CORDIC-SNN: On-FPGA STDP Learning with Izhikevich Neurons

Moslem Heidarpur, *Student Member, IEEE*, Arash Ahmadi, *Senior Member, IEEE*, Majid Ahmadi, *Life Fellow, IEEE*, and Mostafa Rahimi Azghadi, *Member, IEEE*

***Abstract*—This paper proposes a neuromorphic platform for on-FPGA online Spike Timing Dependant Plasticity (STDP) learning, based on the COordinate Rotation DIgital Computer (CORDIC) algorithms. The implemented platform comprises two main components. First, the Izhikevich neuron model is modified for implementation using the CORDIC algorithm, simulated to ensure the model accuracy, described as hardware, and implemented on FPGA. Second, the STDP learning algorithm is adapted and optimized using the CORDIC method, synthesized for hardware, and implemented to perform on-FPGA online learning on a network of CORDIC Izhikevich neurons to demonstrate competitive Hebbian learning. The implementation results are compared with the original model and state-of-the-art to verify accuracy, effectiveness, and higher speed of the system. These comparisons confirm that the proposed neuromorphic system offers better performance and higher accuracy while being straightforward to implement and suitable to scale.**

***Index Terms*—Izhikevich neuron, biological neuron model, CORDIC, digital implementation, neuromorphic, STDP, FPGA, online, on-FPGA, spiking neural network.**

I. Introduction

HIGHLY parallel, energy efficient, fault tolerant, and compact neuromorphic learning systems promise alternative devices for solving engineering problems [1] and powerful tools to understand properties of biological neural networks [2]. Several such systems have already been introduced and used [3]–[7] for various applications such as pattern cognition, signal processing, and autonomous robots [8]–[13].

These neuromorphic systems typically include a large number of neurons, synapses and their interconnecting structure on hardware. They provide real-time simulation, regardless of the size of the network, are parallel, and energy efficient [14]. The performance of such systems at a higher level depends on the neuron, synapse and learning models and at a lower level on the circuits realizing such units [15].

Current neuromorphic research has led to the development of a plethora of models to mimic real neurons with different levels of abstraction in biological details. Biologically-plausible models, such as Hodgkin Huxley [16] describe cellular phenomena and properties of the individual biological components. Such low-level models, impose more computation cost, making it difficult to simulate large-scale networks. On the other hand, biologically-inspired models such as Izhikevich [17] and models in [18]–[22], aim to mimic the biological neurons to the best degree of accuracy. Such models can reproduce most of the firing patterns of real neurons and are easier to couple to other spike-oriented units. Moreover, high-level Integrate and Fire (IF) [23] is another computationally efficient neural model, but cannot exhibit many essential features of the biological neurons as observed in experiments [24]. As far as neuromorphic computing is concerned, simpler models are cheaper, faster, and more energy efficient. Nevertheless, the choice of models depends on the application of the device to be designed. To perform computations with SNNs only a simple IF or Exponential IF (EIF) may be enough to act as a thresholding box. However, for research in neuroscience, biologically plausible models have higher flexibility in mimicking biology. Here, we have chosen the Izhikevich neuron for simulation and Field Programmable Gate Array (FPGA) implementation, because while being computationally efficient, it produces biologically plausible firing patterns.

After selecting the neuron model, a proper Spiking Neural Network (SNN) topology should be chosen. This depends on a number of factors such as the level of abstraction, the targeted application, available hardware, and the learning algorithm. A variety of spiking network topologies have been used in neuromorphic systems such as recurrent [25], feed-forward [26], winner-take-all [27], and probabilistic [28]. Subsequently, the SNN learning method should be selected based on factors such as network topology, whether the learning should be on-chip or off-chip, be supervised or unsupervised, etc. Previous hardware implementations of SNN adopt many of these approaches [29]–[32]. Among them, Spike-Timing Dependent Plasticity (STDP) is the most favored for unsupervised online training of feed-forward networks which is believed to be closer to biology [33]. As a result, many neuromorphic architectures have used various techniques to implement STDP-based spiking networks [34]–[39]. Similarly, this paper uses a novel technique based on CORDIC algorithm, described in the following sections, to realize an online STDP-learning architecture in hardware.

Considering hardware implementation platforms, they could be divided into three major categories as analog [6], [34], [36], [40], [41], digital [4], [7], [42] or mixed analog-digital [3], [35], [37] systems, each with its advantages and disadvantages. Two classes of digital systems are FPGAs and Application Specified Integrated Circuits (ASICs). Comparing these two classes, logic components in FPGA devices could

M. Heidarpur, A. Ahmadi, and M. Ahmadi are with the department of Electrical and Computer Engineering, University of Windsor, Ontario, Canada (e-mail:{heidarp@uwindsor.ca,m.ahmadi@uwindsor.ca, arash.ahamadi@uwindsor.ca).

M. Rahimi Azghadi is with the College of Science and Engineering, James Cook University, Townsville, QLD 4814, Australia (e-mail:{mostafa.rahimiazghadi@jcu.edu.au).

easily change with a configuration bitstream result from HDL synthesizers providing a cheap and flexible platform. In ASIC devices, on the other hand, a simple change in design could result in a new development cycle, which is expensive and prolonged. However, when using FPGAs as the implementation platform, one should take into consideration the limited FPGAs resources, which makes it crucial to employ them effectively for the best performance and the lowest cost.

To that end, the first challenge is to implement the neuron model as efficient and fast as possible. This paper utilizes CORDIC to calculate Izhikevich neuron differential equations. CORDIC is used to exclude the use of multipliers which are area-intensive and slow arithmetic operators in FPGAs. In order to increase the performance and size of the network, several techniques have been previously utilized to decrease the multiplication cost. These include bit serial and reduced range precision multipliers, stochastic-based neurons, replacing multiplication with add & shift operations, and Look Up Tables (LUTs) [43].

A number of FPGA implementations of Izhikevich neuron are available in literature. In [44], a rotate-and-fire digital spiking neuron model has been implemented that can reproduce five type of inhibitory responses as an asynchronous sequential logic circuit. In [45], an asynchronous cellular automata-based neuron model is presented. In [46], the continuous nullclines are approximated to cellular space for a low-cost neuron implementation. Reference [47] presents a piece-wise linear approximation [48] of the Izhikevich model to achieve multiplier-less hardware for lower cost and higher speed. Further, reference [49] utilizes CORDIC algorithm to design a low power digital circuit for this neuron. Compared with previous works, the CORDIC-based method presented here results in neurons requiring fewer resources and operating at a higher frequency.

In addition, to implement the STDP algorithm, the CORDIC exponential core in [50] was adopted to compute STDP function with high precision while requiring low resources.

Different method have been used by researchers to implement STDP algorithm. One of the common methods is to use Address-Event Representation (AER) data protocol [35]. Reference [51] utilizes piece-wise linear approximation (PWL) technique to implement the exponential term in STDP and a counter to store spike events. Moreover, a dedicated plasticity processor was used in [52]. In another paper, authors used a simplified multiplier to reduce the STDP implementation cost [53]. In this work, to implement the STDP algorithm, the CORDIC exponential core in [50] was adopted to compute STDP function with high precision while requiring low resources. To account for the spike timings required for STDP, a shift register was utilized to store the firing times of pre and post-synaptic neurons in order to determine the time differences and calculate synaptic weight updates. This is a novel technique that exploits a distributed memory to realize biological networks.

The rest of this paper is organized as follows. Section II reviews the Izhikevich neuron and STDP learning algorithm and further presents CORDIC modified models, computer simulations, and investigation of accuracy through errors analysis and studying the network behaviors. Section III discusses FPGA implementation procedure and compare achieved result with previous works. Finally, Section IV concludes the paper.

II. CORDIC NEURON AND NETWORK MODEL

A. CORDIC Izhikevich

1) Izhikevich neuron: Izhikevich neuron is a two-dimensional model, which consists of two coupled Ordinary Differential Equations (ODEs) as:

$$\frac{dv}{dt} = 0.04v^2 + 5v + 140 - u + I \tag{1}$$

$$\frac{du}{dt} = a(bv - u) \tag{2}$$

and a reset condition as:

$$if \quad v > 30mv \quad then \quad \begin{cases} v \rightarrow v_r \\ u \rightarrow u_r = u + d. \end{cases} \tag{3}$$

Here, Eqs. 1 and 2 describe membrane potential v, recovery variable u and applied current I. Other dimensionless parameters are

- a: Time scale of the recovery variable;
- b: Sensitivity of the u to v;
- c: After-spike reset value of v;
- d: After-spike reset value of u;

With adjustment of these variables, Izhikevich model is capable of replicating several firing patterns exhibited by biological neurons such as tonic spiking, adaptation, initial or regular bursting, transient spiking, and irregular spiking [54].

2) CORDIC Izhikevich: CORDIC is an iterative algorithm originally developed in [55] and thereafter generalized for calculation of hyperbolic and exponential functions, multiplications, divisions and square roots. CORDIC only requires simple shift and addition operations, which can be cheaply implemented on hardware hence making it an appropriate choice for fast and low-cost hardware implementations.

The algorithm for CORDIC calculation of square term in Eq. 1 is shown in Fig. 1. The FOR loop in line 4 calculates

```
1 square(x)
2 y=x;
3 z=0;
4 for i=-k:n
5 {
6  if (x > 0)
7  {
8   x = x - 2^(-i);
9   z = z + y*2^(-i);
10 }
11 else
12 {
13 x = x + 2^(-i);
14 z = z - y*2^(-i);
15 }
16 }
17 return z;
```

Fig. 1. The CORDIC code for calculation of square function.

the square(x) to the n bit precision. The x register keeps track of rotation direction in each iteration where z accumulates the result. In this approach, calculating to $k+n$ bit precision

is equal to rounding of multiplication to $k+n$ bit without calculating unnecessary bits. Choosing n is a trade-off between computation complexity and precision where k depends on the domain of the square function. Since the membrane potential of the neuron ranges between -100 and 30, k is set to 6 so that its two's power ($2^6 = 64$) is greater than $100/2 = 50$ and therefore the algorithm can keep up splitting the v to reach the value of v^2. To further evaluate the effect of n on the neuron behavior, we define four models with n=6, n=8, n=10 and n=12, naming them *IzhCOR*6, *IzhCOR*8, *IzhCOR*10, and *IzhCOR*12, respectively. This will help to compare simulation and implementation results in terms of deviation from the original model and hardware cost. The indicated code is most useful for a fixed point hardware but it can be modified to make it applicable to floating point hardware as well.

3) Simulation Results: Fig. 2 compares computer simulation of multiplication and CORDIC-based square functions. As this figure shows, two graphs are very close and only by zooming in small range the difference is visible (Error analysis is further presented in the next Section).

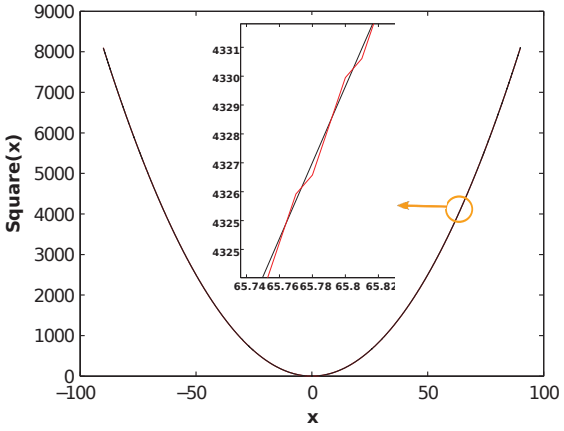


Fig. 2. Computer simulation of multiplication and CORDIC-based square function. Here, black and red lines show multiplication, and CORDIC square functions, respectively. The difference between two lines is only visible by zooming in a small range.

Fig. 3 shows the computer simulation of Izhikevich and modified CORDIC model for different neuronal behaviors. For an identical applied current, responses are very similar and there is no distinctive difference. However, these results only indicate resemblance of models for one specified value of applied current. Therefore, the resemblance of models for a wide range and different random values of stimulation currents are investigated as follows.

First, ODEs in both modified and original models was set equal to zero as

$$\frac{dv}{dt} = 0 \quad and \quad \frac{du}{dt} = 0, \tag{4}$$

to depict nullclines in the phase planes of the systems. The result is displayed in Fig. 4, where the first row (a and c) shows nullclines for low values of stimulation where there are two fixed points. The second row (b and d), on the other hand, shows the responses of the models for higher stimulation current where those fixed points merge and annihilate simultaneously in both CORDIC and Izhikevich model phase plane.

Second, Fig. 5 compares the raster diagram of 1000 randomly coupled instances of original Izhzikevich (a) and the proposed CORDIC neuron models. Here, each dot represents a specific neuron spiking at a specific time. Despite the differences in the details of the two models used, in general they are very much alike. Both Figures 4 and 5 demonstrate that the proposed CORDIC implementation of Izhikevich neuron can show qualitatively similar behavior to the original model. Further quantitative error analysis is presented in the following subsection.

4) Models Numerical Analysis: To investigate the accuracy of the proposed model in generating Izhikevich behavior, two types of time domain errors were examined as follows.

ERRT: Modification in the neuron model may cause difference in spike timing and lag in the spike train of the modified model compared to the original one. For quantitative measuring of this error, first, two spike trains were synced and then time to next spike for original and CORDIC models was considered for the calculation of a timing error (named ERRT) as shown in Fig. 6. Here,

$$ERRT = \left| \frac{\Delta t_c - \Delta t_o}{\Delta t_o} \right| \times 100, \qquad \Delta t = t_{s2} - t_{s1}, \tag{5}$$

where Δt_c, and Δt_o are time intervals between the second (t_{s2}) and first spike (t_{s1}), for CORDIC and original model, respectively.

NRMSD: The Normalized Root Mean Square Deviation (NRMSD) [56] error is also used to measure the similarity of spike shapes in CORDIC and the original model. Low values for this error indicate more resemblance of output spikes. This error is defined as

$$RMSD = \sqrt{\frac{\sum_{i=1}^{n} (v_c(n) - v_o(n))^2}{n}}, \tag{6}$$

and normalized as

$$NRMSD = \frac{RMSD}{v_{max} - v_{min}}, \tag{7}$$

where V_c and V_o are the wave forms of the CORDIC and Izhikevich model, respectively. Here, V_{max} and V_{min} are the maximum and minimum values of V_o in its domain. For instance, for the curve in Fig. 2 at the range of [-100 100], NRMSD was calculated as 5.2177×10^{-5}, confirming a very small error between CORDIC squaring and squaring using normal multiplication operation. To measure the similarity of output spikes, first two spikes were synced as shown in Fig. 6 and thereafter NRMSD was evaluated for the half of time interval between these two spikes. Table I presents values of ERRT and NRMSD for computer simulation of modified CORDIC models. As expected, a higher value of n will result in smaller error values, where the IZHCOR12 has a negligible deviation from the Izhikevich neuron.

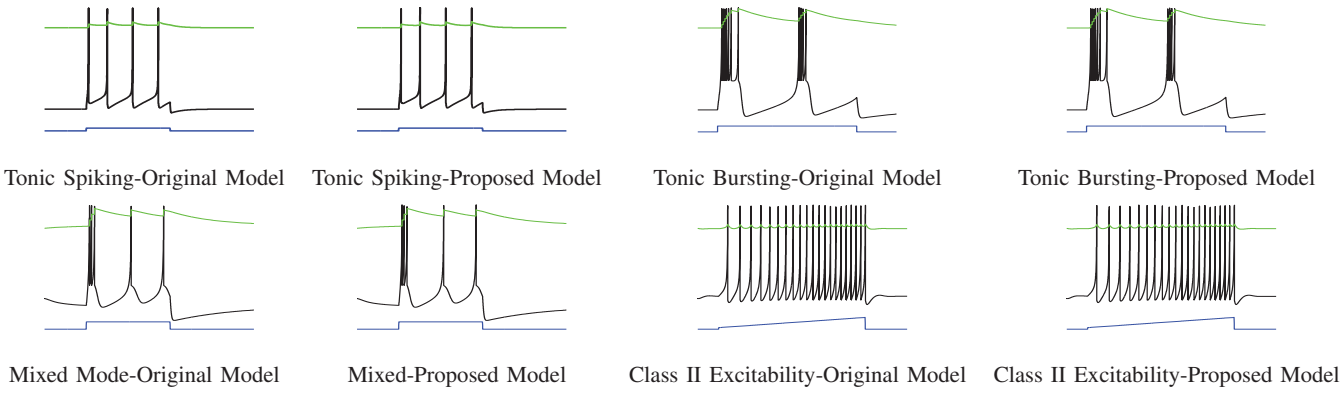


Fig. 3. Computer simulation of the original and the proposed modified CORDIC models for different neuronal behaviors. The black and green lines show membrane potential and recovery variable respectively. The applied current is illustrated by the blue line.

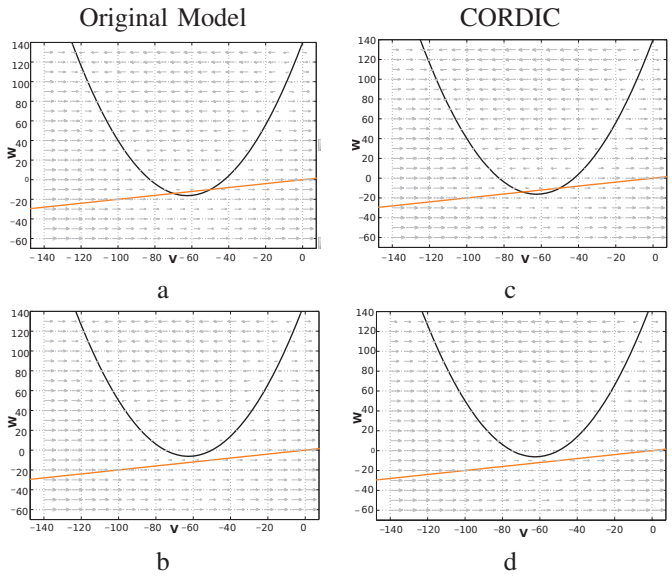


Fig. 4. Nullclines of original and CORDIC model. In the first row, similar to the original model, CORDIC model has two interaction points for low injected current; the second row shows the state of models for higher injected current where those intersection points merged and annihilated.

TABLE I
ERRT AND NRMSD FOR TONIC SPIKING AND REGULAR BURSTING.

	Model	Error Type	Ton. Spiking	Reg. Bursting
Computer simulation	IZHCOR6	Errt	%0.2549	%0.0000
		NRMSD	%0.0034	%0.0705
	IZHCOR8	Errt	%0.2049	%0.0000
		NRMSD	%0.0006	%0.0136
	IZHCOR10	Errt	%0.1025	%0.0000
		NRMSD	%0.0001	%0.0082
	IZHCOR12	Errt	%0.0000	%0.0000
		NRMSD	%0.0000	%0.0063
FPGA implementation	IZHCOR6	Errt	%0.0191	%0.0000
		NRMSD	%0.3951	%2.0631

B. Network Topology and Learning Method

1) Network Topology: In this study, a two-layer spiking neural network as shown in Fig. 7 was formed. The first layer consisting of 20 neurons acts as an input layer while the second one with a single neuron is the output. A uniform random spike train input, with the mean firing rate of 7 Hz, was applied to each input neuron, which made them fire (defined as the state

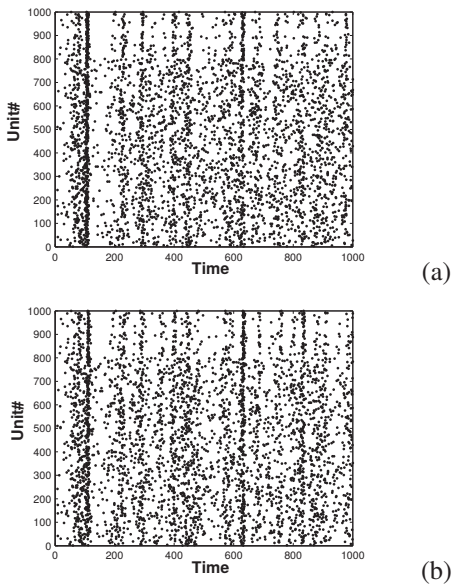


Fig. 5. The spike raster for a population of 1000 tonic bursting neurons which are coupled randomly. The utilized neuron models are (a) original Izhikevich and (b) proposed CORDIC.

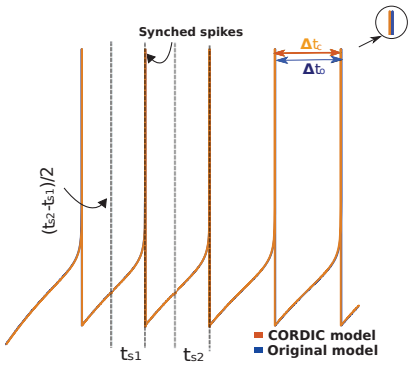


Fig. 6. ERRT: The difference of time interval between two spikes in the original and CORDIC model obtained from computer simulations [50].

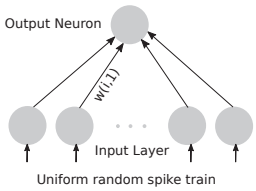


Fig. 7. The topology of the utilized spiking neural network.

where membrane potential become greater than 30mv). For the output neuron, the input current is considered to be the sum of the currents received from the input layer spiking neurons as

$$I_o = \sum_{i=1}^{20} w(i,1) f(i) \tag{8}$$

where $w(i,1)$, which was initially set to 96 (by trial and error), is the synaptic weight connecting the input layer i to the output neuron. The value of $f(i)$ is 1 if the corresponding neuron fires and is 0 otherwise.

2) STDP Learning: In STDP, analogous to biology, the synaptic weight changes when a pre-synaptic neuron fires in a short time before or after the post-synaptic neuron, strengthening or weakening the neuron connection accordingly. Such a change is determined as an exponential of the time difference between two events and is formulated as

$$\begin{cases} w_i(\Delta t) = A_+ \, e^{-\Delta t/\tau_+} & if \quad \Delta t > 0 \\ w_i(\Delta t) = -A_- \, e^{\Delta t/\tau_-} & if \quad \Delta t \leq 0, \end{cases} \tag{9}$$

where $\Delta t = t_{post} - t_{pre}$ is the time span between pre- and post-synaptic spikes. Here, τ_+ and τ_- are STDP learning windows, which determine any time differences greater than them is considered to have a small effect on the synaptic weights and could be disregarded. These windows were set to $\tau_+ = \tau_- = 20ms$ in our experiments. In addition, A_+ and A_- are gain parameters set to 2 and 4 respectively considering the fact that in biology too, synapses tend to be more depressed than potentiated. Overall, these five parameters determine the magnitude of weight change.

Furthermore, as in biological synapses, the weight should be confined between $w_{min} < w < w_{max}$. The STDP mechanism of weakening and strengthening of synapses will eventually lead to a bi-modal distribution of weights, which is a result of competitive Hebbian learning [57]. This rule applies to the utilized network in Fig. 7 as well. STDP learning in this two-layer network leads to a bi-modal weight distribution as shown in Fig. 8. This figure depicts the evolving of network weights over the simulation time to distribute into two extreme weight values of 0 and 200.

3) CORDIC STDP: The main challenges in implementing STDP are its exponential terms and the memory required to store and retrieve spike timing. Here we implemented the exponential function required for STDP, using a modified version of the CORDIC algorithm presented in [50]. The algorithm for calculating the exponential of x (e^x) is shown in Fig. 9. Here, variables x and $expx$ are used to store input and output values, respectively. As part of the algorithm, the pre-calculated values of $e^{(\frac{1}{2})}, e^{(\frac{1}{4})}, \cdots, e^{(\frac{1}{n})}$ are stored in an array, as shown in line

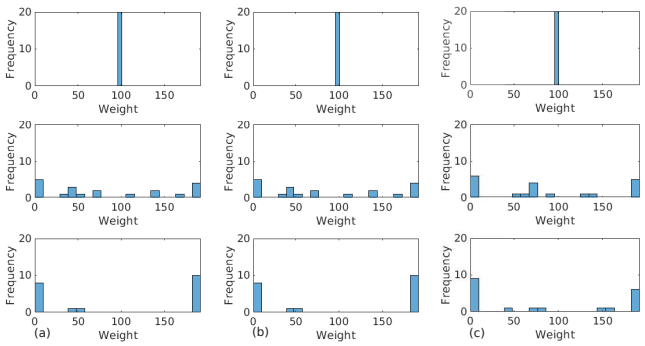


Fig. 8. Weight distribution after STDP learning in the network of (a) original, (b) CORDIC, and (c) 2^x-based approximation model. All the 20 synaptic weights are initially set to 96 as shown in the first row. The second row displays weight distribution and their frequency after half of the simulation time. Finally, the third row depicts the weight distribution at the end of simulation, where the weights have mostly evolved to be either zero or the maximum possible value of $Wmax = 192$.

```
1  //assign initial values
2 z=fraction(x);poweroftwo=0.5;
3 expx=1;
4  //pre-calculated a elements
5 a=[exp((1/2)*(1:n))]
6  //Determine the weights
7  //and calculate products
8 for i from 0 to n do
9  {
10   if ( poweroftwo < z )
11   {
12    z=z-poweroftwo;
13    expx = expx * a(i);
14   }
15   poweroftwo=poweroftwo/2;
16 }
```

Fig. 9. The pseudo code of CORDIC exponential.

5. The *FOR* loop in line 8 calculates the exponential function for the fraction part of x with e^{-n} precision. In this work, n is set to 8, but higher values of n could be selected in the case of the need for higher precision exponential function. However, this will in turn slightly increases implementation cost. Our proposed algorithm is simpler than that of [50], because the range of x, for which we need to calculate the exponential function, is between -1 and 0. Fig. 10 demonstrates the very good approximation in implementing the exponential function achieved using our proposed CORDIC algorithm. In this figure, the blue curve shows the computer simulation of exponential function, while the red curve is the exponential approximation using CORDIC. The NRMSD error calculated for these curve was 2.38×10^{-3}, which further verify the high accuracy of the proposed CORDIC algorithm. To further verify the effectiveness of the proposed CORDIC algorithm in replicating the STDP model, the simple 2^x function was used as another method to approximate exponential function, because the value of x is always negative and in the range of -1 and 0. Such a term could be cheaply implemented on hardware using shift registers.

To test the accuracy of the approximated STDP models compared to the original model, two networks with the topology shown in Fig. 7 were formed. The first network

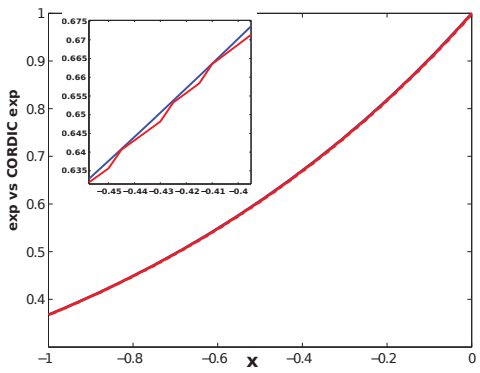


Fig. 10. Software simulation of: exponential function (blue line) and the one calculated by CORDIC algorithm (red line) for n=8 indicating the resemblance of both functions.

consisted of original Izhikevich neurons and original STDP rule, while the second one used CORDIC STDP to connect CORDIC (IZHCOR8) neurons. Next, the same random current was applied to the input layer of both networks. Fig. 8(a) and (b) show the evolution of weight for the original and CORDIC networks, respectively. Due to the high precision of the CORDIC algorithm, the resulted weight distributions are very similar to the network implementing original STDP and Izhikevich models. Furthermore, Fig. 8(c) is the weight distribution achieved using the 2^x function instead of the exponential functions. As seen, the weight distribution is different from the original and CORDIC models but a similar bi-modal distribution could be observed.

III. FPGA IMPLEMENTATION

A. Architecture

1) Izhikevich neuron: This section presents FPGA implementation of the proposed CORDIC Izhikevich neuron. Since the primary objective of this paper is to reduce the implementation cost and improve hardware speed, fixed-point arithmetic was used in our implementations. For solving the Izhikevich Ordinary Differential Equations (ODEs) shown in Eq. 1, and 2, they were discretized and simple Euler method was used that resulted in the following Equations.

$$v[n+1] = (0.04\ CORDIC\_Mul(v[n]) + 5v[n]... \\ +140 - u[n] + I[n])dt + v[n], \tag{10}$$

$$u[n+1] = a(bv[n] - u[n])dt + u[u]. \tag{11}$$

By choosing small step sizes and with the help of the reset equation, which keeps v bounded to help the stability of Euler method, this method produced stable outputs. In addition, multiplications by constant numbers were approximated to the closest possible values with the sum of a series of power of two numbers ($\sum_{-l}^{k} 2^n$), thereby reducing multiplications to simple shifts and adds. Obviously, a higher value of $k+l$ increases the multiplication precision but it also requires more shift registers and adders leading to a higher hardware resource requirement. The Control Data Flow Graph (CDFG) [58] of the Izhikevich CORDIC model is shown in Fig. 11. In this

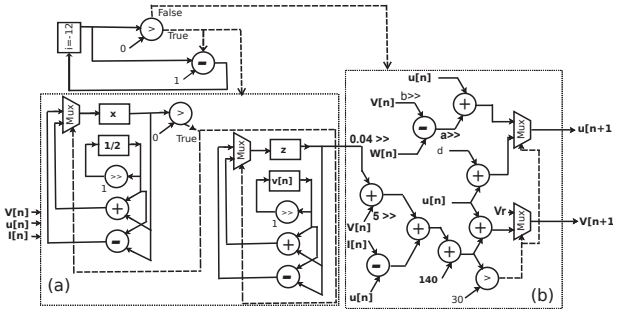


Fig. 11. Control data flow graph for FPGA implementation of CORDIC Izhikevich neuron. First, block (a) calculates the square function as per the pseudo code in Fig. 1 . The counter which is showed at the top of this block, counts from -6 to 5, enabling this block for 12 iterations. In each iteration, $2^{(-i)}$ is added or subtracted from x register based on the sign of x. Eventually, this register's value tends to zero as iterations continues. The same scenario applies to the z register. The value of $2^{(-i)} * v[n]$ will be added or subtracted from the z register depending on the sign of z. After 12 iterations, the multiplication result is ready and the (b) block is enabled. This block solves the Euler method in the eq. 10 and 11. At the last stage of this block, v is compared with the threshold value of 30. If v is grater than this threshold, the multiplexers reset the v and u according to eq. 3. The plain lines show the flow of data and the dashed lines indicate the jumps and decision signals.

figure, operation blocks and registers are represented with circles and rectangles, respectively. In this graph, block A calculates the square function while block B solves the Euler method presented in Eq. 10 and 11. Arithmetic shift operations are shown by "»" and "x«" means shift by "x" position while adding the results, which implements $\sum_{-l}^{k} 2^n$.

For the model to work properly, optimum word length for the architecture should be determined. This can be specified when considering the minimum number of integer bits to correctly represent the range of variables, and the number of fraction bits for the minimum required precision. In addition, extra bits are required to prevent from over and under flow in the shift&add operations. Considering all the requirements, and to avoid overflow and precision loss, 14 and 16 bits were dedicated for the fraction and integer parts, respectively.

2) Network and STDP rule: Similar to the neuron, the Euler method was used to solve the discretized version of Eq. 9 to implement STDP. The CDFG for calculating the exponential term in this equation is presented in Fig. 12. Comparing to the flow graph used for calculating exponentials in [50], this one is simpler because here x is in the range of -1 and 0, and therefore no shift by e is needed. Nonetheless, this design also only uses shift & add operations, so it is hardware friendly.

The flow graph for implementation of the spiking network with STDP learning is shown in Fig. 13. The post-synaptic input current is the sum of the synaptic weights of all the pre-synaptic neurons that fire. As shown in the block (a) of the figure, a 41-bit shift register is used to record the spike timing of pre- and post-synaptic neurons. Every time a neuron fires or is silent, the register shifts to left and the least significant bit updates with 1 (spike) or 0 (silence), accordingly. Here, sampling time is controlled by a counter to act as enable signal for the shift register.

Online STDP learning rule is implemented in block (b) of Fig. 13. Here, the middle bit (Reg[20]) of the 41-bit shift register that records pre-synaptic spike times, enables STDP

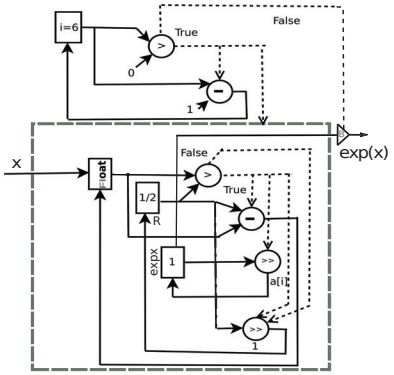


Fig. 12. Control data flow graph for digital implementation of the exponential function for the range of -1 < x < 0 according the pseudo code shown in the Fig 9. Since the value of the input x is always smaller than one, the word length of this architecture was considered as total number of fraction bits. The calculations complete in 6 iterations as the counter at the top of the figure counts from 1 to 6. In each iteration, float register is compared with the 2^{-i}. If it is greater, the register is subtracted from 2^{-i}. Moreover, expx register, which it's initial value is 1, is multiplied by constant $a(i)$ with performing shift and add operations. Upon completion of iterations, the counter enables the out signal.

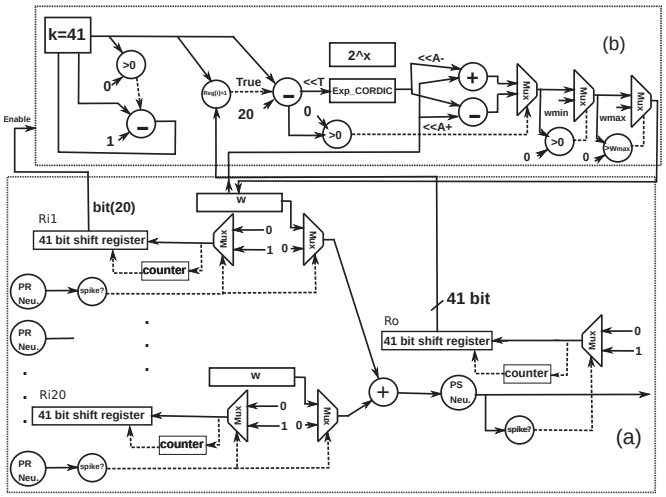


Fig. 13. Control data flow graph for digital implementation of STDP algorithm. Block (a) is the hardware presented for the network in Fig. 7 and recording spike times. Block (b) implements STDP to calculate weight changes and update weights. In this block, either of Exp_CORDIC or $2\hat{x}$ blocks could be used for approximating the STDP exponential term.

mechanism. This is to account for, and enable STDP, in response to future (Reg[19:0]) and past (Reg[40:21]) spike events. If a pre-synaptic neuron spikes, the time of that spike is compared to the time of post-synaptic spikes and the difference will be divided by τ (using shift operations) and passed to the exponential CORDIC calculator unit. Next, based on the sign of the time difference, the new weight will be determined and compared to the boundaries. The same approach could be used for calculating STDP but using the 2^x model. That way, the *exp_cordic* unit (in Fig. 13(b)), should be replaced with a 2^x calculator.

TABLE II
RESOURCES USED TO IMPLEMENT DIFFERENT PROPOSED CORDIC BASED IZHIKEVICH MODELS.

Model	Slice Registers	Slice LUT's	Max Speed (MHz)
IZHCOR6	229	410	183.4
IZHCOR8	232	413	182.7
IZHCOR10	234	418	181.4
IZHCOR12	236	421	180.1

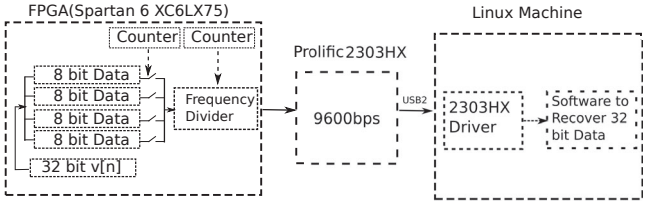


Fig. 14. The method of transferring on-FPGA spiking neuron outputs to PC for analysis.

B. FPGA Implementation

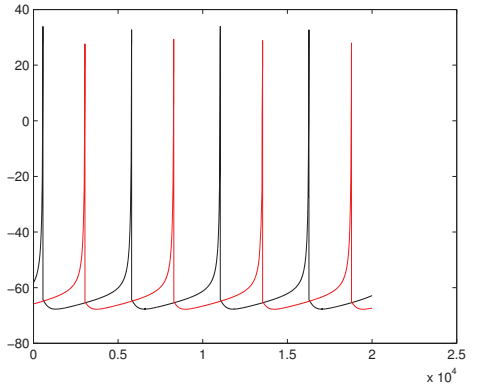
Data flow graphs in Fig. 11 to 13 were described with VHDL hardware description language using Finite State Machines (FSMs). Further, the developed codes were first simulated using Modelsim for validation. Afterwards, the codes were synthesized by XILINX ISE XST synthesizer and implemented on the 45nm technology XILINX Spartan-6 XC6SLX75 FPGA.

Since the utilized FPGA only supports Universal Asynchronous Receiver Transmitter (UART) port, a Prolific 2303HX chip and its driver were used to create virtual UART port in PC, through Universal Serial Bus (USB) port, to transfer the data from FPGA to PC for analysis. Furthermore, a UART transmitter and receiver module was added to the neuron VHDL code and implemented on FPGA as shown in Fig. 14. A counter was used to divide FPGA operation frequency to the chosen baud rate of the UART port (9600 bps). Data stream was structured as one start bit, eight data bits, one stop bit, and no parity. An additional counter was used to break the thirty-bit data in register V[n] into four bytes and send to UART port. Further, software was developed to receive and recover data from virtual UART port on Linux PC. Fig. 15 demonstrates the FPGA implementation and simulation results for two cases of tonic spiking and regular bursting of IZHCOR6. As it can be seen from the figure the FPGA implementation results well resemble the computer simulation results of the original Izhikevich model. To have a better comparison between the simulation and the FPGA outputs, NRMSD and ERRT were calculated as shown in Table I. These errors further confirm that the digital implemented CORDIC neuron has a similar behavior to the original model.

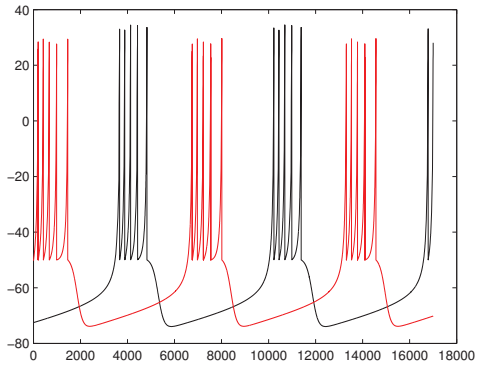
To implement the spiking neural network with STDP learning and demonstrate its bi-modal behavior on FPGA, first, a Linear-Feedback Shift Register (LFSR) unit was designed to generate semi-random input spikes for the first layer neurons in the network shown in Fig. 7. The implemented LFSR is shown in Fig. 16. It is worth noting that, for other applications, the LFSR that generates random currents could be replaced with the spiking output of event-based sensors such as a silicon

TABLE III
COMPARISON BETWEEN PROPOSED METHOD AND PREVIOUSLY PUBLISHED WORKS

Refrence	Slice Registers	Slice LUT's	Max Speed (MHz)	DSPs	NRMSD%	Errt%	Device
Soleimani et al [46].	493	617	241.9	0	-	1.54	Virtex-II Pro XC2VP30
Gomar et al. [59]	388	1279	190	0	4.02	-	Virtex-II Pro XC2VP30
Hayati et al. [60]	476	856	135	0	3.7	-	Virtex-II Pro XC2VP30
Grassia et al. [61]	646	1048	105	22	-	-	Virtex-5 XC5VLX50
Heidarpur et al. [50]	829	1221	134.3	0	0.04	0.39	Spartan-6 XC6SLX9
Shimada et al. [62]	357	1776	Asynchronous	-	-	-	Zync-7000 XC7Z020
This work (IZHCOR6 -Area optimization goal)	229	410	183.4	0	0.003	0.26	Spartan-6 XC6SLX75
This work (IZHCOR6 -Speed optimization goal)	280	469	212.8	0	0.003	0.26	Spartan-6 XC6SLX75



(A)



(B)

Fig. 15. FPGA Implementation of CORDIC Izhikevich (red) and computer simulation of Izhikevich model (black). The FPGA Data was transferred to PC via UART-USB port. (A) Tonic Spiking and (B) Regular Bursting. Please note that the implementation data is scaled and an offset was added to it for closer behavior to the simulation.

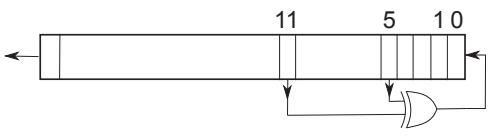


Fig. 16. Linear Feedback Shift Register (LFSR) technique was used to generate semi-random input currents to feed the SNN input neurons.

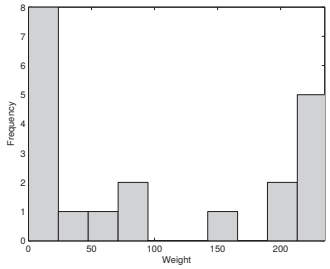


Fig. 17. Bi-modal weight distribution reached after execution of the online on-FPGA STDP learning on a network of Izhikevich neurons.

TABLE IV
TOTAL NUMBER AND HIGHEST SPEED OF CORDIC-BASED AND ORIGINAL (IMPLEMENTED USING DSP 36-BIT MULTIPLIERS) IZHIKEVICH NEURONS THAT CAN BE IMPLEMENTED ON VARIOUS FPGA DEVICES.

Device	CORDIC		DSP Multiplier	
	Number	Speed	Number	Speed
Spartan-6 XC6LX75	110	183 MHz	22	44 MHz
Virtex-5 XC5VSX240T	365	220 MHz	176	102 MHz
Virtex-6 XC6VLX550T	835	332 MHz	144	111 MHz
Virtex-7 XC7VX980T	1490	370 MHz	600	130 MHz

retina or cochlea. Fig. 17 demonstrates the bi-modal behavior reached after stimulating the implemented SNN on FPGA, with randomly generated input spikes generated using the on-FPGA LFSRs.

C. Results and Discussion

Table II shows the amount of resources used to implement different CORDIC models of the Izhikevich neuron and the maximum speed reached using each of these models. As can

TABLE V
UTILIZED RESOURCES TO IMPLEMENT THE CORDIC (IZHCOR6) AND ORIGINAL IZHIKEVICH NEURON

Resource	CORDIC	Original
Slice LUTs	410	370
Slice Registers	229	211
DSPs	0	6

TABLE VI
CORDIC AND ORIGINAL NEURON MODEL ON-FPGA POWER (REPORTED BY XILINX XPOWER ANALYZER FOR THE SAME FREQUENCY)

	CORDIC neuron	Original Model
On-FPGA power	71 mW	73 mW

TABLE VII
TOTAL FPGA UTILIZATION FOR IMPLEMENTATION OF CORDIC AND 2^x ONLINE STDP ON A NETWORK OF CORDIC (IZHCOR8) IZHIKEVICH NEURONS WITH TOPOLOGY OF FIG. 7. THESE RESULTS INCLUDES SEMI-RANDOM INPUT GENERATOR MECHANISM AS WELL.

	Slice Registers	Utilization Perc.	Slice LUT's	Utilization Perc.	Max Speed (MHz)
CORDIC STDP	7,088	7%	10,376	22%	84.1
2^x STDP	7,047	7%	10,234	21%	84.5

be seen in this table, the resource usages of the four different implementations are close but for each higher precision model, extra time is needed to produce the new value of v. This delay can be calculated as:

$$T = \frac{1}{frequency} * n \tag{12}$$

Where T is the total time required to calculate the CORDIC square function and n is the number of the iterations. Considering the IZHCOR6 model (Area optimization goal) and frequency of device as 184 Mhz, total delay to calculate the result will be $6 * 5.5ns = 33ns$. DSP multiplier on the other side, operate at lower frequency of 44 Mhz but it need one clock to complete the results. DSP's total time can be calculated as $1 * 22.7ns = 22.7ns$. Still, the total delay is less than CORCID method. However, the architecture presented in this paper is not only consisted of the neurons. But also include the hardware to store the the spike times and the STDP algorithm to calculate and update the synaptic weights. The DSP multiplier reduces the frequency of the FPGA, resulting in other units to perform much slower. This in turn, increases total delay and reduces the throughput of the system.

In addition, in Table III, the device utilization, speed, NRMSD, and ERRT are compared with some previously published works where a single neuron model is implemented on FPGA. Since the FPGA devices and synthesizer used are different in these works, this table results should be considered relatively. However, it can be seen that the proposed Izhikevich device consumes fewer resources while having higher speed compared to previous works.

Furthermore, Table IV shows the number of CORDIC and original Izhikevich neurons that could be implemented on some FPGAs devices and compares the speed of both methods. The resources utilized for implementation of the CORDIC and DSP based neuron is presented in the Table V. In implementation of CORDIC model, the number of neurons is limited by available LUTs in FPGA. Total number of LUTs in Spartan-6 XC6LX75 is 46648. Therefore, the number of neurons was calculated as:

$$N = \frac{Available\ LUTs}{Utilized\ LUTs} = \frac{46648}{410} \approx 110 \tag{13}$$

In the case of DSP based implementation, the number of neurons is limited by available DSPs. In Spartan-6 XC6LX75, there are 132 DSP slices available. Thus, dividing 132 to the number of utilized DSP slices which is 6, gives maximum number of neurons.

$$N = \frac{Available\ DSPs}{Utilized\ DSPs} = \frac{132}{6} = 22 \tag{14}$$

To implement the original model on FPGAs, 36-bit DSP multipliers were used and the results were truncated to 36 bits. However, multiplication in constants were still performed with shift and add operations, the same way as performed in the proposed CORDIC device. Despite this simplification in the original model, the proposed CORDIC method allowed a higher number of faster neurons to be implemented on all FPGAs.

Power consumption and density is another important concern when designing hardware. It is also, one of major issues that need to be resolved for massive large scale implementation of neuromorphic systems considering that building such systems has been one of the main motivations of this work. To measure the on-FPGA power, first we generated a value change dump file and then the XILINX XPower Analyzer was used to determine the circuits power. For the fair comparison, it is presumed that both circuits work at the same frequency (40 MHz). As it is shown in this table VI, the CORDIC neuron consumes slightly less power than the original neuron model implementation.

To evaluate the cost of the total SNN with STDP learning and random input spike generation, the network with the topology of Fig. 7 consisting of CORDIC Izhikevich neurons (Fig. 11), semi-random input generators (Fig. 16), and STDP learning synapses (Fig. 13 and Fig. 12), was implemented on FPGA. This is the same network that was used to successfully generate the bi-modal weight distribution due to competitive Hebbian Learning of STDP synapses. Table VII reports the total resources and speed of the implemented network. As the table indicates, this STDP learning spiking network only consumes around 29% of available FPGA resources and could therefore be scaled almost 3.5 times on a fairly cheap device like Spartan XA6SLX75.

In addition, the second row in Table VII presents implementation result for 2^x method, which uses lower resources and has higher speed in comparison with the previous methods. However, as discussed earlier, the accuracy of this model is lower than the proposed CORDIC model.

Overall, the above results confirm the reliable functionality of the proposed CORDIC-based SNN with STDP Learning. These results also show that the proposed design can lead to more efficient and faster FPGA-based SNNs compared to the literature. It can therefore contribute to the design and implementation of low-cost and high-speed large-scale digital neuromorphic systems exploring unsupervised STDP learning. It is important to note that FPGA devices utilize more resources for hardware implementation than that of ASICs. Implementing such hardware on silicon will have considerably less cost and have better performance.

IV. CONCLUSION

In this paper, a novel hardware was presented based on the CORDIC method for on-FPGA online STDP learning.

This hardware proved to be accurate while requiring less FPGA resources and having higher speed compared to the original models and state-of-the-art designs. The CORDIC method was utilized because of the simplicity of its structure, since it only uses add and shift operations which could be cheaply implemented on hardware. In order to implement the proposed learning system, first, the CORDIC method was used to implement Izhikevich neurons and its accuracy was analyzed. Second, the STDP algorithm was adopted for online learning and modified using the CORDIC algorithm to improve hardware efficiency. Furthermore, error analysis was performed on computer simulation data to ensure the accuracy of the implemented CORDIC models. Consequently, hardware was designed, described in VHDL, and simulated for both neuron and learning mechanism. Finally, the models were implemented on FPGA to form a spiking neural network composed of Izhikevich neurons and STDP synapses to demonstrate competitive Hebbian learning. The proposed CORDIC-based FPGA spiking network with STDP learning is a step toward simpler and more efficient hardware design for SNN with unsupervized STDP learning implemented on FPGAs and digital platforms.

REFERENCES

[1] K. Boahen, "A neuromorph's prospectus," *Computing in Science and Engg.*, vol. 19, no. 2, pp. 14–28, Mar. 2017. [Online]. Available: https://doi.org/10.1109/MCSE.2017.33

[2] L. A. Pastur-Romay, F. Cedron, A. Pazos, and A. B. Porto-Pazos, "Deep artificial neural networks and neuromorphic chips for big data analysis: Pharmaceutical and bioinformatics applications," *International Journal of Molecular Sciences*, vol. 17, no. 8, 2016.

[3] B. V. Benjamin, P. Gao, E. McQuinn, S. Choudhary, A. R. Chandrasekaran, J. M. Bussat, R. Alvarez-Icaza, J. V. Arthur, P. A. Merolla, and K. Boahen, "Neurogrid: A mixed-analog-digital multichip system for large-scale neural simulations," *Proceedings of the IEEE*, vol. 102, no. 5, pp. 699–716, May 2014.

[4] 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)*, Sept 2011, pp. 1–4.

[5] S. B. Furber, F. Galluppi, S. Temple, and L. A. Plana, "The spinnaker project," *Proceedings of the IEEE*, vol. 102, no. 5, pp. 652–665, May 2014.

[6] J. Schemmel, D. Briiderle, A. Griibl, M. Hock, K. Meier, and S. Millner, "A wafer-scale neuromorphic hardware system for large-scale neural modeling," in *Proceedings of 2010 IEEE International Symposium on Circuits and Systems*, May 2010, pp. 1947–1950.

[7] R. Wang, C. S. Thakur, G. Cohen, T. J. Hamilton, J. Tapson, and A. van Schaik, "Neuromorphic hardware architecture using the neural engineering framework for pattern recognition," *IEEE Transactions on Biomedical Circuits and Systems*, vol. 11, no. 3, pp. 574–584, June 2017.

[8] C. S. Thakur, J. Molin, G. Cauwenberghs, G. Indiveri, K. Kumar, N. Qiao, J. Schemmel, R. Wang, E. Chicca, J. O. Hasler *et al.*, "Large-scale neuromorphic spiking array processors: A quest to mimic the brain," *arXiv preprint arXiv:1805.08932*, 2018.

[9] B. Sen-Bhattacharya, S. James, O. Rhodes, I. Sugiarto, A. Rowley, A. B. Stokes, K. Gurney, and S. B. Furber, "Building a spiking neural network model of the basal ganglia on spinnaker," *IEEE Transactions on Cognitive and Developmental Systems*, pp. 1–1, 2018.

[10] J. P. Dominguez-Morales, A. Rios-Navarro, D. Gutierrez-Galan, R. Tapiador-Morales, A. Jimenez-Fernandez, E. Cerezuela-Escudero, M. Dominguez-Morales, and A. Linares-Barranco, "Multilayer spiking neural network for audio samples classification using spinnaker," in *2017 IEEE International Symposium on Circuits and Systems (ISCAS)*, May 2017, pp. 1–1.

[11] D. Khodagholy, J. N. Gelinas, T. Thesen, W. Doyle, O. Devinsky, G. G. Malliaras, and G. Buzsáki, "Neurogrid: recording action potentials from the surface of the brain," *Nature neuroscience*, vol. 18, no. 2, p. 310, 2015.

[12] S. Scholze, H. Eisenreich, S. Höppner, G. Ellguth, S. Henker, M. Ander, S. Hänzsche, J. Partzsch, C. Mayr, and R. Schüffny, "A 32 gbit/s communication soc for a waferscale neuromorphic system," *INTEGRATION, the VLSI journal*, vol. 45, no. 1, pp. 61–75, 2012.

[13] M. Chu, B. Kim, S. Park, H. Hwang, M. Jeon, B. H. Lee, and B. G. Lee, "Neuromorphic hardware system for visual pattern recognition with memristor array and cmos neuron," *IEEE Transactions on Industrial Electronics*, vol. 62, no. 4, pp. 2410–2419, April 2015.

[14] S. Davies, "Learning in the spiking neural networks," Ph.D. dissertation, Univ. of Manchester, Manchester, 2012.

[15] M. R. Azghadi, N. Iannella, S. F. Al-Sarawi, G. Indiveri, and D. Abbott, "Spike-based synaptic plasticity in silicon: design, implementation, application, and challenges," *Proceedings of the IEEE*, vol. 102, no. 5, pp. 717–737, 2014.

[16] A. L. Hodgkin and A. F. Huxley, "A quantitative description of membrane current and its application to conduction and excitation in nerve," *The Journal of physiology*, vol. 117, no. 4, p. 500, 1952.

[17] E. M. Izhikevich, "Simple model of spiking neurons," *IEEE Transactions on neural networks*, vol. 14, no. 6, pp. 1569–1572, 2003.

[18] R. Brette and W. Gerstner, "Adaptive exponential integrate-and-fire model as an effective description of neuronal activity," *Journal of neurophysiology*, vol. 94, no. 5, pp. 3637–3642, 2005.

[19] R. FitzHugh, "Impulses and Physiological States in Theoretical Models of Nerve Membrane," *Biophysical Journal*, vol. 1, pp. 445–466, jul 1961.

[20] C. Morris and H. Lecar, "Voltage oscillations in the barnacle giant muscle fiber," *Biophysical Journal*, vol. 35, no. 1, pp. 193–213, 1981.

[21] H. R. WILSON, "Simplified dynamics of human and mammalian neocortical neurons," *Journal of Theoretical Biology*, vol. 200, no. 4, pp. 375–388, 1999.

[22] R. M. Rose and J. L. Hindmarsh, "The assembly of ionic currents in a thalamic neuron i. the three-dimensional model," *Proceedings of the Royal Society B: Biological Sciences*, vol. 237, no. 1288, pp. 267–288, 1989.

[23] W. Gerstner and W. M. Kistler, *Spiking neuron models: Single neurons, populations, plasticity*. Cambridge university press, 2002.

[24] W. Gerstner, W. M. Kistler, R. Naud, and L. Paninski, *Neuronal dynamics: From single neurons to networks and models of cognition*. Cambridge University Press, 2014.

[25] P. U. Diehl, G. Zarrella, A. Cassidy, B. U. Pedroni, and E. Neftci, "Conversion of artificial recurrent neural networks to spiking neural networks for low-power neuromorphic hardware," in *2016 IEEE International Conference on Rebooting Computing (ICRC)*, Oct 2016, pp. 1–8.

[26] D. Martí, M. Rigotti, M. Seok, and S. Fusi, "Energy-efficient neuromorphic classifiers," *Neural computation*, vol. 28, no. 10, pp. 2011–2044, 2016.

[27] R. Kreiser, T. Moraitis, Y. Sandamirskaya, and G. Indiveri, "On-chip unsupervised learning in winner-take-all networks of spiking neurons," in *2017 IEEE Biomedical Circuits and Systems Conference (BioCAS)*, Oct 2017, pp. 1–4.

[28] H. Y. Hsieh, P. Y. Li, C. H. Yang, and K. T. Tang, "A high learning capability probabilistic spiking neural network chip," in *2018 International Symposium on VLSI Design, Automation and Test (VLSI-DAT)*, April 2018, pp. 1–4.

[29] J. s. Kim and S. Jung, "Implementation of the rbf neural chip with the on-line learning back-propagation algorithm," in *2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence)*, June 2008, pp. 377–383.

[30] E. Stromatias and J. S. Marsland, "Supervised learning in spiking neural networks with limited precision: Snn/lp," in *2015 International Joint Conference on Neural Networks (IJCNN)*, July 2015, pp. 1–7.

[31] A. M. Sheri, A. Rafique, W. Pedrycz, and M. Jeon, "Contrastive divergence for memristor-based restricted boltzmann machine," *Engineering Applications of Artificial Intelligence*, vol. 37, pp. 336 – 342, 2015. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0952197614002334

[32] F. Grassia, L. Buhry, T. Levi, J. Tomas, A. Destexhe, and S. Saighi, "Tunable neuromimetic integrated system for emulating cortical neuron models," *Frontiers in NEUROSCIENCE*, vol. 5, p. 134, 2011.

[33] A. Morrison, M. Diesmann, and W. Gerstner, "Phenomenological models of synaptic plasticity based on spike timing," *Biological cybernetics*, vol. 98, no. 6, pp. 459–478, 2008.

[34] E. Covi, S. Brivio, M. Fanciulli, and S. Spiga, "Synaptic potentiation and depression in al: Hfo2-based memristor," *Microelectronic Engineering*, vol. 147, pp. 41–44, 2015.

[35] M. R. Azghadi, S. Moradi, D. B. Fasnacht, M. S. Ozdas, and G. Indiveri, "Programmable spike-timing-dependent plasticity learning circuits in

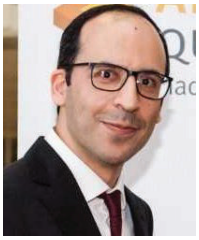
neuromorphic vlsi architectures," *ACM Journal on Emerging Technologies in Computing Systems (JETC)*, vol. 12, no. 2, p. 17, 2015.
[36] C. Mayr, J. Partzsch, M. Noack, S. Hanzsche, S. Scholze, S. Hoppner, G. Ellguth, and R. Schuffny, "A biological-realtime neuromorphic system in 28 nm cmos using low-leakage switched capacitor circuits," *IEEE transactions on biomedical circuits and systems*, vol. 10, no. 1, pp. 243–254, 2016.
[37] R. M. Wang, T. J. Hamilton, J. Tapson, and A. van Schaik, "A mixed-signal implementation of a polychronous spiking neural network with delay adaptation," *Frontiers in neuroscience*, vol. 8, p. 51, 2014.
[38] 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*, pp. 1–14, 2018.
[39] K. Isobe and H. Torikai, "A novel hardware-efficient asynchronous cellular automaton model of spike-timing-dependent synaptic plasticity," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 63, no. 6, pp. 603–607, June 2016.
[40] M. R. Azghadi, S. Al-Sarawi, D. Abbott, and N. Iannella, "A neuromorphic VLSI design for spike timing and rate based synaptic plasticity," *Neural Networks*, vol. 45, pp. 70–82, 2013.
[41] M. R. Azghadi, S. Al-Sarawi, N. Iannella, and D. Abbott, "Tunable low energy, compact and high performance neuromorphic circuit for spike-based synaptic plasticity," *PLoS ONE*, vol. 9, no. 2, p. art. no. e88326, 2014.
[42] C. Lammie, T. Hamilton, and M. R. Azghadi, "Unsupervised character recognition with a simplified fpga neuromorphic system," in *2018 IEEE International Symposium on Circuits and Systems (ISCAS)*, 2018.
[43] L. P. Maguire, T. M. McGinnity, B. Glackin, A. Ghani, A. Belatreche, and J. Harkin, "Challenges for large-scale implementations of spiking neural networks on fpgas," *Neurocomputing*, vol. 71, no. 1, pp. 13–29, 2007.
[44] T. Matsubara, H. Torikai, and T. Hishiki, "A generalized rotate-and-fire digital spiking neuron model and its on-fpga learning," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 58, no. 10, pp. 677–681, Oct 2011.
[45] N. Shimada and H. Torikai, "A novel asynchronous cellular automaton multicompartment neuron model," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 62, no. 8, pp. 776–780, 2015.
[46] H. Soleimani and E. M. Drakakis, "An efficient and reconfigurable synchronous neuron model," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. PP, no. 99, pp. 1–1, 2017.
[47] H. Soleimani, A. Ahmadi, and M. Bavandpour, "Biologically inspired spiking neurons: Piecewise linear models and digital implementation," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 59, no. 12, pp. 2991–3004, Dec 2012.
[48] M. Heidarpur, A. Ahmadi, and N. Kandalaft, "A digital implementation of 2d hindmarsh–rose neuron," *Nonlinear Dynamics*, vol. 89, no. 3, pp. 2259–2272, Aug 2017.
[49] A. Elnabawy, H. Abdelmohsen, M. Moustafa, M. Elbediwy, A. Helmy, and H. Mostafa, "A low power cordic-based hardware implementation of izhikevich neuron model," in *2018 16th IEEE International New Circuits and Systems Conference (NEWCAS)*. IEEE, 2018, pp. 130–133.
[50] M. Heidarpour, A. Ahmadi, and R. Rashidzadeh, "A cordic based digital hardware for adaptive exponential integrate and fire neuron," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 63, no. 11, pp. 1986–1996, Nov 2016.
[51] S. Gomar and M. Ahmadi, "Digital realization of pstdp and tstdp learning," in *2018 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2018, pp. 1–5.
[52] B. Belhadj, J. Tomas, O. Malot, G. N'Kaoua, Y. Bornat, and S. Renaud, "Fpga-based architecture for real-time synaptic plasticity computation," in *2008 15th IEEE International Conference on Electronics, Circuits and Systems*, Aug 2008, pp. 93–96.
[53] C. Lammie, T. J. Hamilton, A. van Schaik, and M. R. Azghadi, "Efficient fpga implementations of pair and triplet-based stdp for neuromorphic architectures," *IEEE Transactions on Circuits and Systems I: Regular Papers*, pp. 1–13, 2018.
[54] E. M. Izhikevich, "Which model to use for cortical spiking neurons?" *IEEE transactions on neural networks*, vol. 15, no. 5, pp. 1063–1070, 2004.
[55] J. E. Volder, "The cordic trigonometric computing technique," *Electronic Computers, IRE Transactions on*, vol. EC-8, no. 3, pp. 330–334, Sept 1959.
[56] R. J. Hyndman and A. B. Koehler, "Another look at measures of forecast accuracy," *International Journal of Forecasting*, vol. 22, no. 4, pp. 679–688, 2006.
[57] C. Borgers, *Spike Timing-Dependent Plasticity (STDP)*. Cham: Springer International Publishing, 2017, pp. 349–359.
[58] S. P. Mohanty, *Low-power high-level synthesis for nanoscale CMOS circuits*. Springer, 2008.
[59] S. Gomar and A. Ahmadi, "Digital multiplierless implementation of biological adaptive-exponential neuron model," *Circuits and Systems I: Regular Papers, IEEE Transactions on*, vol. 61, no. 4, pp. 1206–1219, April 2014.
[60] M. Hayati, M. Nouri, S. Haghiri, and D. Abbott, "Digital multiplierless realization of two coupled biological morris-lecar neuron model," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 62, no. 7, pp. 1805–1814, July 2015.
[61] F. Grassia, T. Levi, T. Kohno, and S. Saïghi, "Silicon neuron: digital hardware implementation of the quartic model," *Artif Life Robotics*, vol. 19, no. 3, pp. 215–219, 2014.
[62] N. Shimada and H. Torikai, "A novel asynchronous cellular automaton multicompartment neuron model," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 62, no. 8, pp. 776–780, Aug 2015.

Moslem Heidarpur received the B.Sc. degree in electrical engineering and M.Sc. degree in electronic engineering from the Department of Electrical Engineering, Razi University, Kermanshah, Iran, in 2012 and 2014. He is now Ph.D. student in the University of Windsor, Canada. His research interests include analog and digital electronic circuit design and optimization, bio-inspired computing, neuromorphic and integrated circuit design.

Arash Ahmadi (M'04–SM'16) received the B.Sc. and M.Sc. degrees in electronics engineering from Sharif University of Technology and Tarbiat Modares University, Tehran, Iran, in 1993 and 1997, respectively, and the Ph.D. degree in electronics from the University of Southampton, U.K., in 2008. He was with Razi University, Kermanshah, Iran, as a Faculty Member. From 2008 to 2010, he was a Fellow Researcher with the University of Southampton. He is currently an Associate Professor in the Electrical Engineering Department, Razi University and visiting scholar at University of Windsor, Canada. His current research interest includes neuromorphic, hardware implementation of signal processing systems, high-level synthesis, bio-inspired computing and memristors.

Majid Ahmadi (S'75–M'77–SM'84–F'02–LF'14) received the B.Sc. degree in electrical engineering from the Sharif University of Technology, Tehran, Iran, in 1971, and the Ph.D. degree in electrical engineering from the Imperial College of Science, Technology and Medicine, London, U.K., in 1977. He has been with the Department of Electrical and Computer Engineering, University of Windsor, Windsor, ON, Canada, since 1980 and he is currently a Distinguished University Professor and an Associate Dean of Engineering for Research and Graduate Studies. He has co-authored the book Digital Filtering in One-D and Two-Dimensions; Design and Applications (New York, Plennum, 1989) and has authored over 500 articles in these areas. His current research interests include digital signal processing, machine vision, pattern recognition, neural network architectures, applications, and VLSI implementation, computer arithmetic, and MEMS.

Dr. Ahmadi is a fellow of IET. He was a recipient of an Honorable Mention Award from the Editorial Board of the Journal of Pattern Recognition in 1992, and the best paper award from the 2011 IEEE International Electro/Information Technology Conference. He received the Distinctive Contributed Paper Award from the Multiple-Valued Logic Conference Technical Committee and the IEEE Computer Society in 2000, the Distinguished University Professorship in 2003, the Faculty of Engineering Deans Special Recognition Award in 2007, and the University of Windsor Award for Excellence in Scholarship, Research, and Creative Activity in 2008. He was the IEEE-CAS representative on the Neural Network Council and the Chair of the IEEE Circuits and Systems Neural Systems Applications Technical Committee in 2000. He has served on the Editorial Board of the Journal of Circuits, Systems, and Computers as an Associate Editor and a Regional Editor from 1992 to 2012, an Associate Editor for the Journal of Pattern Recognition since 1992.



Mostafa Rahimi Azghadi (S'07–M'14) completed his PhD in Electrical & Electronic Engineering at The University of Adelaide, Australia, earning the Doctoral Research Medal, and the Adelaide University Alumni Medal in 2014. From 2012-2014, he was with the Neuromorphic Cognitive System group, Institute of Neuroinformatics, University and Swiss Federal Institute of Technology (ETH) Zurich, Switzerland.

He is currently a lecturer at the College of Science and Engineering, James Cook University, Townsville, Australia, where he researches neuromorphic engineering and brain-inspired architectures, and FPGA-acceleration of machine learning algorithms. Dr. Rahimi was a recipient of several national and international awards and scholarships such as Queensland Young Tall Poppy Science Award in 2017 and South Australia Science Excellence Awards in 2015. He serves as an associate editor of *Frontiers in Neuromorphic Engineering* and *IEEE Access*.