

## Article

# Hardware-Efficient Stochastic Computing-Based Neural Networks with SNN-Isomorphic LIF Activation

Jiho Kim , Kaeun Lim  and Youngmin Kim \* 

School of Electronics and Electrical Engineering, Hongik University, Seoul 04066, Republic of Korea; kimjiho@g.hongik.ac.kr (J.K.); c5292604@g.hongik.ac.kr (K.L.)

\* Correspondence: youngmin@hongik.ac.kr

## Abstract

Recent advances in artificial intelligence have made power efficiency a primary objective in system design. In this context, stochastic computing (SC), which processes probabilistic bitstreams using simple logic, and spiking neural networks (SNNs), a neuromorphic paradigm, have gained prominence as alternative approaches. This study proposes a Stochastic Computing Neural Network (SC-NN) framework that minimizes the intrinsic errors of stochastic computing and leverages the isomorphism between one-count operations on bitstreams and spike-rate computations in spiking neural networks, yielding improvements in accuracy and hardware efficiency. In contrast to earlier studies that utilized independent random number sequences of 10 bits or higher, our study employed a practically implementable 8-bit linear feedback shift Register (LFSR)-based pseudo-random bitstream. Using 4 taps and 255 seeds improves the realism of the hardware. Despite the inherent accuracy ceiling of pseudo-random sequences, the proposed method achieves higher accuracy. Applied to an 8-bit SC-based neural network accelerator, the proposed design improves accuracy by 35% over a conventional FSM baseline, while reducing power and area by 43.8% and 17.2%, respectively, and decreasing delay by 5.5%. These improvements translate to a 2.3× enhancement in the Figure of Merit (FoM), which was further verified through physical layout and FPGA results. Overall, this work introduces a new paradigm that enables simultaneous gains in accuracy and efficiency for low-power AI by suppressing the error sources and embedding the structural similarity between SNNs and SC into the design.

**Keywords:** stochastic computing; spiking neural network; leaky integrate-and-fire; neuromorphic computing; low-power design; hardware accelerator



Academic Editors: Ashiq A. Sakib,  
Kiat Seng Yeo and Ronald F. DeMara

Received: 16 December 2025

Revised: 16 January 2026

Accepted: 10 February 2026

Published: 11 February 2026

**Copyright:** © 2026 by the authors.

Licensee MDPI, Basel, Switzerland.

This article is an open access article distributed under the terms and conditions of the [Creative Commons Attribution \(CC BY\)](https://creativecommons.org/licenses/by/4.0/) license.

## 1. Introduction

As deep-learning inference becomes ubiquitous, even in edge and Internet of Things (IoT) settings, stochastic computing (SC) [1,2], renowned for its area and power efficiency, has regained attention. Conventional binary-integer arithmetic requires complex circuits, such as adders and multipliers [3], which significantly impact power consumption and area. In contrast, SC represents values as probabilistic bitstreams, enabling basic operations such as multiplication and addition to be performed with simple logic gates [4,5].

The strengths of stochastic computing include achieving high parallelism with modest hardware resources [6], exhibiting a high degree of fault tolerance, and reducing memory bandwidth requirements, which in turn improves overall system efficiency [7]. However, a finite bitstream length introduces error in probability estimation and stochastic computing

is sensitive to stochastic sequence correlation (SSC), including correlation between inputs and weights, which can degrade accuracy [8,9]. Within the neuromorphic paradigm, spiking neural networks (SNNs) offer strong potential for low-power operation owing to their sparse, event-driven dynamics [10,11]. They are also capable of performing approximate statistical computation through spike rate integration over time [12]. Nevertheless, SNNs continue to encounter notable challenges, including the difficulty of model conversion and training, overhead from discrete time steps, and loss of numerical precision [13,14]. These complementary characteristics suggest that combining the two paradigms judiciously could alleviate the accuracy–resource trade-off in a new way. Although many Stochastic Computing-based Neural Network (SC-NN) studies have attempted to enhance accuracy through longer bitstreams or more complex state machines, these strategies frequently compromise the fundamental goal of low-power operation [15–18]. To address these limitations, the main contributions of this paper are as follows:

- **Preprocessing Layer:** We address the main error source in bipolar stochastic multiplication by adding a preprocessing layer that removes the “Dead Zone”. This reduces stochastic error by over 113.7×.
- **LIF Activation:** Building on the observation that SNNs are computationally isomorphic to SC, we propose a leaky integrate-and-fire (LIF) model for SC-NNs. This method enhances accuracy and improves the figure of merit (FoM) by more than 2.3×.
- **On-chip SNG:** We replaced an independent random bitstream with a pseudo-random bitstream produced by an implementable 8-bit linear feedback shift register (LFSR) and realized the design on actual SoC hardware.

Collectively, by introducing the minimal preprocessing and a dynamic activation mechanism required for practical SC deployment, we establish a new design paradigm that unifies SC and SNN within a single layer. The remainder of this paper is organized as follows. Section 2 reviews prior work on SC, outlines the operating principles of SNNs, and critically examines the limitations of existing SC-NN approaches. Section 3 presents the proposed SC-NN architecture. Section 4 presents the experimental results. Section 5 concludes with a discussion of the contributions and their implications.

## 2. Background

### 2.1. Stochastic Computing

SC represents real numbers as probabilistic bitstreams and performs arithmetic using only simple combinations of logic gates [19]. In a bitstream of length  $N$ , the frequency of ones encodes the magnitude of the value. When the stream is sufficiently long and the samples are mutually independent, the expected value converges to the target value. LFSRs serve as the standard for digital pseudo-random number generation where high-precision SNGs are critical [20–22], and recent studies have expanded this field by introducing single-device-based sources to achieve high-precision stochastic number generation [23]. A comparator (CMP) or weighted-bit generator (WBG) is then used to encode a fixed-point value  $y$  as the frequency of ones in the bitstream [24]. Increasing the stream length  $N$  reduces the approximation variance, but increases cycle count, power, and area, necessitating an appropriate balance.

SC offers two encoding schemes: unipolar and bipolar. For the unipolar case where  $x$  ranges from 0 to 1, the encoding uses  $P(\text{bit} = 1) = x$ , so the probability equals the value itself. For the bipolar case where  $x$  ranges from  $-1$  to  $1$ , the encoding is  $P(\text{bit} = 1) = (x + 1)/2$ , and decoding recovers  $x = 2P(\text{bit} = 1) - 1$  [2,3]. For example, with the 8-bit stream “01101101”, the count of ones is 5. Unipolar decoding yields  $5/8$ . Bipolar decoding yields  $(5/8) \times 2 - 1 = 2/8$ .

Regardless of bitstream types, SSC is critical to accuracy [25]. A shared LFSR or any strong inter-stream correlation reduces computational performance and typically requires multiple SNGs to break the correlation [26]. However, the number of seed combinations for an 8-bit LFSR is only 256, which is smaller than the minimum typical count of inputs and weights/biases in a neural network. In such cases, one can vary the bitstreams by adjusting the LFSR tap positions. To achieve the maximum period  $2^N - 1$  for a register of width  $N$ , the characteristic polynomial over GF(2) must be primitive and therefore irreducible. Specifically,

- Irreducibility: the polynomial must not factor into a product of lower-degree polynomials.
- Primitivity: the polynomial must generate all nonzero elements of the corresponding finite field.

Additionally, under limited bit-width, correlation can be further reduced by techniques such as inserting NOT/XNOR gating within or around the LFSR path, or by mixing CMP and WBG encoders to diversify the generated bitstreams.

## 2.2. Activation Function Using Stochastic Computing

A key determinant of an artificial neural network's performance is the activation function, which imparts nonlinearity. When SC relies on gate-level linear operations [27], errors arise not only from the bitstream representation but also from the Taylor-series approximation used for nonlinear functions. Polynomial terms require extra stochastic number generators (SNGs) or cycles to control SSC, increasing energy use [4]. Consequently, there are inherent limits to accurately approximating nonlinear activations such as hyperbolic tangent (tanh) and sigmoid and ReLU using gate-only logic.

To address this issue, the SC approach using a finite-state machine (FSM) was first introduced [28]. Given an input stream of length  $N$ , the state increases by  $+1$  when the input bit is '1' and decreases by  $-1$  when the input bit is '0'. A saturating up/down counter evolves the state. When the state crosses preset boundaries [29,30], the circuit outputs 0 or 1, thereby approximating the target function. A larger  $M$  improves fidelity but increases latency and resource usage. Previous studies have demonstrated that this FSM approach facilitates useful nonlinear approximations with relatively small area and power requirements [31–33].

To overcome the limitations of a one-dimensional FSM, a two-dimensional FSM approach has been proposed as shown in Figure 1 [34]. An SNG converts a real-valued input  $X$  into a bitstream with probability  $P_X$ . Along with calibration control stream with probability  $P_K$ , this pair drives an  $M \times N$  FSM that evolves according to the transition rules. Here, parameter  $K$  functions as a calibration term, dependent on the target function, that effectively shifts its input axis. State transitions are controlled by the four configurations induced by  $X$  and  $K$ . According to the transition rules, once a state is asserted, the corresponding weight bitstream  $P_W$  is driven to the output  $Y$ . For each state, the associated weight bitstream  $P_W$  is emitted as output  $Y$ , and the accumulated count is subsequently reported by a counter. Unlike a 1-D FSM, the 2-D FSM offers universality for function implementation in principle [24]. However, it requires comparatively greater hardware resources.

## 2.3. Spiking Neural Network

Artificial neural networks (ANNs) are computing systems inspired by the operating principles of biological neurons in the human brain [35]. Whereas a conventional ANN propagates continuous real-valued activations and applies nonlinear functions to them, an SNN models the biological spiking mechanism and encodes inputs as discrete spikes. Figure 2 illustrates the fundamental operation of an SNN, demonstrating how a pre-synaptic neuron transmits the synaptic weight ( $W$ ) to a post-synaptic neuron via the

frequency of discrete spike signals. Unlike ANNs, a neuron in an SNN emits a spike only when its membrane potential reaches a prescribed threshold voltage  $V_{th}$ . This communication scheme resembles how the brain conveys information via action potentials. Therefore, SNNs align more closely with neural processes than conventional networks [36,37]. Representative neuron models in SNNs include the leaky integrate and fire (LIF) model and the integrate and fire (IF) model [38,39]. As shown in the timing diagram of Figure 2, the pre-synaptic neuron generates spikes at specific instants. With each incoming spike in the internal state, the membrane potential increases in a stepwise manner. Between spikes, the membrane potential decays due to leakage. When the accumulated input raises the membrane potential above a threshold  $V_{th}$ , the neuron fires a spike and subsequently resets its potential [40,41]. This mechanism allows SNNs to approximate biological neural computation more effectively than ANNs.

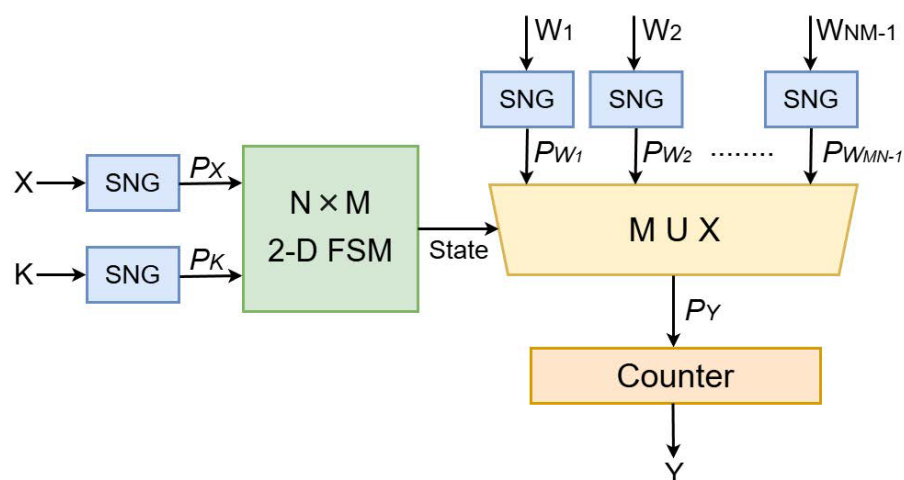


Figure 1. Block diagram of the 2-D FSM-based architecture.

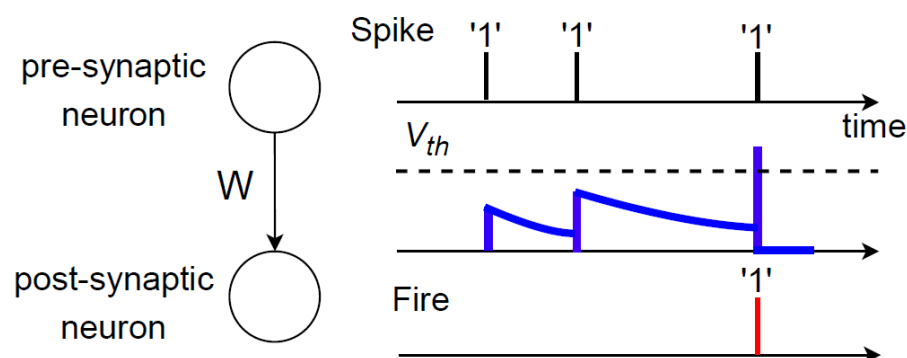


Figure 2. Operating principle of the LIF neuron: input spikes (black),  $V_i$  (blue) and output fire (red).

#### 2.4. Stochastic Computing Neural Network

Research on neural networks implemented with stochastic computing has continued to advance [42–44]. For each input-weight stream pair, unipolar encoding uses an AND gate and bipolar encoding uses an XNOR gate, and a MUX combines the resulting streams to realize the weighted sum. Neuron outputs are then produced using either gate- or FSM-based activations [45]. Training is generally performed in the real-valued domain, and the learned weights and biases are mapped to hardware. More recently, methods that perform learning directly in the SC domain have been proposed, which effectively mitigate quantization error introduced during mapping [46,47].

SC-NNs offer several advantages. Their simple circuits reduce area, routing complexity, and power consumption [48]. They are also robust to errors [49], since bit-level

faults have limited global impact. Furthermore, stream-based processing reduces memory traffic and bandwidth, lowering power [50]. In contrast, achieving higher accuracy often requires longer streams, which increase cycle count, power, and area. In addition, stream correlation degrades accuracy, frequently necessitating multiple SNGs that further increase cost. Attempts to improve accuracy by lengthening streams or adding SNGs also raise circuit complexity, eroding some of SC's intrinsic simplicity and efficiency. Nevertheless, SC-NNs remain attractive as they achieve substantial power savings while requiring low hardware resources.

### 3. Proposed SC-NN Architecture

This study proposes an SC-based neural network with a sequential pipeline composed of a Preprocessing Layer, Fully Connected (FC) Layers, and an Output Layer as shown in Figure 3. The FC widths are 784-100-200-10. To reduce computation and energy, FC1 is shrunk to 100. To compensate for any accuracy loss, FC2 is expanded to 200.

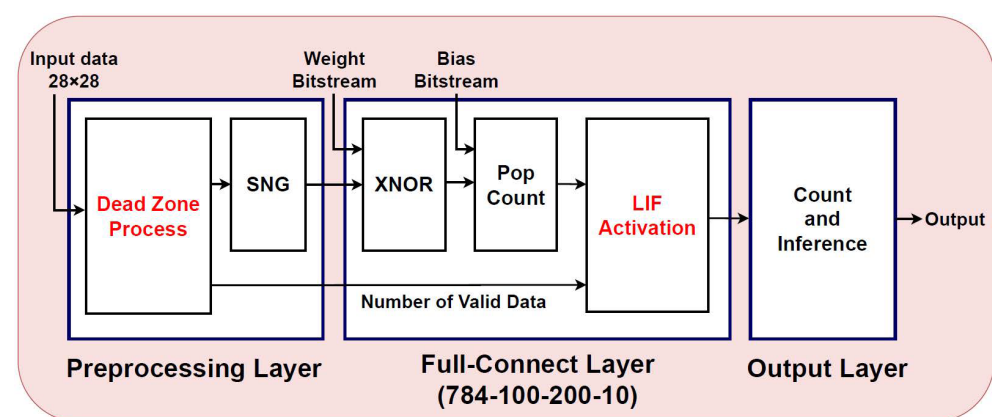


Figure 3. Proposed SC-NN block diagram.

All weights, biases, and input data are quantized to signed 8-bit form. From a deployment perspective, we select 8 bits as the default bit-width by considering the minimum transfer granularity of practical systems. Using 10 or 12 bits introduces packing and unpacking logic in the storage and transfer stages [51]. This increases control complexity, the number of clock cycles, area, and power. For example, moving to 10 bits increases memory bandwidth by roughly 25%. Area and power increase by approximately 25% each. The cycle count becomes  $4\times$  that of the 8-bit case, so the figure of merit (Power  $\times$  Delay  $\times$  Area) worsens by about  $6.25\times$ . Therefore, we adopt 8 bits. In an 8-bit LFSR, we select four out of sixteen available tap sets and vary the seed to generate 1024 distinct sequences, which can be shared across the network [24].

#### 3.1. Proposed Preprocessing Layer

In bipolar SC, multiplication can be achieved using a single XNOR gate, which is advantageous in terms of ultra-low power consumption [9]. Consider real numbers  $X$  and  $Y$ , each taking values from  $-1$  to  $1$ , that are linearly mapped to 8-bit two's complement integers from  $-128$  to  $127$ . Let the bipolar bitstreams corresponding to  $X$  and  $Y$  be denoted by  $P_X$  and  $P_Y$ ; the XNOR-gate error is analyzed as follows. When multiplying completely independent bitstreams  $P_X$  and  $P_Y$ , the Mean Squared Error (MSE) remains substantially consistent even when the absolute values of  $X$  and  $Y$  are near zero, as illustrated in Figure 4a. However, when pseudo-random bitstreams are generated using an 8-bit LFSR and the XNOR MSE is evaluated, the result shown in Figure 4b is obtained. Based on this, the relative error analysis yields the MSRE distribution shown in Figure 4c. It can

be observed that the error increases when the absolute values of  $X$  and  $Y$  are near zero, and surges significantly when both values are close to zero.

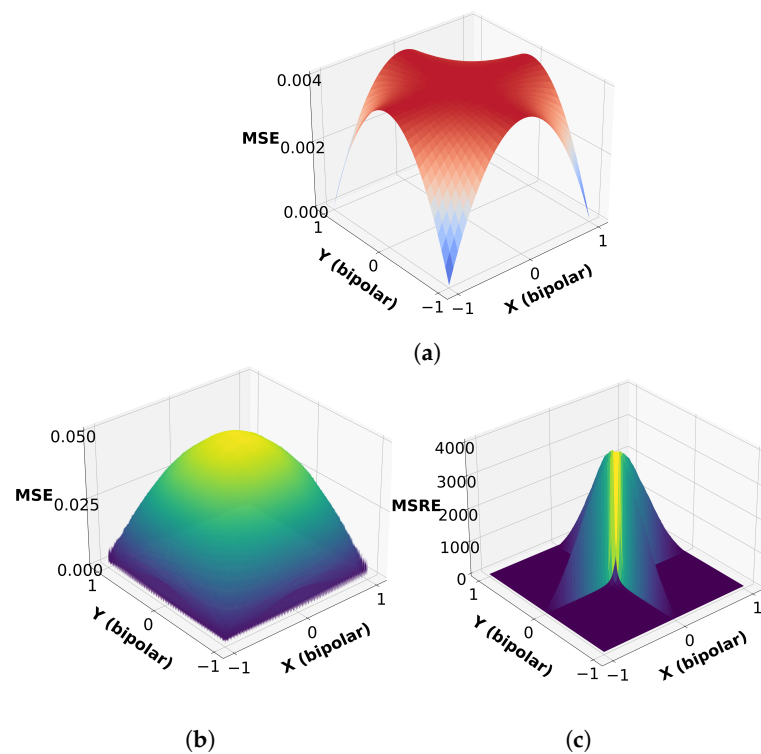


**Figure 4.** Comparison of error for bipolar SC multiplication. (a) Theoretical MSE. Simulated (b) MSE and (c) MSRE.

Accordingly, this work defines the region in which the XNOR-gate error sharply increases as a Dead Zone. The ratio of this width is denoted as  $\tau$ , resulting in a total Dead Zone width of  $2\tau$ . To eliminate this dead zone, we propose a preprocessing layer as illustrated in Figure 5, which prunes signed 8-bit input values falling within the absolute value of  $\tau$ , thereby bypassing computation for these inputs. It is crucial to note that the threshold parameter  $\tau$  is applied uniformly across all network layers. This is because the Dead Zone arises from the inherent mathematical error characteristics of bipolar SC multiplication, rather than the specific data distribution of a layer. If  $\tau$  varies across layers, specifically if a preceding layer has a smaller  $\tau$  than a subsequent layer, valid information preserved upstream is discarded downstream, causing information loss. Conversely, a larger upstream  $\tau$  followed by a smaller downstream  $\tau$  allows noise to propagate, which may be amplified. Consequently, the preprocessing layer counts the number of valid inputs and forwards this statistic as a scalar value to the fully connected layer so that it can be used by the activation for dynamic normalization. Furthermore, to address the Dead Zone along the weight axis, the network is trained to prune weights within the range  $[-\tau, \tau]$ . This effectively eliminates the cross-shaped data distribution near zero, where XNOR gate errors are most prevalent.

Figure 6 shows the results. The overall average error, represented by the blue line, is reduced to the level of the purple line by removing near-zero weights. With the additional application of the preprocessing layer, the error is further reduced to the level of the green line. Quantitatively, the error is  $113.7\times$  lower than the overall and  $10.7\times$  lower than using near-zero weight removal alone. Prior SC-NN preprocessing typically stopped at converting real values to bitstreams via SNGs [45]. In contrast, the proposed preprocessing eliminates terms concentrated in error-prone regions and propagates the count of valid terms, serving as the key differentiator.

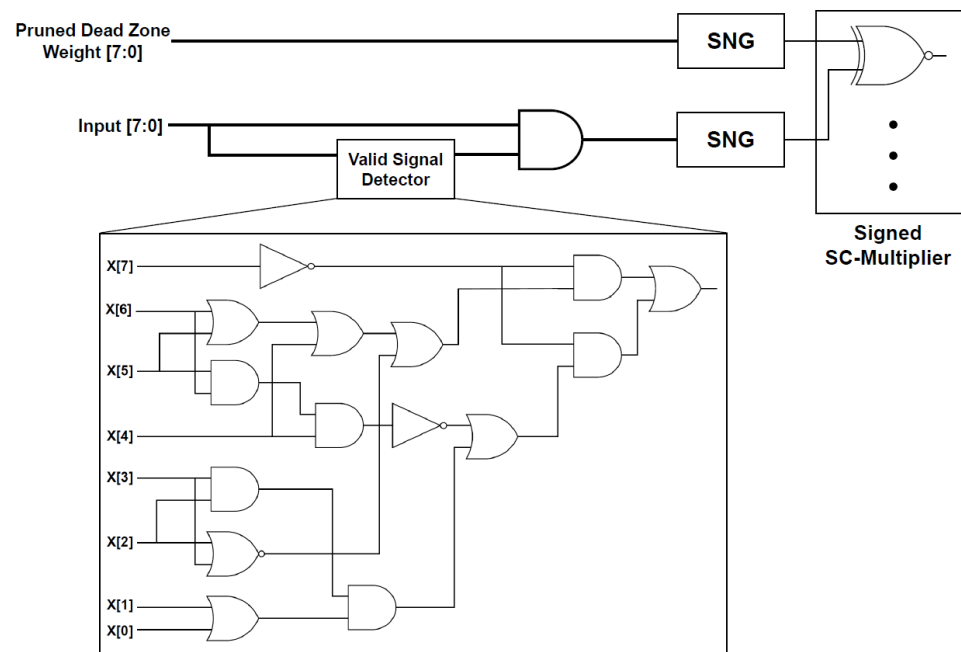


**Figure 5.** Schematic of the preprocessing layer architecture implementing valid signal detector for Dead Zone elimination.

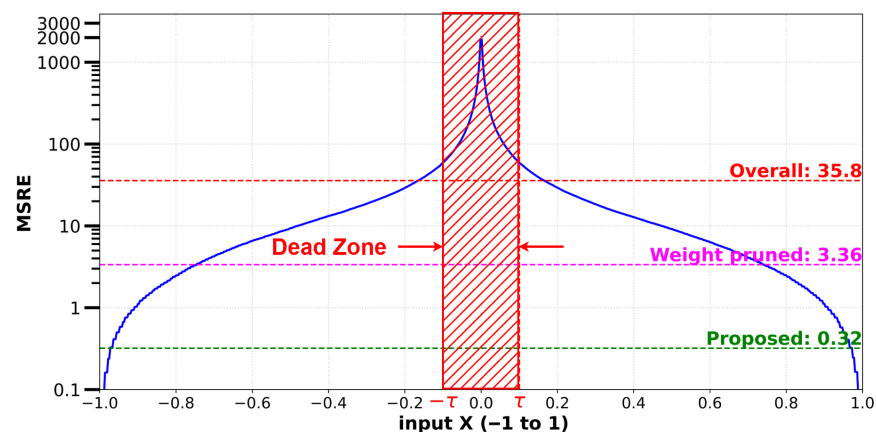


**Figure 6.** Average error comparison between the overall, near-zero weights pruning, and the proposed preprocessing method.

### 3.2. Activation Using Leaky-Integrate Fire Model

Implementations of activation functions with stochastic computing have typically employed FSM [42]. However, the number of states is fixed, and the resulting slope remains steep and fixed, as illustrated in Figure 7. Exploiting the isomorphism between computation on the population of ones in stochastic bitstreams and computation on spike rates in SNNs, we implement a tanh activation in the form of a LIF unit. In the proposed LIF design, the circuit counts the ones and subtracts a baseline determined by the number of valid data items. The LIF also performs per-clock leakage on the accumulated state, enabling an SC-compatible, clock synchronous tanh. Consequently, the design reduces power and area while improving accuracy. The proposed LIF neuron operates as shown in Algorithm 1. For each input vector  $X_t$ , the unit counts the total number of ones, denoted as  $B_t$ . A key feature of this model is its adaptive reference value, which is calculated based on the number of valid data points within the current input  $X_t$ .

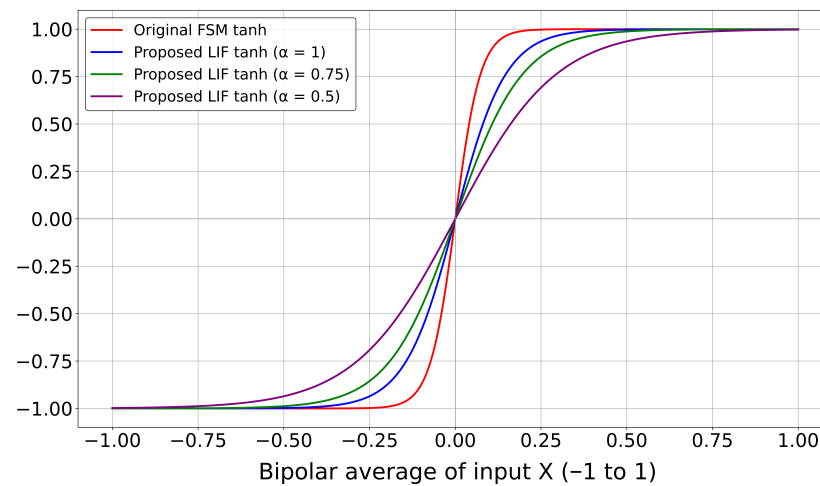


**Figure 7.** Comparison of transfer characteristics for the tanh activation between FSM and proposed LIF implementation.

---

**Algorithm 1** Proposed LIF Neuron

---

```

//  $\mathbf{X}_t$  is the input bit vector of each column.
//  $Y_t$  is Fired value, either 1 or 0.
//  $V_i$  is membrane potential of the neuron.
LIF Update ( $\mathbf{X}_t, Y_t$ )
   $B_t \leftarrow$  The number of 1s in  $\mathbf{X}_t$ 
   $\alpha \leftarrow$  Input scaling //  $\alpha$  is leakage
   $Integrate_t \leftarrow B_t - (\text{Number of valid data in } \mathbf{X}_t)/2$ 
   $V_i(t) \leftarrow V_i(t-1) + Integrate_t * \alpha$ 
  If  $V_i(t) > V_{th}$  then //  $V_{th}$  is threshold value
     $Y_t \leftarrow 1$ 
  else
     $Y_t \leftarrow 0$ 

```

---

This context-dependent reference is subtracted from  $B_t$  to generate an effective input. Instead of applying a traditional leak to the stored potential, our model scales this effective input by a factor  $\alpha$  before integration. This input-scaling mechanism functions analogously to leakage by regulating the neuron's overall excitability and sensitivity to incoming signals. Specifically,  $\alpha$  is a configurable parameter that mathematically dictates the slope of the tanh activation function. Unlike FSM-based approaches that are limited to a fixed slope, our design allows for precise slope modulation. Setting  $\alpha = 1$  results in a sensitivity effectively half that of the conventional FSM, offering a wider dynamic range. Consequently, this parameter provides the mathematical flexibility to implement different activation slopes by simply adjusting the  $\alpha$  value as required by various network models. When the membrane potential reaches or exceeds a threshold value  $V_{th}$ , the unit emits  $Y_t = 1$ ; otherwise, it outputs  $Y_t = 0$ . Unlike standard models, the neuron does not perform a hard reset after firing, allowing it to retain a memory of past inputs. This design allows the firing rate to be precisely regulated. While  $V_{th}$  is primarily established at half the valid data count to maintain zero-centered symmetry, it can be dynamically adjusted to control output sparsity, mirroring the firing threshold mechanism in biological SNNs. Moreover, this synchronous design with its adaptive baseline and tunable parameters improves accuracy while reducing power and area, as it eliminates complex asynchronous control logic and simplifies state register requirements.

The mathematical process by which the above operation approximates the tanh function is as follows. At each clock, the number of ones  $B_t$  in a column follows a binomial

distribution with effective input count  $n$  and single-bit success probability  $p$ . Here,  $p$  is the probability that the XNOR result equals 1 at that clock, as given in (1). When the baseline for valid data is set to half of the incoming data in FC1, to 100 in FC2, and to 50 in FC3, the probability that the output bit is 1 is given in (2), where  $F$  denotes the CDF of the binomial distribution. Using the central limit theorem (CLT), the continuous approximation in (3) is obtained. In the bipolar setting,  $t = 2p - 1$  and  $\phi$  denotes the standard normal CDF. Thus, the expected output equals the tail probability of a binomial CDF, which forms an S-shaped curve symmetric about  $p = 1/2$ . This curve is well approximated by the logistic function  $\sigma$  and, via an affine mapping, by  $\tanh$ , as given in (4). These results indicate that both the LIF dynamics and the FSM mechanism can serve as  $\tanh$  activation.

$$B \sim \text{Bin}(n, p), \quad \Pr(B = b) = \binom{n}{b} p^b (1-p)^{n-b} \quad (1)$$

$$\Pr(Y = 1|p) = \Pr\left(B > \frac{n}{2}\right) = 1 - F\left(\frac{n}{2}; n, p\right) \quad (2)$$

$$\Pr\left(B > \frac{n}{2}\right) \approx \Phi(\sqrt{n}(2p - 1)) = \Phi(\sqrt{nt}) \quad (3)$$

$$\phi(x) \approx \sigma(\alpha x), \quad y(t) \approx \tanh(\kappa t) \quad (4)$$

#### 4. Result Analysis

The proposed SC-NN was evaluated on the MNIST dataset with  $28 \times 28$  inputs using 10,000 test samples. All inputs, weights, and biases were represented as signed 8-bit integers and converted into bitstreams using pseudo-random number sequences generated by LFSRs. To evaluate the robustness of the proposed architecture against stochastic fluctuations and sequence correlations, we performed Monte Carlo simulations across multiple LFSR configurations. Specifically, we utilized 4 distinct primitive polynomial taps and 255 initial seeds, resulting in a total of 1020 unique pseudo-random instances. The accuracy metrics reported in this section represent the mean values derived from these extensive trials. The observed standard deviation of the accuracy was minimal, ranging from 0.05% to 0.25%, confirming that the performance gains are stable and not dependent on favorable random number sequences. The design was synthesized using Synopsys Design Compiler in 65 nm CMOS technology under typical conditions (1.0 V, 25 °C) with a target clock period of 10 ns (100 MHz) to analyze the power, delay, and area overhead of a single module. The layout for each activation module in Fully Connected Layer 1 was implemented via Place and Route (PnR) using Cadence Innovus with commercial 65 nm standard cells. Additionally, FPGA performance was validated using Xilinx Vivado 2023.1.

To assess the contribution of the proposed approach, we compared accuracy with and without the preprocessing layer, and additionally evaluated the method against conventional FSM-based activation and the proposed LIF-based activation; the results are summarized in Table 1. The baseline that does not compensate for XNOR error yields an accuracy of 65.5%. Pruning weights with  $|\omega| < 0.1$  without modifying the input values increases accuracy by 1.9% points. Notably, applying the proposed preprocessing layer alone to prune inputs within the same range, while retaining all weights, achieves an accuracy of 77.4%, demonstrating that eliminating input-dependent errors is highly effective even without weight constraints. When the preprocessing layer is added to eliminate the cross-shaped input values responsible for XNOR gate errors, the accuracy improves to 82% and 88.4% when using FSM and LIF activation functions, respectively. Compared to the FSM model, the LIF model can more precisely control the effective sensitivity of the  $\tanh$ -like response, resulting in higher accuracy. With  $\alpha = 1$ , the LIF model achieves an accuracy that is 6.4% points higher than the FSM model. The impact of varying the Dead Zone ratio  $\tau$  on accuracy is shown at the bottom of Table 1. Larger  $\tau$  degrades

accuracy by reducing the amount of usable data, whereas smaller  $\tau$  increases stochastic-computing error by retaining near-zero values. These results indicate that  $\tau = 0.1$  provides the most favorable trade-off. Overall, the proposed 8-bit SC-NN implemented with pseudo-random hardware demonstrates higher accuracy than previously reported 10-bit hardware implementations [50,52].

**Table 1.** Comparison of SC-NN accuracy on average for the MNIST dataset with FSM-based and proposed LIF-based activation.

| SC-NN Activation                                                    |              | Accuracy |
|---------------------------------------------------------------------|--------------|----------|
| Conventional using FSM-based [45]                                   |              | 65.5%    |
| Near-Zero weight removed using FSM-based [45]                       |              | 67.4%    |
| Proposed preprocessing w/o weight pruning using FSM-based           |              | 77.4%    |
| Proposed preprocessing with weight pruning using FSM-based [45]     | $\tau = 0.4$ | 73.2%    |
|                                                                     | $\tau = 0.3$ | 77.9%    |
|                                                                     | $\tau = 0.2$ | 80.4%    |
|                                                                     | $\tau = 0.1$ | 82.0%    |
| Proposed preprocessing with weight pruning using proposed LIF-based | $\tau = 0.4$ | 77.8%    |
|                                                                     | $\tau = 0.3$ | 79.5%    |
|                                                                     | $\tau = 0.2$ | 82.7%    |
|                                                                     | $\tau = 0.1$ | 88.4%    |

Table 2 reports the hardware resource consumption and delay for each single activation module. Unlike the FSM, the LIF module operates only when valid inputs are received, offering power advantages. Furthermore, unlike the FSM, which relies on the number of state registers to define a fixed slope for the activation function, the LIF model requires fewer registers, resulting in reductions in both area and delay. At the activation module level, the proposed design delivers 43.8% lower power, 5.5% reduced delay, and 17.2% smaller area compared to conventional designs. The FoM is defined as the product of power, delay, and area (PPA), representing the overall hardware efficiency. The power, delay, area, and FoM values are normalized to the FSM-based design, where the metrics for the proposed LIF-based design are presented as relative ratios. This translates to a 2.3× improvement in the FoM normalized by power, delay, and area. Furthermore, validation on the Xilinx xc7a100tcs324-3 demonstrates that the proposed LIF-based activation significantly outperforms the FSM-based activation in terms of hardware efficiency, as illustrated in Table 3. Consistent with the analysis in Table 2, the metrics in Table 3 are normalized to the FSM-based reference. Specifically, the proposed design achieved a 17.6% reduction in power consumption and a 17.1% decrease in critical path delay compared to the FSM baseline. Regarding resource utilization, a significant reduction is observed in both Look-Up Table (LUT) and Flip-Flop (FF) usage. Although both architectures require the same 255 cycles for 8-bit MNIST inference, the reduced critical path delay in the proposed design implies a higher maximum operating frequency, directly leading to lower latency and faster real-time processing.

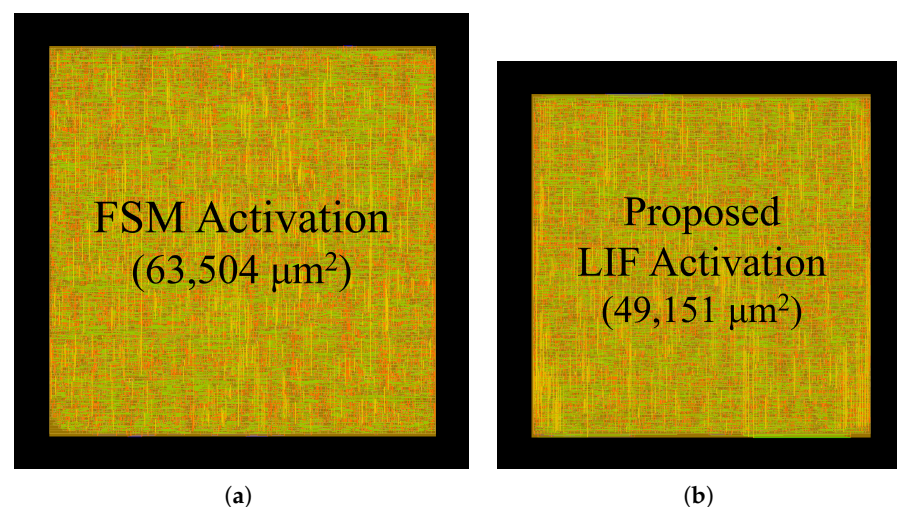
**Table 2.** Comparison of hardware overhead and FoM for a single activation module between FSM-based and proposed LIF-based designs in 65 nm CMOS.

| SC-NN<br>Activation | Power |       | Delay |       | Area                |       | FoM   |       |
|---------------------|-------|-------|-------|-------|---------------------|-------|-------|-------|
|                     | (mW)  | Norm. | (ns)  | Norm. | ( $\mu\text{m}^2$ ) | Norm. | (PPA) | Norm. |
| FSM-based [45]      | 0.774 | 1.0   | 1.798 | 1.0   | 2994.3              | 1.0   | 4167  | 1.0   |
| Proposed LIF-based  | 0.435 | 0.56  | 1.700 | 0.95  | 2479.8              | 0.83  | 1934  | 0.44  |

**Table 3.** Comparison of power, delay, and resource efficiency for FPGA implementation.

| SC-NN<br>Activation | Power |       | Delay |       | Cycle |       | Utilization |     |
|---------------------|-------|-------|-------|-------|-------|-------|-------------|-----|
|                     | (mW)  | Norm. | (ns)  | Norm. | (#)   | Norm. | LUT         | FF  |
| FSM-based [45]      | 17.0  | 1.0   | 3.492 | 1.0   | 255   | 1.0   | 443         | 123 |
| Proposed LIF-based  | 14.0  | 0.82  | 2.895 | 0.83  | 255   | 1.0   | 113         | 114 |

Figure 8a,b show the post-PnR layouts of the baseline FSM module and the proposed LIF module, respectively. Both designs were implemented with a target utilization density of 86–87% to ensure a fair comparison. The measurement results indicate that the total area of the FSM module is  $63,504 \mu\text{m}^2$ , whereas the proposed LIF module occupies  $49,151 \mu\text{m}^2$ . Such a compact physical footprint implies that the proposed approach not only alleviates routing congestion but also enables higher integration density, demonstrating its superior hardware efficiency for SoC implementations.

**Figure 8.** Post-PnR layouts of activation modules in the first fully connected layer using commercial 65 nm standard cells. (a) Baseline FSM module and (b) proposed LIF module.

## 5. Discussion and Conclusions

This study proposes a hardware-efficient SC-NN framework designed to minimize intrinsic calculation errors while maximizing hardware efficiency. Prior SC-NN architectures have predominantly relied on FSM-based activation functions, which entail fixed activation slopes and considerable hardware overhead. We address this research gap by introducing a novel activation paradigm that moves beyond conventional FSMs. By leveraging the structural isomorphism between the population count in stochastic bitstreams and the dynamics of SNNs, we implemented a LIF-based activation function. Concurrently, we identified that the error in bipolar SC multiplication is heavily concentrated in a cross-shaped region and implemented a novel Preprocessing Layer to eliminate this Dead Zone. This dual approach

effectively integrates error mitigation with a dynamic activation mechanism, enhancing both accuracy and efficiency.

This work is significant in that it moves beyond theoretical simulations to practical hardware realization. We employed pseudo-random bitstreams generated by a realistic 8-bit LFSR and verified the SC-NN based on 8-bit integer quantization at the hardware level. Quantitatively, the proposed design demonstrates distinct advantages over traditional FSM-based baselines: it reduces power consumption by 43.8%, area by 17.2%, and delay by 5.5%, ultimately yielding a  $2.3\times$  enhancement in the FoM. Unlike previous SC-NNs that often relied on ideal stochastic sources in software simulation, our results reflect practical hardware constraints and verify that the proposed LIF-based approach offers a superior trade-off between accuracy and hardware cost compared to existing FSM implementations.

Furthermore, the proposed paradigm comprising dead-zone elimination and LIF activation is not limited to specific network topologies but can be applied as a complementary module to enhance the performance of various SC-NN architectures. Although this study primarily verified the architecture on the MNIST dataset using a fully connected topology, the results demonstrate robust scalability. The fundamental validity of the proposed error mitigation and activation schemes suggests that they can be effectively extended to deeper architectures using stochastic computing and more complex datasets, such as CIFAR-10 or ImageNet, which remains a subject for future investigation. Continued research that prioritizes these novel techniques over resource-intensive scaling will further solidify SC-based networks as a practical alternative for edge and mobile devices.

**Author Contributions:** Conceptualization, J.K.; methodology, J.K.; software, J.K.; validation, J.K. and K.L.; formal analysis, J.K.; investigation, J.K.; resources, J.K.; data curation, J.K.; writing—original draft preparation, J.K. and K.L.; writing—review and editing, Y.K.; visualization, J.K. and K.L.; supervision, Y.K.; project administration, Y.K.; funding acquisition, Y.K. All authors have read and agreed to the published version of the manuscript.

**Funding:** This research was supported by the Ministry of Science and ICT (MSIT), Republic of Korea.

**Data Availability Statement:** All data underlying the results are available as part of the article and no additional source data are required.

**Acknowledgments:** This work was partly supported by the Institute of Information & Communications Technology Planning & Evaluation (IITP)—ITRC (Information Technology Research Center) grant funded by the Korea government (MSIT) (IITP-2025-RS-2022-00156225) and by the 2025 Hongik University Innovation Support Program Fund. The EDA tool was supported by the IC Design Education Center (IDEC), Korea.

**Conflicts of Interest:** The authors declare no conflicts of interest.

## References

1. Gaines, B.R. Stochastic computing. In Proceedings of the AFIPS Spring Joint Computer Conference, Atlantic City, NJ, USA, 18–20 April 1967; pp. 149–156.
2. Gaines, B.R. Stochastic computing systems. In *Advances in Information Systems Science*; Tou, J.T., Ed.; Springer: New York, NY, USA, 1969; Volume 2, pp. 37–172.
3. Parhi, K.K. Analysis of stochastic logic circuits in unipolar, bipolar and hybrid formats. In Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS), Baltimore, MD, USA, 28–31 May 2017; pp. 1–4.
4. Parhi, K.K.; Liu, Y. Computing arithmetic functions using stochastic logic by series expansion. *IEEE Trans. Emerg. Top. Comput.* **2019**, *7*, 44–59. [[CrossRef](#)]
5. Liu, S.; Rosselló, J.L.; Liu, S.; Tang, X.; Font-Rosselló, J.; Frasser, C.F.; Qian, W.; Han, J.; Reviriego, P.; Lombardi, F. From multipliers to integrators: A survey of stochastic computing primitives. *IEEE Trans. Nanotechnol.* **2024**, *23*, 238–249. [[CrossRef](#)]
6. Li, P.; Lilja, D.J.; Qian, W.; Bazargan, K.; Riedel, M.D. Computation on stochastic bit streams digital image processing case studies. *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.* **2014**, *22*, 449–462. [[CrossRef](#)]

7. Khataei, A.; Singh, G.; Bazargan, K. Approximate hybrid binary-unary computing with applications in BERT language model and image processing. In Proceedings of the ACM/SIGDA International Symposium on Field Programmable Gate Arrays (FPGA), Monterey, CA, USA, 12–14 February 2023; pp. 165–175.
8. Alaghi, A.; Hayes, J.P. Exploiting correlation in stochastic circuit design. In Proceedings of the IEEE 31st International Conference on Computer Design (ICCD), Asheville, NC, USA, 6–9 October 2013; pp. 39–46.
9. Akhtar, R.; Khanday, F.A. Stochastic computing: Systems, applications, challenges and solutions. In Proceedings of the International Conference on Communication and Electronics Systems (ICCES), Coimbatore, India, 15–16 October 2018; pp. 722–727.
10. Trench, G.; Morrison, A. A system-on-chip based hybrid neuromorphic compute node architecture for reproducible hyper-real-time simulations of spiking neural networks. *Front. Neuroinform.* **2022**, *16*, 884033. [\[CrossRef\]](#)
11. Liang, M.; Zhang, J.; Chen, H. A 1.13μJ/classification Spiking Neural Network Accelerator with a Single-spike Neuron Model and Sparse Weights. In Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS), Daegu, Republic of Korea, 22–28 May 2021; pp. 1–5.
12. Sen, S.; Venkataramani, S.; Raghunathan, A. Approximate Computing for Spiking Neural Networks. In Proceedings of the Design, Automation & Test in Europe Conference & Exhibition (DATE), Lausanne, Switzerland, 27–31 March 2017; pp. 193–198.
13. Kim, E.; Kim, Y. Exploring the potential of spiking neural networks in biomedical applications: Advantages, limitations, and future perspectives. *Biomed. Eng. Lett.* **2024**, *14*, 967–980. [\[CrossRef\]](#)
14. Kim, Y.; Kim, T. An analysis of learning performance changes in spiking neural networks(SNN). *J. Conver. Cult. Technol. (JCCT)* **2020**, *6*, 463–468.
15. Tang, C.; Han, J. Design and Implementation of a Highly Accurate Stochastic Spiking Neural Network. In Proceedings of the IEEE Workshop on Signal Processing Systems (SiPS), Coimbra, Portugal, 20–22 October 2021; pp. 1–6.
16. Lee, V.T.; Alaghi, A.; Hayes, J.P.; Sathe, V.; Ceze, L. Energy-Efficient Hybrid Stochastic-Binary Neural Networks for Near-Sensor Computing. In Proceedings of the Design, Automation & Test in Europe Conference & Exhibition (DATE), Lausanne, Switzerland, 27–31 March 2017; pp. 13–18.
17. Varadharajan, S.K.; Nallasamy, V.; Hayes, J.P.; Sathe, V.; Ceze, L. P-SCADA—A novel area and energy efficient FPGA architectures for LSTM prediction of heart arrhythmias in biot applications. *Expert Syst.* **2021**, *39*, e12687. [\[CrossRef\]](#)
18. Nobari, M.; Jahanirad, H. FPGA-based implementation of deep neural network using stochastic computing. *Appl. Soft Comput.* **2023**, *137*, 110166. [\[CrossRef\]](#)
19. Alaghi, A.; Hayes, J.P. Survey of Stochastic Computing. *ACM Trans. Embed. Comput. Syst.* **2013**, *12*, 1–19. [\[CrossRef\]](#)
20. Justin, R.; Mathew, B.K.; Abe, S. FPGA Implementation of high quality random number generator using LUT based shift registers. *Procedia Technol.* **2016**, *24*, 1155–1162. [\[CrossRef\]](#)
21. Salehi, S.A. Low-Cost Stochastic Number Generators for Stochastic Computing. *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.* **2020**, *28*, 992–1001. [\[CrossRef\]](#)
22. Cerda, J.C.; Martinez, C.D.; Comer, J.M.; Hoe, D.H.K. An efficient FPGA random number generator using LFSRs and cellular automata. In Proceedings of the IEEE 55th International Midwest Symposium on Circuits and Systems (MWSCAS), Boise, ID, USA, 5–8 August 2012; pp. 912–915.
23. Seo, J.; Kang, S.; Kumar, D.; Shin, W.; Cho, J.; Kim, T.; Kim, Y.; Jang, B.; Trivedi, A.; Yoo, H. Random Number Generators and Spiking Neurons from Metal Oxide/Small Molecules Heterojunction N-Shape Switching Transistors. *Adv. Funct. Mater.* **2025**, *35*, 2411348. [\[CrossRef\]](#)
24. Kim, J.; Yang, S.; Kim, Y. Efficient Hardware Implementation of Stochastic Computing with Hybrid Techniques. In Proceedings of the 23rd IEEE International Conference on NEWCAS, Paris, France, 22–25 June 2025; pp. 163–167.
25. Joe, H.; Kim, Y. Compact and Power-efficient Sobel Edge Detection with Fully Connected Cube-network-based Stochastic Computing. *J. Semicond. Technol. Sci.* **2020**, *20*, 436–443. [\[CrossRef\]](#)
26. Neugebauer, F.; Polian, I.; Hayes, J.P. Building a Better Random Number Generator for Stochastic Computing. In Proceedings of the Euromicro Conference on Digital System Design (DSD), Vienna, Austria, 30 August–1 September 2017; pp. 1–8.
27. Liu, Y.; Parhi, K.K. Computing hyperbolic tangent and sigmoid functions using stochastic logic. In Proceedings of the 50th Asilomar Conference on Signals, Systems and Computers, Pacific Grove, CA, USA, 6–9 November 2016; pp. 1580–1585.
28. Brown, B.D.; Card, H.C. Stochastic neural computation. I. Computational elements. *IEEE Trans. Comput.* **2001**, *50*, 891–905. [\[CrossRef\]](#)
29. Chu, S.-I.; Hsieh, C.-E.; Huang, Y.-J. Design of FSM-Based Function With Reduced Number of States in Integral Stochastic Computing. *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.* **2019**, *27*, 1475–1479. [\[CrossRef\]](#)
30. Ichihara, H.; Fukuda, T.; Inoue, T. A Design of Reliable Linear FSMs With Equivalent States in Stochastic Computing. In Proceedings of the IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT), Athens, Greece, 6–8 October 2021.
31. Asadi, S.; Najafi, M.H.; Imani, M. A low-cost FSM-based bit-stream generator for low-discrepancy stochastic computing. In Proceedings of the Design, Automation & Test in Europe Conference & Exhibition (DATE), Grenoble, France, 1–5 February 2021; pp. 908–913.

32. Liu, J. Optimization of Stochastic Computing Based Deep Learning Systems with Parallel Finite State Machine Implementation. In Proceedings of the International Conference on Algorithms and Computing and Systems (ICACS), Rabat, Morocco, 6–8 January 2020; pp. 22–26.
33. Li, P.; Qian, W.; Riedel, M.D.; Bazargan, K.; Lilja, D.J. The synthesis of linear Finite State Machine-based stochastic computational elements. In Proceedings of the Asia South Pacific Design Automation Conference (ASP-DAC), Sydney, Australia, 30 January–2 February 2012; pp. 757–762.
34. Li, P.; Lilja, D.J.; Qian, W.; Bazargan, K.; Riedel, M. The synthesis of complex arithmetic computation on stochastic bit streams using sequential logic. In Proceedings of the International Conference on Computer-Aided Design (ICCAD), San Jose, CA, USA, 5–8 November 2012; pp. 480–487.
35. Rosenblatt, F. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychol. Rev.* **1958**, *65*, 386–408. [[CrossRef](#)]
36. Isik, M. A Survey of Spiking Neural Network Accelerator on FPGA. *Tsinghua Sci. Technol.* **2020**, *25*, 443–457.
37. Rodrigues, J.; Cajueiro, J.P.C.; Ranhel, J. Neural Encoding and Spike Generation for Spiking Neural Network implemented in FPGA. In Proceedings of the International Conference on Electronics, Communications and Computers (CONIELECOMP), Choluta, Mexico, 25–27 February 2015; pp. 55–61.
38. Yang, Y.; Kim, Y. Approximate digital leaky integrate-and-fire neurons for energy efficient spiking neural networks. *IEIE Trans. Smart Process. Comput.* **2020**, *9*, 252–259. [[CrossRef](#)]
39. Samadzadeh, A.; Far, F.S.T.; Javadi, A.; Nickabadi, A.; Chehrehghani, M.H. Convolutional Spiking Neural Networks for Spatio-Temporal Feature Extraction. *Neural Process. Lett.* **2023**, *55*, 6979–6995. [[CrossRef](#)]
40. Kuang, Z.; Wang, J.; Yang, S.; Yi, G.; Deng, B.; Wei, X. Digital implementation of the spiking neural network and its digit recognition. In Proceedings of the Chinese Control and Decision Conference (CCDC), Nanchang, China, 3–5 June 2019; pp. 3621–3625.
41. Corradi, F.; Adriaans, G.; Stuijk, S. Gyro: A Digital Spiking Neural Network Architecture for Multi-Sensory Data Analytics. In Proceedings of the Workshop on Drone Systems Engineering and Rapid Simulation and Performance Evaluation, Amsterdam, The Netherlands, 18 January 2021; pp. 9–15.
42. Liu, Y.; Liu, S.; Wang, Y.; Lombardi, F.; Han, J. A Survey of Stochastic Computing Neural Networks for Machine Learning Applications. *IEEE Trans. Neural Netw. Learn. Syst.* **2021**, *32*, 2809–2824. [[CrossRef](#)] [[PubMed](#)]
43. Chen, L.; Xiong, X.; Luo, Z. Design and Implementation of Spiking Neural Network Based on Stochastic Computing. *IEEE Access* **2022**, *10*, 22113.
44. Ardakani, A.; Leduc-Primeau, F.; Onizawa, N.; Hanyu, T.; Gross, W.J. VLSI Implementation of Deep Neural Network Using Integral Stochastic Computing. *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.* **2017**, *25*, 2688–2699. [[CrossRef](#)]
45. Kim, K.; Kim, J.; Yu, J.; Seo, J.; Lee, J.; Choi, K. Dynamic energy-accuracy trade-off using stochastic computing in deep neural networks. In Proceedings of the 53rd ACM/EDAC/IEEE Design Automation Conference (DAC), Austin, TX, USA, 5–9 June 2016; pp. 1–6.
46. Liu, Y.; Liu, S.; Wang, Y.; Lombardi, F.; Han, J. A stochastic computational multi-layer perceptron with backward propagation. *IEEE Trans. Comput.* **2018**, *67*, 1273–1286. [[CrossRef](#)]
47. Nishi, Y.; Nomura, K.; Marukame, T.; Mizushima, K. Stochastic binary synapses having sigmoidal cumulative distribution functions for unsupervised learning with spike timing-dependent plasticity. *Sci. Rep.* **2021**, *11*, 18282. [[CrossRef](#)]
48. Nguyen, V.-T.; Luong, T.-K.; Duc, H.L.; Hoang, V.-P. An Efficient Hardware Implementation of Activation Functions Using Stochastic Computing for Deep Neural Networks. In Proceedings of the IEEE 12th International Symposium on Embedded Multicore/Many-core Systems-on-Chip (MCSoc), Hanoi, Vietnam, 12–14 September 2018; pp. 233–236.
49. Liu, Y.; Liu, L.; Lombardi, F.; Han, J. An Energy-Efficient and Noise-Tolerant Recurrent Neural Network Using Stochastic Computing. *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.* **2019**, *27*, 2213–2221. [[CrossRef](#)]
50. Sanni, K.; Garreau, G.; Molin, J.L.; Andreou, A.G. FPGA implementation of a Deep Belief Network architecture for character recognition using stochastic computation. In Proceedings of the 49th Annual Conference on Information Sciences and Systems (CISS), Baltimore, MD, USA, 18–20 March 2015; pp. 1–5.
51. Ji, Y.; Ran, F.; Ma, C.; Lilja, D.J. A hardware implementation of a radial basis function neural network using stochastic logic. In Proceedings of the Design, Automation & Test in Europe Conference & Exhibition (DATE), Grenoble, France, 9–13 March 2015; pp. 880–883.
52. Nguyen, D.-A.; Ho, H.-H.; Bui, D.-H.; Tran, X.-T. An Efficient Hardware Implementation of Artificial Neural Network based on Stochastic Computing. In Proceedings of the 5th NAFOSTED Conference on Information and Computer Science (NICS), Ho Chi Minh City, Vietnam, 23–24 November 2018; pp. 237–242.

**Disclaimer/Publisher’s Note:** The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.