Hyperdimensional Computing Using Time-To-Spike Neuromorphic Circuits

Graham Bent
Crime and Security Institute, Cardiff University
grahamabent@gmail.com

Chris Simpkin
Crime and Security Institute, Cardiff University
simpkinc@cardiff.ac.uk

Yuhua Li
School of Computer Science, Cardiff University
liy180@cardiff.ac.uk

Alun Preece
Crime and Security Institute, Cardiff University
preecead@cardiff.ac.uk

***Abstract*—Vector Symbolic Architectures (VSA) can be used to encode complex objects, such as services and sensors, as hypervectors. Such hypervectors can be used to perform efficient distributed service discovery and workflow orchestration in communications constrained environments typical of the Internet of Things (IoT). In these environments, energy efficiency is of great importance. However, most hypervector representations use dense i.i.d element values and performing energy efficient hyperdimensional computing operations on such dense vectors is challenging. More recently, a sparse binary VSA scheme has been proposed based on a slot encoding having M slots with B bit positions per slot, in which only one bit per slot can be set. This paper shows for the first time that such sparse encoded hypervectors can be mapped into energy-efficient *time-to-spike* Spiking Neural Network (SNN) circuits, such that all the required VSA operations can be performed. Example VSA SNN circuits have been implemented in the Brian 2 SNN simulator, showing that all VSA binding, bundling, unbinding, and clean-up memory operations execute correctly. Based on these circuit implementations, estimates of the energy and processing time required to perform the different VSA operations on typical SNN neuromorphic devices are estimated. Recommendations for the design of future SNN neuromorphic processor hardware that can more efficiently perform VSA processing are also made.**

***Index Terms*—Vector Symbolic Architecture, Hypervectors, Spiking Neural Networks, Neuromorphic Processing**

I. Introduction

Vector Symbolic Architectures (VSAs) are a family of bio-inspired methods for representing and manipulating concepts and their meanings using fixed-size vector representations in a high-dimensional vector space as described by [1]. Eliasmith has shown how these vector representations can be used to perform 'brain like' neuromorphic cognitive processing [2] and coined the phrase 'semantic pointer' for such a vector since it acts as both a 'semantic' description of the concept and a 'pointer' to the concept. As such, they are said to be semantically self-describing. VSAs are capable of supporting a large range of cognitive tasks such as: Semantic composition and matching; Representing meaning and order, [3]; Analogical mapping [4]; and Logical reasoning, [3]. Consequentially they have been used in natural language processing, [3], and cognitive modelling [2]. VSAs use vectors of very high dimensionality (D), i.e., hypervectors (HVs). For example, Plate's Holographic Reduced Representations (HRR) [5] use real-number vectors typically having ($512 \leq D < 2048$). Whereas Kanerva's Binary Spatter Codes (BSC) [6] are bit-string HVs, typically having $D \approx 10,000$. A hierarchical BSC binding and bundling scheme was presented in [7]–[9], which showed how sensor and service descriptions could be represented as HVs that, in turn, can be bound and bundled into higher-level HVs that represent sensor-service workflows. Using these workflow HV representations, the possibility to perform efficient distributed service discovery and workflow orchestration in communications and energy-constrained environments that are typical of the Internet of Things (IoT) has been demonstrated [7]–[12]. In such environments, performing the VSA operations in an energy-efficient manner is essential and in [11] it was demonstrated that some, (but not all) of the required VSA operations can be performed on BSC HVs using ultra-low energy-efficient Phase Change Memory Devices (PCM) [13]. The motivation for this paper was to investigate the possibility of performing all of the required VSA operations using energy-efficient devices, SNN neuromorphic processing devices.

Eliasmith has demonstrated how spike-rate encoded SNN circuits can implement convolution/deconvolution in an HRR based VSA for cognitive processing [2]. Whilst this is an elegant solution, it requires large numbers of spikes to be processed, and as a result, this is not an energy-efficient mechanism. Similarly, although BSC operators are much simpler, implementation using SNNs is not energy efficient due to their very high dimensionality requiring many spikes per operation. This paper uses the sparse hypervector (SHV) encoding model proposed by Laiho [14] to address this challenge. The model uses a slot encoding mechanism with M slots and B bits per slot, of which only one bit is set per slot. A recent paper, [15], compares various alternative SHV models and concludes that slot encoding has the desired properties for VSA manipulations, outperforming the other methods evaluated. The main contribution of this paper is to show how the model proposed in [14] can be mapped onto an SNN circuit using M neurons and where the corresponding bit positions represent a *time-to-spike* encoding rather than the spike rate encoding used by [2]. The next section gives a brief overview of the mathematical properties of VSA operations and shows how complex object representations can be built and queried using VSA.

II. VSA binding and bundling Overview

Unlike classical computing, which operates on bits through logical operations and the four arithmetic operations of addition, subtraction, multiplication and division, VSAs deal with HVs through three operations, *bundling* (a superposition operator denoted '+'), *binding* and *unbinding* (permutation/multiplication operators denoted '$\odot$' and '$\oslash$' respectively) and a HV *normalisation* operator. VSA '*concept/compound*' vectors built using these operations can then be compared for similarity and have their sub-vector contents probed using vector comparison operators such as *cosine similarity* and *Hamming Distance/Similarity*.

Binding permutes HVs to a different part of the HV space making them orthogonal to all other vectors with high probability so that, if $V = R \odot A$ and $W = R \odot B$ then R, A and B will have no similarity to V or W. However, comparing V with W will produce the same match value as comparing A with B because *binding* preserves distance within the hyperspace [16, page 147]. *Cyclic-shift* binding, denoted 'ρ' is a unary version of binding/unbinding whereby instead of a multiplicative permutation between *role* and *filler* HVs, a HV is shifted right (*binding*) or left (*unbinding*) to produce an output that is orthogonal to its input HV. Binding/unbinding and cyclic-shift are associative, commutative and distribute over *bundling*. Binding and unbinding are also invertible for many VSAs, including BSCs and Laiho's SHVs.

Bundling combines groups of *role-filler* pairs into a single same sized compound HV that bears similarity to each of its role-filler constituents. It represents the group of sub-HVs as a whole, analogous to a set or data record, while simultaneously storing each role-filler within its HV elements. Consider the bundled set of role-filler pairs in Eq. 1

$$Z_v = V_r \odot A_v + W_r \odot B_v + X_r \odot C_v \tag{1}$$

To recover the sub-HVs from Z_v it must be unbound with the appropriate role HV, for example to extract A_v perform:

$$V_r \oslash Z_v = A_v + V_r \oslash W_r \odot B_v + V_r \oslash X_r \odot C_v \tag{2}$$

$$= A_v + noise \tag{3}$$

By designating the role HVs in Eq. 1 as '*position*' HVs it is easy to see that Eq. 1 can be used to represent sequences. An alternate approach for the creation of sequences is to combine bundling with the cyclic-shift operator, for example

$$Z_v = \rho^1(A_v) + \rho^2(B_v) + \rho^3(C_v) \tag{4}$$

The exponentiation operator applied to ρ indicates the number and direction of the cyclic-shift steps. Each bundled sub-HV can be retrieved in order by applying ρ in the opposite direction.

$$A_v \approx \rho^{-1}(Z_v) = A_v + \rho^1(B_v) + \rho^2(C_v) \tag{5}$$

In [7], [8], [12] an alternate binding/bundling scheme is described that employs both permutation and cyclic-shift binding to enable the creation of practically unlimited hierarchically nested sequences overcoming the limitations that occur when using Eq. 1 or Eq. 4 for hierarchical encoding.

Bundled VSA HVs preserve vector dimension regardless of the number of combined sub-attribute HVs. However, the dimensionality in use limits the ability to decode sub-HVs within a bundled vector. In the Laiho model, cyclic-shift binding is achieved by a simple right-cyclic-shift of the SHV slot positions, and cyclic-shift unbinding is a simple left-cyclic-shift of the slots. Role-filler binding is achieved by adding the bit positions of the corresponding slots in the role SHV (e.g., bit position b_1) and filler SHV (e.g., bit position b_2) and then computing the sum of the two bit positions modulus the number of possible bit positions B (i.e., $b_n = (b_1 + b_2) mod\, B$). The bit at this position (b_n) is then set in the slot. Unbinding is a modulus subtraction operation (i.e., $b_1 = (b_n - b_2) mod\, B$). The SHV bundling operation is performed by slotwise addition operation by counting the number of bits that occur in the same position within each slot from all of the bundled SHVs. Sparsity is preserved by computing for each slot the bit position corresponding to the maximum number of co-occurring bits (i.e., the argmax) and setting the bit at this position in the final bundled SHV. If there are multiple bit positions with the same number of co-occurring bits, the bit at one of these positions is selected randomly or by a well-defined algorithm (e.g., lowest bit index). Therefore, the resulting bundled SHV has the same sparse structure as the component SHVs of which it is composed. It can be shown that for M slots and B bit positions per slot, to a first approximation, $D = M \log_2 B$, where D is the dimension of the equivalent dense HV.

III. Mapping sparse binary VSA operators onto SNN circuits

Mapping the sparse slot encoding vector model into an SNN representation is performed by treating each slot as a neuron and the time-to-spike as the bit position. In the following sections, a symbolic representation of an SNN circuit is used, and this is illustrated in Figure 1(a). In the circuit representation, neurons are shown as triangles, each with an input dendrite and an output axon. The axon of any neuron can connect to the dendrite of one or more other neurons via synaptic connections. The synaptic connection between an axon and a dendrite can introduce a time delay to the incoming spike before it stimulates the neuron associated with the dendrite to add or subtract from the neuron voltage.

How the neuron responds to the stimulus voltage depends on the neuron model. Sections III-A and III-B describe how SNN circuits can perform all of the required VSA operations using relatively simple neuron models. Section IV demonstrates results from simulations of the different circuits using the Brian 2 SNN simulation tool [17], therefore this paper uses Brian 2 terminology to describe the neuron model behaviour. All the neuron models in the following sections are defined by systems of differential equations controlled by user-defined parameters. Each neuron model also defines one or more spike threshold voltage levels that result in the neuron producing one or more output spikes on its axon whenever a spike voltage

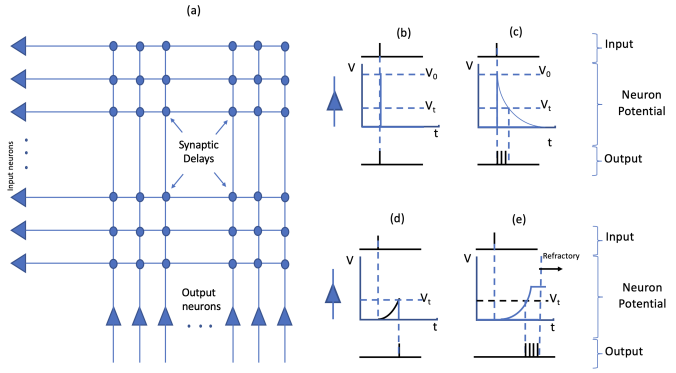

Fig. 1. Spiking Neural Network Model showing: (a) the model structure; different action potential effects (b) to (e)

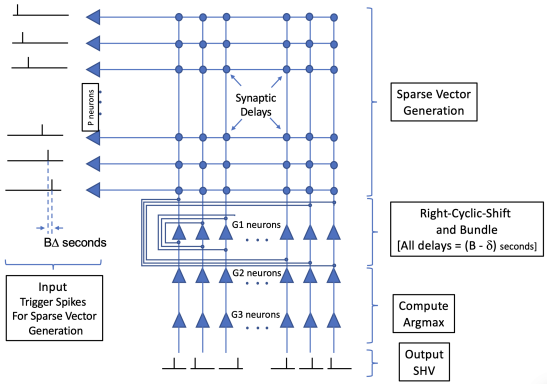

Fig. 2. Cyclic shift binding and bundling circuit

threshold is crossed. The Brian 2 neurons also allow the user to specify refractory properties. After the neuron fires a spike, it becomes refractory for a specific duration, and further spiking is inhibited during the refractory period. Figure 1 (b) - (e) shows several simple example neuron behaviours that are employed in the VSA circuit models and simulation results described in the remainder of this section and Section IV. Each neuron model shows an input spike and the resulting changes in neuron potential and the resulting output spike sequence.

A. Cyclic Shift Binding and Bundling Circuit

This section describes how the Laiho model can be used to implement cyclic-shift binding and bundling. Example results are presented in Section IV, and the reader should refer to these results to aid the understanding of the circuit descriptions.

1) Cyclic-Shift Binding and Bundling Circuit: Figure 2 shows the cyclic shift binding and bundling circuit. The circuit comprises three main components: SHV Generation, Right Cyclic Shift Binding and Bundling and a final argmax computation from which the encoded bundled SHV is output. The circuits are described using neuron groups. In the following circuit descriptions, the labelling of the different neuron groups corresponds to the naming convention used in the Brian 2 simulations presented in Section IV.

a) Sparse Hypervector Generation: SHV generation comprises two neuron groups, P and G1. Each of the G1 neurons corresponds to one slot in the Laiho model, so there are M G1 neurons. There is one P neuron for each vector in the SHV library, and so there are L P-neurons. The P-neuron axons are connected in a matrix structure to the G1 dendrites via synapses as shown in Figure 2 and so there are $M * L$ synaptic connections. The firing of a P-neuron is the trigger for generating a single corresponding SHV. Each P neuron axon connects to all G1 neuron dendrites through the corresponding synaptic connections. Each synaptic connection introduces a delay with a value between 0 and $B\Delta$ seconds, where Δ is the spike-time-delay between spikes corresponding to consecutive bit positions in that neuron representing the slot.

After the delay, the corresponding neuron is stimulated, and its neuron voltage is increased by V_s. All the G1 neurons have a threshold voltage of V_t ($V_t \leq V_s$). The reset condition for the G1 neuron model is similar to that shown in Figure 1 (b), i.e., the neuron has an exponential decay with a time constant τ seconds, chosen such that only a single spike is generated when a stimulus of V_s is received in any Δ time period. Therefore, the firing of the P1 neuron causes all G1 neurons to fire, each with a time-to spike delay determined by the corresponding synaptic delay. The spike firing pattern of the G1 neurons represents the required SHV encoding with one spike per neuron such that the time to spike represents the bit position in the Laiho model.

b) Right Cyclic Shift Binding and Bundling: Right Cyclic Shift binding of the spike encoded SHV is performed by connecting the output axon of each G1 neuron to the dendrite of the next neighbouring neuron in the group via further synaptic connections. As shown, the last neuron in the group has a feedback connection to the first neuron. The synaptic delay on these connections is $B\Delta - \delta$ seconds, where δ is the minimum neuron processing delay between a stimulus event on the dendrite and the generated output spike. This processing time also determines the maximum spike frequency of the neuron. Time $B\Delta$ is the cycle time between each binding and bundling operation. At the end of the first cycle, the feedback ensures that the encoded SHV has been right-cyclic shifted and adds its stimulus of V_s volts to the G1 neuron group. A second P neuron fires after $B\Delta$ seconds and after the corresponding synaptic delays, V_s volts are added to the corresponding G1 neuron voltages. If the two stimulus events occur more than Δ seconds apart, then the neuron decay constant is such that two spikes will be generated in the cycle, and these, in turn, will be fed back with the $B\Delta - \delta$ seconds time delay for the next cycle. When a third P neuron now fires at the beginning of the next cycle, another SHV is generated. If none of the three stimulus events occurs within the same Δ seconds period, then three spikes will be generated in that cycle. This process can be repeated up to the maximum binding capacity of an SHV with M slots and B bits per slot.

If stimulus events on the dendrite occur in the same Δ seconds period, corresponding to two bits in the same position, then the neuron voltage will be increased by V_s volts for each

stimulus in that time window. This voltage increase results in the neuron generating a short burst of spikes at a frequency of $1/\delta$ Hz (i.e., the maximum spike frequency as shown in Figure 1 (c), where the number of spikes is proportional to the size of the stimulus event. These spikes are then fed back to the next neuron resulting in a greater stimulus for that neuron in the next cycle. Section IV-A1 provides simulation examples that illustrate these activities.

c) Argmax Computations: The argmax computation is achieved using the two neuron groups, G2 and G3, as shown in Figure 2. The objective of the computation is to produce, after N bundling cycles, a single output spike from each of the G3 neurons where the spike from each G3 neuron occurs in the Δ second time period where the most spikes from all of the cyclically shifted SHVs occur. Both G2 and G3 have the same number of neurons (M) as the G1 group. The G2 neurons receive the output spikes from the G1 neurons in each cycle; however, the refractory properties of the G2 neurons are used to prevent any neuron firing for the first $N-1$ cycles. In the Nth cycle, the G1 neurons output all of the $M * N * (N-1)/2$ spikes that have been generated from bundling the N SHVs. The G2 neurons now process these spikes. The G2 neurons use a varying threshold to identify the time slot where the maximum number of spikes occur. When the first stimulus occurs, the corresponding G2 neuron fires and its neuron voltage is reset to V_0. The neuron voltage threshold V_t is then set to the measured neuron voltage at the time of the spike. This mechanism ensures that this will be the only spike unless a larger stimulus occurs within the $B\Delta$ slot time. If a larger stimulus occurs, then the G2 neuron will spike again, and the threshold will be increased such that no further spikes will occur without an even larger stimulus. The G3 neuron layer must select the last spike produced from the corresponding G2 neurons, and this is achieved using a neuron model with a linear decay and a time constant of $(V_1 - V_0)B\Delta$ seconds. At the beginning of each cycle, the neuron voltage is set to V_1 volts and a low threshold of V_0. On any stimulus, the neuron voltage is reset to V_1 so that, after the last spike in the Nth cycle, it will decay to spike at the corresponding time in the next cycle period. Section IV-A1 also provides simulation examples that illustrate these activities.

2) Cyclic Shift Unbinding and Clean-Up Memory Circuit: The corresponding Sparse Cyclic Shift unbinding and clean-up memory circuit is shown in Figure 3. It comprises a left-cyclic-shift operation followed by a parallel clean-up memory process that generates a spike corresponding to the best matching library SHV. The input to the circuit is the sparse bundled SHV generated by the binding and bundling circuit. The left cyclic shift operation is performed by a single group of M neurons (G4) with axon connections which produce a left cyclic shift of the received SHV with a cycle time of $B\Delta$ seconds. After each cycle, the resulting SHV is a noisy representation of the corresponding bundled SHV. A subset of the bits will still match each of the original SHVs bundled, starting with the last SHV to be bound and unbinding in the reverse order to the bundling.

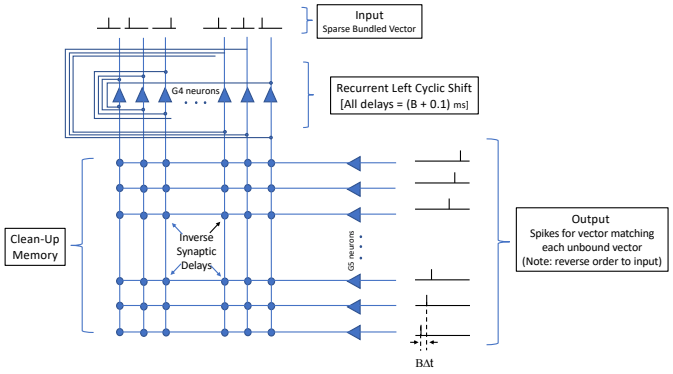

Fig. 3. Cyclic shift unbinding and clean-up memory circuit

The clean-up memory circuit comprises M input neurons (G4) and L output neurons (G5). The circuit is required to mirror the action of the SHV generation circuit such that when a noisy version of the SHV input from the G4 neurons matches the SHV that would be generated when the n^{th} P-neuron (P_n) spikes, the corresponding n^{th} G4 neuron ($G4_n$) in the clean-up memory is the only neuron that is sufficiently stimulated to spike. If a clean version of the corresponding SHV is required, the output from the L G5 neurons would then feed into a sparse SHV generation circuit replacing the P-neurons.

To achieve this objective, the axons of each G4 neuron connect to all L dendrites of the G5 neurons via synaptic connections. The synaptic delays are now calculated such that if the noisy unbound SHV is to match, the additional synaptic time delays will ensure that when m out of the M bits match, the corresponding G5 neuron will be stimulated with a voltage of mV_s volts in the last Δ second time slot of the current unbinding cycle. Expressing the time delays in the synaptic connections of the SHV generation as an $M * L$ matrix (P_matrix), the corresponding time delays of the clean-up memory $M * L$ matrix would be the transpose of the $(P\_matrix - B\Delta)$. The threshold V_t of the G5 neurons is chosen to ensure that only those neurons that match above the noise threshold generate a spike. So if N SHVs are bundled in the order P_n to P_1 the unbound output sequence will be in the reverse order $G5_1$ to $G5_n$ as shown in Figure 3. An important feature of this clean-up memory circuit is that matching against all L library SHVs is performed in parallel but requires L synaptic connections on a single dendrite. Simulation results for the cyclic shift circuits are presented in Section IV-A1.

B. Role-Filler Binding and Bundling

In this section, SNN circuits for role-filler binding and bundling are presented. Role-filler binding requires a circuit that can perform a modulus addition between the times of arrival of two input spikes and produce a single output spike at that position, see Figure 4(a). The neuron model for this, shown in Figure 4(b), uses a linearly increasing neuron voltage that is then held constant on the arrival of the first spike and then linearly decays on the arrival of the second spike. The neuron has two spike thresholds, voltages V_0 and V_t and the

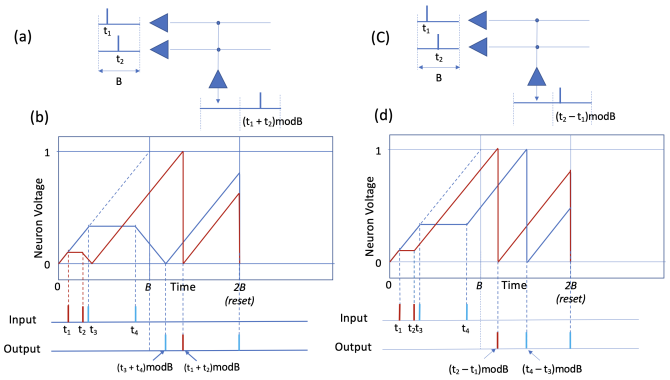

Fig. 4. Role-Filler binding operations

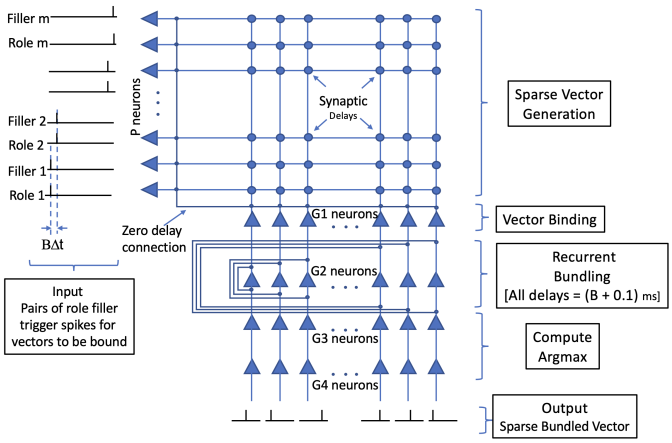

Fig. 5. Role-filler binding and bundling circuit

time constant of the linear increase or decrease is $(V_t - V_0)/B\Delta$. If the neuron voltage falls below the lower threshold V_0, the neuron fires and the neuron voltage reverts to the linear increase. If the upper threshold is exceeded, the neuron fires and the neuron voltage is reset to V_0. If two spike times (e.g., t_1, t_2) sum to less than $B\Delta$ the lower threshold is crossed at time $t_1 + t_2$ and the neuron fires and the neuron voltage reverts to a linear increase which reaches the upper threshold at time $B\Delta + t_1 + t_2$. This behaviour is equivalent to the required modulus addition in the next cycle. If the two spike times (e.g., t_3, t_4) sum to more than $B\Delta$, then the lower threshold is reached at time $B\Delta + t_3 + t_4$ which is the required modulus addition in the next cycle time window and the neuron fires. The neuron voltage then reverts to a linear increase, but this gets reset at time $2B\Delta$ and the process can repeat.

Role filler unbinding requires a circuit that can perform a modulus subtraction between the arrival times of two input spikes and produce a single output spike at that position. This is illustrated in Figure 4(c) and (d). To perform this operation, the neuron model is similar to the modulus addition, but now the neuron model produces a linearly increasing neuron voltage that is then held constant on the arrival of the first spike and then continues to linearly increase on the arrival of the second spike. In this case, the output spikes will always occur in the second cycle time window at $B\Delta + t_1 - t_2$ and $B\Delta + t_3 - t_4$ which are the required times for the modulus subtraction.

1) Role-Filler Binding and Bundling Circuit: Figure 5 shows the role-filler binding and bundling circuit. The model uses the same SHV generation circuit as the cyclic shift binding and bundling circuit. However, now the role and filler SHVs are generated concurrently with each pair being generated after a time delay of $2B\Delta$ seconds. There is also an additional connection from all P-neurons to all G1 neurons with zero time delay on the synaptic connections. This connection generates a reset spike that initialises the neuron voltage to V_0 at the start of each cycle. In this circuit, the G1 neurons perform the role filler binding operation using the modulus addition neuron model described above.

The bundling operation is now performed by an additional neuron group G2. These neurons have recurrent connections that feedback the output spike to the same neuron with a delay of $(2B\Delta - \delta)$ seconds to coincide with the arrival of the next bound SHV from the G1 layer. The G2 neuron model is identical to the G1 neurons in the cyclic-shift-binding and bundling circuit, resulting in an output SHV with approximately $N(N-1)/2$ spikes per neuron after N cycles. The argmax layer is also identical to that used for the cyclic-shift and is performed by the G3 and G4 neuron layers. At the end of N bundling cycles, the required role-filler bound and bundled SHV will be the SHV encoded by the M output spikes from the G4 neuron group.

2) Role-Filler Unbinding and Clean-Up Memory Circuit: Role-filler unbinding is the process of concurrently processing the bundled SHV with either a *role* SHV to obtain the noisy corresponding *filler* SHV or with the *filler* SHV to obtain the corresponding *role* SHV. The resulting noisy SHV is then processed by the clean-up memory to obtain an index to the corresponding clean SHV, which can then be recovered. The circuit shown in Figure 6 is used to achieve this. Here the bundled SHV is added to the sparse SHV library using the time delays for each neuron for the bound SHV encoding.

To obtain a filler SHV from the bound SHV, the P-neuron corresponding to role SHV is fired concurrently with the P neuron corresponding to the bundled SHV. The two SHVs then stimulate the G4 neurons that perform the modulus subtraction operation, and the noisy resulting encoded SHV is injected into the clean-up memory. The clean-up memory performs precisely the same operation as the cyclic shift circuits and the G6 neuron corresponding to the bound filler SHV fires. The process takes $B\Delta$ seconds when the next unbinding operation can be performed.

IV. Results Using Brian 2 Simulations

All of the results in this section have been generated using the Brian 2 simulator for spiking neural networks. Example results are described, and links to a Github repository containing commented python code for each model are provided. The results presented for the different circuits are intended to illustrate how the circuits perform and so circuit parameters

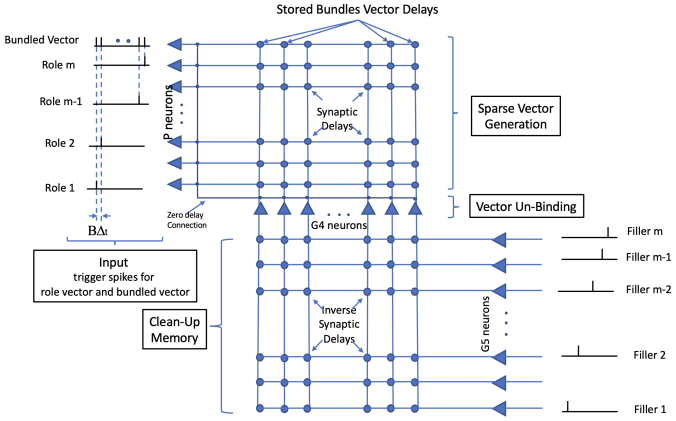

Fig. 6. Role-filler unbinding and clean-up memory circuit

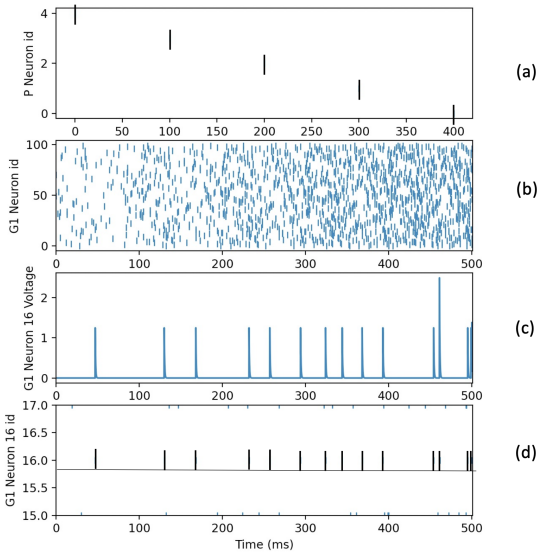

Fig. 7. Output from various monitoring points of the cyclic shift binding and bundling circuit. (a) Input spikes from the P1 neurons; (b) Output spikes from all G1 neurons; (c)Neuron voltage from neuron 16; (d) Output spikes from neuron 16

have been chosen to make the results easier to interpret. Therefore the following circuits use M =100, B=100 and L = 1000 and the number of bundled vectors is limited to N=5. The Brian 2 simulations use the default timings with a maximum spike frequency of 10KHz. Hence δ is 0.1ms. The time period for each bit position Δ is chosen to be 1ms. In the neuron models used, stimulus voltages in the range of 1 to 2 volts were used to make the simulation output easy to interpret. The reader is encouraged to use the Brian 2 models to explore the circuit performance for different values of the parameters which conform to the empirical outcomes obtained in the jupyter sample code when bundling larger numbers of SHVs. In the following section, the results from the cyclic-shift circuits are presented.

A. Cyclic Shift Results

1) Cyclic Shift Results: The cyclic shift circuit described in Section III-A1 is implemented in the Brian 2 neuromorphic simulator as two separate networks. The first network (*net1*) implements the SHV generation and the cyclic shift binding and bundling, see Figure 2. The second network (*net2*) implements the unbinding and clean-up memory circuit, see Figure 3. The corresponding Brian 2 model can be found at Compliant-Cyclic-Shift-net2 (**https://github.com/vsapy/DTIN07**).

a) Binding and Bundling Results: Figures 7 and 8 show the output from *net1* at various monitoring points in the circuit. In Figure 7(a), the input P neurons generate a sequence of pulses with a 100ms separation, where the y-axis is the neuron-id of the P neuron group. The first spike at time zero is generated for the fourth P-neuron ($P_{id} = 4$) and stimulates the corresponding G1 neurons after the appropriate time delays. The resulting SHV encoding can be seen in the first 100ms of Figure 7(b), which also shows the spikes for all 100 G1 neurons over five bundling cycles. In each cycle, the spikes are right cyclically shifted with a delay of 99.9 ms and so when the next P neuron fires ($P_{id} = 3$) at time 100ms, the stimulus of the corresponding G1 neurons will be from both spike events. This is shown in the time period $100ms - 200ms$ in Figure 7(b). The process then continues until all spikes have been bundled, resulting in the now densely encoded SHV between 400ms and 500ms. Figure 8(a) and (b) illustrates what happens for a single G1 neuron ($G1_{16}$). If the bundled vectors do not have spikes that occur in the 1ms time period, then each neuron stimulus is 1.25 volts. The neuron has a maximum spike frequency of 1KHz and each neuron only has time to generate one spike while the voltage is above the spike threshold. When the neuron stimulus from two or more spikes occurs in the same 1ms time period, as in the time period between 400 and 500ms, the neuron stimulus is doubled and two spikes are generated while the threshold is exceeded. This still results in 5 spikes being generated in this cycle.

The G1 neuron spikes are forwarded to the corresponding G2 neuron and the resulting G2 neuron voltage is as shown in Figure 8(d). The initial stimulus at 455ms exceeds the initial voltage threshold shown in Figure 8(e) and the neuron spikes, as shown in Figure 8(f). The spike threshold voltage is increased to this neuron voltage, preventing any later single spike stimulus from generating a spike. However, in this case, the neuron voltage resulting from the spike burst stimulus at 462ms is greater than this threshold and the neuron fires at this time as well. The spike threshold is increased again, so only a longer spike burst stimulus (i.e., greater argmax) would generate a later spike.

To output only the last spike to occur in the cycle, the G3 neuron group uses a linear decay as shown in Figure 8(g). Each time a spike is received from the G2 neuron, the G3 neuron voltage is reset and so from the time of the last spike in the cycle, the voltage decays to create a single spike in the next cycle, which in this case is the 500ms to 600ms time window as shown in Figure 8(h). The output from the *net1* circuit is the required sparse SHV encoding shown in 9(a) using a start time of zero to show the encoded spike time delays for the bundled

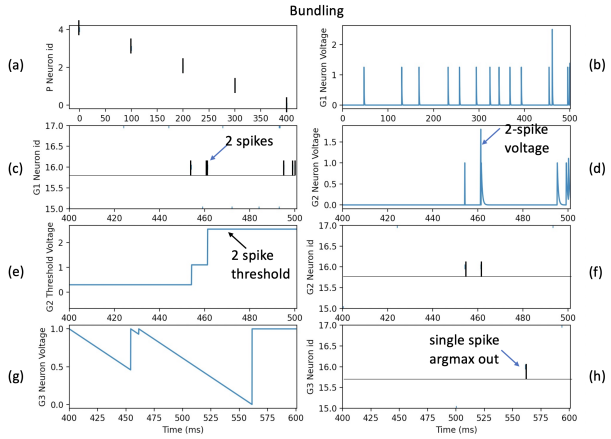

Fig. 8. Output from various monitoring points of the cyclic shift binding and bundling circuit for a single neuron. (a) Input spikes from the P1 neurons; (b) G1 neuron voltage; (c) Output spikes from G1 neuron; (d) G2 neuron voltage; (e) G2 neuron threshold voltage; (f) Output spikes from G2 neuron; (g) G3 neuron voltage; (h) single spike output from G3 neuron

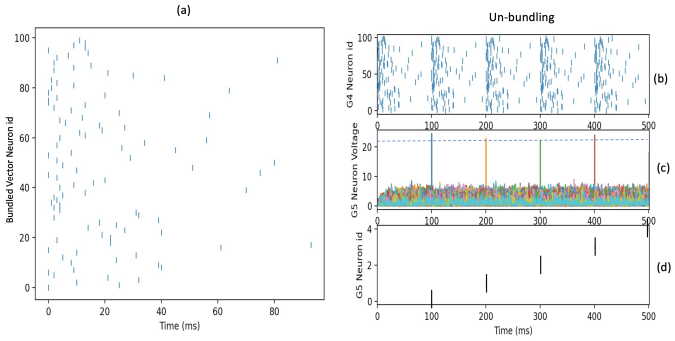

Fig. 9. Output from various monitoring points of the cyclic shift unbinding and un-bundling circuit. (a) Input spikes to G4 neuron group- bundled SHV output from G3 neuron group; (b) Output spikes from G4 neuron group; (c) G5 neuron group voltages; (d) Spike output from G5 neuron group showing index of matching library SHVs

SHV. Since the argmax operation chooses the first spike to occur where there is no later greater number of co-occurring spikes, the resulting bundled SHV is largely compressed in time except for neurons where there are co-occurring spikes.

b) Unbinding and Clean-Up Memory Results: The *net2* simulation implements the unbinding and clean-up circuit shown in Figure 3. The input to the circuit is the bundled encoded SHV shown in Figure 9(a). The effect of the left cyclic shift unbinding operation performed by the G4 neuron layer is shown in Figure 9(b) over a 500 ms period where each spike is shifted down by one neuron position in each cycle. Each of these SHVs is passed into the clean-up memory and the resulting neuron voltage in the G5 neuron layer is shown in Figure 9(c). The different colours identify the L different neurons in the SHV library. To illustrate how the circuit performs, the stimulus voltage from a single spike, V_t, has been set to 1 volt so that the measured voltage in volts corresponds to the number of co-occurring spikes.

In this example, random time delays are used to generate the library SHVs and so the neuron voltage from any non-matching SHV arriving at any 1ms time window will be random. The expected number of matching bits can be determined empirically using the slot encoding model directly (see the conventional python calculation at the start of each jupyter notebook in the GitHub library). In this case, the predicted number of matches for each of the 5 SHVs when unbound and compared to the clean SHV was 24,22,22,23,18 bits, respectively which match exactly the measured voltages from the clean-up memory circuit as shown in Figure 9(c).

To determine an appropriate spike threshold for the G5 neurons, a noise threshold V_n of 4 s.d above the expected noise is used, and so $V_n = 5volts$ in this case. The lower threshold for the co-occurring spikes to be detected (V_e) is set at 4 s.d below the expected mean and so $V_e = 5.39$ volts. Provided $V_n \leq V_e$, the spike threshold can be set at between V_n and V_e. Otherwise, successful unbinding of all N SHVs is not possible with the required error rate. In this case, the threshold voltage was set at 5.39 volts and the threshold equation only allows spikes in a 5ms time period around the expected time of the stimulus event. In Figure 9(d), the resultant firing of the G5 neurons is shown. As the bundled SHV unbinds, the neuron firing order is the reverse of the bundling order because the clean-up memory mirrors the SHV generation memory.

B. Role-Filler Circuit Results

Due to this paper's page limit requirements and because an explanation of the role-filler results would essentially be a repeat of the cyclic-shift results walkthrough, the reader is encouraged to run the jupyter notebook versions of the code available at Compliant-Role-Filler-net2 (**https://github.com/vsapy/DTIN07**) in order to inspect the role-filler circuit results.

V. Discussion

A. Estimated Energy Efficiency of the SNN Circuits

The results from the Brian 2 simulations have demonstrated that it is possible to implement VSA operations using energy-efficient SNN processing devices. The clean-up memory circuits can efficiently perform the SHV comparison operations as a parallel operation, where the number of comparisons is only limited by the number of synaptic connections that a neuron can support.

The amount of energy consumed by a neuromorphic processing device is a function of the number of spikes required to perform the computation; therefore, the *time-to-spike* implementation of the Laiho model [14] offers significant energy savings over SNN implementations that use conventional dense VSA models including Eliasmith's spike-frequency model.

As a guide, the TrueNorth experimental SNN processor consumes an estimated 45pJ per spike [18]. Thus, as an example, using a cyclic-shift unbinding circuit using $M = 1000$ neurons and $B = 1024$, for each unbinding operation, the energy cost is in the region of 45nJ and would take $B\Delta$ seconds to complete. Similar estimates can be made for the binding and bundling

circuits. In contrast, estimates for a CMOS cyclic-shift circuit are typically in the region of 100pJ per bit, [19]. Thus, the cyclic shift operation on a equivalent $D = 10000$ bit BSC HV (i.e., $D = M \log_2 B$) would require in the region of 1 μJ. Hence, the SNN process would potentially be approximately 20x more energy efficient. However, there is a clear trade-off between energy efficiency gain and processing time latency.

In the case of the clean-up memory circuit, the cost of performing the matching operations will depend on the cost of the synaptic delays rather than simply the number of spikes. Current SNN devices such as TrueNorth do not support time delays on each synaptic connection and so the energy cost of performing the clean-up memory operations will therefore depend on the efficiency with which future SNN neuromorphic devices can perform the required time delay synaptic processing. In terms of baseline comparison, an experimental Phase Change Memory (PCM) device has demonstrated the capability to perform a 300 vector clean-up memory for 10,000-bit HVs using just 9.44 nJ per query. An improvement in the total energy efficiency of x 117.5 compared to the equivalent CMOS implementation, [13]. These figures give some indication of the energy efficiencies that would be required for synaptic time delay processing for an SNN processor that could perform an equivalent SHV clean-up memory. However, this requirement would need to be balanced against the fact that the SNN processor could also perform all the other required VSA SHV operations.

VI. Conclusion

This paper has demonstrated that by using sparse encoded hypervectors, all VSA hyperdimensional computing operations can be performed using *time-to-spike* SNN circuits. Energy estimates are provided for performing typical VSA functions on neuromorphic processing devices, and energy efficiencies of x20 are estimated compared to performing the same operations using standard CMOS operation on 10,000 bit BSC HVs. Further energy savings can be made by reducing the number of neurons, but this is at the cost of extending the required processing time for each VSA operation. Recommendations for future SNN neuromorphic processor design to support the required synaptic time delays have been made.

Current work is focused on simplifying the binding and bundling circuits to minimise further the number of spikes required to perform these operations. Extending the VSA circuits to perform hierarchical binding and bundling, which requires cyclic-shift and role-filler operations to be combined into a single circuit, is also being investigated. Future work will investigate how SNN circuits can be developed to process semantic SHVs, using VSA to perform analogical mapping and abductive reasoning as a novel form of cognitive processing. The full capabilities that can be enabled by VSA operations in SNN circuits are still to be explored, however, this paper shows that hyperdimensional computing using *time-to-spike* SNN neuromorphic circuits offer the potential to develop a new generation of energy-efficient artificial intelligence capabilities.

Acknowledgment

This research was sponsored by U.K. Ministry of Defence under Agreement Number W911NF-16-3-0001. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the U.K. Ministry of Defence or the U.K. Government. The U.K. Governments are authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation hereon.

References

[1] P. Kanerva, "Hyperdimensional computing: An introduction to computing in distributed representation with high-dimensional random vectors," *Cognitive computation*, vol. 1, no. 2, pp. 139–159, 2009.

[2] C. Eliasmith, "How to build a brain: from function to implementation," *Synthese*, vol. 159, no. 3, pp. 373–388, 2007.

[3] D. Widdows and T. Cohen, "Reasoning with vectors: A continuous model for fast robust inference," *Logic Journal of the IGPL*, vol. 23, no. 2, pp. 141–173, 11 2014.

[4] R. W. Gayler, "Vector symbolic architectures answer jackendoff's challenges for cognitive neuroscience," *arXiv preprint cs/0412059*, 2004.

[5] T. A. Plate, *Distributed representations and nested compositional structure*. University of Toronto, Department of Computer Science, 1994.

[6] P. Kanerva, "Binary spatter-coding of ordered k-tuples," in *Artificial Neural Networks — ICANN 96*. Springer, 1996, pp. 869–873.

[7] C. Simpkin, I. Taylor, G. A. Bent, G. de Mel, and R. K. Ganti, "A scalable vector symbolic architecture approach for decentralized workflows," in *COLLA 2018 International Conference on Advanced Collaborative Networks, Systems and Applications*, 2018, pp. 21–27.

[8] C. Simpkin, I. Taylor, G. A. Bent, G. de Mel, S. Rallapalli, L. Ma, and M. Srivatsa, "Efficient orchestration of node-red iot workflows using a vector symbolic architecture," *Future Generation Computer Systems*, vol. 100, pp. 70–85, 2019.

[9] C. Simpkin, I. Taylor, G. A. Bent, D. Harbourne, A. Preece, and R. Ganti, "Constructing distributed time-critical applications using cognitive enabled services," *Future Generation Computer Systems*, vol. 111, pp. 117–131, 2020.

[10] C. Simpkin, I. Taylor, A. Preece, G. A. Bent, R. Tomsett, R. K. Ganti, and O. Worthington, "Coalition c3 information exchange using binary symbolic vectors," in *IEEE Military Communications Conference*, 2019, pp. 784–789.

[11] G. Bent, C. Simpkin, I. Taylor, A. Rahimi, G. Karunaratne, A. Sebastian, D. Millar, A. Martens, and K. Roy, "Energy efficient 'in memory' computing to enable decentralised service workflow composition in support of multi-domain operations," in *SPIE Proceedings*, vol. 11746, 2021, pp. 414–435.

[12] C. Simpkin, "A vector symbolic approach for cognitive services and decentralized workflows," Ph.D. dissertation, Cardiff University, Cardiff, Dec. 2021. [Online]. Available: https://orca.cardiff.ac.uk/146996

[13] G. Karunaratne, M. Le Gallo, G. Cherubini, L. Benini, A. Rahimi, and A. Sebastian, "In-memory hyperdimensional computing," *Nature Electronics*, vol. 3, no. 6, pp. 327–337, 2020.

[14] M. Laiho, J. H. Poikonen, P. Kanerva, and E. Lehtonen, "High-dimensional computing with sparse vectors," in *2015 IEEE Biomedical Circuits and Systems Conference (BioCAS)*, 2015, pp. 1–4.

[15] E. P. Frady, D. Kleyko, and F. T. Sommer, "Variable binding for sparse distributed representations: Theory and applications," *CoRR*, vol. abs/2009.06734, 2020.

[16] P. Kanerva, "Hyperdimensional computing: An introduction to computing in distributed representation with high-dimensional random vectors." *Cognitive Computation*, vol. 1, no. 2, pp. 139–159, 2009.

[17] M. Stimberg, R. Brette, and D. F. M. Goodman, "Brian 2: an intuitive and efficient neural simulator," *bioRxiv*, 2019.

[18] P. Merolla, J. Arthur, F. Akopyan, N. Imam, R. Manohar, and D. S. Modha, "A digital neurosynaptic core using embedded crossbar memory with 45pj per spike in 45nm," in *2011 IEEE Custom Integrated Circuits Conference (CICC)*, 2011, pp. 1–4.

[19] N. Joshi and R. Jangir, "Design of low power and high speed shift register," 2019.