to the mini-columns and stored into *Addr_Reg*. When the input packet is stored in the *Addr_Reg* and the LFSR generates an address for a location in the received packet, a matching score is stored in the synapses' registers which are modeled by $n_{sp} \times 1$ serial-in-parallel-out shift register. Once all inputs are received, the outputs of the synapses' registers are presented to the memristive crossbar word-line where the proximal synapse permanences are stored. The input voltages to the crossbar will be converted into current through the memristor and the output is collected at the crossbar bit-line. The output of the crossbar, which modulates the mini-column overlap score to current, is then boosted. Boosting is done via the use of a sense memristor (M$_s$). However, upon the completion of computing the overlap score ($V_{\alpha j} \equiv \alpha_j$), its value, which is sampled by the sense memristor, is then presented to a WTA circuit (detailed description of the WTA is provided in [6]). The WTA performs a $kmax$ operation on $V_{\alpha j}, \forall j$ followed by a thresholding, to generate the final jth mini-column output, (Λ_j), as given in (1) and (2):

$$V_{\alpha_j} = \frac{\sum_{i=1}^{n_s} g_i \ V_i}{g_s + \sum_{i=1}^{n_s} g_i} \tag{1}$$

$$\Lambda_j = \begin{cases} 1, & V_{x_j} > V_{th}, \ where \ V_{x_j} = f(V_{\alpha j}) \\ 0, & Otherwise, \end{cases} \tag{2}$$

where V_i denotes the ith input voltage, g_i refers to the conductance of the ith memristor, and g_s is the conductance of the sense memristor. However, once the final output of each mini-column is generated, the learning phase of the spatial pooler starts. As alluded to earlier, the learning in HTM follows Hebbian rule [30], which is implemented using Ziksa [33], as discussed in [6]. Then, the mini-columns' status is relayed to their associated cells to start the next phase, temporal memory. Although the cells are encapsulated within the mini-columns and are considered a part of it, for the sake of clarity and simplicity we dealt with them separately.

Fig. 3-(right) demonstrates the process of computing the overlap score and tuning the proximal synaptic connections

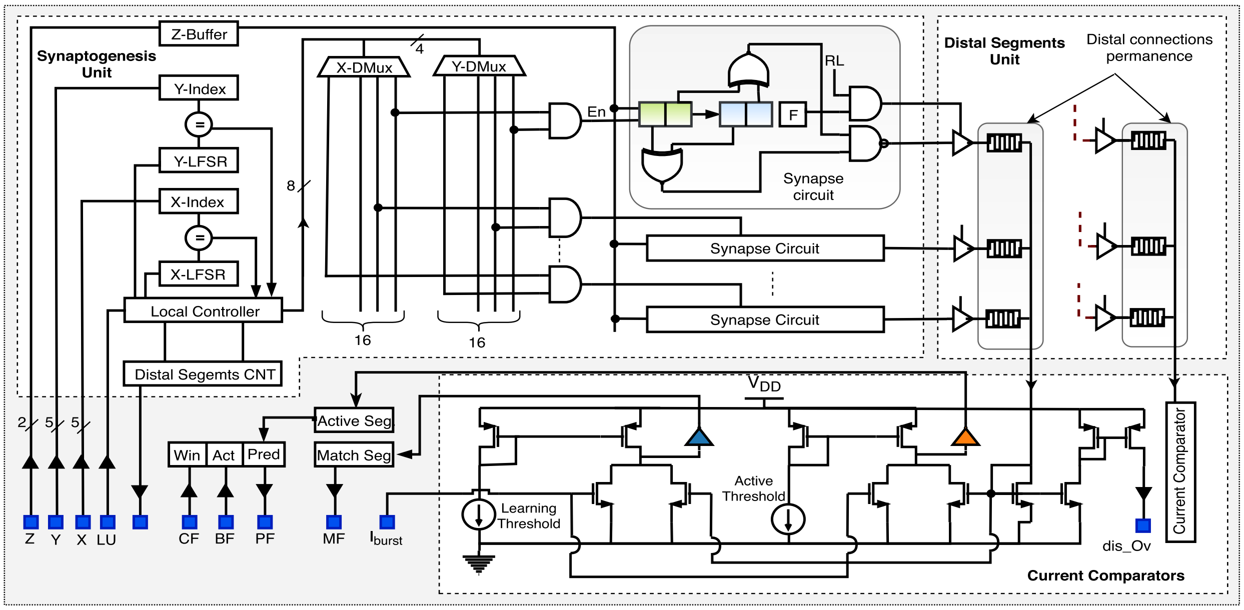

Fig. 4. The circuit diagram depicting the HTM cell with synaptogenesis unit, which can generate or prune distal segments; a distal dendritic segment to hold the permanence values of the distal connections; and current comparators to evaluate the distal segment activation level and consequently the cell status (predictive or unpredictive).

for a given mini-column while receiving feed-forward input, shown in Fig. 3a. Since the mini-column has a large number of proximal connections, for the purpose of demonstration, we randomly picked only two. The changes in proximal connection permanence for both HTM-SW and HTM-HW models are shown in Figs. 3b and 3c, respectively. Here, it can be observed that any changes in the synapse's permanence below the permanence threshold, P_{th}, in the HTM-SW model has no impact on the overlap score, unlike the HW model where there is no explicit threshold blocking the memristors from contributing to overlap score value. Furthermore, the change in the HTM-HW model synaptic permanence (memristors' conductances) tends to be non-linear as compared to the HTM-SW counterpart. However, selecting a memristor device with high conductance range and switching dynamics as required by the HTM theory made the synapses with high conductance states dominate the changes in the overlap level (see Fig. 3d). This eventually results in almost analogous overlap score[6] variation for both the SW and HW models.

3.2 HTM Cell

The cells in HTM enable the network to capture the temporal patterns, modeling the input representations within their context, and predicting the upcoming events. The HTM cell circuit developed in this work is composed of synaptogenesis unit,[7] distal segments unit, and current comparators, shown in Fig. 4. The synaptogenesis unit is responsible for forming and pruning distal synaptic pathways with the previous active cells. The distal segments unit possesses the permanence values which describe the growth level of the individual distal synaptic pathways, while the current comparators are utilized to evaluate the segments activation level and to determine their states (active or matching) accordingly.

Initially, the cells start with no distal synapses. Once the HTM begins processing the incoming patterns, the distal synapses start forming in the synaptogenesis unit. Given an HTM region arranged into a 3D space, where the x and y axes index the mini-columns in the region and the z axis indexes the cells. When the region receives an input, this causes activation of a population of cells within the region, and in this context, it is referred to as A^t_{3D}. If $a^t_{xyz} \in A^t_{3D}$, where a^t_{xyz} is an active cell located at xyz, a^t_{xyz} will form connections with the active cells in A^{t-1}_{3D}. Let's assume that the number of active cells in A^{t-1}_{3D} is 4.2 percent of n_c. Then, if $n_c = 961$, ≈ 40 cells will be active in each time step, assuming no bursting takes place. The active cell at time t establishes connections with the 40 cells that were active at $t-1$ by forming a distal segment. A cell in HTM can have around 10 or more distal segments, and this enables the network to learn the temporal transitions in sequences. Recall that forming and pruning distal connections in hardware platforms requires high interconnect dynamics which are lacking in most of the existing platforms, especially ASIC designs, hence the virtual description of the synapse became a common approach [22], [34]. However, describing the synapses virtually, in most cases, demands a high memory usage to store the sender/receiver addresses. For instance, in HTM's context (assuming there are 961 mini-columns in the region with 4 cells each), if we assume that the address of each cell is represented with 12 bits and the distal connection permanence is represented with 16 bits, having 10 segments with 60 distal connections in each cell costs 16.8 kb of memory per cell and more than 64.57 Mb for the entire network. Lets assume that the addresses and the permanences are stored in a DRAM implemented in 45nm process. If the energy cost per 32 bits of off-memory access takes 640 pJ [35], having 40 active cells at each time step leads to a total energy

6. The overlap scores for the HTM-HW and HTM-SW models are not reported up to scale for the purpose of comparison.

7. One may share the synaptogenesis unit between multiple cells of the same mini-column to cut-down resources and reduce power consumption, but at the expense of increasing the latency.

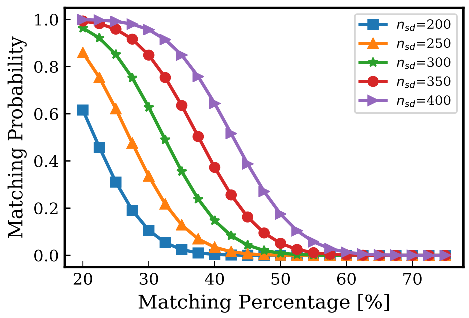

Fig. 5. The matching probability between a distal segment's addresses generated by LFSRs and the addresses of the active cells in the previous time step for various segment sizes.

consumption of 15.36 μJ (first-order approximation). Running the system at 8 MHz can result in a power consumption of 122.88 W just to access the memory, which is a prohibitive amount of power especially for edge devices with limited power budget.

One possible solution to overcome the above challenge is to reduce the memory usage in each cell. This can be done through modeling the synaptic permanence using analog memristors and leveraging the randomness in forming the distal synaptic connections to generate the addresses rather than storing them. A possible approach to do so is generating the distal segment addresses through the use of LFSRs. To demonstrate this, let's assume that the cell c_{242} is currently active and trying to establish a connection with another cell, c_{333}, which was active in the previous time step. The cell c_{242} will receive a packet that holds c_{333} location into 3D space, in this example 333. Upon receiving the address, the cell, c_{242}, begins the matching process in which the cell identifies whether there is a possibility to establish a distal connection with the cell c_{333}. The matching process starts by enabling the X-LFSR to generate 16 addresses within one clock cycle.[8] The same is applied for the Y-LFSR. While the LFSRs generate their random values, the cell translates any matches between the generated random numbers and received the Cartesian locations into flags stored into 4-bit registers, which are later decoded by X-DMUX and Y-DMUX. Here, a match means there is a distal connection established between the two cells. It is important to mention here that following such an approach makes the process of forming distal connection probabilistic, while in HTM network it is deterministic. However, in HTM, the cells that are currently active form connections with a subset (typically 50 percent) of the cells that were active in the previous time step, and in our design this is achieved naturally through our adopted probabilistic approach. Now, in order to estimate the likelihood of matching between distal segment addresses (randomly generated) and the addresses of the active cells, (3) can be used, where n_{sd} is the maximum number of synapses in a distal segment. Let the distal segment size for a given cell be 256. Given 961 mini-columns with 40 actives at each time step, there is a 0.847 likelihood that at least 20 percent of the generated random address matches those of the previous active cells. This likelihood can be significantly increased beyond 0.95 when the segment size[9] is increased, as shown in Fig. 5.

$$P_{match} = \frac{\sum_{i=10}^{n_w} \binom{n_w}{i} \times \binom{n_c - n_w}{n_{sd} - i}}{\binom{n_c}{n_{sd}}}. \tag{3}$$

After finishing the matching process and activating the X-DMUX and Y-DMUX, all the possible combinations of 16 X-addresses and 16 Y-addresses are achieved through the AND gate array. The output of logic '1' for an AND gate, let's say gate number 5, may indicate an active cell in the location ($x = 1$ and $y = 5$). The output of the AND gate enables the corresponding 'green' 2-bit register to load the Z-address, and this represents the cell distal synapse that is currently connected to an active cell at time $t - 1$, whereas the previously formed distal synapses are stored in the 'blue' 2-bit register. However, once the registers are loaded, they are compared and the results are relayed to the distal segment memristors (only when evaluating the cellular activities detected by distal segment). For the distal segment unit, this cell architecture leverages the union propriety of the SDR representation to considerably reduce the cell architecture complexity. The main concept behind the union property is storing several patterns using one representation. This can be translated into having one universal distal segment for each cell rather than multiple of them. The universal segment grows as the cell learns more temporal information. It is important to mention here that merging the segments can increase the possibility of false triggering of cell segments and incorrect predictions. However, this is less likely to happen if we limit the number of patterns (M) a segment can learn, while setting the number of mini-columns and cells to be large enough. For instance, in this work, we used 961 mini-columns with 4 cells each. If we stored 30 patterns in a segment and set the matching threshold for any two given patterns to 5, according to [36], the probability of a false match is 6.408×10^{-14} as calculated using (4).

$$P_{fp} = \left[1 - \left(1 - \frac{n_w}{n_c}\right)^M\right]^{n_w}. \tag{4}$$

The output current that is collected at the distal segment bitline is received by the current comparator unit. Then, the current gets mirrored[10] to be compared with two reference currents: active threshold and learning threshold. If the segment current is more than the active threshold, the segment is set to be in an active state and consequently the cell state changes to predictive for the next time step. On the contrary, current less than the active threshold and more than the learning threshold, marks the cell as a matching cell. A matching cell has a high probability to be selected to represent the input when bursting takes place. It is important to

8. The cells' LFSRs are clocked with 128 MHz, while the system clock is 8 MHz.

9. Increasing the distal segment size cost more cycles to generate more random addresses and additional memristor devices for each new added synapse.

10. The bursting mini-columns' cells generate additional current that should add up to the segment current during the evaluation step. However, the bursting mini-columns are evaluated globally, at the region level, and their contribution to the segment activation is done through I_{burst}.

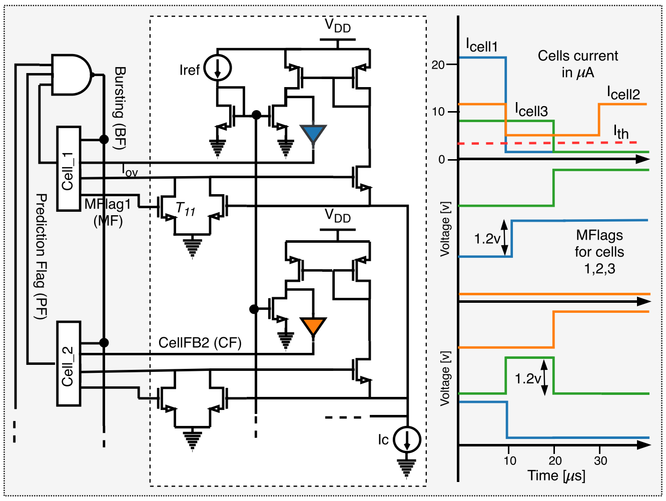

Fig. 6. (left) Competitive circuit that enables the cells within one mini-column to interact with each other when a massive firing activity takes place in the mini-column. (right) Waveform diagram illustrating the competitive circuit for the cell to select the best matching cell.

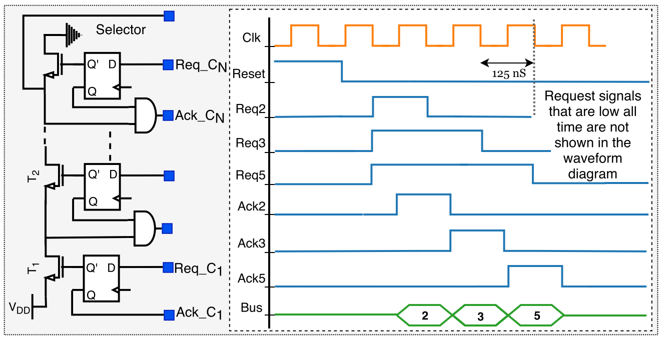

Fig. 7. (left) A synthetic synaptic arbiter (SSR) circuit consisting of buffers to store the simultaneous requests from the winning mini-columns, a series of nMOS pass transistors to monitor the status of the individual mini-column's requests, and a feedback circuit to clear mini-column requests once served. (right) Waveform diagram demonstrating a part of the SSR operation while processing several concurrent requests sent from several mini-columns located within the same row of the HTM region.

mention here that the prior discussed operations are carried out within the cells, but running the temporal memory successfully also requires the cells within the mini-columns to interact with each other to identify whether bursting is necessary. If bursting takes place in a mini-column, all the cells within the mini-column are set to be active and one cell is selected to learn the current input pattern. Typically, this is done either by selecting the best matching cell or least used cell. The former occurs only when a cell has a sufficient number of potential synapses that are connected to active cells in the previous time step i.e., a cell has a matching segment. Choosing the best matching segment involves selecting the cell with the highest matching level (distal current). This implies mirroring all cells' output current to another unit, namely the competitive circuit (a modified current based winner-take-all circuit originally proposed in [37]), so that the cell with the highest output current is chosen (see Fig. 6-(left)). In the case when there are no matching segments, the least used cell is chosen as a winning cell. Selecting the least used cells is done via selecting the cells with the least number of distal segments. Since this implementation deals with one universal merged segment, a counter in the cell is used to monitor the flags of added segments and consequently the number of merged segments in each universal one. Fig. 6-(right) demonstrates the operation of the cells competitive circuit. Here, three cells are competing to select the best matching cells. Two scenarios are considered. The first of which (interval 0-10 μs), all cells have high overlapping current (all MFlags = '0') so that they are in competition. Since cell$_1$ has the highest overlapping current, it is selected as a winner. In the second scenario (interval 10-40 μs), cell$_1$ has less current than the 'Active Threshold', for this reason it is excluded from the competition. This is accomplished via switching T_{11} to an ON state, and this eventually blocks cell$_1$ current which is mirrored to the WTA circuit.

4 Synthetic Synapses Representation

The cells in the HTM network interact with each other during the temporal memory phase. This interaction is essential to enable the network to predict the upcoming events. As alluded to earlier, the cells' interaction is enabled through the distal segments which are established and evolved while learning temporal information. In hardware, this translates into thousands of interconnects that are continuously changing in their conductivity level and locations. Due to the fact that interconnects in VLSI systems are rigid in nature and do not support this level of reconfigurability, memory units can be used to virtually formulate these connections and to describe their strength as in [24], [34]. Although such an approach is effective as it endows the network with the necessary dynamic to learn spatial and temporal information, it does not suit edge devices which have stringent area and energy constraints. Thus, we are presenting the SSR communication scheme that heavily relies on random generators and memristor devices rather than conventional memory units to form synaptic connections and to define their growth levels. This results in significant savings in terms of resources and energy consumption.

Two aspects associated with the SSR are addressed in this work: forming synaptic connections using LFSRs (discussed earlier in Section 3.2) and controlling the data transfer among cells through regulating the access to the H-Tree bus. Considering the same HTM system with A^{t-1} active cells in the previous time step and A^t active cells in the current time step, during the temporal memory phase, every cell in the network with enough strong connection to A^{t-1} cells can be depolarized for the next time step and become predictive. The challenge here is how to transfer the A^{t-1} cells' addresses to all other cells in the network efficiently. Let all the mini-columns with active cells at time $t-1$ place a request at the input of the outgoing tri-state gates (see Fig. 2). Then, each set of tri-states belonging to the same row are activated simultaneously through the selector. When a row is selected all its tri-state buffers associated with the mini-columns are activated, allowing the mini-columns to send requests to the arbiter and to receive acknowledgements. The arbiter circuit is shown in Fig. 7-(left). It comprises of buffers, a series of nMOS pass transistors, and a feedback circuit. The buffers are used to store the simultaneous requests from the selected mini-columns. The series of pass transistors are used to monitor the status of the individual mini-column requests, whereas the feedback circuit is used to acknowledge the mini-columns after their requests are served. In Fig. 7-(right), a waveform diagram illustrates the operation of the arbiter, selector, and other units in the developed system while processing information sent from a row with 5 mini-columns. Initially, all the winning mini-columns' (in this example: 2, 3, and 5) requests are directed toward the arbiter and stored in the buffers (DFF). When the DFF-3, for instance, receives Req_3, it waits in a queue until Req_2 is served. Once Req_2 is served, the voltage drop at T_2 drain will be high. This will trigger the feedback circuit to send ack_3 signal to mini-column 3, which in turn clears its request and broadcasts the address of its active cell(s). Serving the requests of all the active cells in the HTM region leads to a latency given by:

$$t_{cc} = \sum_{i=1}^{\sqrt{n_c}} \left(\sum_{j=1}^{\sqrt{n_c}} \Lambda[i][j] + 1 \right). \quad (5)$$

Recall that the SSR conveys the same concepts of the AER and the enhanced AER, but it is designed to serve intra-chip communication while offering the following advantages:

- In AER, the neuron potential duration must be ≈500 times more than the event duration for transmission to time-multiplex the transmission channel [22]. There is no need for such a constraint in the SSR.
- The enhanced AER demands memory units on both sides, sender and receiver, to hold neuron addresses that are virtually connected (connecting 32 x 32 cells requires 11 Mb RAM [24]). For a sparse network like the HTM, this is very overwhelming in terms of memory usage. However, in the SSR, the addresses are generated rather than stored. This serves two advantages: smaller storage units are used and random selection is achieved.
- The SSR is synchronous and its capacity, the maximum rate of sample transmission (considering the worst case scenario and the adopted network architecture), is 4 MSamples/sec. In the AER case, its capacity for SNN with approximately the same network size is 2.5 MSamples/sec [38].
- The SSR uses priority arbiter, which applies a queuing mechanism to access the H-Tree (or channel) bus, whereas AER utilizes an arbitration mechanism to access the channel. The latter is known to lengthen the communication cycle period and reduce channel capacity [38].
- The AER is deemed an effective approach for inter-chip communication, where neuronal information is communicated by means of encoded events. At the targeted destination, the encoded events are typically decoded and routed to the proper accessible neurons. The encoder size here is highly dependent on the number of neurons, whereas in the SSR, the decoding process complexity is defined by the number of synapses associated with the targeted neurons. This property is extremely beneficial for sparse networks like HTM.
- The enhanced AER offers better flexibility in updating the synaptic connections individually. The opposite is true for the SSR, in which changing the seeds of LFSRs enables the cell to form a new set of synaptic connections.

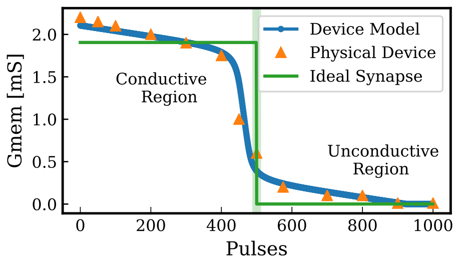

Fig. 8. Fitting the memristor model to the physical device behavior while modulating the device conductance with a train of pulses.

5 Experimental Methodology

In order to assess the performance of the proposed mixed-signal HTM system, two models are created. The first is a golden model (HTM-SW) that runs the HTM system without any constraints. This model is used to find the optimal network performance for a given task. The second model (HTM-HW) is an emulation of the hardware design under predefined circuit constraints. Here, the circuit constraints[11] are achieved after the individual components of the design are simulated and verified within Cadence Virtuoso environment. Prior to that, all the digital units are verified for functionality in Cadence SimVision. During simulation, the supply voltage is set to 1.2 v and the system clock is at 8 MHz. The system also has 128 MHz high-speed clock to drive the LFSRs of the cells. When it comes to emulating HTM synaptic connections' strength, a representative non-linear Verilog-A memristor model [39] with a modified Z-window function [6] is utilized.[12] The device conductance changes as a function of the state variable, w, is described in (6) and (7),[13] where D is the device thickness, and G_{on} and G_{off} define the memristor conductance limits. Emulating the synaptic behavior of HTM using memristors turns out to be challenging. This is because the synapses in HTM are binary in nature, i.e., they exhibit the same properties if they are above the permanence threshold regardless of the synapse's growth level and vice versa. In 2017, Jiang *et al.* proposed a memristor device to implement the k-nearest neighbour algorithm and that exhibits properties required for HTM [40]. Fig. 8 illustrates the experimental behavior of the physical device as a function of the applied pulses, fitted to the memristor model. Here, it can be observed that the memristor has minor changes in conductance level on either side of the permanence threshold (highlighted in green), while the changes are extreme in the middle. To some extent, this captures the binary nature of the ideal synapse in HTM. It is important to mention here that in order to optimize the HTM system performance and maintain low power consumption,

11. Memristor device non-idealities considered during the simulation are: 10 percent cycle-to-cycle variability (memristor resistance) and device-to-device variability (write variation).

12. τ, δ, k, and p are constants to control the window function shape. The nominal values used in this work are: τ=200, δ=0.5, k =1, and p=4.

13. k_off, k_{on}, α_{on}, and α_{off} are constant, and v_{off} and v_{on} are the memristor threshold voltage.

TABLE 1
The Memristor Device Parameters Used in the Mini-Column and Cell Designs

Parameter	Value [mini-column]	Value [cell]
Proximal memristor range	150 kΩ - 10 MΩ	150 kΩ - 10 MΩ
Memristor threshold	±0.95 v	±0.95 v
No. of switching pulses[a]	51	51
Training voltage	1.1 v	1.1 v
Sense memristor range	20 kΩ-80 kΩ	-

[a]*The number of pulses required to transition from* G_{on}/G_{off} *to* G_{off}/G_{on}

the following assumptions were made: 1) the memristor device exhibits semi-symmetrical behavior when switching from low/high conductance to high/low; 2) the memristor device offers fast switching speed and high conductance range. Table 1 shows all the device parameters used for proximal and distal synaptic connections.

$$G_{mem} = \frac{w}{D} \times G_{on} + \left(1 - \frac{w}{D}\right) \times G_{off} \tag{6}$$

$$\frac{\Delta w}{\Delta t} = \begin{cases} k_{off} \cdot \left(\frac{v(t)}{v_{off}} - 1\right)^{\alpha_{off}} . f_z(w), & 0 < v_{off} < v \\ 0, & v_{on} < v < v_{off} \\ k_{on} \cdot \left(\frac{v(t)}{v_{on}} - 1\right)^{\alpha_{on}} . f_z(w), & v < v_{on} < 0 \end{cases} \tag{7}$$

$$f_z(w) = \frac{k[1 - 2(\frac{w}{D} - \delta)]^p}{e^{\tau(\frac{w}{D} - \delta)^p}} \tag{8}$$

6 Results and Discussion

6.1 Time-Series Prediction

The prediction accuracy of the proposed HTM system is evaluated using real-world streaming data. Given an input dataset of length n_n, where each data point presented to the HTM system at time t is represented by y^t, while the corresponding predicted value is given by $\hat{y}^t$, the mean absolute percentage error (MAPE) can be computed as in (9).

$$MAPE = \frac{\sum_{t=1}^{n_n} |y^t - \hat{y}^t|}{\sum_{t=1}^{n_n} |y^t|}. \tag{9}$$

Fig. 9 illustrates a snapshot of the Hot-Gym dataset [41], the power consumption in a gym, over a small period. The power consumption is recorded at every hour for 4 months (total samples count = 4390). Here, the HTM system is used to predict the power consumption for the next 2 and 5 hours. Initially, the golden software model, HTM-SW, is used in the prediction. Then, the same prediction is made using the HTM-HW model.[14] Fig. 10a shows the accumulated MAPE recorded at every 250 samples. It can be seen that the initial value of the MAPE is really high, but over time it decreases as the network learns patterns and uses the acquired knowledge to make valid predictions in the future. However, the overall MAPE of the software model,

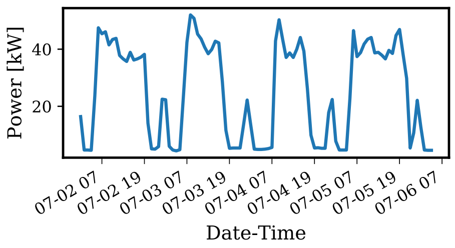

Fig. 9. A snapshot of the power consumption of the Hot-Gym dataset [41] recorded every hour over approximately 4 days.

assuming the first 500 samples presented to the network are dedicated to learning, is calculated to be 0.154 ± 0.0014 (0.171 ± 0.002 for 5-step prediction), while the hardware equivalent is 0.174 ± 0.002 (0.205± 0.0046 for 5-step prediction). This degradation may be attributed to the unsymmetrical characteristics of the memristor devices leading to disparity in the network learning and forgetting rate. Applying the union property to the distal segment and forming its distal synapses using LFSRs might have negative consequences as well, especially when making higher order predictions.

6.2 Latency

The latency is measured as the time required for the HTM network to process an SDR input generated by the encoder. In this context, HTM processes SDR inputs of the Hot-Gym dataset, where each input is encoded with 512-bit binary vector. The spatial pooler and temporal memory phases here are performed simultaneously[15] and in a pipelined fashion to minimize the latency, which is estimated to be 11.64 μs. Fig. 10b shows the latency of the CMOS digital HTM (system clk = 100 MHz) and the proposed mixed-signal HTM (system clk = 8 MHz) as a function of the network size, given by the number of mini-columns. One can notice that the latency in the digital HTM is always higher than the mixed-signal counterpart. This can be attributed to several reasons. The first is the need for the initialization phase in the digital HTM design to set the synaptic connections' permanences, particularly the proximal synapses, prior to receiving any input. The initialization of the synaptic connections' permanence is achieved for free in the mixed-signal design as the memristors after the formation process have random conductance with Gaussian distribution [43]. Second, tuning the synaptic connections, proximal or distal, is performed simultaneously at the cell and mini-column levels, but within them it is sequential because the permanence values are stored in distributed SRAMs, where the read/write operations take several clock cycles. In the mixed-signal design, on the contrary, the tuning process is performed concurrently even within the mini-columns or cells and usually it takes two clock cycles. Finally, in the digital HTM, the winning mini-columns that represent the input are decided in a sequential fashion to cut down the resource cost and power consumption. This in turn translates to longer latency that is proportional to the number of mini-columns. In the

14. HTM-HW model is also benchmarked using other datasets such as NYC-Taxi [42]. The achieved MAPE for the 2nd and 5th order predictions are 0.0996 ± 0.0014 and 0.156 ± 0.0084, respectively.

15. Spatial pooler and temporal memory operate simultaneously when the H-Tree bus is exploited by either of them.

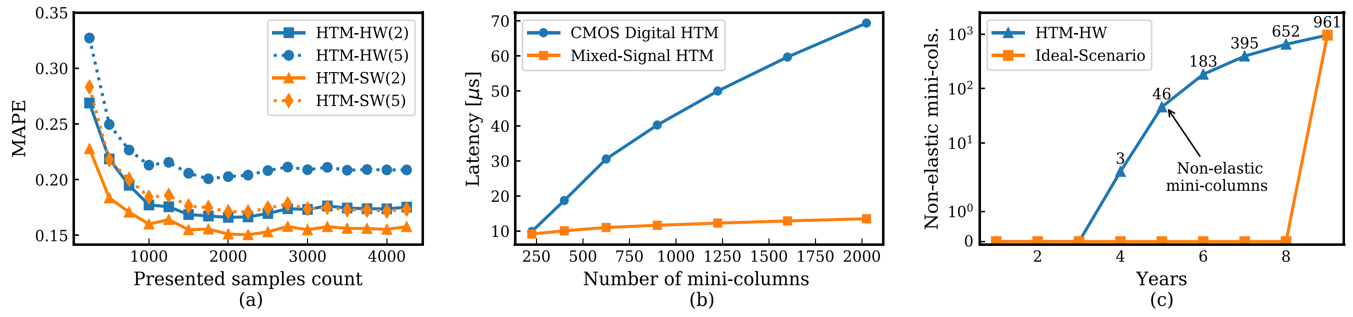

Fig. 10. (a) MAPE for predicting the power consumption in a gym for the next 2 and 5 hours using HTM software (HTM-SW) and HTM hardware (HTM-HW) models. (b) Latency of the digital and mixed-signal HTM as a function of the network size, given by the number of mini-columns. (c) Elasticity (lifespan) of the overall HTM mini-columns in the ideal and real-world scenarios.

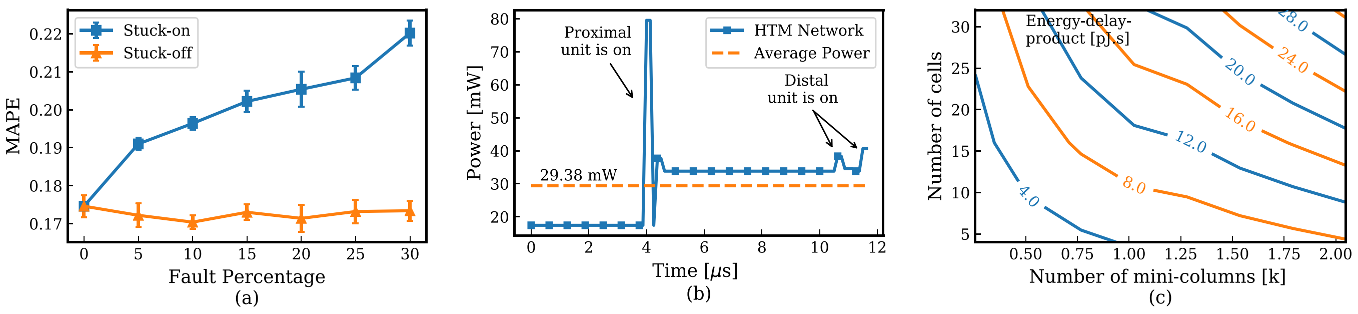

Fig. 11. (a) The MAPE of the HTM-HW predicting two steps ahead in time for the Hot-Gym dataset while experiencing various types of stuck-at faults. (b) The total power consumption of the developed HTM system as it processes and predicts time-series data from Hot-Gym dataset. (c) Contour of energy-delay-product for the developed HTM system as a function of network size.

mixed-signal design, a WTA circuit [6] is used, which processes all the inputs concurrently.

6.3 Network Reliability and Lifespan

The memristor device write endurance, which is the number of times a memory cell can be overwritten successfully, turns out to be a crucial factor in determining network sustainability for learning. The memristor devices, particularly oxide-based devices, have a typical endurance range between $10^6 - 10^{12}$ [44]. This low endurance reduces the network reliability for online learning and continuous adaptation especially when the network is densely connected and all neurons need to be updated continuously. For the HTM network, this is not the case as cell/mini-column activities are sparse in nature and the learning is confined only to the active ones. This feature endows the network longer elasticity (lifespan) in comparison to other networks. In order to estimate the elasticity of mini-columns in the HTM network, we need to estimate their successful training rounds (L_r) and likelihood of activation, as given by (10), where E_d is the memristor device endurance.

$$L_r = \frac{E_d \times n_c}{n_w}. \tag{10}$$

In the ideal scenario, mini-columns in the HTM network are activated with equal likelihood by patterns detected at the proximal segments. Thus, the number of successful learning rounds that can be made, given $E_d = 10^9$, $n_c = 961$, and $n_w = 40$, is 240×10^8. This is equivalent to ≈8 years of successful continuous learning performed at a rate of 10ms. Comparatively, this is ≈24 times more than a conventional network with no sparse activities, and X[16] times more than the SNNs.[17] In spite of the fact that SNNs are asynchronous and sparse in nature, usually their neurons fire and their synaptic connections are tuned multiple times while processing a single input. This is because each input is stochastically encoded as a stream of spikes. However, the previous comparison hypothesizes that the HTM mini-columns' activations are perfectly regularized by incorporating the homeostasis plasticity mechanism (or boosting). In real-world scenarios, this is not the case, because the mini-columns' activations are highly affected by input space statistics. Fig. 10c is an example demonstrating an estimation of the developed system elasticity (lifespan) for the Hot-Gym dataset. Here, we see that after year 4, a gradual loss in mini-columns' elasticity starts to occur. Even after 8 years of work, ≈309 mini-columns are still elastic and have the capability to acquire new information. However, the overall network performance at that time would be limited.

6.4 Device Failure and Network Robustness

There are various types of memristor defects that may affect network performance, and usually they occur due to process variation [45], [46]. Examples of device defects are ageing faults, endurance degradation faults, switching delay faults, and stuck-at faults [47], [48]. Here, we will emphasize the stuck-at fault as it is ubiquitous and has high impact on

16. X is not specified here because it is highly affected by the input and the encoding approach.

17. SNNs are usually trained with spike-time-dependent-plasticity (STDP) rules. STDP requires neurons to be tuned based on the time difference between the pre and post synaptic neurons' spikes.

network performance [48]. Two types of stuck-at faults are studied. The first investigates the impact of stuck-on (high-conductance state) on HTM system performance while making two step ahead prediction for the Hot-Gym dataset. The second focuses on the stuck-off (low-conductance state) effect. Fig. 11a illustrates the averaged MAPE over 5 runs for the HTM-HW prediction as a function of the faulty[18] device percentage for the aforementioned cases. It can be seen that the stuck-off fault has a positive marginal impact on the network performance as it leads to an increase in the network sparsity level. In contrast, the stuck-on increases the MAPE by 1.7 percent and it can go up to 4.9 percent when the fault percentage is 30 percent. This degradation in performance arises from the fact that the SDR classifier is implemented using a softmax classifier with weighted synapses realized using a memristive crossbar. Having 10 percent of stuck-on fault in crossbar means on average, every row and column in the crossbar has 55 and 344 defective devices, respectively. This eventually makes the softmax classifier output nodes unable to distinguish various pattern activities and fire excessively.

During the fault analysis, it is also found that applying the fault solely to the spatial pooler results in a marginal change in the system performance. This is because each input sample presented to the HTM is spatially represented by a small population of active mini-columns, and having a slight change in the representation pattern, which may result from the fault, has very low impact. Furthermore, using k-winner mechanism mitigates the changes that may occur in spatial patterns.

6.5 Power Consumption and Distribution

The average total power consumption of the developed HTM system while predicting time-series data from the Hot-Gym dataset is estimated to be 28.94 mW and 29.38 mW[19] when the online learning is enabled. The high power consumption during the training is due to the use of high voltage (memristor training voltage ≈1.1v) and extra clock cycles to modulate the memristor devices. Fig. 11b demonstrates the estimated total power consumption over time. Initially, 17.18 mW is consumed while transferring the input SDRs through the H-Tree[20] to the mini-columns and establishing the proximal synaptic connections, which take place in simultaneous fashion. The power then abruptly increases due to the activation of the proximal segments to compute the mini-column overlap scores. Once the winning mini-columns are selected, the spatial pooler learning phase starts, in which the memristors associated with proximal synapses are modulated. Meanwhile, the prior active cells' addresses are routed to each cell in the winning mini-columns to compute their distal segments' overlap scores. Computing distal segments' overlap scores give rise to another abrupt increase in the power consumption (at time $\approx 10.6\,\mu$s). However, this increase is much smaller than the one occurred while computing the overlap scores of the mini-columns. This is because computing the cells' overlap scores is confined only to the cells within the winning mini-column while other cells are disabled through clock-gating. After computing the cells' overlap scores, the cells of the winning mini-columns locally compete to represent the input contextually. The selected active cells form the lateral connections with the neighboring cells and tune their distal connections accordingly. One may observe from the previous discussion that tuning and computing the overlap scores here turn out to be the most power-hungry operations as there are more than 45.15 k synapses involved in the network computations. One possible way to minimize the power is to modify network size or segregate the above operations into multiple stages at the mini-column or cell levels, but this will be at the expense of increasing the overall network latency. Fig. 11c illustrates the contour of energy-delay-product measured in pJ.s which can be used to pick the optimal network architecture for a given power consumption and latency requirement.

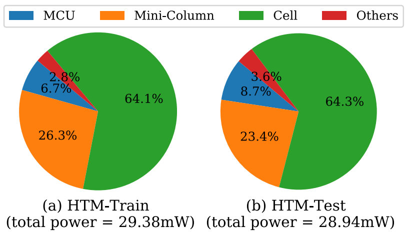

Fig. 12. The distribution of the power consumption for the building blocks of the proposed HTM system during training and testing modes.

Fig. 12 shows the distribution of the power consumption among the different entities of the proposed HTM system during the training and testing modes. It implies that in the HTM-Test, most of the power consumption is devoted to the HTM cells as they are more complicated and have a large number of synaptic connections. During the training mode, HTM-Train, the cells and mini-columns pull further power to modulate their synaptic connections. On the contrary, the MCU and other units (arbiter and selector, excluding the H-Tree) consume a small fraction of the total power as they are less complex and have limited memory usage.

In an endeavour to compare our work with previous HTM implementations in literature (see Table 2), we found that performing relative comparisons is a challenging process due to the lack of similarity in network architectures, technology nodes, operating frequency, etc. Thus, we attempt to bring all networks to the same size in terms of the mini-columns and cell count. Also, we hypothesize that the size of the networks can be scaled linearly and the same is applied to their power consumption. Starting with Krestinskaya et. al. [20],[21] here we scaled only the number of mini-columns and the single pixel processing elements (total = 961x1) as detailed information about the distal segments and their sizes are not reported, and this results in 17.45 X improvement. In the case of fully CMOS digital design, 77.02 X is achieved when compared to our previous work in [17], and 31.75 X and 22.29 X when compared to the work done by Li Weifu *et al.* [11], [18]. In contrast to other previous works, the power

18. The fault is applied to the mini-columns' proximal connections and SDR classifier weights.

19. The approach used to estimate the power consumption is described in our previous work [6].

20. The H-Tree structure might be buffered with full-swing and reduced swing buffers, proposed in [31], to minimize the power consumption further.

21. The authors in this paper also consider linear scaling for the network size and the power consumption.

TABLE 2
A Comparison of the Proposed HTM System With Previous Work

Algorithm	Memristive HTM [20]	PIM HTM [17]	Digital HTM [18]	PE HTM [11]	This work
Task	Classification	Classification&Prediction	Prediction	Image recognition	Classification&Prediction
Operating Frequency	-	100 MHz	100 MHz	100 MHz	Dual 8-128 MHZ
Proximal Segment Size	9	16	1	40	31
Distal Segments x Size	-	5x10	-	12x16	Shared 256
Total Power consumed	13.34 mW[b]	417 mW	516 mW[a]	4.1 W	29.38 mW
Dataset	AR, TIMIT, & ORL	MNIST	MNIST	KTH	MNIST[d] & Hot-Gym
Mini-columns x cells	25xX[c]	100x3	400x2	2048x32	961x4
Latency (ms)	-	0.0057	0.0045	6.04	0.0116
Technology node	TSMC 180 nm	TSMC 65 nm	Nangate 45 nm	GF 65 nm	IBM 65 nm

[a]*In [18], the power consumption is reported for a single processing element (PE) without considering the register files. Thus, we linearly scaled the power for an HTM network of size 400x2.*
[b]*In this reference, the temporal memory power is reported for single pixel processing. This value is multiplied by the total number of mini-columns to estimate a total power of an HTM region with 25 mini-columns with one cell each.*
[c]*X denotes unknown number of cells.*
[d]*Further details about MNIST results are provided in [6].*
One may note that these implementations are on different substrates, thereby this table offers a high-level reference template for HTM hardware rather than an absolute comparison.

consumption reported in [18] does not consider the register files, which are usually the most power-hungry components in the design. In the case of [11], it is unclear if the register files power consumption is included. It is important to mention here that, in most cases, the overall networks' synaptic connections have not been included in the aforementioned scaling process as there is no clear approach to estimate the power consumption for the individual synaptic primitives. However, since our design uses more synaptic connections, equating our design with previous works in terms of the synaptic connections count may result in further improvement in power consumption.

7 Conclusion

This paper proposes a memristor-based mixed-signal architecture of the HTM network including the spatial and temporal aspects of the algorithm. The proposed architecture incorporates several plasticity mechanisms such as synaptogenesis, neurogenesis, etc. that endow the network a high-degree of plasticity with lifelong learning and minimal energy dissipation. The high-level behavioral model of the architecture is verified for time-series data prediction. It is found that the MAPE of the hardware model is more than that in the software counterpart by 1.129X. This degradation is mainly attributed to the memristor devices' non-idealities and the use of synthetic synapses representation. The proposed architecture is also evaluated for latency and lifespan. We found that the mixed-signal implementation is $\approx$3.46X faster than the pure CMOS implementation and it is less affected by network scale, while the network elasticity (lifespan) can be up to 8 years, assuming that learning occurs every 10 ms. When it comes to network robustness, it is observed that the HTM network is robust to device failure, but this is not the case for its SDR classifier, which is impacted by stuck-on faults. Furthermore, it is observed that the power consumption in the proposed architecture is dominated by the cells, particularly the proximal and distal segments. Thus, in our design, we strive to limit their use to a minimum number of cycles and thereby reduce the average total power consumption of the network to 29.38 mW.

Acknowledgments

This work was supported in part by Seagate Technology.

References

[1] J. Hawkins and S. Blakeslee, *On Intelligence: How a New Understanding of the Brain Will Lead to the Creation of Truly Intelligent Machines*. New York, NY, USA: Macmillan, 2005.
[2] J. Hawkins, D. George, and J. Niemasik,"Sequence memory for prediction, inference and behaviour," *Philos. Trans. Royal Society B: Biol. Sci.*, vol. 364, no. 1521, pp. 1203–1209, 2009.
[3] D. George and J. Hawkins, "Towards a mathematical theory of cortical micro-circuits," *PLoS Comput. Biol.*, vol. 5, no. 10, 2009, Art. no. e1000532.
[4] J. Hawkins and S. Ahmad, "Why neurons have thousands of synapses, a theory of sequence memory in neocortex," *Frontiers Neural Circuits*, vol. 10, 2016, Art. no. 23.
[5] D. E. Padilla-Baez, "Analysis and spiking implementation of the hierarchical temporal memory model for pattern and sequence recognition," Ph.D. dissertation, Inst. Telecommun. Res., School Inf. Technol. Math. Sci., Univ. South Australia, Australia, 2015.
[6] A. M. Zyarah and D. Kudithipudi, "Neuromemrisitive architecture of HTM with on-device learning and neurogenesis," *ACM J. Emerg. Technologies Comput. Syst.*, vol. 15, no. 3, 2019, Art. no. 24.
[7] K. L. Rice, T. M. Taha, and C. N. Vutsinas, "Hardware acceleration of image recognition through a visual cortex model," *Optics Laser Technol.*, vol. 40, no. 6, pp. 795–802, 2008.
[8] J. Xing, T. Wang, Y. Leng, and J. Fu, "A bio-inspired olfactory model using hierarchical temporal memory," in *Proc. 5th Int. Conf. Biomed. Eng. Inform.*, 2012, pp. 923–927.
[9] N. O. El-Ganainy, I. Balasingham, P. S. Halvorsen, and L. A. Rosseland, "On the performance of hierarchical temporal memory predictions of medical streams in real time," in *Proc. 13th Int. Symp. Med. Inf. Commun. Technol.*, 2019, pp. 1–6.
[10] A. Lavin and S. Ahmad, "Evaluating real-time anomaly detection algorithms–the numenta anomaly benchmark," in *Proc. IEEE 14th Int. Conf. Mach. Learn. Appl.*, 2015, pp. 38–44.
[11] W. Li, "Design of hardware accelerators for hierarchical temporal memory and convolutional neural network," Ph.D. dissertation, Dept. Elect. Eng., North Carolina State Univ., Raleigh, NC, USA, 2019.
[12] O. Krestinskaya, I. Dolzhikova, and A. P. James, "Hierarchical temporal memory using memristor networks: A survey," *IEEE Trans. Emerg. Topics Comput. Intell.*, vol. 2, no. 5, pp. 380–395, Oct. 2018.
[13] L. Streat, D. Kudithipudi, and K. Gomez, "Non-volatile hierarchical temporal memory: Hardware for spatial pooling," 2016, *arXiv:1611.02792*.
[14] M. Kerner and K. Tammemäe, "Hierarchical temporal memory implementation on FPGA using LFSR based spatial pooler address space generator," in *Proc. IEEE 20th Int. Symp. Des. Diagnostics Electron. Circuits Syst.*, 2017, pp. 92–95.

[15] T. Ibrayev, A. P. James, C. Merkel, and D. Kudithipudi, "A design of HTM spatial pooler for face recognition using memristor-CMOS hybrid circuits," in *Proc. IEEE Int. Symp. Circuits Syst.*, 2016, pp. 1254–1257.

[16] A. M. Zyarah, "Design and analysis of a reconfigurable hierarchical temporal memory architecture," Master thesis, Dept. Elect. Eng., Rochester Inst. Technol., Rochester, NY, 2015.

[17] A. M. Zyarah and D. Kudithipudi, "Neuromorphic architecture for the hierarchical temporal memory," *IEEE Trans. Emerg. Topics Comput. Intell.*, vol. 3, no. 1, pp. 4–14, Feb. 2019.

[18] W. Li and P. Franzon, "Hardware implementation of hierarchical temporal memory algorithm," in *Proc. 29th IEEE Int. System-on-Chip Conf.*, 2016, pp. 133–138.

[19] D. Fan, M. Sharad, A. Sengupta, and K. Roy, "Hierarchical temporal memory based on spin-neurons and resistive memory for energy-efficient brain-inspired computing," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 27, no. 9, pp. 1907–1919, Sep. 2015.

[20] O. Krestinskaya, T. Ibrayev, and A. P. James, "Hierarchical temporal memory features with memristor logic circuits for pattern recognition," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 37, no. 6, pp. 1143–1156, Jun. 2018.

[21] B. V. Benjamin *et al.*, "Neurogrid: A mixed-analog-digital multichip system for large-scale neural simulations," *Proc. IEEE*, vol. 102, no. 5, pp. 699–716, May 2014.

[22] M. Mahowald, "Vlsi analogs of neuronal visual processing: a synthesis of form and function," Ph.D. dissertation, California Inst. Technol., Pasadena, CA, 1992.

[23] R. J. Vogelstein, F. Tenore, R. Philipp, M. S. Adlerstein, D. H. Goldberg, and G. Cauwenberghs, "Spike timing-dependent plasticity in the address domain," in *Proc. Advances Neural Inf. Process. Syst.*, 2003, pp. 1171–1178.

[24] D. H. Goldberg, G. Cauwenberghs, and A. G. Andreou,"Probabilistic synaptic weighting in a reconfigurable network of VLSI integrate-and-fire neurons," *Neural Netw.*, vol. 14, no. 6–7, pp. 781–793, 2001.

[25] Y. Cui, C. Surpur, S. Ahmad, and J. Hawkins, "A comparative study of HTM and other neural network models for online sequence learning with streaming data," in *Proc. Int. Joint Conf. Neural Netw.*, 2016, pp. 1530–1538.

[26] Y. Cui, S. Ahmad, and J. Hawkins,"Continuous online sequence learning with an unsupervised neural network model," *Neural Comput.*, vol. 28, no. 11, pp. 2474–2504, 2016.

[27] Y. Cui, A. Subutai, and J. Hawkins, "The HTM spatial pooler − A neocortical algorithm for online sparse distributed coding," *Frontiers Comput. Neurosci.*, vol. 11, 2017, Art. no. 111.

[28] P. Földiak,"Forming sparse representations by local anti-hebbian learning," *Biol. Cybern.*, vol. 64, no. 2, pp. 165–170, 1990.

[29] S. Ahmad and J. Hawkins, "Properties of sparse distributed representations and their application to hierarchical temporal memory," 2015, *arXiv:1503.07469*.

[30] D. O. Hebb, *The Organization of Behavior: A Neuropsychological Theory*, J. Wiley; Chapman & Hall, 1949.

[31] F. H. A. Asgari and M. Sachdev, "A low-power reduced swing global clocking methodology," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 12, no. 5, pp. 538–545, May 2004.

[32] S. Purdy, "Encoding data for HTM systems," 2016, *arXiv:1602.05925*.

[33] A. M. Zyarah *et al.*, "Ziksa: On-chip learning accelerator with memristor crossbars for multilevel neural networks," in *Proc. IEEE Int. Symp. Circuits Syst.*, 2017, pp. 1–4.

[34] A. M. Zyarah and D. Kudithipudi, "Reconfigurable hardware architecture of the spatial pooler for hierarchical temporal memory," in *Proc. 28th IEEE Int. Syst.-on-Chip Conf.*, 2015, pp. 143–153.

[35] S. Han, "Efficient methods and hardware for deep learning," Ph.D. dissertation, Stanford Univ., Stanford, CA, 2017.

[36] S. Ahmad and J. Hawkins, "How do neurons operate on sparse distributed representations? a mathematical theory of sparsity, neurons and active dendrites," 2016, *arXiv:1601.00720*.

[37] J. Lazzaro, S. Ryckebusch, M. A. Mahowald, and C. A. Mead, "Winner-take-all networks of O(N) complexity," in *Proc. Advances Neural Inf. Process. Syst.*, 1989, pp. 703–711.

[38] K. A. Boahen, "Communicating neuronal ensembles between neuromorphic chips," in *Neuromorphic Systems Engineering*. Berlin, Germany: Springer, 1998, pp. 229–259.

[39] S. Kvatinsky, M. Ramadan, E. G. Friedman, and A. Kolodny, "Vteam: A general model for voltage-controlled memristors," *IEEE Trans. Circuits Syst. II: Express Briefs*, vol. 62, no. 8, pp. 786–790, Aug. 2015.

[40] Y. Jiang, J. Kang, and X. Wang, "RRAM-based parallel computing architecture using k-nearest neighbor classification for pattern recognition," *Sci. Reports*, vol. 7, 2017, Art. no. 45233.

[41] 2013. [Online]. Available: https://github.com/numenta/nupic/tree/master/examples/opf/clients/hotgym/prediction/one_gym

[42] "Passanger demand for New York city taxis," 2016. [Online]. Available: http://www.nyc.gov/html/tlc/html/about/trip_record_data.shtml

[43] M. Prezioso, F. Merrikh-Bayat, B. Hoskins, G. C. Adam, K. K. Likharev, and D. B. Strukov, "Training and operation of an integrated neuromorphic network based on metal-oxide memristors," *Nature*, vol. 521, no. 7550, 2015, Art. no. 61.

[44] M. Coll *et al.*, "Towards oxide electronics: A roadmap," *Appl. Surface Sci.*, vol. 482, pp. 1–93, 2019.

[45] S. Kannan, N. Karimi, R. Karri, and O. Sinanoglu, "Detection, diagnosis, and repair of faults in memristor-based memories," in *Proc. IEEE 32nd VLSI Test Symp.*, 2014, pp. 1–6.

[46] L. P. Romero *et al.*, "Training fully connected networks with resistive memories: impact of device failures," *Faraday Discussions*, vol. 213, pp. 371–391, 2019.

[47] S. Kumar *et al.*, "Oxygen migration during resistance switching and failure of hafnium oxide memristors," *Appl. Phys. Lett.*, vol. 110, no. 10, 2017, Art. no. 103503.

[48] V. Ravi and S. Prabaharan, "Fault tolerant adaptive write schemes for improving endurance and reliability of memristor memories," *AEU-Int. J. Electron. Commun.*, vol. 94, pp. 392–406, 2018.

Abdullah M. Zyarah received the BSc degree in electrical engineering from the University of Baghdad, Iraq, in 2009, and the MSc degree in electrical engineering from the Rochester Institute of Technology, USA, in 2015. He is a lecturer with the Department of Electrical Engineering, University of Baghdad. He specializes in digital and mixed-signal designs and his current research interests include Neuromorphic architectures for energy constrained platforms and biologically inspired algorithms. Currently, he is working toward the PhD degree with the Neuromorphic AI Lab research group in the Department of Computer Engineering, Rochester Institute of Technology.

Kevin Gomez is a technologist with the research group at Seagate. His research interests include computer architecture post Dennard scaling and human brain inspired computing.

Dr. Dhireesha Kudithipudi (Senior Member, IEEE) is a professor and founding director of the AI Consortium with the University of Texas, San Antonio and Robert F McDermott chair in engineering. Her research interests include neuromorphic AI, low power machine intelligence, brain-inspired accelerators, and use-inspired research. Her team has developed comprehensive neocortex- and cerebellum-based architectures with nonvolatile memory, hybrid plasticity models, and ultra-low precision architectures. She is passionate about transdisciplinary and inclusive research training in AI fields. She is the recipient of the Clare Booth Luce Scholarship in STEM for women in highered (2018) and the 2018 Technology Women of the Year in Rochester.