Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2024.0429000

Energy-Efficient Izhikevich Neuron Design Using Approximate CORDIC-Based Multipliers for Low-Power Neuromorphic Hardware

VAN-VU LUYEN[1]*, THANH-DAT NGUYEN[1]*, QUANG-THAI PHAM[1], DUY-ANH NGUYEN[1], KHANH N. DANG[2], VAN-HAI PHAM[1,3] THANH-TOAN DAO[1]

[1]Department of Electronic Engineering, University of Transport and Communications, No.3 Cau Giay Street, Hanoi, Vietnam
[2]School of Computer Science and Engineering, University of Aizu, Aizuwakamatsu, Japan
[3]Hanoi Open University, B101, Nguyen Hien Street, Hanoi, Vietnam
*These authors contributed equally to this work

Corresponding author: Duy-Anh Nguyen (e-mail: anhnd@utc.edu.vn).

This work was supported in part by the UoA Competitive Research Fund P-21. This work is partially supported by the Minister of Education and Training (MOET), grant no. B2025-GHA-02.

ABSTRACT The Izhikevich neuron model is a widely used spiking neuron model that combines biologically plausible behavior with computational efficiency. As hardware implementations often suffer from high power and area usage due to multiplication operations, CORDIC-based multipliers have been a promising solution. However, as the energy efficiency of CORDIC-based multipliers is still limited, further improvement is needed. This paper proposes an approximate-adder-based CORDIC approach for implementing the Izhikevich neuron, replacing multipliers with shift-add operations and exact adders with approximate ones to reduce resource usage. Simulation and ASIC implementation results demonstrate significant improvements in energy efficiency with 14.56% increase in throughput and 10.23% area reduction, while compromising the neuron dynamics by only 0.07% in Mean Relative Error (MRE). In simulation, the membrane potential traces of the proposed model closely match those of the standard HOMIN model across all neuron behaviors, and the optimal configuration of 9 CORDIC iterations achieves the best trade-off between accuracy and efficiency.

INDEX TERMS Izhikevich neuron, CORDIC, hardware accelerator, approximate computing, low-power computing, neuromorphic engineering, spiking neural network.

NOMENCLATURE

a	Izhikevich parameter
α_i	Bit-shift scaling factor ($\alpha_i = 2^{-i}$)
b	Izhikevich parameter
c	Reset value of membrane potential (mV)
d	Recovery variable adjustment
E_λ	Cumulative error after λ iterations
ε_i	Adder-induced error in the i-th iteration
η	Adder type index
g	Sign of z_{scaled}
g_i	Control sign
γ	Signed adder deviation
h	Total number of spikes considered
I	Input current (normalized units)
i	Iteration index
k	Normalization factor ($k = 2^{N-1}$)
λ	Number of iterations
λ^*	Optimal iteration
$l_{\text{model},i}$	Spike times of tested variant
$l_{\text{HOMIN},i}$	Spike times of standard HOMIN neuron
m	Integer bits in fixed-point format
μ_γ	Mean of adder deviation γ
n	Fractional bits in fixed-point format
N	Number of bits for representation
$Q_{m.n}$	Fixed-point format (m integer, n fractional bits)
S_{accurate}	Exact adder output
S_{approx}	Approximate adder output
σ_γ	Standard deviation of adder deviation γ
t	Timestep
θ	Neuron type index

The source code and simulation scripts used in this work are publicly available at https://github.com/klab-aizu/IZH-ApproxCORD.

u	Membrane recovery variable
$u[t]$	Recovery variable at timestep t
v	Membrane potential (mV)
$v[t]$	Membrane potential at timestep t (mV)
x	Input value to CORDIC algorithm
y	Accumulator in CORDIC algorithm
z_{scaled}	Scaled intermediate value in CORDIC

I. INTRODUCTION

Spiking Neural Networks (SNNs) have emerged as the third generation of neural networks [1]–[3], offering a biologically plausible and energy-efficient approach to neural computation [4], [5]. Unlike traditional artificial neural networks that rely on continuous-valued activations, SNNs communicate through discrete spikes over time, closely mimicking the behavior of biological neurons in the brain. This event-driven mechanism not only enhances computational sparsity [6], [7] but also enables real-time processing with ultra-low power consumption, making SNNs particularly attractive for edge AI and neuromorphic applications [8]–[10]. The increasing demand for intelligent, low-power solutions in edge computing and IoT has fueled research into SNNs, driving advancements in both algorithmic models and neuromorphic hardware platforms [11]–[16].

At the core of SNNs lies the spiking neuron model, which governs how input stimuli are integrated and transformed into output spikes. Among the many models proposed—such as Integrate-and-Fire (IF) [2], [17] and Hodgkin-Huxley (H-H) [18]—the Izhikevich model is especially notable for its ability to reproduce a wide variety of biologically realistic firing patterns, including tonic spiking, bursting, and adaptation [19], [20], using a compact and computationally efficient formulation. While the IF model is computationally lightweight but biologically simplistic, and the H-H model is highly accurate but computationally demanding, the Izhikevich model strikes a balance between biological realism and efficiency [21], making it highly suitable for real-time hardware implementation in large-scale neuromorphic systems [8], [22], [23].

The Izhikevich neuron model is widely used in hardware-oriented SNN designs as it achieves a practical trade-off between biological accuracy and computational cost [8], [11]. However, one of its main hardware bottlenecks arises from the quadratic term $0.04v^2$ (where v represents the membrane potential) in the update equation [24].

This term requires a multiplication operation, which is costly in conventional digital hardware in terms of area, power, and delay. This challenge is particularly acute in resource-constrained edge devices, where neuromorphic architectures like TrueNorth have demonstrated the benefits of replacing multiplications with simpler operations [25]. Consequently, designing low-power and hardware-efficient alternatives to conventional multipliers is critical to enabling real-time and scalable SNN deployment [26]–[28].

To overcome the hardware cost associated with multipliers in the Izhikevich neuron model, several hardware-efficient computation techniques have been explored. Among these, the Coordinate Rotation Digital Computer (CORDIC) algorithm [29], [30] presents a promising alternative, as it eliminates the need for dedicated multipliers by utilizing only iterative shift and add operations [26]. This property makes CORDIC particularly attractive for neuromorphic edge computing and IoT applications, where minimizing energy and area overhead is paramount for deployment [25], [31].

One key advantage of the CORDIC approach lies in its ability to simplify control logic while maintaining computational flexibility. Specifically, the linear mode of the CORDIC algorithm can approximate square and multiplication functions, allowing it to directly compute the quadratic term $0.04v^2$ present in the Izhikevich neuron dynamics [27]. Compared to traditional multiplier-based designs, CORDIC-based architectures have been demonstrated to achieve higher operating frequencies and lower resource utilization [32], [33], making them a scalable solution for energy-efficient spiking neural network implementations [11], [24].

Biological neural systems inherently tolerate a certain degree of imprecision, which has motivated the adoption of approximate computing in neuromorphic circuits [34], [35]. Recent studies have applied approximate adders within the CORDIC datapath to reduce switching activity and hardware cost [36]. For instance, in our earlier work on approximate computing, integrating 1-error or 2-error approximate adders into a CORDIC-based multiplier resulted in notable hardware benefits, including up to 19.3% power reduction, 11.5% area savings, and 14.7% frequency improvement compared to a conventional exact CORDIC multiplier [37]. These results demonstrated that carefully selected approximation techniques can effectively improve energy and area efficiency in arithmetic units without compromising application-level functionality. This highlights the potential of approximate computing as a viable strategy for achieving meaningful energy–accuracy tradeoffs in hardware-efficient neuromorphic designs.

In this work, we propose a novel hardware architecture for the Izhikevich neuron using a CORDIC-based multiplier enhanced with approximate adders. Building upon our previous work on low-power CORDIC design, this architecture leverages both the shift-add efficiency of linear-mode CORDIC [27] and the area/power advantages of approximate arithmetic [34]. To further optimize resource utilization on FPGA platforms, we integrate 1-error and 2-error approximate adders within the CORDIC datapath, which reduce switching activity without significantly affecting neuron behavior.

Additionally, motivated by the HOMIN model [38] and hardware-oriented reformulations of Izhikevich dynamics [24], we decompose the update equations into modular blocks amenable to pipelining and parallelism. Insights from prior approximate CORDIC work [37] in image processing applications guided our design methodology for integrating approximate arithmetic into the neuromorphic domain. This makes our architecture highly suitable for large-scale SNN deployments in edge-AI and neuromorphic systems.

Our contributions are summarized as follows:

- Design and integrate a CORDIC-based multiplier tailored for the Izhikevich model, replacing the traditional multiplier unit.
- Further enhancement of the multiplier by incorporating an approximate adder to reduce hardware cost and power consumption.

The remainder of this paper is organized as follows: Section II reviews related works. Section III introduces the Izhikevich neuron model, its hardware-oriented simplification, and the CORDIC algorithm for efficient multiplier-less computation. Section IV presents the proposed design. Section V reports the evaluation results, and Section VI provides further discussion. Finally, Section VII concludes the paper.

II. RELATED WORKS

Hardware implementations of spiking neural networks (SNNs) have been investigated across multiple directions to balance computational accuracy, scalability, and resource efficiency. In particular, prior research has focused on three relevant areas: hardware-oriented realizations of the Izhikevich neuron, CORDIC-based computation for multiplier-less arithmetic, and approximate arithmetic techniques for power- and area-constrained systems.

Early hardware realizations of the Izhikevich neuron model adopted fixed-point arithmetic instead of floating-point to simplify computations and enable real-time execution on FPGAs [16], [39]–[41], [41], [42]. These designs leveraged existing DSP blocks, achieving high performance but at the cost of increased power consumption and hardware area, especially when scaling to many neurons [27], [43].

Subsequent FPGA/ASIC architectures focused on parallelism, pipelining, and time-multiplexing to maximize throughput for large-scale spiking neural networks [44], [45]. Despite throughput gains, these implementations remained dependent on accurate multipliers, which impose significant hardware constraints and limit network scalability [27].

In many FPGA and ASIC implementations of spiking neuron models, accurate multipliers are used to evaluate nonlinear terms in the membrane potential update equations. While such designs ensure high numerical precision, they incur substantial hardware costs, including high DSP block utilization, increased power consumption, and longer critical paths [25]–[27]. These drawbacks become more pronounced when scaling to large neuron populations, where the number of required multiplications grows rapidly [23], [40]. This dependency on costly multipliers has motivated the search for hardware-friendly alternatives such as CORDIC algorithms and piecewise-linear approximations which replace multiplications with iterative shift-and-add operations, aiming to maintain neuron dynamic fidelity while reducing hardware overhead [46], [47].

CORDIC is widely employed in hardware to evaluate trigonometric, hyperbolic, exponential, logarithmic, square-root, and other transcendental functions using only iterative shift-and-add operations [29], [30], [48]. Its multiplier-less nature makes it particularly attractive for resource-limited platforms such as FPGA and ASIC, where reducing DSP block usage is critical [31], [49]. In addition to transcendental functions, CORDIC's linear mode supports vector magnitude, square, and multiplication approximations, enabling its integration into a wide range of digital signal processing, communication, and neuromorphic computing applications [27], [50]. It has also been applied in neural network accelerators to approximate activation functions such as sigmoid and tanh with compact, low-complexity circuits [51]. However, the co-integration of CORDIC structures and approximate arithmetic has received limited attention, leaving potential gains in throughput and resource efficiency underexploited [52]. Moreover, applications within spiking neuron models are sparse; in particular, to the best of our knowledge, multiplier-less Izhikevich implementations leveraging such co-design have not been systematically investigated.

Approximate adders and multipliers have been extensively studied to reduce hardware cost and energy consumption in error-resilient systems [53], [54]. By exploiting the inherent tolerance to noise and computation errors, these designs achieve an effective balance between energy efficiency and computational accuracy [55]. Such trade-offs are particularly advantageous in neuromorphic and edge-computing systems, where stringent power and area budgets must be met [56], [57]. In neural accelerators, approximate arithmetic has been applied to both ANN and SNN implementations, significantly lowering resource usage and latency on constrained hardware [58], [59]. These approaches have also demonstrated competitive performance in ultra-low-power embedded systems for real-time applications [60]. Nevertheless, approximate computing techniques have been largely confined to artificial neural networks or signal processing, with limited exploration in biologically inspired spiking neuron models that demand precise dynamic behavior [58], [61].

Critically, there is no known hardware-efficient implementation of the Izhikevich neuron that combines CORDIC-based operations with approximate arithmetic—highlighting a clear gap and opportunity for innovation.

III. BACKGROUND

A. IZHIKEVICH NEURON

The Izhikevich neuron model is a computationally efficient spiking neuron model that combines biological realism with low computational complexity, making it particularly suitable for large-scale simulations of neural networks. Proposed by Eugene M. Izhikevich in 2003 [20], it reproduces a wide variety of spiking and bursting behaviors observed in biological neurons using only two coupled differential equations:

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

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

where v represents the membrane potential, u is a membrane recovery variable accounting for the activation of K^+

ionic currents and inactivation of Na^+ ionic currents, and I is the input synaptic or injected current. The parameters a, b, c, and d determine the dynamics of the neuron and can be tuned to replicate different neuronal firing patterns. When the membrane potential v reaches a threshold (typically 30 mV), the model enforces the reset conditions:

$$\text{if } v \geq 3\,\text{mV}, \quad \text{then} \quad \begin{cases} v & \leftarrow -6.5, \\ u & \leftarrow u + d, \end{cases} \tag{3}$$

where c is the reset value of the membrane potential and d adjusts the recovery variable. Thanks to its simple yet powerful formulation, the Izhikevich model can efficiently replicate complex neuronal behaviors such as regular spiking, fast spiking, thalamo-cortical bursting, and resonator-like firing.

B. MODIFIED IZHIKEVICH NEURON

To reduce complexity, HOMIN [38] simplifies and scales the equations. The membrane potential range is scaled down by a factor of 10, and constants are replaced by powers of 2 to enable simple bit-shift operations. The typical values of the parameters are a = 0.02, b = 0.2, c = -65mV, and d = 8.0. By scaling, we got the following equations:

$$\frac{dv}{dt} = \frac{1}{4}v^2 + 2v + 15 - u + I \tag{4}$$

$$\frac{du}{dt} = 2^{-6}(2^{-2}v - u), \tag{5}$$

$$\text{if } v \geq 3\,\text{mV}, \quad \text{then} \quad \begin{cases} v & \leftarrow c, \\ u & \leftarrow u + d, \end{cases} \tag{6}$$

By modifying the equation of Izhikevich neuron, computing u and v become less complicated. As most operations are converted to shift-bit and addition (or subtraction), only the v^2 is still challenging. To handle this, this work targets using CORDIC multiplier to simplify the calculation.

C. CHALLENGES IN HARDWARE IMPLEMENTATION

Although the HOMIN neuron reduces most computations to shifts and additions, the quadratic term v^2 remains a hardware "bottleneck". Conventional multipliers incur high area, power, and latency costs, which become prohibitive in large neuromorphic systems [27].

A more efficient alternative is to use CORDIC, which approximates multiplication with iterative shift–add operations, avoiding full multipliers [52]. Integrating a CORDIC-based squaring unit into HOMIN enables efficient and scalable hardware while preserving biological accuracy.

To further reduce cost, approximate arithmetic offers favorable trade-offs between efficiency and precision [53]–[55]. Accordingly, we enhance the CORDIC squaring with approximate adders [62] and dynamic early termination, improving energy efficiency without compromising neuronal fidelity.

IV. ENERGY-EFFICIENT IZHIKEVICH NEURON DESIGN WITH APPROXIMATE CORDIC-BASED MULTIPLIERS

In this work, we propose a hardware-efficient implementation of the HOMIN neuron by replacing the quadratic term v^2 with a CORDIC-based squaring unit. This approach eliminates the need for costly multipliers while preserving the essential dynamics of the neuron model. Building on this foundation, we introduce the integration of approximate adders Section IV-C to further reduce hardware cost and energy consumption. Together, these techniques yield a scalable and energy-efficient neuron architecture that maintains biological fidelity while significantly improving hardware feasibility for large-scale neuromorphic systems.

A. FINAL CORDIC IZHIKEVICH IMPLEMENTATION

$$\frac{dv}{dt} = \frac{1}{4}CORDICSQ(v[t]) + 2v[t] + 15 - u[t] + I \tag{7}$$

$$\frac{du}{dt} = 2^{-6}(2^{-2}v[t] - u[t]), \tag{8}$$

Where *CORDICSQ* is the square function by CORDIC in the previous section.

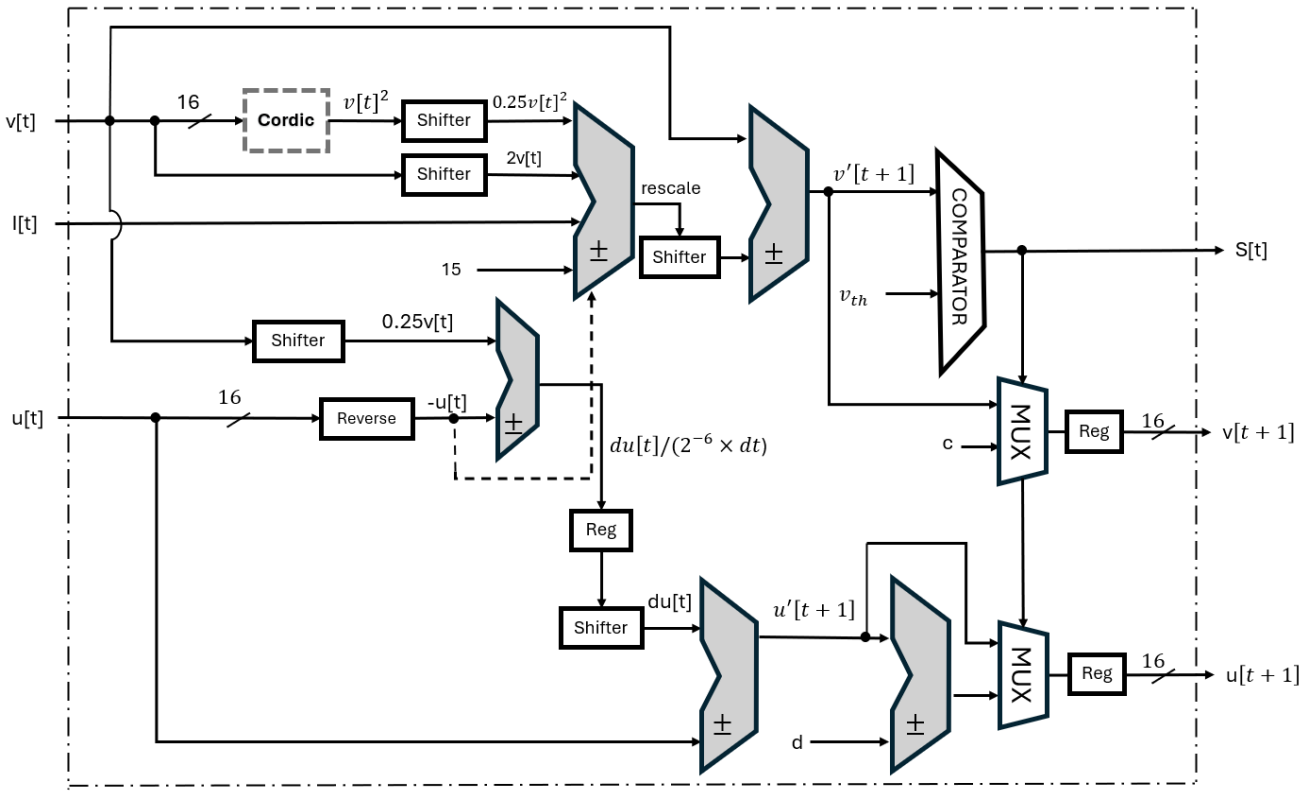
$$\text{if } v[t] \geq 3\,\text{mV}, \quad \text{then} \quad \begin{cases} v[t] \leftarrow c, \\ u[t] \leftarrow u[t] + d, \end{cases} \tag{9}$$

Fig. 1 presents the hardware architecture of the proposed HOMIN neuron. Besides the CORDIC-based squaring unit, the core circuit consists of multiple shifters and adders to perform multiplications by constant values, such as $\frac{1}{4} \times v[t]^2$ and $2 \times v[t]$ Multiplications by constant values can be efficiently implemented using shifters and adders instead of multipliers, significantly reducing hardware cost and power consumption with only minor accuracy degradation.

Considering the numerical range of the membrane potential v and recovery variable u, a fixed-point format with $Qm.n$ bits (m integer bits and n fractional bits) is adopted. This provides a total bit width of $(m + n)$ bits, allowing sufficient dynamic range while minimizing quantization error.

The selection of m and n is driven by two key factors. First, the physiological range of the membrane potential in the HOMIN model is approximately $v[t] \in [-8, 3]$, requiring at least 4 integer bits plus 1 sign bit to avoid overflow ($2^4 = 16 > 8$). Second, regarding the fractional precision, the HOMIN model must accurately reproduce complex spiking dynamics such as spike latency, resonator, rebound spike, and rebound burst. To maintain biological fidelity across all these behaviors while minimizing quantization error. Based on these premises, we adopt Q6.9 format (1 sign bit + 6 integer bits + 9 fractional bits, totaling 16 bits), which provides sufficient dynamic range (± 64) and precision ($2^{-9} \approx 0.002$) to accurately reproduce all neuron behaviors.

In this study, a $Qm.n$-bit fixed-point quantization is used for the membrane potential $v[t]$ and recovery variable $u[t]$, i.e., 1 bit for the sign, m bits for the integer part, and n bits for the fractional part. In the hardware implementation, the



The

FIGURE 1: Proposed Izhikevich neuron architecture.

CORDIC-based squaring unit combined with the approximate adder only supports 8-bit input precision. Therefore, the membrane potential $v[t]$, originally represented in Q6.9 fixed-point format (16 bits), must be down-scaled to Q4.4 (8 bits) before multiplication. This conversion removes 2 integer bits and 5 fractional bits, thus reducing both the representable range and precision of $v[t]$. The squaring operation $v[t] \times v[t]$ is then carried out in the Q4.4 domain using the CORDIC-based squaring unit with approximate adder, and the result is converted back to Q6.9 for subsequent computations in the HOMIN update equations. Importantly, the physiological range of $v[t]$(approximately $[-8, 3]$) remains fully representable within Q4.4, ensuring that the scaling process does not compromise the biological fidelity of the neuron model.

B. CORDIC MULTIPLIER ARCHITECTURE

Our CORDIC multiplier architecture is shown in Fig. 2 and Algorithm 1. We use a basic sequential architecture to update y and z_{scaled} through multiple iterations, where the updated values are determined by the current iteration number and the sign of z_{scaled}.

To help illustrate the algorithm for CORDIC-based squaring, consider a simple example where the input is $x = 5$. We want to compute $x^2 = 25$ using $\lambda = 3$ iterations.

Initialization:

- $k = 4$ (normalization factor)
- $z_{\text{scaled}} = 5/4 = 1.25$
- $y = 0$ (accumulator)

Iteration 1 ($i = 0$):

- $g = \text{sign}(1.25) = +1$
- $y = 0 + 1 \cdot (5 \cdot 2^0) = 0 + 5 = 5$
- $z_{\text{scaled}} = 1.25 - 1 \cdot 2^0 = 1.25 - 1 = 0.25$

Iteration 2 ($i = 1$):

- $g = \text{sign}(0.25) = +1$
- $y = 5 + 1 \cdot (5 \cdot 2^{-1}) = 5 + 2.5 = 7.5$
- $z_{\text{scaled}} = 0.25 - 1 \cdot 2^{-1} = 0.25 - 0.5 = -0.25$

Iteration 3 ($i = 2$):

Algorithm 1: CORDIC-based Squaring of $x \cdot x$

Require: Input value x; Number of iterations λ; Number of bits for representation N

Ensure: Approximate square $x \cdot x$

1: Calculate the normalization factor $k \leftarrow 2^{N-1}$
2: Initialize $z_{\text{scaled}} \leftarrow x/2^k$
3: Initialize $y \leftarrow 0$
4: for $i \leftarrow 0$ to $\lambda - 1$ do
5: $\quad g \leftarrow \text{sign}(z_{\text{scaled}})$
6: $\quad y \leftarrow y + g \cdot (x \cdot 2^{-i})$
7: $\quad z_{\text{scaled}} \leftarrow z_{\text{scaled}} - g \cdot 2^{-i}$
8: end for
9: Rescale: $x^2 \leftarrow y \cdot 2^k$
10: return $x^2 \leftarrow y \cdot 2^k$

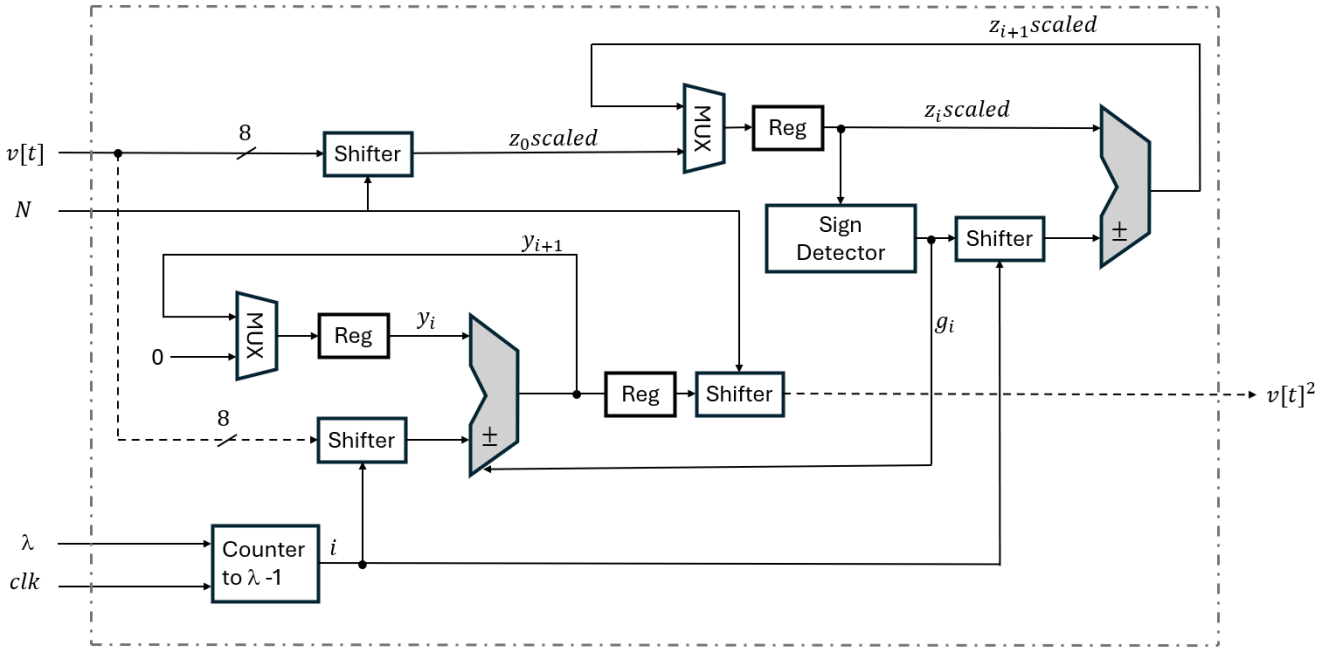


FIGURE 2: CORDIC Multiplier Architecture.

- $g = \text{sign}(-0.25) = -1$
- $y = 7.5 + (-1) \cdot (5 \cdot 2^{-2}) = 7.5 - 1.25 = 6.25$
- $z_{\text{scaled}} = -0.25 - (-1) \cdot 2^{-2} = -0.25 + 0.25 = 0$

Final Result:

- Rescale: $x^2 = 6.25 \times 4 = 25$
- Expected result: $5^2 = 25$

This example demonstrates how the algorithm approximates multiplication using only shift and add operations.

Fig. 3 illustrates the *Fully Unrolled CORDIC*-based squaring unit. All iterative steps of the CORDIC algorithm are unrolled into N parallel hardware stages, where N corresponds to the number of iterations. The squaring operation $v[t]^2$ is thus implemented in a pipelined structure. Each stage contains three functional blocks: a Shifter i for multiplying or dividing by 2^{-i}, a Sign Detector to determine the convergence direction and select addition or subtraction, and an Adder/Subtractor that performs the corresponding arithmetic operation. These N stages progressively refine intermediate values until the final squared result is obtained. The number of stages is chosen to achieve the desired precision, and the main advantage of this architecture is its minimal latency, as computation completes once the pipeline is filled.

C. APPROXIMATE ADDER INTEGRATION

In this paper, we proposed using an approximate adder to calculate the update of y in each iteration. Algorithm 2 shows the proposed multiplier.

Since the signed adders for the updates of y are restricted to 16-bit adders only, we restricted the multiplier to use an 8-bit input. In this study, we used a set of 16-bit signed approximate adders from EvoApproxLibLITE [62], focusing on key metrics such as mean absolute error (MAE), error probability (EP), and hardware cost. We evaluated all the 16-bit signed adders from the library.

Algorithm 2: CORDIC-based Squaring of $x \cdot x$ with Approximate Adder.

Require: Input value x; Number of iterations λ; Number of bits for representation N
Ensure: Approximate square $x \cdot x$
1: Calculate the normalization factor $k \leftarrow 2^{N-1}$
2: Initialize $z_{\text{scaled}} \leftarrow x/2^k$
3: Initialize $y \leftarrow 0$
4: for $i = 0$ to $\lambda - 1$ do
5: $\quad g \leftarrow \text{sign}(z_{\text{scaled}})$
6: $\quad y \leftarrow \text{ApproximateAdd}(y, g \cdot (x \cdot 2^{-i}))$
7: $\quad z_{\text{scaled}} \leftarrow z_{\text{scaled}} - g \cdot 2^{-i}$ {with exact adder}
8: end for
9: Rescale: $x^2 \leftarrow y \cdot 2^k$
10: return x^2

D. OPTIMAL ITERATION SELECTION BASED ON MRE ANALYSIS

To determine the optimal number of CORDIC iterations that achieves the highest accuracy, the MRE is used as the evaluation metric. The MRE quantitatively measures the deviation in spike timing between the tested neuron model and the standard HOMIN model, and is defined as:

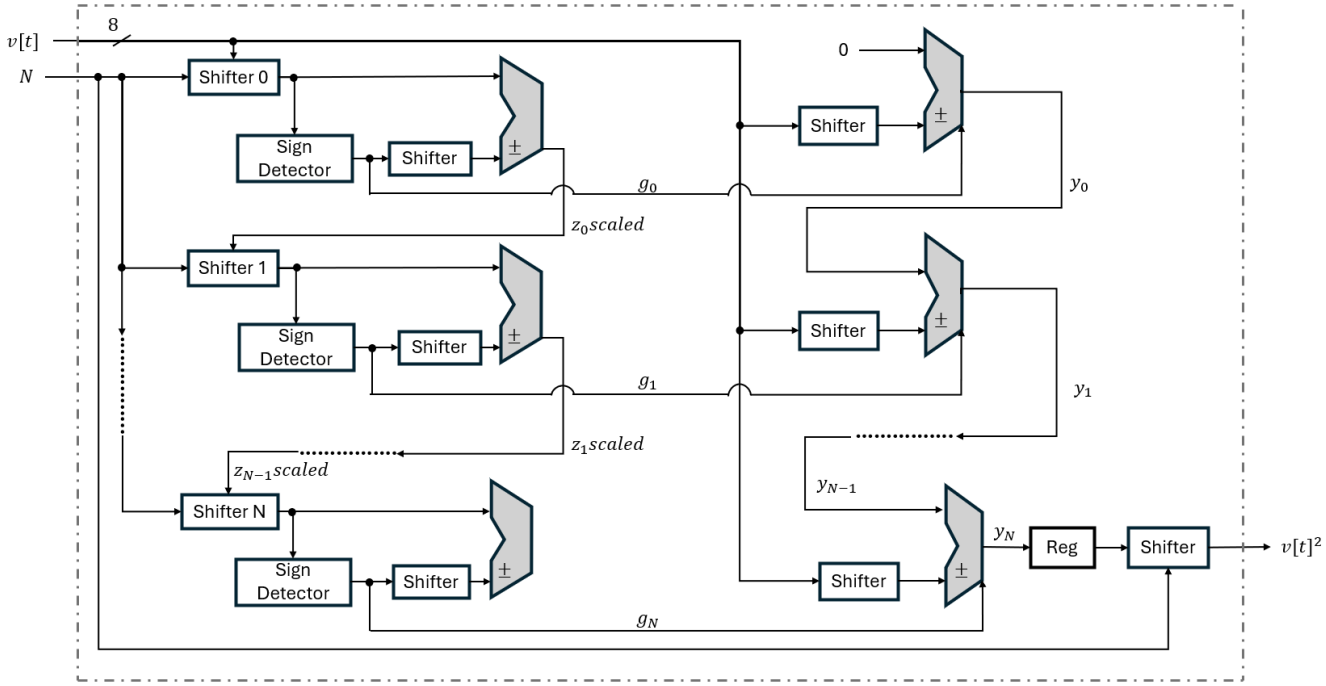


FIGURE 3: Modified Fully Unrolled CORDIC Multiplier Architecture.

$$\text{MRE}(\%) = \frac{1}{h}\sum_{i=1}^{h}\frac{l_{\text{model},i} - l_{\text{HOMIN},i}}{l_{\text{HOMIN},i}} \times 100\%, \qquad (10)$$

where $l_{\text{model},i}$ are the spike times of the tested variant (CORDIC-based or CORDIC with approximate adder HOMIN model), $l_{\text{HOMIN},i}$ are the spike times of the standard HOMIN neuron, and h is the total number of spikes considered.

The MRE values are collected across all neuron models (Regular Spiking, Initial Bursting, Chattering, and Low-Threshold Spiking) and approximate adder configurations. For each CORDIC iteration count, the average MRE is computed, and the iteration yielding the lowest mean error is selected as the optimal configuration.

The overall procedure for determining the optimal iteration based on MRE analysis is summarized in Algorithm 3.

Algorithm 3: Finding the Optimal CORDIC Iteration Based on Average MRE

Require: MRE values $\text{MRE}_{\theta,\eta,\lambda}$ for all neurons θ, adders η, and number of iterations λ
Ensure: Optimal iteration λ^*
1: for each iteration λ do
2: Compute average error: $\text{avgMRE}[\lambda] = \text{mean}(\text{MRE}_{\theta,\eta,\lambda})$
3: end for
4: $\lambda^* = \arg\min_\lambda \text{avgMRE}[\lambda]$
5: return λ^*

Algorithm 3 summarizes the process used to determine the optimal number of CORDIC iterations. For each iteration count λ, the MRE values obtained from all neuron types and approximate adder configurations are aggregated, and the average error $\text{avgMRE}[\lambda]$ is computed. The iteration index that yields the minimum average error, denoted as λ^*, is then selected as the optimal iteration count. This approach ensures that the chosen configuration provides consistently high accuracy across different neuron dynamics and hardware approximation schemes.

E. ERROR ANALYSIS

To quantify the impact of approximate adders within the HOMIN model, we define the signed adder deviation, denoted by γ, which represents the difference between the output of an approximate 16-bit adder and that of an exact adder for the same input operands:

$$\gamma = S_{\text{approx}} - S_{\text{accurate}} \qquad (11)$$

where S_{approx} and S_{accurate} denote the approximate and exact addition outputs, respectively.

Rather than being a fixed constant, the deviation γ is modeled as a random variable characterized by its mean μ_γ and standard deviation σ_γ. These parameters capture the bias and variability of the approximate adder's error distribution. In principle, they can be obtained by exhaustively evaluating all $2^{16} \times 2^{16}$ input operand pairs and computing the resulting deviations.

Although individual deviations may be small, iterative algorithms such as CORDIC can accumulate these deviations

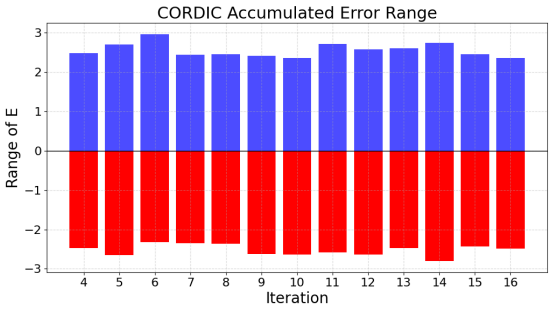


FIGURE 4: Range of accumulated CORDIC error E_λ over iterations $\lambda = 4$–16 using the add16se_2TN approximate adder.

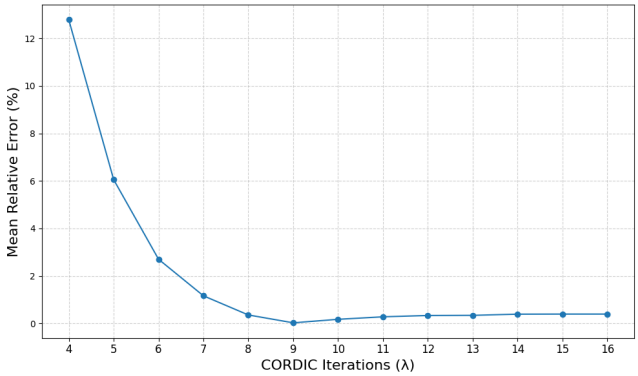


FIGURE 5: MRE (%) versus CORDIC iteration count ($\lambda = 4$–16) for the HOMIN Regular Spiking neuron using the add16se_2TN approximate adder at $I = 15$.

across multiple cycles. Therefore, statistically characterizing γ is essential for understanding potential aggregate error.

During the CORDIC iterations used to compute v^2, each iteration i introduces a per-iteration error, ε_i, representing the effective error contributed by the approximate adder in that iteration. The error direction is determined by $g_i \in \{\pm 1\}$.

$$\varepsilon_i = g_i \cdot \gamma \tag{12}$$

After bit-shift scaling by α_i (with $\alpha_i = 2^{-i}$), the cumulative error after λ iterations is:

$$E_\lambda = \sum_{i=0}^{\lambda-1} \alpha_i \, \varepsilon_i = \sum_{i=0}^{\lambda-1} 2^{-i} \, \varepsilon_i \tag{13}$$

Assuming that the deviation signs follow a symmetric random distribution, i.e., $P(g_i = \pm 1) = 0.5$, the expected cumulative error is zero, and the variance can be computed accordingly.

To visualize how the approximate-adder deviations propagate through the CORDIC iterations, we performed a Monte-Carlo simulation. In this simulation, errors ε_i were generated based on the measured $(\mu_\gamma \approx 0.25, \sigma_\gamma \approx 0.433)$ of the approximate adder add16se_2TN. For each iteration count $\lambda = 4$ to 16, the cumulative error E_λ was computed separately. The resulting plot shows the range of E_λ for each λ, illustrating how the cumulative error varies across different numbers of CORDIC iterations.

The simulation results in Fig. 4 indicate that the cumulative error E_λ of the CORDIC using the add16se_2TN adder remains nearly constant across the iteration count $\lambda = 4$ to 16. The per-iteration errors, generated independently, contribute only small variations to the total cumulative error due to the scaling factor 2^{-i}, resulting in an almost uniform error range without any noticeable upward or downward trend. Therefore, increasing the number of iterations does not significantly change the amplitude of the cumulative error, and the observed error range remains effectively the same across different iteration counts.

While E_λ characterizes arithmetic-level deviations in CORDIC, the system-level impact of errors introduced by CORDIC iterations and approximate adders is evaluated through the spike-time MRE of the HOMIN model. To this end, we perform simulations with CORDIC iteration counts ranging from $\lambda = 4$ to 16 and evaluate the resulting errors using the spike-time MRE of the HOMIN Regular Spiking neuron, enabling a direct assessment of how arithmetic approximations affect the spiking behavior of the Izhikevich-based neuron model.

Fig. 5 shows the spike-time MRE of the HOMIN model as a function of the CORDIC iteration count λ. As λ increases from 4 to 16, the spike-time MRE consistently decreases and gradually converges to a small value. This trend indicates that increasing the number of CORDIC iterations improves the temporal accuracy of spike generation. The observed convergence further suggests that the nonlinear dynamics of the HOMIN neuron effectively mitigate the impact of errors introduced by CORDIC computation and approximate adders, resulting in stable spiking behavior at the system level.

V. EVALUATION

A. EVALUATION METHODOLOGY

Software simulation and validation were conducted in Python, employing the EvoApproxLibLITE [62] library to implement approximate adders within the CORDIC multiplication process. The CORDIC-based multiplication was then integrated into the HOMIN neuron model and simulated across different neuron behaviors to evaluate accuracy and spiking dynamics. The validation encompassed three main aspects: time-domain simulations of the membrane potential for the standard HOMIN, HOMIN with CORDIC, and HOMIN with CORDIC combined with the approximate adder; evaluation of the MRE across CORDIC iterations (4–16) for multiple approximate adder configurations relative to the reference model; and large-scale raster plot analysis of 1000 randomly connected excitatory neurons to assess network-level behavior consistency. We note that the specific adders tested were chosen for availability in EvoApproxLibLITE [62]. The key contribution of our work is a systematic, library- and

hardware-independent approach to select the approximate adder and CORDIC iterations based on accuracy-efficiency trade-offs.

The Izhikevich neuron is also written in SystemVerilog. We instantiate the combination of EvoApproxLibLITE and a CORDIC-based multiplier within the HOMIN neuron model, exploring a wide range of adder options (2TN, 3BD, and others) and several CORDIC variants, including iterative designs with 9 and 16 iterations and fully unrolled realizations with matching iteration counts. ModelSim is employed for functional simulation, Synopsys Design Compiler for RTL-to-gate synthesis, and Synopsys PrimeTime for estimating output power and timing characteristics. The design is implemented using 45nm CMOS NANGATE.

B. SOFTWARE EVALUATION

In this section, we present a comparative evaluation of the standard HOMIN model, the CORDIC-based HOMIN model, and the CORDIC with approximate adder HOMIN model. The analysis is carried out at both the single-neuron level and the network level, focusing on membrane potential traces, spike timing accuracy measured by the MRE, and population dynamics captured by raster plots.

1) Membrane potential traces

The comparison begins at the level of the membrane potential traces, which reflect the dynamic evolution of the neuron's internal voltage in response to external input. This provides a direct way to assess how closely the CORDIC-based implementations reproduce the spiking behavior of the standard HOMIN model.

Fig. 6 presents four representative neuron firing patterns — Regular Spiking (d=8.0), Initial Bursting (d=5.0), Chattering (d=1.125), and Low-Threshold Spiking (d=0.375) — each showing both the membrane potential (zoomed around spike events) and the corresponding spike output over the same time interval. This presentation allows direct comparison of both membrane potential shapes and spike timing across the three HOMIN variants (standard, CORDIC, and CORDIC with approximate adder add16se_2TN) under identical conditions. Notably, the differences between the variants are relatively small, indicating that the CORDIC approximations do not significantly alter the neuron firing patterns. However, to quantitatively assess the differences, particularly in spike timing, a metric such as the Mean Relative Error (MRE) can be used to provide a concrete measure of deviation between the models.

2) Spike Timing Accuracy

To quantitatively evaluate the accuracy of spike timing across different configurations, the MRE metric was employed. The MRE was computed for CORDIC iteration counts ranging from 4 to 16 and evaluated across four distinct neuron behaviors: Regular Spiking (RS, $d = 8.0$), Initial Bursting (IB, $d = 5.0$), Chattering (CH, $d = 1.125$), and Low-Threshold Spiking (LTS, $d = 0.375$).

As shown in Fig. 7, the HOMIN-CORDIC and HOMIN-CORDIC + approximate adder models produce almost identical MRE values across all neuron behaviors, demonstrating that the spike timing characteristics remain highly consistent even when approximate adders are introduced. Furthermore, the MRE steadily decreases with increasing CORDIC iteration counts for all neuron types, indicating improved numerical precision with deeper iterations.

Quantitatively, the results show close agreement across adders. For Regular Spiking, the average MRE is 1.96% with a maximum of 12.76%. For Initial Bursting, the average is 1.25% and the maximum is 6.42%. For Chattering, the average MRE is 1.83% with a maximum of 12.77%, while for Low-Threshold Spiking, the average is 0.84% and the maximum is 2.60%.

These results show that approximate adders generally preserve model accuracy, though their error remains dependent on the selected iteration count. The overall MRE levels remain nearly identical to those of the exact CORDIC model, and the general trend of error reduction with increasing iterations is preserved. However, it is also observed that at higher iteration counts, the MRE may slightly increase due to *iterative error accumulation*, i.e., the propagation and build-up of approximation errors through successive CORDIC stages.

In Fig. 8, raster plots of network simulations with 1000 randomly connected excitatory neurons are shown for three configurations: the standard Python-based HOMIN model without CORDIC, the Python-HOMIN model with CORDIC and the add16se_2TN approximate adder, and the Verilog-HOMIN implementation with CORDIC and the add16se_2TN approximate adder. All configurations were initialized using identical synaptic weights and connectivity matrices to ensure a fair comparison. Overall, the raster plots reveal highly consistent spiking patterns across the three cases, characterized by regular, synchronous firing events spanning the neuronal population. The incorporation of CORDIC and approximate arithmetic leads only to minor variations in spike timing, while preserving the global firing structure and population dynamics. These results indicate that the proposed hardware-oriented approximations do not significantly alter the qualitative behavior of the network.

3) Optimal Number of Iterations

To identify the most optimal number of CORDIC iterations, the MRE values were aggregated across all neuron types (RS, IB, CH, and LTS) and all approximate adder configurations. Using Algorithm 3, the average MRE for each iteration count, ranging from 4 to 16, was computed, and the iteration yielding the lowest overall error was selected as the optimal configuration.

The analysis indicates that, when applying the mean-based selection criterion defined in Algorithm 3, the procedure yields an iteration count of 9, where the average MRE reaches its minimum value of 0.0705%. Although this result shows that nine iterations perform well under the evaluated conditions, it should not be interpreted as a universally opti-

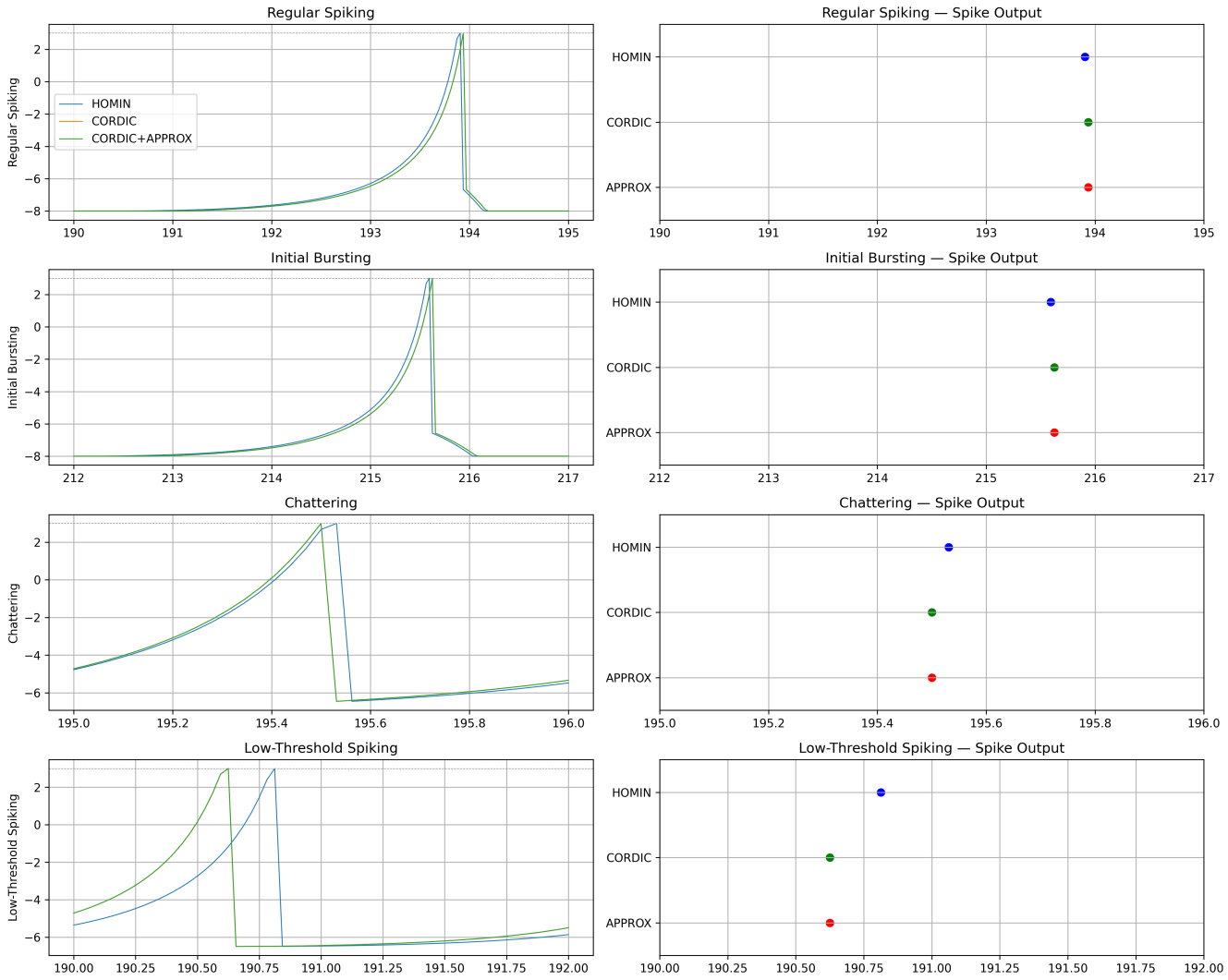


FIGURE 6: Python simulations of the membrane potential as a function of time for four neuron types: Regular Spiking, Initial Bursting, Chattering, and Low-Threshold Spiking. Spike outputs of each neuron are additionally shown. All models use a scaled input current of $I = 15$. The CORDIC+approximate model employs the add16se_2TN adder.

mal choice. The appropriate number of iterations may vary depending on several factors, such as the specific neuron model, the adder architecture, and the underlying approximation behavior. When these factors are fixed—for example, when targeting a particular neuron model with a given adder design—a distinct optimal iteration count can be determined.

C. HARDWARE EVALUATION

For the hardware evaluation, we examine 10 variants of the HOMIN model based on different adder types—2TN, 2U6, 2UB, 2UY, 2YM, 3BD, 32T, 36D, 349, and EXACT—as well as 4 multiplier variants: running the multiplier with CORDIC iterations of 9 and 16, and the fully-unrolled versions with CORDIC iterations of 9 and 16. The results, including throughput, energy per spike, and area, are illustrated in Fig. 9.

As illustrated in Fig. 9, 3BD-based models exhibit superior stability and balanced trade-offs across energy, area, and throughtput, making them the most efficient overall. Other adder variants also show certain improvements compared with the exact design, although these benefits are less pronounced than the clearer gains achieved by 3BD.. In contrast, 2TN and 2U6 show irregular fluctuations in energy and throughput, indicating limited scalability. Although fully unrolled designs consume more energy and area, they deliver clear throughput gains, confirming the benefit of full parallelization. Other variants—2U6, 2UB, 2UY, 2YM, 32T, 36D, and 349—maintain consistent results within the performance range defined by 2TN and 3BD. The energy increase remains reasonable in view of the throughput improvement of 14.56% and the area reduction of 10.23% measured against the exact implementation. Overall, intermediate complexity adders such as 3BD and 32T provide the best balance between ac-

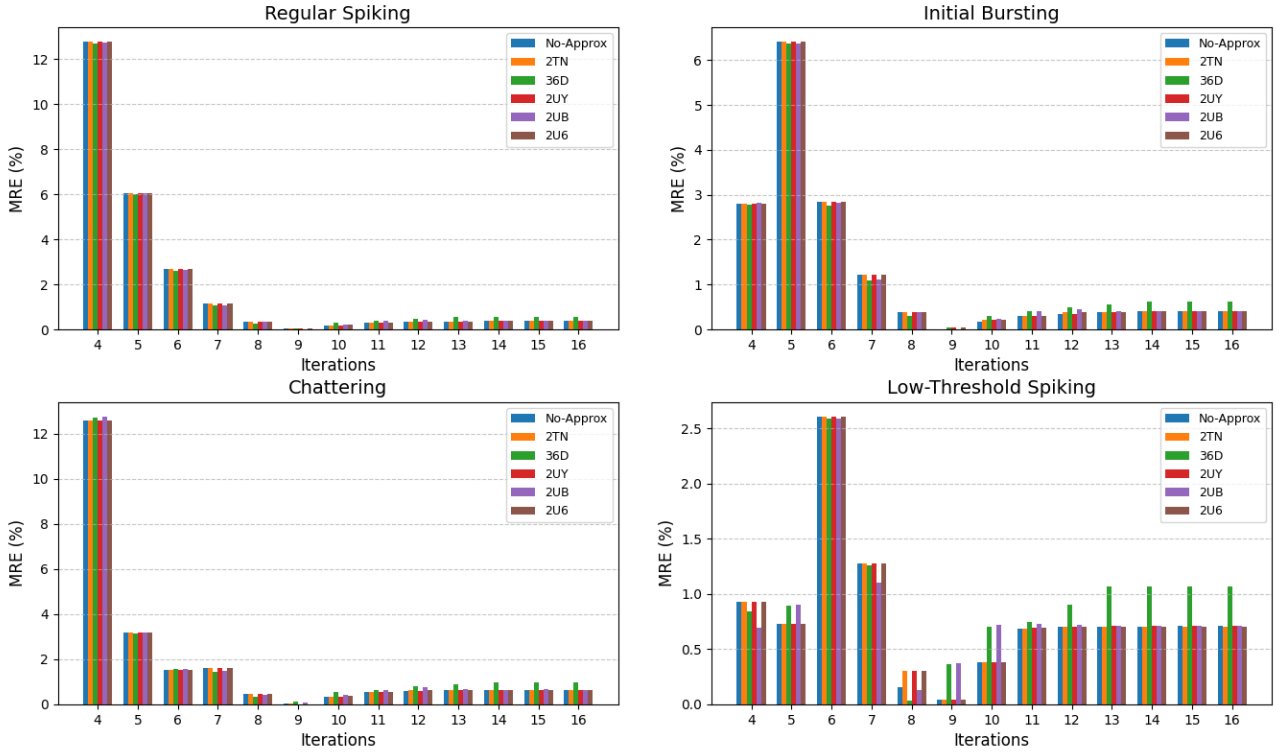


FIGURE 7: MRE% versus CORDIC iterations (4–16) for HOMIN with CORDIC and HOMIN with CORDIC + approximate adders (add16se_2TN, add16se_36D, add16se_2UY, add16se_2UB, and add16se_2U6), relative to the standard HOMIN model, at $I = 15$.

curacy, performance, and efficiency, while others offer stable but less optimal trade-offs.

D. COMPARISON

The v^2 term is implemented using a CORDIC-based multiplier with approximate adders. Since this block contributes a noticeable portion of the overall hardware cost, we further analyze its implementation in terms of area, power, and achievable frequency, and compare it against representative approximate squaring designs. Table 1 shows that the CORDIC-based multiplier variants incorporating approximate adders achieve up to about 11–12% area reduction and nearly 20% power reduction compared with the exact design, while maintaining low average error for most variants. Several variants (such as 2UB and 2YM) reach Fmax values of up to 562 MHz (corresponding to clock periods below 2 ns), indicating strong peak throughput capability. In this work, throughput is evaluated based on the reported Fmax and the iterative CORDIC structure, which supports configurable iteration counts from 9 to 16; to ensure a conservative and fair comparison, 16 iterations are assumed in Table 1. It is acknowledged that the throughput reported at 100 MHz represents a conservative operating point and may be lower than that of designs optimized for higher operating frequencies. However, the availability of high Fmax provides significant performance headroom, allowing throughput to be increased when required by operating closer to the timing limit, at the expense of increased area and power consumption. This explicitly reflects the inherent trade-off between throughput and hardware cost. The design of Raghuram et al. [63] exhibits competitive area–power characteristics at a 25 MHz operating frequency (40 ns period). As no Fmax is reported and the architecture is purely combinational, throughput is derived assuming single-cycle operation at the reported operating frequency. In contrast, under the same throughput-based evaluation assumptions, our designs achieve substantially higher peak throughput due to their higher attainable operating frequencies (Fmax ranging from 474 MHz to 562 MHz). The scaled results from Deepa et al. [64] further emphasize the favorable area–power trade-off of the proposed CORDIC-based approach.

In Table 2, our 3BD design demonstrates substantial area efficiency improvements: 60.1% reduction versus Elnabawy et al. [66] (1,345 vs. 3,372 μm^2) and 49.1% reduction versus Liu et al. [47] (1,345 vs. 2,640 μm^2). Regarding power consumption, while our design consumes higher power than scaled Elnabawy et al. [66] (0.526 vs. 0.06 mW), it operates at a significantly higher frequency (384.6 MHz vs. 9.1 MHz original), delivering approximately 42$\times$ higher throughput. This translates to superior energy efficiency per operation and throughput per watt for real-time neuromorphic applications. Compared to scaled Liu et al. [47], our power is comparable (0.526 vs. 0.50 mW, only 5.2% higher) while achieving sub-

TABLE 1: Hardware evaluation of CORDIC-based multiplier variants and reference approximate squaring design

Design	Area (μm^2)	Power (μW)	Avg. error (%)	F – Fmax (MHz)	Throughput – Peak Throughput (Mops)
Exact	617	114	0.24	100 – 490	6.25 – 30.62
2TN	615	112	0.39	100 – 490	6.25 – 30.62
2U6	593	108	0.44	100 – 476	6.25 – 29.75
2UB	555	99	1.88	100 – 535	6.25 – 33.44
2UY	606	111	0.09	100 – 474	6.25 – 29.62
2YM	562	92	3.76	100 – 562	6.25 – 35.12
32T	546	96	4.48	100 – 513	6.25 – 32.06
36D	556	98	1.62	100 – 508	6.25 – 31.75
Raghuram et al. [63] - P1	594.1	107.3	–	25	25
Raghuram et al. [63] - P2	584.3	106.9	–	25	25
Raghuram et al. [63] - P3	589.6	106.7	–	25	25
Raghuram et al. [63] - P4	577.5	107.7	–	25	25
Deepa et al. [64] 45nm equiv[1]	1,111	45.7	–	–	–

[1] Scaled to 45nm equivalent using technology scaling equations from [65].

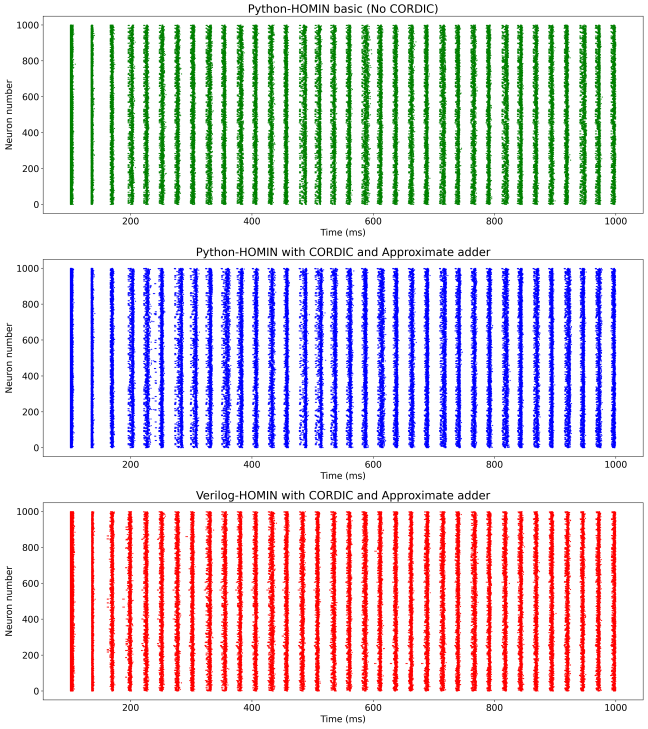


FIGURE 8: Raster plots of 1000 randomly connected excitatory neurons for the HOMIN model using the add16se_2TN approximate adder. From top to bottom: Python implementation without CORDIC; Python implementation with CORDIC and the add16se_2TN approximate adder; Verilog implementation with CORDIC and the add16se_2TN approximate adder.

stantial area savings of 49.1%, indicating a favorable trade-off for high-speed, low-power neuromorphic hardware.

VI. DISCUSSION

Following are some discussion points for this work:

First, this work focuses solely on the neuron design without evaluating its performance in specific applications. Nevertheless, given the low MRE in the range of 0.0073%–0.0509% at the optimal configuration of 9 iterations of CORDIC multiplier, the resulting impact on practical applications is expected to be minimal. Moreover, the analysis shows that this iteration count consistently provides the lowest error across all tested neuron types, balancing numerical accuracy with computational efficiency. As also demonstrated in Fig. 8, the resulting impact on practical applications is expected to be minimal.

Second, one design consideration lies in the selection of the number of CORDIC multiplier's iterations. While more iterations can improve accuracy, they also increase hardware cost, and fewer iterations save resources at the expense of precision. Therefore, an automated optimization flow could help determine the optimal trade-off for a given design target.

Another limitation comes from the approximation library used for the adders. The restricted set of available approximate adder designs reduces flexibility in tailoring the Izhikevich neuron implementation, especially for varying bit-width requirements. This constraint could be addressed either by adopting alternative libraries with more diverse approximate adders or by concatenating accurate adders with approximate ones to achieve finer control over accuracy and resource usage.

Although there are some limitations on this work, we have demonstrated an efficient design for Izhikevich neuron with a novel CORDIC multiplier.

VII. CONCLUSION

This paper presents an approximate-adder-based CORDIC design for implementing the Izhikevich neuron, replacing multipliers with shift-add operations and exact adders with approximate ones to reduce resource usage. The proposed approach improves energy efficiency and hardware cost while maintaining stable performance. Simulation and synthesis results show up to 10.23% improvement in area over the exact design. The optimized configurations also provide a 14.56% increase in throughput compared with the exact baseline. The fully unrolled configuration achieves about 1.5×

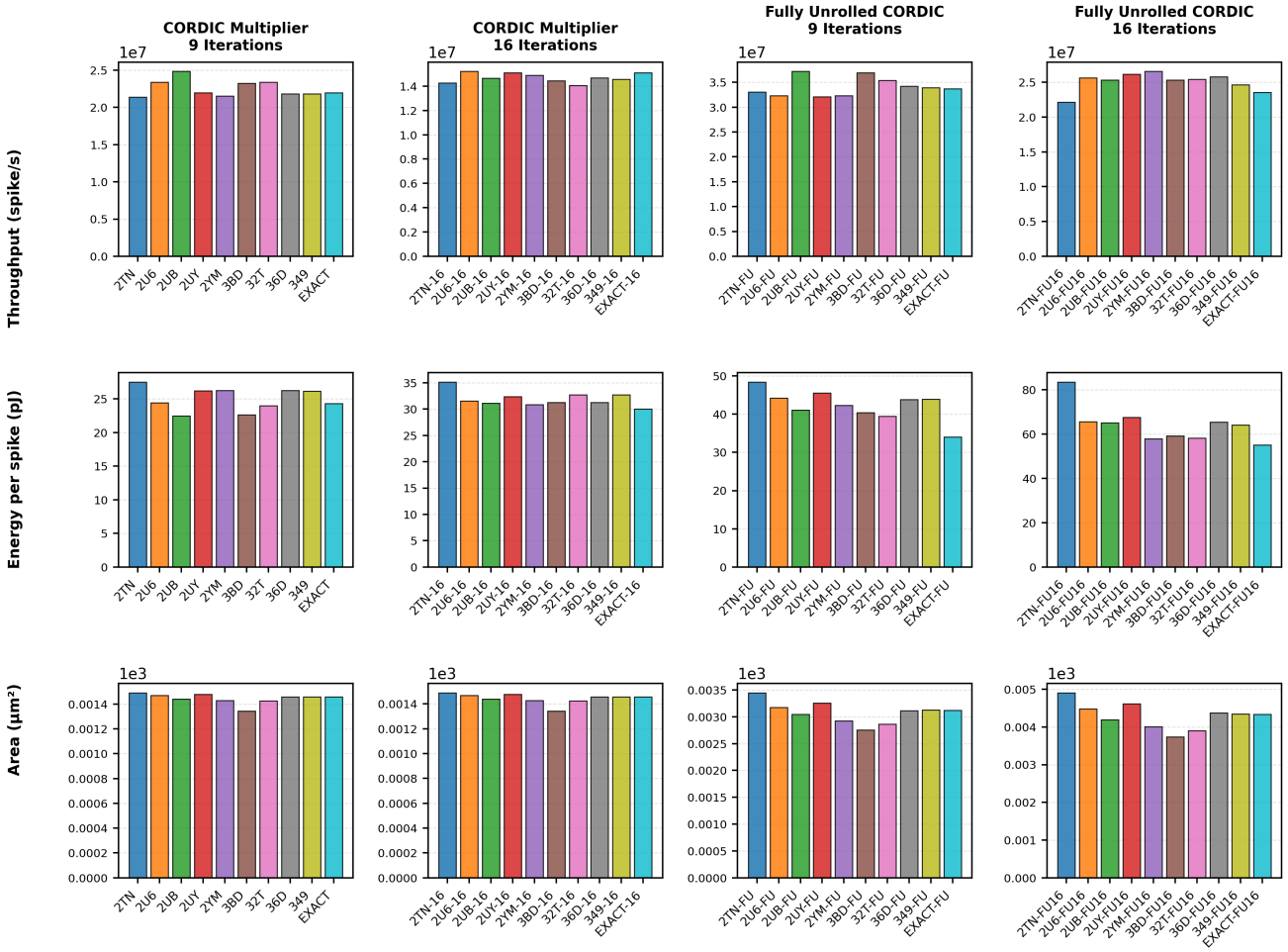


FIGURE 9: Bar charts comparing
CORDIC multiplier and approximate adder configurations in the HOMIN model.

higher throughput with moderate energy overhead. Overall, the design achieves a favorable balance among accuracy, performance, and efficiency, making it suitable for large-scale, energy-efficient neuromorphic hardware. Future work will focus on integrating the proposed neuron into application-level neuromorphic systems to evaluate its real-world performance and resilience. Additional efforts will be devoted to developing automated optimization tools for parameter selection and exploring broader approximation design libraries to further enhance flexibility and efficiency.

REFERENCES

[1] W. Maass, "Networks of spiking neurons: the third generation of neural network models," *Neural networks*, vol. 10, no. 9, pp. 1659–1671, 1997.

[2] A. B. Abdallah and K. N. Dang (authors), *Neuromorphic computing principles and organization (2nd edition)*, 2nd ed. Springer, 2025.

[3] D.-A. Nguyen, X.-T. Tran, K. N. Dang, and F. Iacopi, "A low-power, high-accuracy with fully on-chip ternary weight hardware architecture for Deep Spiking Neural Networks," *Microprocessors and Microsystems*, vol. 90, p. 104458, 2022.

[4] K. Yamazaki, V.-K. Vo-Ho, D. Bulsara, and N. Le, "Spiking neural networks and their applications: A review," *Brain sciences*, vol. 12, no. 7, p. 863, 2022.

[5] K. Boahen, "A neuromorph's prospectus," *Computing in Science & Engineering*, vol. 19, no. 2, pp. 14–28, 2017.

[6] K. Roy, A. Jaiswal, and P. Panda, "Towards spike-based machine intelligence with neuromorphic computing," *Nature*, vol. 575, no. 7784, pp. 607–617, 2019.

[7] 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, 2014.

[8] M. Davies, N. Srinivasa, T.-H. Lin, G. Chinya, Y. Cao, S. H. Choday, G. Dimou, P. Joshi, N. Imam, S. Jain *et al.*, "Loihi: A neuromorphic manycore processor with on-chip learning," *Ieee Micro*, vol. 38, no. 1, pp. 82–99, 2018.

[9] Kobayashi, Ryoji, Nguyen, Ngo-Doanh, N. A. V. Doan, and K. N. Dang, "Energy-efficient spiking neural networks using approximate neuron circuits and 3D stacking memory," in *2024 IEEE 17th International Symposium on Embedded Multicore/Many-core Systems-on-Chip (MCSoC)*. IEEE, 2024, pp. 421–425.

[10] P. A. Merolla, J. V. Arthur, R. Alvarez-Icaza, A. S. Cassidy, J. Sawada, F. Akopyan, B. L. Jackson, N. Imam, C. Guo, Y. Nakamura *et al.*, "A million spiking-neuron integrated circuit with a scalable communication network and interface," *Science*, vol. 345, no. 6197, pp. 668–673, 2014.

[11] C. D. Schuman, S. R. Kulkarni, M. Parsa, J. P. Mitchell, P. Date, and B. Kay, "Opportunities for neuromorphic computing algorithms and applications," *Nature Computational Science*, vol. 2, no. 1, pp. 10–19, 2022.

[12] Nguyen, Ngo-Doanh, A. B. Ahmed, A. B. Abdallah, and K. N. Dang, "Power-Aware Neuromorphic Architecture With Partial Voltage Scaling

TABLE 2: Hardware comparison of neuron designs

Design	Technology	Area (μm^2)	Power (mW)	Fmax(MHz)	Max delay (ns)
Ours (3BD)	45nm ASIC	1,345	0.526	384.6	2.5
Ours (3BD FU)	45nm ASIC	2,756	1.49	212.3	4.71
Elnabawy et al. [66]	130nm ASIC	25,894	0.40	9.1	–
Elnabawy et al. (scaled) [66]	45nm equiv.[1]	3,372	0.06	–	–
Liu et al. [47]	65nm ASIC	3,999.6	0.90	–	1.55
Liu et al. (scaled) [47]	45nm equiv.[1]	2,640	0.50	–	–

[1]Scaled to 45nm equivalent using technology scaling equations from [65].

3-D Stacking Synaptic Memory," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 31, no. 12, pp. 2016–2029, 2023. [Online]. Available: https://doi.org/10.1109/TVLSI.2023.3318231

[13] Kobayashi, Ryoji, N.-D. Nguyen, A. B. Abdallah, N. Anh Vu Doan, and K. N. Dang, "ApproxiMorph: Energy-efficient Neuromorphic System with Layer-wise Approximation of Spiking Neural Networks and 3D-Stacked SRAM," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2025, (accepted).

[14] Nguyen, Ngo-Doanh, X.-T. Tran, A. B. Abdallah, and K. N. Dang, "An in-situ dynamic quantization with 3D stacking synaptic memory for power-aware neuromorphic architecture," *IEEE Access*, vol. 11, pp. 82 377–82 389, 2023.

[15] Nguyen, Ngo-Doanh, K. N. Dang, A. B. Ahmed, A. B. Abdallah, and X.-T. Tran, "NOMA: A novel reliability improvement methodology for 3-D IC-based neuromorphic systems," *IEEE Transactions on Components, Packaging and Manufacturing Technology*, 2024 (in-press). [Online]. Available: https://doi.org/10.1109/TCPMT.2024.3488113

[16] K. N. Dang, N. A. V. Doan, and A. B. Abdallah, "MigSpike: A migration based algorithms and architecture for scalable robust neuromorphic systems," *IEEE Transactions on Emerging Topics in Computing*, vol. 10, no. 2, pp. 602–617, 2021.

[17] P. Dayan and L. F. Abbott, *Theoretical neuroscience: computational and mathematical modeling of neural systems*. MIT press, 2005.

[18] A. L. Hodgkin and A. F. Huxley, "A quantitative description of membrane current and its application to conduction and excitation in nerve," *Bulletin of mathematical biology*, vol. 52, no. 1, pp. 25–71, 1990.

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

[20] ——, "Which model to use for cortical spiking neurons?" *IEEE transactions on neural networks*, vol. 15, no. 5, pp. 1063–1070, 2004.

[21] 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.

[22] G. Indiveri, B. Linares-Barranco, T. J. Hamilton, A. v. Schaik, R. Etienne-Cummings, T. Delbruck, S.-C. Liu, P. Dudek, P. Häfliger, S. Renaud *et al.*, "Neuromorphic silicon neuron circuits," *Frontiers in neuroscience*, vol. 5, p. 73, 2011.

[23] K. Cheung, S. R. Schultz, and W. Luk, "Neuroflow: a general purpose spiking neural network simulation platform using customizable processors," *Frontiers in neuroscience*, vol. 9, p. 516, 2016.

[24] M. N. Andabili, S. Nazari, and T. Moosazadeh, "Chaotic dynamics analysis and digital hardware design of the izhikevich neuron model," *Scientific Reports*, vol. 15, no. 1, p. 16766, 2025.

[25] F. Akopyan, J. Sawada, A. Cassidy, R. Alvarez-Icaza, J. Arthur, P. Merolla, N. Imam, Y. Nakamura, P. Datta, G.-J. Nam *et al.*, "Truenorth: Design and tool flow of a 65 mw 1 million neuron programmable neurosynaptic chip," *IEEE transactions on computer-aided design of integrated circuits and systems*, vol. 34, no. 10, pp. 1537–1557, 2015.

[26] S. S. Sarwar, S. Venkataramani, A. Raghunathan, and K. Roy, "Multiplier-less artificial neurons exploiting error resiliency for energy-efficient neural computing," in *2016 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2016, pp. 145–150.

[27] M. Heidarpur, A. Ahmadi, M. Ahmadi, and M. R. Azghadi, "Cordic-snn: On-fpga stdp learning with izhikevich neurons," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 66, no. 7, pp. 2651–2661, 2019.

[28] Kobayashi, Ryoji and K. N. Dang, "An efficient hardware implementation of spiking neural network using approximate Izhikevich neuron," in *2024 9th International Conference on Integrated Circuits, Design, and Verification (ICDV)*. IEEE, 2024, pp. 13–18.

[29] J. E. Volder, "The cordic trigonometric computing technique," *IRE Transactions on electronic computers*, no. 3, pp. 330–334, 2009.

[30] J. S. Walther, "A unified algorithm for elementary functions," in *Proceedings of the May 18-20, 1971, spring joint computer conference*, 1971, pp. 379–385.

[31] S. C. Inguva and J. B. Seventline, "Lh-cordic: Low power fpga based implementation of cordic architecture." *International Journal of Intelligent Engineering & Systems*, vol. 12, no. 2, 2019.

[32] J. Wang, Z. Peng, Y. Zhan, Y. Li, G. Yu, K.-S. Chong, and C. Wang, "A high-accuracy and energy-efficient cordic based izhikevich neuron with error suppression and compensation," *IEEE Transactions on Biomedical Circuits and Systems*, vol. 16, no. 5, pp. 807–821, 2022.

[33] R. Xu, Z. Jiang, H. Huang, and C. Dong, "An optimization of cordic algorithm and fpga implementation," *International Journal of Hybrid Information Technology*, vol. 8, no. 6, pp. 217–228, 2015.

[34] J. Han and M. Orshansky, "Approximate computing: An emerging paradigm for energy-efficient design," in *2013 18th IEEE European test symposium (ETS)*. IEEE, 2013, pp. 1–6.

[35] G. Indiveri and S.-C. Liu, "Memory and information processing in neuromorphic systems," *Proceedings of the IEEE*, vol. 103, no. 8, pp. 1379–1397, 2015.

[36] M. Osta, A. Ibrahim, and M. Valle, "Fpga implementation of approximate cordic circuits for energy efficient applications," in *2019 26th IEEE International Conference on Electronics, Circuits and Systems (ICECS)*. IEEE, 2019, pp. 127–128.

[37] N. Thanh Dat, L. Van Vu, P. Quang Thai, N. Duy Anh, D. Khanh N., L. Nguyen Khanh, and T. Dao Thanh, "Low-power cordic multiplier design using approximate arithmetic for energy-efficient computing," *Transport and Communications Science Journal*, vol. 77, no. 1, pp. 30–42, 2026.

[38] A. J. Leigh, M. Mirhassani, and R. Muscedere, "An efficient spiking neuron hardware system based on the hardware-oriented modified izhikevich neuron (homin) model," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 67, no. 12, pp. 3377–3381, 2020.

[39] M. Hopkins and S. Furber, "Accuracy and efficiency in fixed-point neural ode solvers," *Neural computation*, vol. 27, no. 10, pp. 2148–2182, 2015.

[40] D. Pani, P. Meloni, G. Tuveri, F. Palumbo, P. Massobrio, and L. Raffo, "An fpga platform for real-time simulation of spiking neuronal networks," *Frontiers in neuroscience*, vol. 11, p. 90, 2017.

[41] Nguyen, Ngo-Doanh and K. N. Dang, "A novel yield improvement approach for 3D stacking neuromorphic architecture," in *2023 IEEE 16th International Symposium on Embedded Multicore/Many-core Systems-on-Chip (MCSoC)*. IEEE, 2023, pp. 392–398.

[42] P.-K. Dong, K. N. Dang, D.-A. Nguyen, and X.-T. Tran, "A lightweight neuromorphic controlling clock-gating based multi-core cryptography platform," *Microprocessors and Microsystems*, vol. 106, p. 105040, 2024.

[43] Z. Karaca, N. Korkmaz, Y. Altuncu, and R. Kılıç, "An extensive fpga-based realization study about the izhikevich neurons and their bio-inspired applications," *Nonlinear Dynamics*, vol. 105, no. 4, pp. 3529–3549, 2021.

[44] V. Bandeira, V. L. Costa, G. Bontorin, and R. A. Reis, "Low latency fpga implementation of izhikevich-neuron model," in *International Embedded Systems Symposium*. Springer, 2015, pp. 210–217.

[45] S. Çağdaş and N. S. Şengör, "A folded architecture for hardware implementation of a neural structure using izhikevich model," in *International Conference on Artificial Neural Networks*. Springer, 2022, pp. 508–518.

[46] M. F. Tolba, A. H. Elsafty, M. Armanyos, L. A. Said, A. H. Madian, and A. G. Radwan, "Synchronization and fpga realization of fractional-order

izhikevich neuron model," *Microelectronics Journal*, vol. 89, pp. 56–69, 2019.
[47] H. Liu, M. Wang, L. Yao, and M. Liu, "Hardware implementation of an approximate simplified piecewise linear spiking neuron," *Electronics*, vol. 12, no. 12, p. 2628, 2023.
[48] R. Andraka, "A survey of cordic algorithms for fpga based computers," in *Proceedings of the 1998 ACM/SIGDA sixth international symposium on Field programmable gate arrays*, 1998, pp. 191–200.
[49] M. Chinnathambi, N. Bharanidharan, and S. Rajaram, "Fpga implementation of fast and area efficient cordic algorithm," in *2014 International Conference on Communication and Network Technologies*. IEEE, 2014, pp. 228–232.
[50] G. Naga Jyothi, K. Debanjan, and G. Anusha, "Asic implementation of fixed-point iterative, parallel, and pipeline cordic algorithm," in *Soft Computing for Problem Solving: SocProS 2018, Volume 1*. Springer, 2019, pp. 341–351.
[51] I. Tsmots, O. Skorokhoda, and V. Rabyk, "Hardware implementation of sigmoid activation functions using fpga," in *2019 IEEE 15th International Conference on the Experience of Designing and Application of CAD Systems (CADSM)*. IEEE, 2019, pp. 34–38.
[52] B. Khurshid, "High-performance cordic-based approximate mac architectures for fpga platforms," *Integration*, vol. 101, p. 102338, 2025.
[53] Y. Kim, Y. Zhang, and P. Li, "Energy efficient approximate arithmetic for error resilient neuromorphic computing," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 23, no. 11, pp. 2733–2737, 2014.
[54] S. Venkataramani, S. T. Chakradhar, K. Roy, and A. Raghunathan, "Approximate computing and the quest for computing efficiency," in *Proceedings of the 52nd Annual Design Automation Conference*, 2015, pp. 1–6.
[55] S. Mittal, "A survey of techniques for approximate computing," *ACM Computing Surveys (CSUR)*, vol. 48, no. 4, pp. 1–33, 2016.
[56] O. Spantidi, G. Zervakis, I. Anagnostopoulos, H. Amrouch, and J. Henkel, "Positive/negative approximate multipliers for dnn accelerators," in *2021 IEEE/ACM International Conference On Computer Aided Design (ICCAD)*. IEEE, 2021, pp. 1–9.
[57] G. Armeniakos, G. Zervakis, D. Soudris, and J. Henkel, "Hardware approximate techniques for deep neural network accelerators: A survey," *ACM Computing Surveys*, vol. 55, no. 4, pp. 1–36, 2022.
[58] Y. Wang, H. Zhang, K.-I. Oh, J.-J. Lee, and S.-B. Ko, "Energy efficient spiking neural network processing using approximate arithmetic units and variable precision weights," *Journal of Parallel and Distributed Computing*, vol. 158, pp. 164–175, 2021.
[59] S. Rehman, W. El-Harouni, M. Shafique, A. Kumar, J. Henkel, and J. Henkel, "Architectural-space exploration of approximate multipliers," in *2016 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. IEEE, 2016, pp. 1–8.
[60] F. Conti and L. Benini, "A ultra-low-energy convolution engine for fast brain-inspired vision in multicore clusters," in *2015 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2015, pp. 683–688.
[61] S. Sen, S. Venkataramani, and A. Raghunathan, "Approximate computing for spiking neural networks," in *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2017*. IEEE, 2017, pp. 193–198.
[62] V. Mrazek, R. Hrbacek, Z. Vasicek, and L. Sekanina, "Evoapprox8b: Library of approximate adders and multipliers for circuit design and benchmarking of approximation methods," in *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2017*, 2017, pp. 258–261.
[63] C. N. Raghuram, G. K. Kumar, J. K. Mahankali, S. H. Yasmine, A. Jahnavi, P. K. Kumar, and K. Srinivas, "Design of energy-efficient approximate squaring circuits for error-resilient signal and image processing applications," *Ain Shams Engineering Journal*, vol. 16, no. 9, p. 103523, 2025.
[64] A. Deepa and C. Marimuthu, "Design of a high speed vedic multiplier and square architecture based on yavadunam sutra," *Sādhanā*, vol. 44, no. 9, p. 197, 2019.
[65] A. Stillmaker and B. Baas, "Scaling equations for the accurate prediction of cmos device performance from 180nm to 7nm," *Integration*, vol. 58, pp. 74–81, 2017. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167926017300755
[66] 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.

VAN-VU LUYEN is currently with the Department of Electronic Engineering, University of Transport and Communications, Vietnam. He is currently working as a research assistant with the University while pursuing his bachelor's degree in engineering. His research interests include Deep Learning, Neuromorphic Engineering and Approximate Computing techniques.

THANH-DAT NGUYEN is currently with the Department of Electronic Engineering, University of Transport and Communications, Vietnam. He is currently working as a research assistant with the University while pursuing his bachelor's degree in engineering. His research interests include Hardware Architectures for Deep Learning and Neuromorphic Engineering.

QUANG-THAI PHAM is currently with the Department of Electronic Engineering, University of Transport and Communications, Vietnam. He is currently working as a research assistant with the University while pursuing his bachelor's degree in engineering. His research interests include energy efficient approximate computing techniques for neuromorphic engineering.

DUY-ANH NGUYEN is with the Department of Electronic Engineering, University of Transport and Communications, Vietnam. He received his Ph.D. from the VNU University of Engineering and Technology and the Joint Technology Innovation and Research Centre between Vietnam National University, Hanoi (VNU) and the University of Technology Sydney; his Master's degree from Western Jiao Tong University, China; and his Bachelor's degree from Nanyang Technological University, Singapore. His research interests include system-on-chip design and hardware accelerators for artificial intelligence.

KHANH N. DANG (Member, IEEE) is currently an Associate Professor at the University of Aizu, Japan. He received his M.Sc. from the University of Paris-XI in 2014 and his Ph.D. from the University of Aizu in 2017. His research interests include 3D integrated circuits, neuromorphic engineering, and carbon-neutral AI computing.

VAN HAI PHAM P.V.Hai received the B.Sc. degree in 2007 and earned my M.Sc. degree in 2010 from the Ha Noi Open University, Vietnam. He is currently at Ha Noi Noi Open University. His primary research interests include electronic engineering, structural health monitoring (SHM), computer vision, artificial neural networks, FPGA, and image processing. He is dedicated to advancing these fields through her academic research and practical applications.

THANH-TOAN DAO D.T. Toan received the B.Sc. and M.Sc. degrees from the University of Transport and Communications, Vietnam, and the Ph.D. degree from the Japan Advanced Institute of Science and Technology in 2012. Currently, he is an Associate Professor in the Department of Electronic Engineering, University of Transport and Communications, Vietnam. He has authored over 30 articles and is a co-inventor of 2 patents. His recent research interests include emerging semiconductors, IoT for transportation, and autonomous vehicles.