# A High Accuracy and Hardware Efficient Approximate Computing based Leaky integrate-and-fire Neuron Model

Haihang Xia, Shengdi Wang, Yuqin Zhao, Haotian Liu, Charith Abhayaratne, and Tiantai Deng
*School of Electronic and Electrical Engineering, University of Sheffield, Sheffield, United kingdom*
hxia10@sheffield.ac.uk, t.deng@sheffield.ac.uk

***Abstract*—The adoption of Spiking Neural Networks (SNNs) has grown significantly, driven by their potential to enhance image processing, robotics, and motor control. These applications typically demand both high performance and low power consumption, especially when deployed on edge devices. Achieving high-performance and low-power SNNs in hardware remains challenging due to their computational complexity and large-scale design. Balancing accuracy and speed often increases power and resource usage, making efficient implementation essential. Optimizing a single neuron model—a fundamental unit replicated thousands of times—is key to improving overall hardware efficiency. The Leaky Integrate-and-Fire (LIF) model, widely used in SNNs, offers a more efficient alternative to other neuron models by improving computational and energy efficiency.**

**This paper presents a LIF neuron model design based on approximate multiplication. Given the high robustness of SNNs, they are well-suited for approximate computing. The proposed design significantly reduces multiplication complexity with only 2.6099% error. The minimal error has little impact on SNN performance, as shown by similar training results between approximate and precise LIF-based SNNs across various datasets and network sizes. To further demonstrate the advantages of our AM-based LIF neuron model, we carry on a test through a Xilinx FPGA. The FPGA implementation results demonstrate that the AM-based LIF neuron achieves a 8.26% to 84.36% reduction in Look-Up Table utilization, a 17.04% to 86.49% reduction in slice register utilization, and achieving a 10.31-fold increase in energy efficiency relative to the state-of-the-art LIF neuron model.**

***Index Terms*—Spiking Neural Networks, Leaky integrate-and-fire model, FPGA, approximate computing.**

## I. Introduction

In recent years, SNNs have experienced rapid development due to their potential for a wide range of applications, including image processing, event-based data analysis, robotics, and communication. Compared to Convolution Neural Networks (CNNs) and Transformers [1], SNNs offer significantly better power efficiency and lower hardware resource utilization, making them particularly well-suited for deployment on hardware-constrained and energy-limited platforms such as edge devices. In addition, SNNs feature unique learning algorithms, enabling them to be applied across diverse application domains and to effectively process complex spatio-temporal information [2]. As a result, SNNs are increasingly regarded as a promising paradigm for next-generation neural networks [3] [4] [5].

The neuron model is the basic component in SNNs. An SNN neuron model is a mathematical description that simulates the variation in potential difference across a biological neuron's cell membrane [6]. To balance computational efficiency with biological plausibility, a variety of neuron models have been proposed for SNN, for example, the Leaky Integrate-and-Fire (LIF) model [7], the Hodgkin-Huxley (HH) model [8], and the Izhikevich model [9]. Among them, the LIF neuron model is the most widely used due to its simple computation while maintaining a reasonable level of biological plausibility [10]. However, the LIF neuron model still involves complex operations such as iterative multiplication, addition, and exponentiation. These power- and resource-intensive computations can significantly impact performance-power efficiency in hardware implementations [11].

Simulating SNNs on von Neumann machines is typically inefficient due to asynchronous network activity, which leads to quasi-random access to synaptic weights in both time and space [5]. Additionally, Application-Specific Integrated Circuits (ASICs) are less commonly used due to their high development cost, long design cycles, and lack of reconfigurability. As a result, FPGAs have become a preferred platform for deploying and testing, owing to their low cost, high parallelism, and programmability [12].

In order to reduce the hardware resource utilization of the LIF neuron model, J.Kim *et al.* use Template-Scaling-Based Exponential Function Approximation (TS-EFA) to emulate the LIF neuron model [13]. TS-EFA slightly reduces the latency introduced by the CORDIC iteration and eliminates the hardware resources utilization and power consumption associated with lookup-table-based approximation computing. Although the TS-EFA method reduces latency, it requires a large amount of RAM [13], resulting in high hardware resource utilization and increased power consumption [15].

To improve performance and energy efficiency, this paper proposes a novel LIF neuron design based on approximate multiplication (AM) and validates it through FPGA implementation. Approximate multiplication reduces computational complexity by trading a small amount of accuracy for simplified operations, leading to lower hardware utilization and reduced power consumption. Given the high robustness of SNNs and the high accuracy of the proposed AM-based LIF neuron model, its adoption does not degrade SNNs training and inference accuracy. Moreover, in the hardware implementation of the AM-based LIF neuron model, conventional mul-

tiplication is replaced with a linear approximate multiplication technique, which significantly improves power efficiency and reduces hardware resource utilization. The main contributions of this paper include:

1) An optimized approximate multiplier is proposed for the LIF neuron model, achieving a high accuracy of 97.39% while significantly reducing hardware complexity, with 59.4% fewer slice registers and 53.4% lower LUTs utilization compared to conventional multipliers.
2) The proposed AM-based LIF neuron model was tested at both the individual neuron level and the SNN network level through a software-based emulator. The network demonstrated an impressive accuracy of 92.68% on the MNIST dataset and 97.68% on FashionMNIST.
3) The AM-based LIF neuron model was implemented on an FPGA, demonstrating competitive results in energy efficiency and resource utilization. Compared to state-of-the-art designs, it achieved a reduction in slice register utilization by 17.04% to 86.94% and a reduction in LUTs utilization by 8.26% to 84.36%. Additionally, the energy consumption per neuron iteration was reduced by 90.29%, and energy efficiency was improved by a factor of 10.31.

The rest of this paper is organized as follows. Section II introduces the proposed approximate multiplication and the AM-based LIF neuron model. Section III provides a detailed description of the software-level testing of the proposed design. Section IV discusses the hardware design and implementation. Section V evaluates the FPGA implementation results and compares them with state-of-the-art designs. Finally, Section VI concludes the paper.

## II. AM-Based LIF Neuron Model

### A. LIF Neuron Model

The LIF neuron model is the most widely used due to its simple mathematical calculations while retaining a degree of biological plasticity [12]. In order to make the LIF neuron model meet the requirements of the hardware implementation, we discretize it to obtain Eq.1 [14].

$$V[t+1] = e^{-\frac{1}{\tau}} \cdot (V[t] - E_{rest}) + E_{rest} + I \tag{1}$$

The neuron's membrane potential is represented by $V[t]$, where $\tau$ is the time constant and $E_{rest}$ is the reset potential. $I$ represents the current stimulus input to the neuron membrane; in SNN, $I$ comes from the output of the previous neuron. Thus, the expression for $I$ is shown in Eq.2.

$$I = \sum_{i=0}^{n} \omega_i f_i \tag{2}$$

Where $\omega$ is the weight between neurons. If the pre-synaptic neuron generates a spike to the post-synaptic neuron, the parameter $f$ equals 1; otherwise, $f$ equals 0.

$$if\ V > V_{th},\ then\ V = E_{rest} \tag{3}$$

Furthermore, when V is sufficiently large to reach the threshold $V_{th}$, the neuron generates a spike, and V subsequently returns to the reset potential, as shown in Eq. 3.

### B. Approximate Multiplication

Approximate computation represents a classical trade-off between accuracy loss and increased utilization of hardware resources. In this context, the proposed approach introduces only a minimal impact on accuracy and, therefore, does not affect the performance of the LIF-based SNN. The approximate computing algorithms proposed in this paper are based on linearly approximated logarithmic multiplication [16]. Assume that there are two N-bit fixed-point numbers, A and B. The first '1' in the numbers A and B occurs at positions $k_a$ and $k_b$, where $(N-1) \geq k_a, k_b \geq 0$. According to Mitchell multiplication, the linear approximate product of A and B can be obtained by Eq.4. Where $\widetilde{A}$ and $\widetilde{B}$ is the fractional part of integer A and B.

$$\tilde{P} = \widetilde{A \times B} \approx \begin{cases} 2^{k_a+k_b}(1+\tilde{A}+\tilde{B}), & \tilde{A}+\tilde{B} < 1 \\ 2^{k_a+k_b+1}(\tilde{A}+\tilde{B}), & \tilde{A}+\tilde{B} \geq 1 \end{cases} \tag{4}$$

Since the logarithmic curve is a convex function, the values obtained through linear approximation are always less than the exact values. Therefore, the error can be calculated by $\tilde{P}$ minus the exact value. And the average error can be calculated as Eq.5.

$$\begin{aligned} \text{Average}_{\text{Error}} &= \frac{1}{(1-0)(1-0)} \int_0^1 \int_0^1 (\tilde{P} - P)\ d\tilde{A}d\tilde{B} \\ &\approx -(0.08333) \times 2^{k_a+k_b} \end{aligned} \tag{5}$$

As the error is always negative, the approximate multiplication can be modified as Eq.6.

$$\tilde{P} = \widetilde{A \times B} \approx \begin{cases} 2^{k_a+k_b}(1+\tilde{A}+\tilde{B}+c), & \tilde{A}+\tilde{B} < 1 \\ 2^{k_a+k_b+1}(\tilde{A}+\tilde{B}+\frac{c}{2}), & \tilde{A}+\tilde{B} \geq 1 \end{cases} \tag{6}$$

### C. AM-based LIF Neuron Model

The proposed LIF neuron model can be written as Eq.7 if we replace the multiplication in LIF by our AM approach.

$$\begin{aligned} V[t+1] =& AM(e^{-\frac{1}{\tau}},\ V[t] - E_{rest}) + E_{rest} + I \\ & if\ V[t] \leq V_{th},\ then\ V[t+1] = E_{rest} \end{aligned} \tag{7}$$

## III. Software Emulation & Evaluation of AM-based LIF Neuron Model

### A. Evaluation of Approximate Multiplication

A 16-bit software emulator of the proposed approximate multiplier (AM) was developed in Python to replicate the hardware behavior, with its accuracy and standard deviation evaluated using one million randomly selected operand pairs (A and B) within the range of [0, 65535].

By comparing the emulator results with the exact computed values, the average relative error was found to be 2.6099%, with a standard deviation of only 1.846% and a maximum error of 8.3%. The error distribution, shown in Fig.1, indicates a uniform pattern that prevents isolated high-error cases from

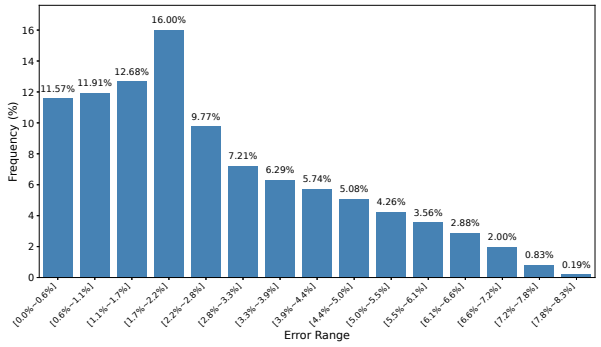
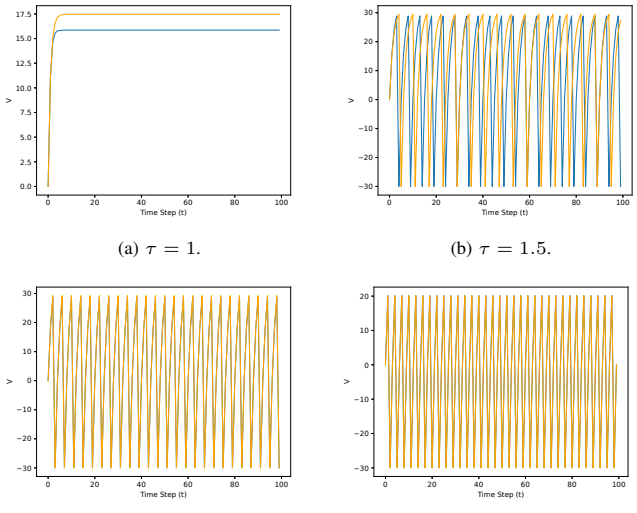



Fig. 1. Approximate Multiplication Error Distribution Graph.

adversely affecting overall system performance. Moreover, the proposed method achieves higher accuracy than the iterative error compensation method (4.43%) while maintaining lower computational complexity [17].

### *B. Evaluation of AM-based LIF Neuron Model*

An emulator of a 16-bit AM-based LIF neuron model has been developed using Python for emulation. This emulator serves to assess the impact of errors introduced by the approximation on neurons.




(a) $\tau = 1$. (b) $\tau = 1.5$.

(c) $\tau = 2$. (d) $\tau = 2.5$.

Fig. 2. Comparison of the original LIF neuron model with the AM-based LIF neuron model. The orange line shows the original model, and the blue line shows the AM-based LIF neuron model.

Fig. 2 shows the different performances of AM-based LIF neuron models and normal LIF neuron models with the same parameters, where $I$ equals 30, $E_{rest}$ equals -30, and $V_{th}$ equals 30. It can be seen that when $\tau$ is too small, a larger $I$ input is required to generate a spike. Furthermore, the degree of difference between the two curves varies when the time constant $\tau$ takes different values. To accurately quantify the differences between the neuron model implemented with AFM and the model obtained through exact computation, time error and Normalized Root Mean Square Deviation were used for evaluation [18].

**Time error (ERRT):** The error from the approximate computation in the neuron may cause a difference in spike timing and lag compared with the original model. The time between two neuron spikes was recorded with the same neuron model parameters and outputs, and synchronized neurons. The ERRT is calculated as follows:

$$ERRT = \left| \frac{\Delta t_a - \Delta t_o}{\Delta t_o} \right| \times 100\% \tag{8}$$

As Fig.3 shows, $\Delta t_a$ means the time interval between spikes in the approximate computing neuron model. Correspondingly, $\Delta t_o$ this means the time interval between spikes in the original neuron model.

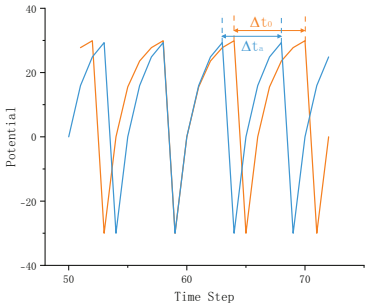



Fig. 3. ERRT: An example diagram to show the time error measurement. The blue line shows the original model, and the yellow line shows the proposed neuron model.

**Normalized Root Mean Square Deviation error (NRMSD):** The NRMSD is used to measure the similarity of spike shapes in approximated and original models. NRMSD is defined as

$$NRMSD = \frac{\sqrt{\frac{\sum_{i=1}^{n} (v_a(n) - v_o(n))^2}{n}}}{v_{max} - v_{min}}$$

Where $v_a$ and $v_o$ are the membrane potentials of the approximated and original neuron models. $v_{max}$ and $v_{min}$ are the maximum and minimum values in the $v_o$ domain.

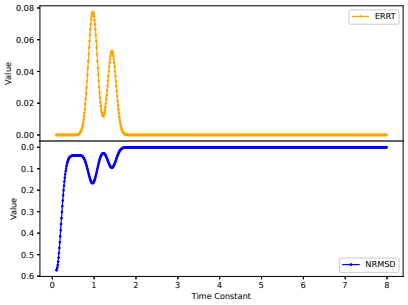



Fig. 4. ERRT and NRMSD of AM-based LIF neuron model.

Fig.4 illustrates the variation of ERRT and NRMSD with the time constant $\tau$ under fixed input current $I$. As $\tau$ increases, both metrics decrease and may even approach zero. This trend arises because $\tau$ is positively related to $e^{-\frac{1}{\tau}}$, leading to more stable spike generation in the LIF neuron model (e.g., Fig.2(c), (d)). When $\tau$ is small, spike generation becomes unstable or ceases entirely (Fig.2(a)), resulting in higher accumulated errors. Notably, in cases where no spikes occur, ERRT becomes zero. In practice, $\tau$ is typically chosen as an integer greater than 2 in LIF-based SNNs.

### C. Evaluation of AM-based LIF Spiking Neural Networks

To assess the impact of approximation-induced errors on training accuracy, both 1-layer and 6-layer SNNs were constructed using exact and AM-based LIF neuron models. As illustrated in Fig.5, the training results exhibit high consistency, demonstrating similar accuracy and convergence behavior across both simple and complex network architectures. Specifically, the 1-layer SNN was trained on the MNIST dataset, while the 6-layer SNN was evaluated using the FashionMNIST dataset. On the MNIST dataset, the AM-based LIF SNN and the exact LIF SNN achieved maximum test accuracies of 92.68% and 92.61%, respectively. On the FashionMNIST dataset, the AM-based LIF SNN and the exact LIF SNN achieved maximum training accuracies of 97.68% and 97.78%, respectively. Furthermore, across both datasets, the training curves of the AM-based models closely align with those of their exact counterparts. This demonstrates that, owing to the inherent robustness of SNNs, approximation-induced errors have a negligible impact on the accuracy of both simple and complex network architectures.

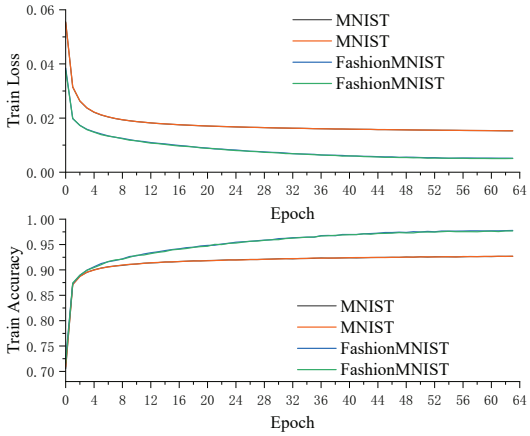



Fig. 5. Training result of AM-based LIF SNN and original LIF SNN.

All Python-based emulators, SNN testing codes, and additional figures not presented in the paper are available in the following GitHub repository: AM-based-LIF-SOCC-2025.

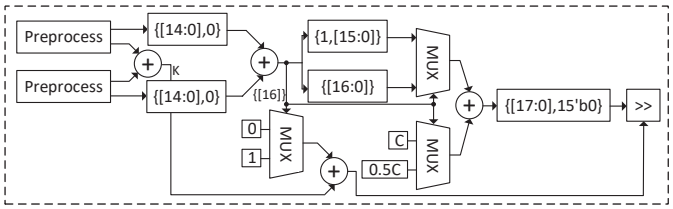
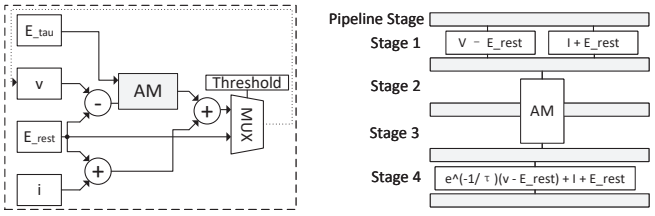



Fig. 6. Hardware design structure of the proposed approximate multiplier.

## IV. Hardware Implementation of AM-based LIF Neuron Model

### A. Hardware Design of AM-based LIF Neuron Model

Fig. 6 shows the data flow for the approximate multiplier. The computation begins with the pre-processing module, which extracts the fractional part of the operand and obtains $\tilde{A}$ and $\tilde{B}$. These two values are then summed, and the adder's carry-out is used to determine whether the sum exceeds 1. Furthermore, the carry-out is also used to decide whether to add 1 to the left-shifted digit and the error offset. The difference from the formula is that the $k$ values in preprocessing are actually $16-k_a$ and $16-k_b$. This is done to simplify subsequent shift operations and avoid subtraction. In effect, both operands are left-shifted by 16 bits during preprocessing. To improve accuracy, overflow bits are preserved in both the addition and error compensation stages, resulting in an 18-bit intermediate value effectively left-shifted by 17 bits. Therefore, the final result requires only a right shift of either $k_a+k_b-15$ or $k_a+k_b+1-15$. To simplify the computation and avoid conditional operations, the design eliminates the need to determine whether the final shift should be to the left or right based on the sign of $k_a+k_b-15$ or $k_a+k_b+1-15$. Instead, 15 zero bits are appended to the least significant end of the 18-bit value before applying the right shift of $k_a+k_b$ or $k_a+k_b+1$. After the right shift operation, the overflow bits are discarded to produce the final 32-bit output result.




(a) Hardware Structure. (b) Pipeline structure.

Fig. 7. The hardware design structure of AM-based LIF neuron model.

Fig.7(a) shows the control data flow of the AM-based LIF neuron model. Furthermore, the pipeline structure of the AM-based LIF neuron model is shown in Fig.7(b). Since the time constant $\tau$ usually remains unchanged in network computations, $e^{-\frac{1}{\tau}}$ can be precomputed and directly used as an input, thereby reducing hardware implementation resources. First, the LIF neuron performs the calculations of $V-E_{rest}$ and $I+E_{rest}$ in parallel. Then, the approximate multiplication of $E_{tau}$ and $V-E_{rest}$ is completed over two clock cycles,

followed by the accumulation of all computation results. Finally, the result is checked to determine whether it exceeds the threshold $V_{th}$, after which it is fed back into the LIF neuron for the next iteration of the membrane potential update.

## V. FPGA Implementation Result and Evaluation

To evaluate the hardware design of the AM-based LIF neuron model, tests were performed at both the multiplier level and the neuron level. Furthermore, the evaluation included comparisons with state-of-the-art LIF neuron models in terms of hardware resource utilization, power consumption, and computation speed. The AM-based LIF neuron model and the AM-based multiplier were developed using Verilog HDL and implemented on the Xilinx Zynq UltraScale+ FPGA board. The hardware for the AM-based LIF neuron model was developed and evaluated on Vivado 2023.2.

TABLE I
Comparison of Approximate Multiplier with Dadda Multiplier

| | **Approximate Multiplier** | **Dadda Multiplier** |
|---|---|---|
| FPGA Platform | Zynq UltraScale+ | Zynq UltraScale+ |
| Bit-width | 16-bit fixed-point | 16-bit fixed-point |
| Slice Registers | 26 | 64 |
| Slice LUTs | 163 | 350 |
| Maximum Operating Frequency | 307.13 MHz | 175.96 MHz |
| Pipeline | 2 | 2 |
| Latency | 6.512 ns | 11.366 ns |
| Power @100 MHz | 3 mW | 8 mW |

### *A. Evaluation of Approximate Multiplier for LIF Neuron Model Design*

Table I compares the performance of the approximate multiplier and the Dadda multiplier. Both of them are implemented in the same bit-width (1, 6, 9) and on the same FPGA platform. The approximate multiplier uses 46.6% LUTs of the Dadda tree multiplier. And the approximate multiplier's maximum operation frequency is 174.5% of the Dadda tree multiplier. When all the multipliers are working at 100 MHz, the power consumption of the approximate multiplier is only 37.5% of the Dadda tree multiplier. Moreover, the latency of the approximate multiplier is only 57.3% of the Dadda tree with the same pipeline stage number.

### *B. Hardware Overhead Evaluation of AM-based LIF Neuron Model Design*

Table II compares the implementation results of the AM-based LIF neuron model with other state-of-the-art LIF neuron models. The AM-based LIF neuron model saves 44.90% of the LUTs from [22], 84.36% of the LUTs from [21], and 83.62% of the LUTs from [20], as shown in Table II. Although [13] implemented 512 neurons using only 314 LUTs, which is 57% more than the AM-based LIF neuron model, it utilized a 1003×64 b memory. By pre-computing most of the data and storing it in memory, [13] was able to use very few LUTs. However, a 6.4 KB RAM area is approximately equivalent to 52,429 6-input LUTs, and reading data from RAM introduces significant power consumption and latency. Moreover, the AM-based LIF neuron model also uses significantly fewer slice registers than other LIF neuron models. The AM-based LIF neuron model, therefore, minimizes the implementation area of a single neuron and greatly enhances hardware resource utilization. The AM-based LIF neuron model is a more appealing option for implementing large-scale SNNs, which typically comprise a large number of neurons.

### *C. Running Speed Evaluation of AM-based LIF Neuron Model Design*

Table II shows that the AM-based LIF neuron model achieves a maximum operating frequency of 267.38 MHz, representing an improvement of 98.94% over [20] and 40.73% over [21]. Although its frequency is 46.52% lower than that of [22], this is primarily due to [22]'s smaller bit width and significantly higher hardware resource utilization, with 81.50% and 160.71% more LUTs and slice registers, respectively. Moreover, a lower bit width in the LIF neuron can compromise its precision, thereby reducing its representational capability. Therefore, this work adopts a 16-bit width as a trade-off between performance and accuracy. This means the AM-based LIF SNN will have a much faster speed and be more suitable for online learning.

### *D. Energy Efficiency Evaluation of AM-based LIF Neuron Model Design*

The power consumption of the AM-based LIF neuron model is only 4 mW, which is just 1.73% of that in [22]. Although [22] operates at a higher frequency, the performance gains from the increased frequency do not outweigh the rise in energy consumption. Furthermore, in terms of energy consumption per neuron iteration and overall energy efficiency, the AM-based LIF neuron model achieved an 90.29% reduction in energy consumption and a 10.31-fold improvement in energy efficiency compared to [22]. Although works such as [13] and [19] utilize additional memory, they do not report key parameters such as power consumption and operating frequency. In general, LUT- and slice register-based implementations provide faster access and lower power consumption compared to memory-based approaches.

In summary, the AM-based LIF neuron model demonstrates strong performance, particularly in energy efficiency and resource utilization. This enables SNNs implemented with this model to operate on resource-constrained hardware platforms or scale to larger networks while maintaining low power consumption.

## VI. Conclusion

This paper presents a novel design for the LIF neuron model. The proposed model has been tested and evaluated at multiple levels, including the multiplier level, single neuron level, neural network level, and also implemented on FPGA in RTL with appropriate tests and evaluations. In this design, the AM method achieves high accuracy, with a low error rate of 2.6099%. Furthermore, the AM-based LIF neuron model preserves precision with minimal timing errors, quantified

TABLE II
OVERALL COMPARISON OF THE AM-BASED LIF NEURON MODEL AND PRIOR DESIGNS

| | **TCAS1'2021** [13] | **NEUROCOMPUTING'20** [19] | **TCAS1'2016** [20] | **TCAS1'2014** [21] | **TBioCAS'2020** [22] | **This Work** |
|---|---|---|---|---|---|---|
| **Model** | **LIF** | **LIF** | **AdEx LIF** | **AdEx LIF** | **AdEx LIF** | **Modified LIF** |
| FPGA Platform | Virtex-7 | NR | Spartan-6 | Virtex-2 pro | Zynq UltraScale+ | Zynq UltraScale+ |
| FPGA Technology | 28 nm | NR | 45 nm | 130 nm | 16 nm | 16 nm |
| Bit-width | 16-bit fixed-point | NR | 20-bit fixed-point | 37-bit fixed-point | 10-bit fixed-point | 16-bit fixed-point |
| Neuron number | 512 | 1 | 1 | 1 | 1 | 1 |
| Slice Registers | 296 | 135 | 829 | 388 | 292 | **112** |
| Slice LUTs | 314 | 218 | 1221 | 1279 | 363 | **200** |
| DSP block | 4 | NR | **0** | **0** | NR | **0** |
| Memory | 1003 $\times$ 64 b | 15 kb | NR | **0** | NR | **0** |
| Maximum Operating Frequency | NR | NR | 134.4 MHz | 190 MHz | **500 MHz** | 267.38 MHz |
| Pipeline | 8 | 2 | NR | NR | NR | 4 |
| Latency | NR | NR | NR | NR | NR | **14.96 ns** |
| Power | NR | NR | NR | NR | 231 mW (500 MHz) | **4 mW(100 MHz)** |
| Energy Consumption | NR | NR | NR | NR | 462 pJ/MPI | **44.88 pJ/MPI** |
| Energy Efficiency | NR | NR | NR | NR | 2.16 GOPS/W | **22.28 GOPS/W** |

NR means not reported; MPI means Membrane Potential Iteration

by ERRT and NRMSD, thereby exerting no adverse effects on SNNs training. Compared to state-of-the-art LIF neuron models, the FPGA implementation of the AM-based LIF neuron model not only achieves significantly lower hardware resource utilization and minimal per-neuron implementation area, but also demonstrates exceptionally low power consumption. Specifically, it achieves a 17.04% to 86.49% reduction in slice register utilization, an 8.26% to 84.36% reduction in LUT utilization, a 90.29% decrease in energy consumption per neuron iteration, and a 10.31-fold improvement in energy efficiency.

## REFERENCES


[1] Z. Mei *et al.*, "TEA-S: A Tiny and Efficient Architecture for PLAC-Based Softmax in Transformers," *IEEE Trans. Circuits Syst. II-Express Briefs*, vol. 70, no. 9, pp. 3594-3598, April 2023.

[2] Q. Al-Taai *et al.*, "Towards an excitable microwave spike generator for future neuromorphic computing," in *2021 16th European Microwave Integrated Circuits Conference(EuMIC)*, 2022, pp. 386-389.

[3] W. Maass, "Networks of spiking neurons: The third generation of neural network models," *Neural Netw.*, vol. 10, no. 9, pp. 1659-1671, Dec. 1997.

[4] Y. Liu *et al.*, "FPGA-NHAP: A General FPGA-Based Neuromorphic Hardware Acceleration Platform With High Speed and Low Power," *IEEE Trans. Circuits Syst. I-Regul. Pap.*, vol. 69, no. 6, pp. 2553-2566, Mar. 2022.

[5] P. Michael and P. Thomas, "Deep Learning With Spiking Neurons: Opportunities and Challenges," *Front. Neurosci.*, vol. 12, pp. 409662, Oct. 2018.

[6] J. L. Lobo *et al.*, "Spiking Neural Networks and Online Learning: An Overview and Perspectives," *Neural Netw.*, vol. 121, pp. 88-100, July 2019.

[7] W. Gerstner and W. M. Kistler, *Spiking Neuron Models: Single Neurons, Populations, Plasticity.* Cambrisge, U.K.: Cambridge Univ. Press, 2002.

[8] A. L. Hodgkin and A. F. Huxley, "A quantitative description of membrane current and its application to conduction and excitation in nerve," *J. Physiol.*, vol. 117, no. 4, pp. 500-544, Aug. 1952.

[9] E.M. Izhikevich, "Which model to use for cortical spiking neurons?" *IEEE Trans. Neural Netw.*, vol. 15, no. 5, pp. 1063-1070, Sept. 2004.

[10] S. Li *et al.*, "FEAS: A Faster Event-driven Accelerator Supporting Inhibitory Spiking Neural Network," in *2021 12th International Symposium on Parallel Architectures, Algorithms and Programming*, 2021, pp. 14-18.

[11] W. Ye, Y. Chen and Y. Liu "The Implementation and Optimization of Neuromorphic Hardware for Supporting Spiking Neural Networks With MLP and CNN Topologies," *IEEE Trans. Comput-Aided Des. Integr. Circuits Syst.*, vol. 42, no. 2, pp. 448-461, Feb. 2023.

[12] E. Z. Farsa *et al.*, "A Low-Cost High-Speed Neuromorphic Hardware Based on Spiking Neural Network," *IEEE Trans. Circuits Syst. II-Express Briefs*, vol. 66, no. 9, pp. 1582-1586, Sept. 2019.

[13] J. Kim *et al.*, "Hardware-Efficient Emulation of Leaky Integrate-and-Fire Model Using Template-Scaling-Based Exponential Function Approximation," *IEEE Trans. Circuits Syst. I-Regul. Pap.*, vol. 68, no. 1, pp. 350-362, Jan. 2021.

[14] J. Wu *et al.*, "Efficient Design of Spiking Neural Network With STDP Learning Based on Fast CORDIC," *IEEE Trans. Circuits Syst. I-Regul. Pap.*, vol. 68, no. 6, pp. 2522-2534, June 2021.

[15] C. Walden *et al.*, "Monolithically Integrating Non-Volatile Main Memory over the Last-Level Cache," *ACM Trans. Archit. Code Optim.*, vol. 18, no. 4, pp. 1544-3566, July. 2021.

[16] J. N. Mitchell, "Computer Multiplication and Division Using Binary Logarithms," *IRE Transactions on Electronic Computers*, vol. EC-11, no. 4, pp. 512-517, Aug. 1962.

[17] X. Wu *et al.*, "Design of Energy Efficient Logarithmic Approximate Multiplier," in *2023 5th International Conference on Circuits and Systems (ICCS)*, 2023, pp. 129-134.

[18] M. Heidarpur *et al.*, "CORDIC-SNN: On-FPGA STDP learning with Izhikevich neurons," *IEEE Trans. Circuits Syst. I: Regular Papers*, vol. 66, no. 7, pp. 2651-2661, July 2019.

[19] F.Perez-Pena, M. A. Cifredo-Chacon, and A. Quiros-Olozabal, "Digital neuromorphic real-time platform," *Neurocomputing*, vol. 371, pp. 91-99, Jan. 2020.

[20] M. Heidarpour, A. Ahmadi and R. Rashidzadeh, "A CORDIC Based Digital Hardware for Adaptive Exponential Integrate and Fire Neuron," *IEEE Trans. Circuits Syst. I-Regul. Pap.*, vol. 63, no. 11, pp. 1986-1996, Nov. 2016.

[21] S. Gomar, A. Ahmadi, "Digital multiplierless implementation of biological adaptive-exponential neuron model," *IEEE Trans. Circuits Syst. I-Regul. Pap.*, vol. 61, no. 4, pp. 1206-1219, April 2014.

[22] S. Xiao *et al.*, "Low-Cost Adaptive Exponential Integrate-and-Fire Neuron Using Stochastic Computing," *IEEE Trans. Biomed. Circuits Syst.*, vol. 14, no. 5, pp. 942-950, Oct. 2020.