International Journal of Computing and Digital Systems
ISSN (2210-142X)
Int. J. Com. Dig. Sys. 14, No.1 (Sep-2023)

http://dx.doi.org/10.12785/ijcds/140152

Neuromorphic Processor Design and FPGA Implementation for Handwritten Digits Employing Spiking Neural Network

Nagavarapu Sowmya[1], **Jitendra Kumar**[2], **Pradyut Kumar Biswal**[3], **Shirshendu Roy**[4] **and Subhrajit Pradhan**[5]

[1,2]*Department of Electronics and Communication Engineering, GIETU, Gunupur, India*
[3]*Department of Electronics and Communication Engineering, IIIT, BBSR, India*
[4]*Department of Electronics and Communication Engineering, DSU, Bengaluru, India*
[5]*Department of Electronics and Communication Engineering, GIFT, BBSR, India*

Received 15 May 2022, Revised 15 Jul. 2023, Accepted 31 Jul. 2023, Published 01 Sep. 2023

Abstract: Spiking Neural Network (SNN) is very popular and effective in modelling the physical neurons compared to other models of the neural network. Besides the software implementation of the neuromorphic processors, hardware implementation of the neuromorphic processors is also very important in order to apply it in real-time domain. In this work, a hardware efficient architecture of the neuromorphic processor is proposed. The proposed architecture is efficient in terms of low usage of memory elements and other hardware resources. Virtex-6 field programmable gate array (FPGA) development board is used to validate the proposed design. Fixed data format of width 18 is used in this work and 10-bit is reserved for the fractional part. The proposed architecture is applied to detect the handwritten digits. In this work, MNIST database is used to train and validate the SNN. The proposed architecture achieves 90% accuracy when used to recognize the handwritten digit data.

Keywords: Spiking Neural Network (SNN), Spike Timing Dependent Plasticity (STDP), Leaky Integrate and Fire (LIF), Field Programmable Gate Array (FPGA), Hand written digit recognition

1. INTRODUCTION

The human brain is considered as the most complex, energy-efficient system as it is responsible for managing the body functions, interpretation of information, taking proper actions and mostly for controlling the central part of the mind. Scientists have surveyed a lot and tried to introduce a network system that can mimic the biological brain and predict the probability of the output. Thus, artificial neural networks (ANN) are developed. There are various kinds of neural networks like, convolution neural networks (CNN), deep neural networks (DNN), feed-forward neural networks (FNN), recurrent neural networks (RNN) and spiking neural networks (SNN). Spiking are actually the one which closely replicate the human brain. The goal of SNN is to link the gap among neuroscience and machine learning. These proofs made researchers to focus on brain-inspired computing as a new approach to deal with growing multi-layered arithmetic calculations. Thus, in current era of artificial intelligence (AI), SNNs have been evolved and attracted everyone due to its asynchronous event-driven calculation and parallel architecture feature [1]. SNNs provide better efficiency in terms of energy and hardware compared to traditional ANNs. SNNs might use lateral inhibition technique (IT), which means that every neuron transfers its firing rate of spikes not only to next layer but also to the nerve cell in the same layer. The dismissal in the similar layer is considered as an indication to lower the membrane potential.

SNNs have been evolved in wide variety of applications like classification of images, detecting the objects, navigation and motor control [2]. In any platform, users choose between their implementations on the basis of speed, power, area and cost. Moreover, SNNs can be developed using software, analog/digital hardware. SNNs can be implemented on central processing unit (CPUs), graphics processing units (GPUs) or on other microcontrollers using Python or C language. Secondly, dedicated hardwares like FPGA, application specific integrated circuit (ASIC), digital signal processing (DSP) used to design these networks. However, same importance is given to analog hardware implementation of these networks such as CMOS technologies for designing of SNN. The main reason behind the software based and analog based models of SNNs is that they are attractive and can handle large number of data but they are inefficient in terms of energy and speed. As a result, parallel digital neuromorphic systems running on FPGA and ASICs

E-mail address: nagavarapu.sowmya@giet.edu, jitendrakumar@giet.edu, pradyut@iiit-bh.ac.in, shirshenduroy88@gmail.com, subhrajit.pradhan@gmail.com

Algorithm 1 SNN model based on STDP Learning Rule

Input: External input spikes vector (E) and different constants.
Output: The weight matrix (W).

```
 1: Initialization Set $r^0 = y$, $\Lambda^0 = [\ ]$, $\Omega = [\ ]$ and $i = 0$.
 2: for k = 1:M do
 3:   for t = 1:$t_m$ do
 4:     for i = 1:N do
 5:       $V_{min}(i) = V_{min}(i-1) + K_{syn}\sum_{j=1}^{N} W(j,i)S(j) + K_{ext}E(i) - V_{leak}$
 6:     end for
 7:     for i = 1:N do
 8:       if $V_{min}(i) \leq V_{th}$ then
 9:         $S(i) = 1$; $V_{min}(i) = V_{rest}$
10:       else
11:         $S(i) = 0$;
12:       end if
13:     end for
14:     for $i = L_{first} : L_{last}$ do
15:       if S(i) = 1 then
16:         for $j = L_{first} : L_{last}$ do
17:           if S(j) = 1 then
18:             $A_p(j,i) = A_p(j,i) + offset1$
19:             $A_q(j,i) = A_q(j,i) + offset1$
20:           else
21:             $A_p(j,i) = A_p(j,i)e^{t/\tau} + offset1$
22:             $A_q(j,i) = A_q(j,i)e^{t/\tau} + offset1$
23:           end if
24:           $W(j,i) = W(j,i) + A_p(j,i) + A_q(j,i) + offset3$
25:         end for
26:       end if
27:     end for
28:   end for
29: end for
```

are becoming popular [3], [4].

Neuromorphic machines owe the concept of SNNs and works when electric signals or spikes passes through artificial neurons. Neuromorphic computing works on creation of brain's architecture and data processing capabilities with new computer hardware chips and software algorithms. This system uses computational hardware pattern on human brain by considering the connection of neurons which make it possible to encode the information and are efficient than standard computer chips. SNNs with neuromorphic processor incorporates parallel based architecture which can model the dynamics of neurons in real time using software algorithm. In general, the dynamics of neurons is achieved by various neuron models such as Izhikevich neuron model (INM) [5], Hodgkin-Huxley model (HH) [6] and Leaky integrate and fire (LIF) [7] neuron model. Along with that, spike timing dependent plasticity (STDP) rule [8] is used to calculate and update the membrane potential, synaptic weights and firing rate of neuron which are the major parameters taken into consideration while building the neuromorphic processor. LIF and STDP plays a major role in implementation of SNN for a digital neuromorphic processor as they closely analyze the behaviour of human brain and nervous system very efficiently. In comparison to von Neumann architectures, the neuromorphic system promises high processing speed capability with low power consumption as they can resolve the issues based on their unfasten parallelism and less energy consumption with respect to spiking dynamics [9], [10].

Many works on development of architecture for neuromorphic processor have been reported in literature. Biao F, et al. [11] proposed a low power design for digital recognition with the help of multi-kernel parallel technique. A real-time digital neuromorphic system to simulate the large conductance based on SNN is reported in [12]. A simple and computationally efficient design of spike response model with STDP learning has been proposed [13]. An organized quantitative, qualitative and guidelines for optimal temporal encoding of SNNs is presented [14]. A hardware emulator for an RRAM-based neuromorphic chip on FPGA [15] is demonstrated. Labelled data related procedure [16] is formulated for multilayer SNNs to compute the gradient components. A small and low power neural processing system [17] is reported for diagnosis of pathological conditions. Hong T., et al. [18] worked on implementing the neuromorphic computing system on FPGA for image classification on handwritten-digits. A digital neuromorphic system of the pair-based and triplet-based STDP [19] is demonstrated by linear approximations. Wenzhen G., et al. [1] depicted a parameter optimization scheme and neuromorphic platform for digital design using Euler and Runge-Kutta method on FPGA. A supervised learning algorithm to study timed multiple spikes in a multi-layer SNNs is illustrated in [20]. A parallel neuromorphic processor architecture for SNNs on FPGA is proposed [7]. Zhang et al. [21] reported a compact, programmable, unique and scalable neuromorphic design. An overview on implementation of spike response model and temporal coding for SNNs is described [22]. A wide-ranging survey of the study and inspirations for neuromorphic computing is reported [23]. J. S. et al., [24] presented a comparative analysis of STDP learning systems with the help of shift register and counter. The software and hardware implementation on decoupling the SNN calculation work from the network gateways to provide multiple network gateways is demonstrated [25]. The Hebbian model of development and learning for pre and post synaptic neurons using STDP was demonstrated [26]. The processing of information by neurons and neural systems using integrate and fire neuron model was described [27]. P. U. Diehl et al. proposed the unsupervised learning of digit recognition using STDP [28]. An event driven MNIST data for SNN using Poisson's distribution has been described [29]. The triple core digital neuromorphic processor's design on an Altera Quartus II FPGA is reported [30]. The paper [31] presented neurons and synapses with STDP integrated circuits on 90 nm CMOS with lowest energy level. LIF neuron circuits for second generation Brain Scales mixed signal 65 CMOS neuromorphic hardware along with the winner take mechanism of STDP for cortical processing was demonstrated [32]. Supervised learning in SNN using triggered Normalized Approximate Descent algorithm for the problem of handwritten digit recognition was demonstrated [33]. Review of low power SNNs accelerators with both software algorithm and neuromorphic hardware implementations and unsupervised STDP learning rule would be a possible solution to adapt changes has been demonstrated [34], [35], [36].

Technical details of the research methodology of this research are summarized by the following steps.

- Step 1: Handwritten digit data acquired from standard MNIST database.
- Step 2: MNIST digit data is then converted into csv file.
- Step 3: SNN input must be in spike form, thus the digit image data is converted into spike trains using Poisson's distribution in MATLAB.
- Step 4: STDP learning algorithm is used to train and test the handwritten digit images for $N = 212$ and 1591.
- Step 5: STDP algorithm is used during training phase, to update the membrane potential, regularly monitor the firing activity, and update the synaptic weights.
- Step 6: During testing phase, the updated weights is used to detect any digit.
- Step 7: Finally, the input digit image and the output digit image are plotted using MATLAB.

Methodology for hardware implementation of the proposed neuromorphic processor are summarised by the following steps

- Step 1: All the parameters of STDP learning algorithm used in MATLAB are converted using conventional 2's complement method for 18-bit word size in Verilog.
- Step 2: The individual blocks of neuromorphic processor and its global timing and control signals are implemented in Verilog.
- Step 3: The proposed processor is implemented for $N = 212$ and 1591.
- Step 4: The parallelization of the proposed architecture is achieved by folding the architecture by factor of f_k.
- Step 5: The hardware implementation of SNN on FPGA is carried out for $K = 128, 256$ and 512.
- Step 6: Virtex-6 field programmable gate array (FPGA) development board is used to validate the proposed design.
- Step 7: The size of the registers, memory blocks and power consumption are carried out and compared with other works.

The paper is organized as follows. The theoretical details of SNN are explained in Section 2. STDP algorithm for updating the weights and the weight matrix are also defined in this section. The proposed architecture is described in details in Section 3. The performance of the planned neuromorphic processor is analyzed in Section 4. Comparative analysis of the architecture is also carried in this section. Concluding remarks on the proposed work are made in Section 5.

2. THEORETICAL BACKGROUND

A. *SNN Topology*

The projected spiking neural network shown in Figure 1 is employed for letter or digit recognition that enters the input layer as encoded external input spikes. This topology involves N number of neurons. In this SNN model, M number of neurons are used in the input layer, L number of neurons are used in output layer and H number of inhibitory neurons are used. Out of H inhibitory neurons, six are for input layer and one is used for output layer. Inhibitory neurons establish strong negative feedback to the excitatory neurons to meet winner-take-all situation in the network. The synaptic weights associated with inhibitory neurons are fixed, whereas the feed-forward nerve cell that connects input and output layer are plastic. These plastic synapses have the ability to change their strength on the basis of aided biologically-inspired STDP learning rule. During the learning procedure, each synapse adjusts its weight on the basis of relative timing of spikes with respect to presynaptic and postsynaptic neurons [26]. In our work, LIF is used to determine the neuron's dynamics as it is well-suitable for implementing SNN for designing the proposed neuromorphic processor. The generalized LIF model is described by the following Euler equation [27].

$$\tau \frac{dV_{min}(t)}{dt} + V_{min}(t) = R.I(t) \tag{1}$$

where V_{min} represents the cell membrane's potential, τ is the time constant, R is the resistance and I is the current. For digital hardware implementation, the LIF neurons are shortened and digitalized as follows:

$$V_{min}(t) = V_{min(t-1} + \sum_{i=0} nI_{tr} - V_{leak} \tag{2}$$

where I_{tr} tells the rate at which the neurons transmit the chemical message to the human brain and V_{leak} is the constant leakage value for the neuron's membrane potential i.e., V_{min}.

B. *Algorithm of Spiking Neural Networks Based on STDP Rule*

The Algorithm 1 describes SNN learning process on the basis of STDP rule. Handwritten images are taken from

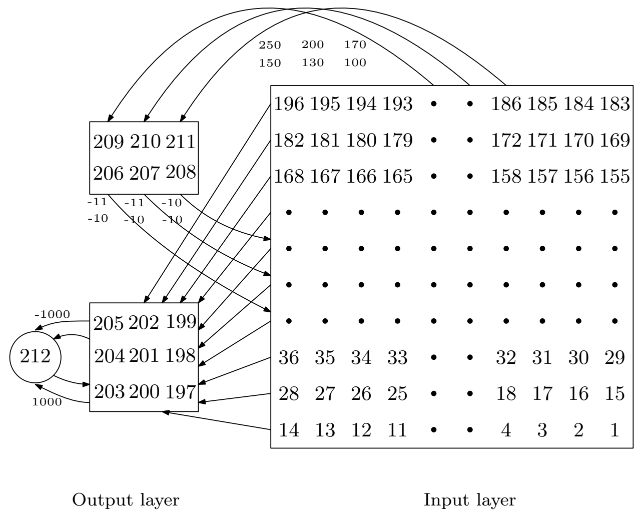


Figure 1. Spiking neural network for character recognition as mentioned in [7].

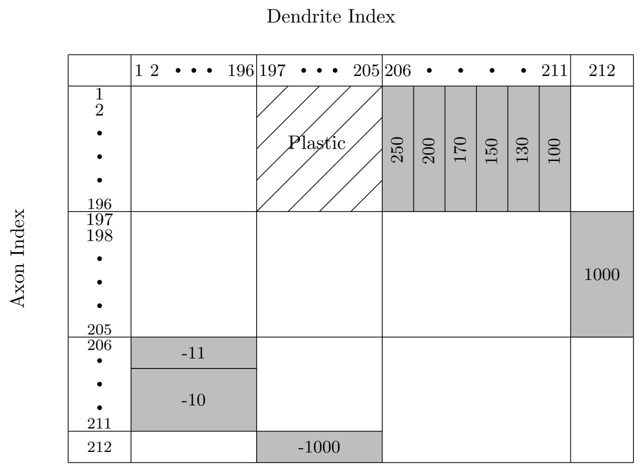


Figure 2. The connections between neurons as mentioned in [7].

MNIST dataset to identify the performance during testing and training phase . An image consists of M pixels and each pixel is converted to spike strain of length N according to the technique mentioned [28], [30]. Then these strains are taken as an input to the algorithm. V_{min} is the potential gradient of the membrane, W is weight, E is the external input spike, S specifies if a neuron excites or not and N is the number of neurons. L_{first} to L_{last} are the indices of first and last excitatory neurons in the output layer respectively. Similarly, M_{first} and M_{last} indicates first and last excitatory neuron in the input layer respectively. The SNN algorithm runs for certain number of iterations (t_m) for one pixel to settle down to a correct output. For each iteration, the membrane potential V_{min} gets updated by incrementing the scaled version of synaptic weights (W) based on firing flag (S). V_{leak} acts as a constant leakage voltage for V_{min}. K_{ext} is a random number that represents amplitude of external input spikes and also produces irregular inserted currents. Once the V_{min} gets updated, then it is equated with V_{th} to check the firing activity of neuron. If V_{min} is greater than or equal to V_{th}, then the neuron fires and flag S is set. Simultaneously, V_{min} is reset to the resting potential V_{rest}. If V_{min} is less than V_{th} then S is reset.

After the updation of membrane potential, the change in weight is calculated using STDP rule. During the firing of specific neuron in present iteration, the pre-synaptic neurons tend to receive the most recent firing times. Each iteration is denoted as 't' and 'τ' represents the time taken by each iteration. A_p and A_q finds the maximum number of synaptic changes. Post updation of weights leads to new iteration.

C. The Weight Matrix

The weights matrix shown in Figure 2 uses an ideal $N \times N$ crossbar array to visualize the interconnections between neurons. Every column signifies the networks among a specific neuron and all its synapses. The weights corresponding to pre-synaptic neurons (M_{first} to M_{last}) and post synaptic neurons (L_{first} to L_{last}) are only needs to be updated. Thus, these weights are called plastic. An example of this weight matrix is shown in Figure 2 for $N = 212$, $M = 196$ and $L = 9$. During the training phase, weights correspond to this plastic region to be updated in each iteration. The remaining synaptic weights associated with inhibitory neurons are constants and it is observed that there is no connection between neurons within the same layer.

3. PROPOSED NEUROMORPHIC PROCESSOR

In this work, a parallel neuromorphic processor is proposed which is scalable to any number of neurons. The proposed neuromorphic processor shown in Figure 3 consists of four major blocks called NU block, STDP block, LAU block and W_logic block. NU block is responsible for generating and storing firing flags whereas LAU block is used to update the membrane potential based on weights and firing flags. STDP block is used to update the weights in the training process. A neuromorphic processor control (NMP) block is used to generate all the control signals. W_logic block is used to efficiently handle the weights. This processor has two modes, training and execution. In the training process, the processor is learned means the with the help of STDP unit, weights get updated here. In the execution step, the updated weights are used to detect any digit. Thus, in the execution step, STDP block is bypassed. The parallelization of the proposed architecture is achieved by folding the architecture by factor of f_k. Total number of neurons are divided into f_k groups. Thus, weights and flags are read in f_k phases from NU block and the W_logic blocks respectively. This way the architecture supports any number of neurons. The value of f_k is 2 and 13 for $N = 212$ and 1591 respectively if $K = 128$ is chosen. Each block is described in detail below.

A. NU Block

The proposed NU block is shown in Figure 4. The potential of the membrane corresponding to a nerve cell is compared with the threshold value (V_{th}). Firing activity

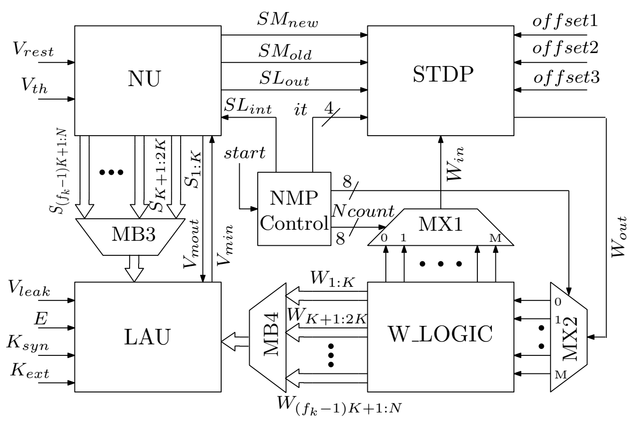


Figure 3. Proposed Neuromorphic Processor.

flags $(S_1, S_2, \ldots, S_K)$ are set if V_{min} is greater than or equal to V_{th}. These firing activity flags are stored initially in a memory named S_mem. Once the identification of firing flags is finished, S_mem is read. Output of the S_mem goes to a register bank where N 1-bit registers are connected in series. f_d is the control register block that stores the data when the control signal is high. Computation of three type of firing flags is shown in Figure 4 which are SM_{new}, SM_{old} and SL_{out}. SM_{new} flags are generated in a particular iteration corresponding to excitatory neurons in the input layer. SM_{old} flags are SM_{new} flags generated in the previous iteration. SL_{out} flags represents the firing activity flags corresponding to the excitatory neurons in the output layer. Initially, when L count is zero then SL_{int} control signal becomes high and this signal chooses Sin as SM_{new}. Once the reading of SM_{new} flags is completed then one value of SL_{out} flag is read serially through S_{M+1} flag. Fig. 4 stores the existing membrane potential and firing flags where S acts as a controlling parameter for all global timing and control signals. Whereas firing time, existing membrane potential, and firing flags was illustrated [7]. This is why the architecture of NU, STDP units are different than the architectures mentioned [7].

B. LAU Block

LAU block, shown in Figure 5, helps in updating the membrane potential (V_{min}). This block takes membrane potentials and firing activity flags $(S_1, S_2, \ldots, S_K)$ from NU block as inputs. This block also receives weights from the W_logic block. The spike strains corresponding to an image pixel is also input to this block. A buffer module is used which receives weights $(W_1, W_2, \ldots, W_K)$ at one end and firing activity flags $(S_1, S_2, \ldots, S_K)$ at another end. The output of this module is considered as input to the adder tree. Output of the adder tree goes to another adder for accumulation. The accumulation of weights takes f_k cycles. This accumulated output is then multiplied by a k_{syn} by another multiplier. Simultaneously, V_{mout} and V_{leak} are passed to an adder. A multiplexer is used which has two inputs named k_{ext} and 0. The multiplexer is controlled by the external input spikes (E). This way the constant $kext$ is

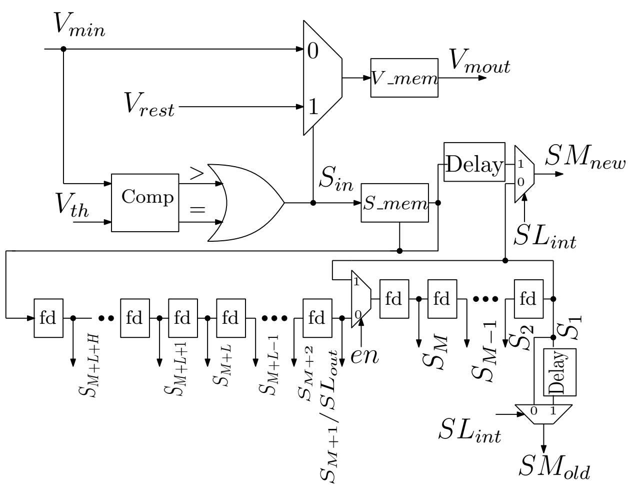


Figure 4. Proposed architecture of NU block.

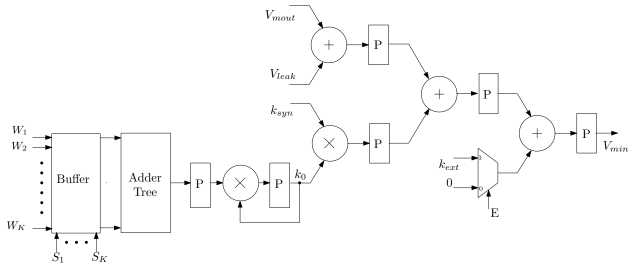


Figure 5. Proposed architecture of LAU block.

added to the equation randomly. Lastly, the output of the LAU block (V_{min}) is again used as an input to the NU block and stored in V_mem. The LAU block generates one V_{min} sample in f_k cycles.

C. STDP Unit

The proposed STDP unit is shown in Figure 6. This unit helps in updating the synaptic weights on the basis of pre-synaptic and post-synaptic neurons. A_{pi} and A_{qi} are the synaptic parameters that identifies the changes in weights for each iteration. The input A_{pi} comes from Ap_mem memory block which is then sent to right shift (RSH1) block as shown in Figure 6. RSH1 block is used for right shifting the input data by 1 bit. Similarly, A_{qi} is read from Aq_mem memory block and it is also passed through a RSH1 block. A lookup table (LUT) block stores the value of the expression $e^{(it/\tau)}$where τ is a constant and it is the iteration count which is input to the STDP block from the controller block. Output from the LUT block goes to a tri-state buffer. The tri-state buffer is controlled by the SM_{new} control input. Scaled A_{pi} and A_{qi} values are multiplied by output of the tri-state buffer. The STDP block receives the scalar weight inputs from the W_logic block. The weights are also scaled by a RSH1 block. In the path of A_{pi}, A_{qi} and Win three constants $offset1$, $offset2$ and $offset3$ are added respectively. A_{po} and A_{qo} values are written into Ap_mem and Aq_mem respectively. The output of the STDP

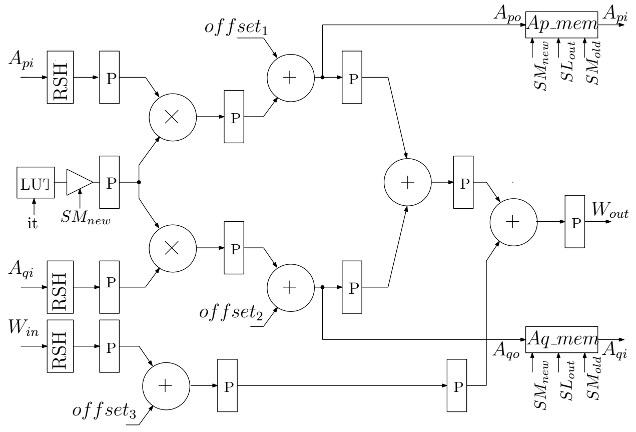


Figure 6. Proposed architecture of STDP unit.

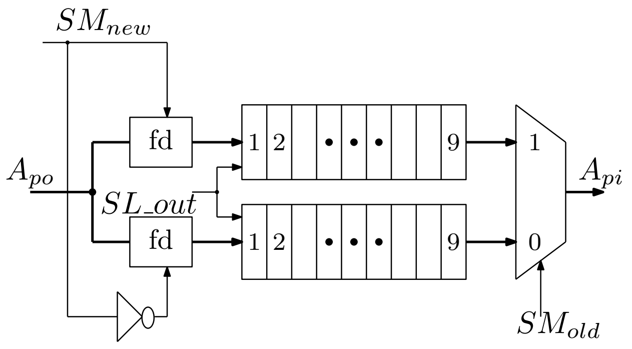


Figure 7. Proposed strategy to store Ap and Aq values.

block is W_{out} which is again written into W_mem.

D. Scheme for Storing A_p and A_q

The values of A_p and A_q are needed to be stored in memory properly. It is observed that in any given iteration only two values of A_p and A_q are produced. Thus, it is not required to store the entire matrix of A_p and A_q. In this work, the A_p and A_q values are efficiently stored in Ap_mem and Aq_mem. The scheme of storing the A_p and A_q values is shown in Figure 7. The firing flags SM_{new} which comes from NU block goes to a f_d block and the inverted version of SM_{new} goes to another f_d block. The unique values of A_{po} are stored in the control registers. But SL_{out} flag decides whether these values will be stored in the memory or not. The reading of the A_p or A_q values are done through a 2:1 multiplexer. The multiplexer is controlled by the previous value of firing activity flag (SM_{old}).

E. W_logic Block

All the neurons in the input or in output layer are connected through the weights. This connectivity is shown in Figure 2. All the columns of the weight matrix take participation in the operation but only the weights belonging to plastic region are updated. Also, the weights which does not belong to plastic region are constant. This is why it is not required to store the complete weight matrix.

In this work, a novel strategy has been used to reduce the number of digital storages to store the weights. In this strategy, memory blocks are only used to store the weights that belong to plastic region as shown in Figure 2. The proposed W_logic block is shown in Figure 8. The W_logic block is divided into two sections. In the first section, a memory block, W_mem is designed which stores the weights in the plastic region. In this block, M number of dual ports RAMs are used. Each RAM has capability to store L words.

In the second section, a novel control strategy is applied to pass the constant weights. This strategy is shown in Figure 8. Two multiplexer banks (MB) are designed. MB1 selects between output of W_mem and 0. MB2 selects between output of MB1 and output of another Read Only Memory (ROM). ROM stores the constant value of weights corresponding to inhibitory neurons. Two control signals $ctrl$ and $ctrl1$ controls MB1 and MB2 respectively to generate W_1 to W_M. The constants 250, 200,170,150,130 and 100 are stored in ROM as shown in Figure 2. A counter is used to count up to N and output of this counter (N_{count}) is compared with N by COMP1 block to generate weights from $W_M + 1$ to $W_M + L$. Similarly, N_{count} is compared with M to generate weights from $W_M + L + 1$ to $W_M + L + 6$. The COMP3 block is used to compare N_{count} with $L_{last} + M + L$ to generate W_N (i.e., $W_M + L + H$). This process of generating the constant weights based on comparison is shown in Table I.

TABLE I. The passing of constants for efficient evaluation of weights based on N_{count}.

$1 \leq N_{count} \leq M$	W_1 to W_M
$M \leq N_{count} \leq M + L$	W_{M+1} to $W_{M+L)}$
$M + L \leq N_{count} \leq M + L + (H - 1)$	W_{M+L+1} to $W_{M+L+(H-1)}$
$N_{count} = M + L + H$	W_1 to $W_N = W_{M+L+H}$

4. PERFORMANCE EVALUATION OF THE PROPOSED DESIGN

The objective of this work is to propose a neuromorphic processor which is efficient in terms of hardware resources, timing complexity and also in terms of dynamic power consumption. Besides this, the architecture should also achieve certain acceptable accuracy. In order to validate the design, image pixels are first converted to strain of pulses and these bits are serially fed to the FPGA using serial to parallel interface (SPI) protocol. The proposed architecture is implemented on Virtex-6 FPGA device using fixed point arithmetic. 18-bit is used to represent a data where 10-bits are used to represent the fractional parts. The proposed processor is first tested by taking $N = 212$ neurons where $M = 196$, $L = 9$ and $H = 7$. In this case, $f_k = 2$ clock cycles are needed to accumulate the weights according to flags for $K = 128$. After successful validation of this prototype, the proposed design is again tested for $N = 1591$ number of neurons. In this case, $M = 784$, $L = 800$, $H = 7$ and $f_k = 13$ for $K = 128$. The proposed design is validated by detecting handwritten digits. The MNIST database is used to train the SNN. Once the SNN is trained, the architecture

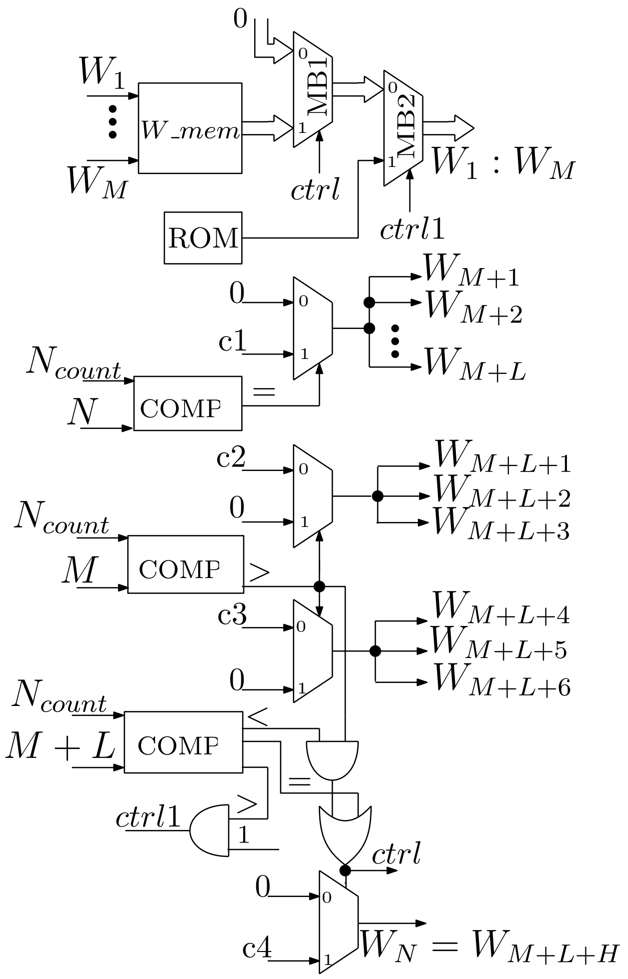


Figure 8. Proposed scheme for evaluation of weights.

is verified by taking the digits from the same MNIST set. The proposed work achieves 90% accuracy for N =1591 and 92% accuracy for N = 212. The comparison of input handwritten digits obtained from MATLAB with recognized digits by FPGA is illustrated in Figure 9. The performance of the proposed design is described below.

A. Hardware Complexity

The parallel architecture mentioned in [7] is not scalable as it is difficult to design adder tree for huge number of neurons. Fully parallel architecture of neuromorphic processor becomes a challenge if number of neurons is beyond 200 [35]. Thus, a foldable LAU unit is proposed in this paper. This architecture is f_k times folded because of which it is scalable to any number of neurons. Whereas M-BRAMs are used to store the weights and four memory elements are used to store synaptic parameters values. The limits would be long computation time with increase in number of neurons. The purpose behind choosing the specific number K is based on the excitatory neurons in the input and output layer along with the neurons in the plastic region and inhibitory neurons of SNN topology.

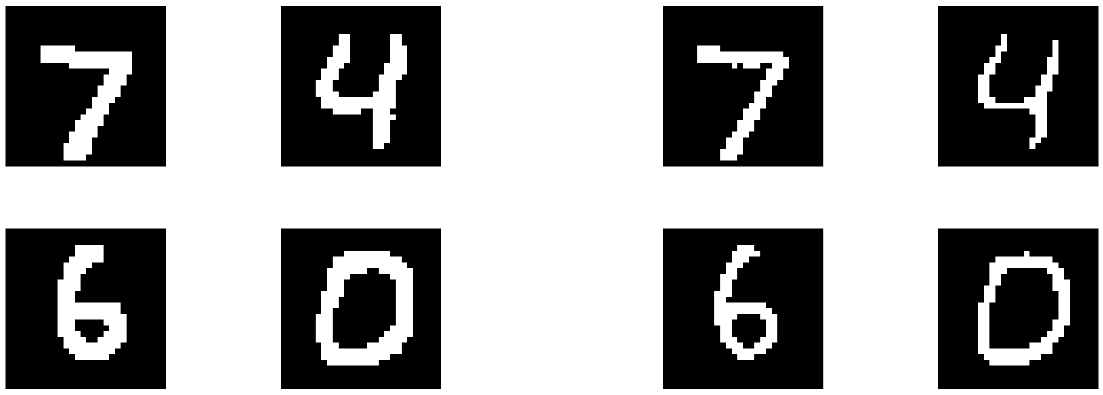
The introduced architecture is parallel and hardware efficient than the architecture reported [7] and is more efficient version of STDP algorithm. Here, the memory consumption for synaptic weights is optimized. In addition to that, this architecture needs only $72 * L$ bits to store the synaptic parameter (A_p and A_q) matrices, compared to $6272 * L$ bits used [7]. In this work, hardware implementation of SNN on FPGA is carried out for K = (128, 256 and 512) whereas same parallelization factor (K = 128) has been applied [7]. The design in [7] is not fully parallelized and runtime is not improvised when the number of neuron processing units gets increasing for the larger communication. Thus, in present work, the architecture is f_k -folded and runtime has been improvised compared to earlier reported literature [7]. Earlier research presented many architectures and did not imply on the parallel architecture of neuromorphic processor [7], but to improve the speed we need to adopt the parallel architectures. LAU for the parallel architecture [7] involves an adder tree which is having N inputs. This scenario is not realizable when size of neurons is huge. The proposed folded LAU architecture, Neuron Unit and STDP units are different from the architecture mentioned [7]. Earlier research doesn't consider the concept of registering the time flags. Instead, the proposed method uses registering the flag status, and reducing the memory requirement to store the A_p and A_q elements. The hardware complexity of each block is summarized in Table II.

B. Timing Complexity

The proposed architecture is parallel and f_k times folded compared to recent research [35] and makes this architecture faster than previous serial design. Out of the four major blocks, LAU block starts processing first. The time complexity of this block is $T_{LAU} = f_k \times N + l_{LAU}$ where l_{LAU} is the latency of the LAU block and it is equal to $(k + 4)$ where $K = 2^k$. The NU block starts processing simultaneously and thus no extra time is needed to write the flags. Once the flags are registered, the STDP module starts processing. The timing complexity of this module is $T_{STDP} = L \times M + l_{STDP}$ where l_{STDP} is the latency of this module and it is equal to 5 clock cycles. The algorithm for SNN is iterated for maximum t_m iterations. Thus, the total timing complexity for the proposed architecture to process an image with M pixels is:

$$T_{SNN} = \begin{cases} Mt_m(f_k N + l_{lau} + LM + l_{stdp}, & \text{for training} \\ Mt_m(f_k N + l_{lau}, & \text{for testing} \end{cases} \tag{3}$$

According to this equation of timing complexity, the proposed processor takes 7.94 seconds for training when a 28×28 handwritten digit is taken as an input and 0.8 seconds to test a handwritten image. In both cases viz. in training and testing phase, the value of t_m = 16 and f_k = 4 is considered. Earlier result includes the value of f_k = 2 or 13, and K = 128 for N = 212 or N = 1591 respectively. f_k = 4 is further considered to estimate and verify high speed with affordable power. The proposed architecture achieves maximum frequency of 125 MHz. Thus, execution time of the proposed processor is computed by taking T_{clk}= 8ns.



(a) Original handwritten digit from the MNIST dataset (b) Output of the FPGA implementation of SNN

Figure 9. Comparison of input handwritten digits obtained from MATLAB with recognized digits by FPGA

TABLE II. Block wise hardware complexity.

Blocks	18-bit Comp.	1-bit Mux.	1-bit Reg.	Memory (Words)	Multiplier	Add./Sub.
NU	1	21	$\approx N$	$(N+1)$	0	0
LAU	0	18	$18K-90$	0	1	$4K-4$
STDP	0	36	$72L+288$	0	2	5
W_logic	3	$36M+72$	0	ML	0	0
Others	0	$36(M-1)+19K(f_k-1)$	0	0	0	0

*: 18-bits are considered as one word.

The architecture is routed on the FPGA device which is Virtex-6 FPGA using Xilinx 14.7 EDA tool. The best timing parameter is achieved after many trials by meeting all the setup and hold time constraints.

C. Power Consumption

Power consumption is an important design parameter to evaluate the performance of an architecture and is measured using Xilinx XPower Analyzer. The consumption of power is accredited to utilization of memory blocks and hardware resources. Dynamic power is measured by considering all the switching activity of the design. The proposed design consumes 418 mW and is higher than that of serial neuromorphic architectures [7] because of more hardware resource requirements for minimum time consumption.

D. Comparative Analysis

The comparison among different SNN implementation methods with our work is summarized in Table III. In this simulation work, MNIST handwritten digit image is converted into spike trains using Poisson spike distributions. The resultant spike train is sent to SNN followed by STDP learning algorithm. The Euler method was implemented for neuron dynamics. The long simulation time is due to the huge number of iterations and large number of neurons as we have trained and tested SNN by taking N = 212 and 1591. Consequently, the existing membrane potential also gets updated for subsequent increasing iteration steps. This work achieved an accuracy of 90% for 800 output neurons. The long simulation time is for solving the differential equations of Euler and Runge-Kutta (RK3) method where an accuracy of 89.5% and 89.7%, is achieved respectively [1]. Whereas in [7], Euler method was used to implement SNN in MATLAB for 800 output neurons and achieved 90% accuracy. Frenkel et al. [35] implemented 0.086-mm^2 64k synapse 256-neuron online-learning digital spiking neuromorphic processor in 28 nm FDSOI CMOS and obtained 84.5% efficiency. An accuracy of 88.6% is achieved for SNN implementation in software simulation [28].

The hardware requirement of the proposed architecture is compared with the existing works in Table IV. The introduced architecture is parallel and hardware efficient than the architecture reported [7], [1]. In addition to that, this architecture needs only $72 * L$ bits to store the synaptic parameters (A_p and A_q) matrices, compared to $6272 * L$ bits used [7]. The number of slice registers and slice LUTs used [7] is almost double compared to the same used in present work for K = 128. This justifies the extra usage of DSP and memory blocks. In this work, hardware implementation of SNN on FPGA is carried out for K = 128, 256 and 512 whereas same parallelization factor (K = 128) has been applied [7], [1]. Recent literature [1], reports the NAND gate counts for both slices and BRAMs followed by an estimation method [36] to compare the area occupation of different designs. As a result, the memory size shown in

TABLE III. Comparison of our work with previous works in terms of accuracy.

Methodology	Accuracy (%)	Ref.
Euler Method, FPGA	92, 90	This work
Euler and RK3 Method	89.5, 89.7	[1]
Euler Method (MATLAB)	90	[7]
CMOS	84.5	[35]
FPGA	88.6	[28]

TABLE IV. Comparison of the proposed architecture with the previous work in terms of hardware resources.

Architectures	This Work			Std. Mul. [7]	Apprx. Mul. [7]	Euler Method [1]	RK3 Method [1]
	K=128	K=256	K=512				
Slice LUTs	32,349	32,384	51,865	97,287	93,232	45,531 32987	88,053 54313
Slice Registers	29,043	29,053	30,091	58,826	58,345	-	-
BRAM	395	396	396	34	34	289,101	289,101
DSP	20	155	538	-	-	-	-
Runtime (ns)	8.12	8	7.94	16.8	16.8	0.053	0.029

Table IV is comparatively smaller with respect to present work. The design [7] is not fully parallelized and runtime is not improvised when the number of neuron processing units gets increasing for the larger communication. Thus, in present work, the architecture is f_k -folded and runtime has been improvised compared to reported literature [7]. The proposed architecture is almost two times faster than the architecture [7] as shown in Table IV. Whereas, with the parameter optimization scheme, genetic algorithm, and parallel architecture design [1], the runtime has been improvised compared to present work which is shown in Table IV. However, present work has less hardware complexity and better accuracy than the reported literature.

5. CONCLUSION

In this work, a hardware and memory efficient folded parallel architecture of neuromorphic processor is proposed. The architecture uses very few memory elements compared to that of recent implementations of neuromorphic processor to accommodate the weights. This efficient usage of memory elements leads to the lower power consumption. The architecture is parallel but scalable to any number of neurons. Scalability factor is introduced by partially folding the architecture by the factor of f_k. The speed of the architecture can be increased by changing the value of f_k or decreasing the folding factor. Further increase in the value of f_k will increase the resources and the power requirement.

References

[1] W. Guo, H. E. Yantır, M. E. Fouda, A. M. Eltawil, and K. N. Salama, "Toward the optimal design and fpga implementation of spiking neural networks," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 33, no. 8, pp. 3988–4002, 2021.

[2] R. M. Morillas and P. Ituero, "Stdp design trade-offs for fpga-based spiking neural networks," in *2020 XXXV Conference on Design of Circuits and Integrated Systems (DCIS)*. IEEE, 2020, pp. 1–6.

[3] P. Soleimani Abhari and F. Razaghian, "Hardware implementation of lif and hh spiking neuronal models," *Signal Processing and Renewable Energy*, vol. 3, no. 1, pp. 35–42, 2019.

[4] S. Xiang, Y. Zhang, J. Gong, X. Guo, L. Lin, and Y. Hao, "Stdp-based unsupervised spike pattern learning in a photonic spiking neural network with vcsels and vcsoas," *IEEE Journal of Selected Topics in Quantum Electronics*, vol. 25, no. 6, pp. 1–9, 2019.

[5] F. Yu, G. Feng, J. Xiong, J. Wu, J. Wang, and A. He, "An asynchronous pipeline based implementation of membrane potential of if model," in *2018 IEEE International Conference on Electron Devices and Solid State Circuits (EDSSC)*. IEEE, 2018, pp. 1–2.

[6] M. Saggar, T. Meriçli, S. Andoni, and R. Miikkulainen, "System identification for the hodgkin-huxley model using artificial neural networks," in *2007 International Joint Conference on Neural Networks*. IEEE, 2007, pp. 2239–2244.

[7] Q. Wang, Y. Li, B. Shao, S. Dey, and P. Li, "Energy efficient parallel neuromorphic architectures with approximate arithmetic on fpga," *Neurocomputing*, vol. 221, pp. 146–158, 2017.

[8] A. Taherkhani, A. Belatreche, Y. Li, and L. P. Maguire, "A supervised learning algorithm for learning precise timing of multiple spikes in multilayer spiking neural networks," *IEEE transactions on neural networks and learning systems*, vol. 29, no. 11, pp. 5394–5407, 2018.

[9] R. Tapiador-Morales, A. Linares-Barranco, A. Jimenez-Fernandez, and G. Jimenez-Moreno, "Neuromorphic lif row-by-row multiconvolution processor for fpga," *IEEE transactions on biomedical circuits and systems*, vol. 13, no. 1, pp. 159–169, 2018.

[10] F. Hadaeghi, "Neuromorphic electronic systems for reservoir computing," *Reservoir Computing: Theory, Physical Implementations, and Applications*, pp. 221–237, 2021.

[11] B. Fang, Y. Zhang, R. Yan, and H. Tang, "Spike trains encoding optimization for spiking neural networks implementation in fpga," in *2020 12th International Conference on Advanced Computational Intelligence (ICACI)*. IEEE, 2020, pp. 412–418.

[12] S. Yang, J. Wang, B. Deng, C. Liu, H. Li, C. Fietkiewicz, and K. A. Loparo, "Real-time neuromorphic system for large-scale conductance-based spiking neural networks," *IEEE transactions on cybernetics*, vol. 49, no. 7, pp. 2490–2503, 2018.

[13] T. Iakymchuk, A. Rosado-Muñoz, J. F. Guerrero-Martínez, M. Bataller-Mompeán, and J. V. Francés-Víllora, "Simplified spiking neural network architecture and stdp learning algorithm applied to image classification," *EURASIP Journal on Image and Video Processing*, vol. 2015, pp. 1–11, 2015.

[14] B. Petro, N. Kasabov, and R. M. Kiss, "Selection and optimization of temporal spike encoding methods for spiking neural networks," *IEEE transactions on neural networks and learning systems*, vol. 31, no. 2, pp. 358–370, 2019.

[15] T. Luo, X. Wang, C. Qu, M. K. F. Lee, W. T. Tang, W.-F. Wong, and R. S. M. Goh, "An fpga-based hardware emulator for neuromorphic chip with rram," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 39, no. 2, pp. 438–450, 2018.

[16] N. Zheng and P. Mazumder, "Online supervised learning for hardware-based multilayer spiking neural networks through the modulation of weight-dependent spike-timing-dependent plasticity," *IEEE transactions on neural networks and learning systems*, vol. 29, no. 9, pp. 4287–4302, 2017.

[17] F. C. Bauer, D. R. Muir, and G. Indiveri, "Real-time ultra-low power ecg anomaly detection using an event-driven neuromorphic processor," *IEEE transactions on biomedical circuits and systems*, vol. 13, no. 6, pp. 1575–1582, 2019.

[18] T. Hong, Y. Kang, and J. Chung, "Insight: An fpga-based neuromorphic computing system for deep neural networks," *Journal of Low Power Electronics and Applications*, vol. 10, no. 4, p. 36, 2020.

[19] M. Nouri, M. Jalilian, M. Hayati, and D. Abbott, "A digital neuromorphic realization of pair-based and triplet-based spike-timing-dependent synaptic plasticity," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 65, no. 6, pp. 804–808, 2017.

[20] D. Zhao, Y. Zeng, T. Zhang, M. Shi, and F. Zhao, "Glsnn: A multi-layer spiking neural network based on global feedback alignment and local stdp plasticity," *Frontiers in Computational Neuroscience*, vol. 14, p. 576841, 2020.

[21] C. Zhang, G. Qiao, S. Hu, J. Wang, Z. Liu, Y. Liu, Q. Yu, and Y. Liu, "A versatile neuromorphic system based on simple neuron model," *AIP Advances*, vol. 9, no. 1, p. 015324, 2019.

[22] A. Rosado-Muñoz, M. Bataller-Mompeán, and J. Guerrero-Martínez, "Fpga implementation of spiking neural networks," *IFAC Proceedings Volumes*, vol. 45, no. 4, pp. 139–144, 2012.

[23] C. D. Schuman, T. E. Potok, R. M. Patton, J. D. Birdwell, M. E. Dean, G. S. Rose, and J. S. Plank, "A survey of neuromorphic computing and neural networks in hardware," *arXiv preprint arXiv:1705.06963*, 2017.

[24] J. Sim, S. Joo, and S.-O. Jung, "Comparative analysis of digital stdp learning circuits designed using counter and shift register," in *2019 34th International Technical Conference on Circuits/Systems, Computers and Communications (ITC-CSCC)*. IEEE, 2019, pp. 1–4.

[25] R. Lent, "Towards cognitive networking using fpga-based acceleration," *IEEE Journal of Radio Frequency Identification*, vol. 5, no. 3, pp. 222–231, 2021.

[26] S. Song, K. D. Miller, and L. F. Abbott, "Competitive hebbian learning through spike-timing-dependent synaptic plasticity," *Nature neuroscience*, vol. 3, no. 9, pp. 919–926, 2000.

[27] A. N. Burkitt, "A review of the integrate-and-fire neuron model: I. homogeneous synaptic input," *Biological cybernetics*, vol. 95, pp. 1–19, 2006.

[28] P. U. Diehl and M. Cook, "Unsupervised learning of digit recognition using spike-timing-dependent plasticity," *Frontiers in computational neuroscience*, vol. 9, p. 99, 2015.

[29] M. Fatahi, M. Ahmadi, M. Shahsavari, A. Ahmadi, and P. Devienne, "evt_mnist: A spike based version of traditional mnist," *arXiv preprint arXiv:1604.06751*, 2016.

[30] Y. Qi, B. Zhang, T. M. Taha, H. Chen, and R. Hasan, "Fpga design of a multicore neuromorphic processing system," in *NAECON 2014-IEEE National Aerospace and Electronics Conference*. IEEE, 2014, pp. 255–258.

[31] J. M. Cruz-Albrecht, M. W. Yung, and N. Srinivasa, "Energy-efficient neuron, synapse and stdp integrated circuits," *IEEE transactions on biomedical circuits and systems*, vol. 6, no. 3, pp. 246–256, 2012.

[32] S. A. Aamir, Y. Stradmann, P. Müller, C. Pehle, A. Hartel, A. Grübl, J. Schemmel, and K. Meier, "An accelerated lif neuronal network array for a large-scale mixed-signal neuromorphic architecture," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 65, no. 12, pp. 4299–4312, 2018.

[33] S. R. Kulkarni and B. Rajendran, "Spiking neural networks for handwritten digit recognition—supervised learning and network optimization," *Neural Networks*, vol. 103, pp. 118–127, 2018.

[34] D.-A. Nguyen, X.-T. Tran, and F. Iacopi, "A review of algorithms and hardware implementations for spiking neural networks," *Journal of Low Power Electronics and Applications*, vol. 11, no. 2, p. 23, 2021.

[35] C. Frenkel, M. Lefebvre, J.-D. Legat, and D. Bol, "A 0.086-mm ̃212.7-pj/sop 64k-synapse 256-neuron online-learning digital spiking neuromorphic processor in 28-nm cmos," *IEEE transactions on biomedical circuits and systems*, vol. 13, no. 1, pp. 145–158, 2018.

[36] V. Berg, J.-B. DORe, and S. Mayrargue, "Silicon area of fbmc receivers for cmos 65nm and comparison to ofdm receivers," in *2017 European Conference on Networks and Communications (EuCNC)*. IEEE, 2017, pp. 1–5.

Nagavarapu Sowmya completed her M. Tech in Electronics and Communications Engineering from Gandhi Institute of Engineering and Technology University, Gunupur, India in 2019. Currently, she is continuing her Ph.D. at GIET University, Gunupur, India. Her research areas are VLSI design, Biomedical, Neural networks.

Jitendra Kumar received the Ph.D. degree in nanoelectronics from Indian Institute of Technology, Guwahati, India, in 2019. He is currently working as an Assistant Professor in GIET University Gunupur, Odisha, India. His research interests include MEMS/NEMS, Sensor modelling and VLSI design.

Pradyut Kumar Biswal received the M. Tech in electronics and Ph.D. degree in electronics engineering from the Visvesvaraya National Institute of Technology, Nagpur, India, and Indian Institute of Technology, Kharagpur, India, in 2002 and 2011, respectively. He is currently an Assistant Professor with the IIIT Bhubaneswar, India. His research interest includes VLSI Architecture design for signal and image processing algorithms, biomedical signal and image processing, and hyperspectral image processing.

Shirshendu Roy has received the B.E. and M.E. degrees in ECE from the IIEST, Shibpur, India, in 2010 and 2016 respectively. He has earned Ph. D degree from the department of ECE, NIT Rourkela. Presently he is working as assistant professor in Dayananda Sagar University, Bengaluru. His current research interest includes compressed sensing, FPGA based implementation of algorithms (signal, image or video processing algorithms, machine learning algorithms, artificial neural networks etc.), low-power architecture design, ASIC, application of FPGA for IoT applications.

Subhrajit Pradhan Subhrajit Pradhan completed his Ph.D. at Berhampur University, India in Optical Communications. He is currently working as a Principal in GIET, Ghangapatana, Bhubaneswar.