An On-Chip Trainable Neuron Circuit for SFQ-Based Spiking Neural Networks

Beyza Zeynep Ucpinar, Mustafa Altay Karamuftuoglu, Sasan Razmkhah, and Massoud Pedram, *Fellow, IEEE*

***Abstract*—We present an on-chip trainable neuron circuit. Our proposed circuit aims at bio-inspired spike-based time-dependent data computation for training spiking neural networks (SNN). The thresholds of neurons can be increased or decreased depending on the desired application-specific spike generation rate. This mechanism is scalable and provides us with a flexible circuit structure design. We simulated the trainable neuron structure under different operating scenarios with thermal noise included. The circuits are designed and optimized for the MIT LL SFQ5ee fabrication process. For a 16-input neuron with four different threshold values, all of the circuit parameter margins are above 20% (±10%) with a 3G sample per second throughput.**

***Index Terms*—Adjustable neuron, on-chip training, SFQ, spiking neural network.**

I. Introduction

NEUROMORPHIC computing is the foundation of deep learning and artificial intelligence (AI) that draws inspiration from the structure and functioning of the human brain [1]. Deep neural networks (DNNs) have proven to be an excellent model for learning systems. However, the training of such networks can be cumbersome. One class of DNNs, known as the Spiking neural networks (SNN), are similar in function to the biological brain regarding their learning style and use of discrete spikes for information transfer between neurons [2], [3]. A class of superconductor circuits, single flux quantum (SFQ) logic [4], [5], also uses discrete spike-like pulses for computing. Therefore, it achieves with orders of magnitude lower power and higher speed than state-of-the-art CMOS. Hence, SFQ is a good candidate for implementing SNN architecture.

Traditional Artificial Neural Networks (ANN) primarily rely on continuous-valued activation to perform neuromorphic computing. However, the human brain, comprising billions of interconnected neurons communicating through synapses, employs discrete spikes or pulses for communication [6]. SNN adopts a more brain-like approach by adopting pulsed communication. These spiking pulses encode the timing and frequency of neuronal activation, allowing for more biologically plausible computations [7]. One remarkable aspect of neuromorphic computing is its event-driven processing methodology. Unlike conventional computing systems that process data continuously, neuromorphic systems respond exclusively to significant changes or events in the input data. This event-driven approach significantly reduces overall computation time, resulting in exceptional energy efficiency. Despite these advantages [8], [9], SNN training[10], [11], especially on-chip, remains a challenge. Reference [12] shows a useful resource for SNN training called snnTorch, an extension of the PyTorch framework. On-chip training of SNNs yields lower accuracy than ANNs due to its forward learning approach.

The neuron is the core part of neural networks [13]. A neuron circuit includes an accumulator and a threshold unit called soma. When the summation of the inputs exceeds the threshold value, the neuron fires and generates an SFQ pulse; otherwise, the neuron remains silent as described in (1). There are several works implementing neuron circuit design with superconductors, primarily focused on inference applications[14], [15]. However, on-chip trainability, especially for SNN, is an important feature that needs to be developed.

$$\mathcal{O}_i = \begin{cases} 1, & \sum_{k=1}^{N} w_k \times x_k \geq T_i \\ 0, & \sum_{k=1}^{N} w_k \times x_k < T_i \end{cases} \tag{1}$$

where the w_k is the weight parameter, x_k denotes the input value of the i^{th} neuron, and T_i is the neuron's threshold value. Depending on the application, the threshold value can vary and take different values. Neurons become more or less sensitive to inputs based on their threshold value. In conventional SNNs, the threshold value is kept constant, which can limit network performance. However, adapting the threshold value can substantially enhance SNN's accuracy [16].

Adjusting a neuron's threshold value provides many benefits [17], [18], such as flexibility and adaptability to the network. Different stages in a neural network may require different thresholds for varying signal sensitivity. By adjusting the threshold value of individual neurons, neurons within the same layers, or neurons within the same kernels, the network's behavior can be finely tuned to specific input patterns. Diehl et al. [19] claim that an adaptive membrane threshold mechanism must be employed to prevent single neurons from dominating the response pattern.

Manuscript received 26 September 2023; revised 20 November 2023, 19 December 2023, and 14 January 2024; accepted 16 January 2024. Date of publication 5 February 2024; date of current version 26 February 2024. This work was supported by National Science Foundation through the DISCoVER Expedition under Grant 2124453. *(Corresponding author: Beyza Zeynep Ucpinar.)*

The authors are with the Ming Hsieh Department of Electrical and Computer Engineering, University of Southern California, Los Angeles, CA 90007 USA (e-mail: ucpinar@usc.edu; karamuft@usc.edu; razmkhah@usc.edu; pedram@usc.edu).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TASC.2024.3359164.

Digital Object Identifier 10.1109/TASC.2024.3359164

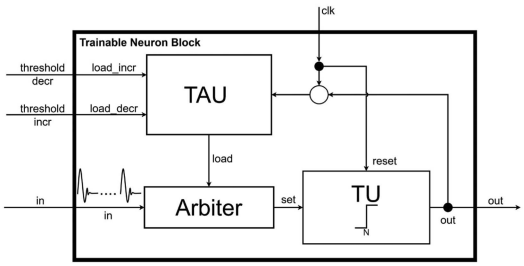

Fig. 1. Neuron Circuit Block Diagram with TAU, TU, and arbiter. The TA has increment and decrement pins that adjust the load value. The arbiter then applies this load value with the input signals to the TU, generating the output. Each output triggers the TAU to reload the data to the arbiter.

The authors increased the accuracy of the proposed architecture from 93% to 95%.

Zhong et al. [20] shows the occurrence of overfitting can be effectively suppressed by using an adaptive threshold. The adaptive threshold reduces the number of excitation pulses. Hence, it is advantageous when optimizing the energy consumption of the chip. The authors report 96% accuracy on the MNIST dataset. Shaban et al. [21] propose an adaptive threshold neuron method with fast convergence, high accuracy, and flexibility. They reached 96.1% accuracy on the SMIST dataset. Chen et al. [22] claim that the ratio of threshold to weights (RTTW), the balance between weight and threshold values, affects the accuracy. With the adaptive threshold method, they achieved 93.93% accuracy. Having appropriate threshold values can enable the reduction of unnecessary computations, leading to lower dynamic power consumption of the neural network on the chip.

There are several SFQ-based spiking element implementations. [23], [24], [25]. The works presented in [26] and [27] mainly focus on the characteristics of superconductor spiking neurons. In [28], fan-in and fan-out limitations of the Spiking Neural Network were studied. Also, [29], [30] demonstrates bio-inspired neuron implementations based on nanowires.

This work introduces a novel feature for SNNs: adjustable neuron thresholds. These thresholds can be modified individually during training or for specific inference networks, ensuring high-margin values. This structure also allows neurons to handle excitatory and inhibitory inputs while changing the load value of the neuron. The adjustable threshold structure has a footprint of $120 \times 90\ \mu m^2$ for the Threshold Adjustment Unit (TAU) and $60 \times 30\ \mu m^2$ for each Threshold Unit (TU). The threshold adjustment time is only 40 ps due to the circuit's synchronous nature.

II. Methodology

The proposed neuron circuit consists of TAU, TU, and Arbiter as seen in in Fig. 1. TU determines the uppermost achievable threshold value, a significant parameter in the system which is an even number. The TAU loads the initial data depending on the desired threshold value and changes the ground state of the TU. This operation is akin to introducing a bias level into a system. The TAU has the increment and decrement inputs and generates the load data for changing the TU's internal states and the system's threshold value. The Arbiter unit merges the load data from the TAU and the input data. Essentially, the Arbiter unit combines the load and input data and subsequently conveys this combined information to the set input of the TU. The distinctive characteristic of the Arbiter is its capacity to merge data without causing any loss of SFQ pulses, addressing a critical concern in the system's performance and reliability.

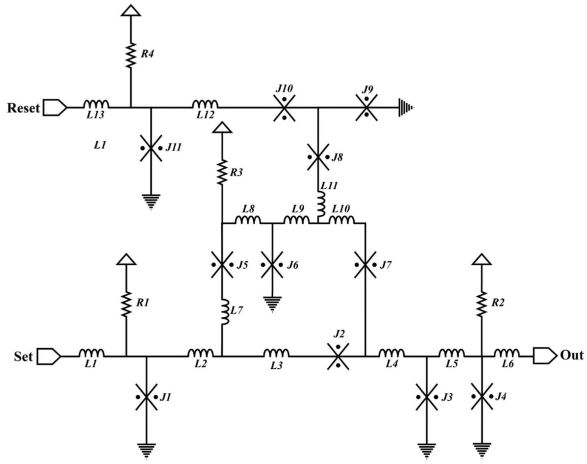

Fig. 2. Schematic of single TU unit.(L1 = 2.098 pH, L2 = 2.1967 pH, L3 = 1.11 pH, L4 = 1.51 pH, L5 = 5.45 pH, L6 = 2.23 pH, L7 = 2.11 pH, L8 = 1.3 pH, L9 = 4.26 pH, L10 = 1.58 pH, L11 = 1.4 pH, L12 = 2.47 pH, L13 = 3.25 pH, J1 = 181 μA, J2 = 104 μA, J3 = 97 μA, J4 = 200 μA, J5 = 106 μA, J6 = 98 μA, J7 = 149 μA, J8 = 100 μA, J9= 100 μA, J10 = 100 μA, J11 = 219 μA, R1 = 16.94 Ω, R2 = 14.33 Ω, R3 = 27.54 Ω, R4 = 14.046 Ω).

The adjusted threshold value is calculated as follows: *Adjusted_Threshold = Max_Threshold − Load.* The hardware determines the *Maximum Threshold Value*, and the *Load Value* comes from the TAU. If the purpose is to change the threshold values layer by layer or kernel by kernel, then the increment and decrement pins of the neurons in the same layer can be connected.

III. Circuit Implementations

A. Thresholding Unit

TU operates asynchronously; hence, it has high-speed characteristics. The structure employed for thresholding is based on Toggle Flip Flop (TFF). TFF acts as a frequency divider and has two states: S1 and S2. The idle state is S1, and the input changes the state from S1 to S2. When the next input comes, it toggles back to the S1 state. The state machine is set to S1 whenever the reset arrives. TFFs are cascadable, and adding each TFF results in twice the frequency division. A single TFF suffices for implementing a threshold of 2, while two TFFs are utilized for achieving a threshold of 4, and so on. Thanks to its asynchronous nature, the throughput is limited by the TFF recovery time, which is over 100 GHz. The schematic of one TU is given in Fig. 2. The circuit has Set and Reset inputs. L3-JJ2-JJ7-L10-L9-L8-JJ5-L7 creates a loop, and when an input pulse comes from the Set input, it will be stored in the loop but won't generate any output pulse. Upon receiving the second pulse, the current starts flowing on L4 and triggers output junctions; this causes an output at the output port. However, when the Reset signal comes, the pulse

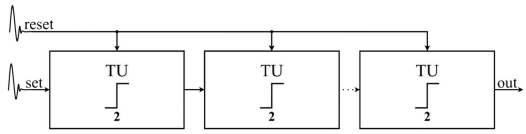

Fig. 3. Threshold unit cascading structure. The threshold unit consists of a series of RTFFs. Adding one RTFF increases the maximum threshold by two.

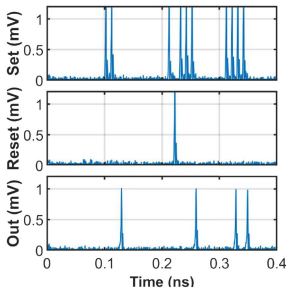

Fig. 4. Simulations result of a TU with one RTFF, which means the threshold value of two.

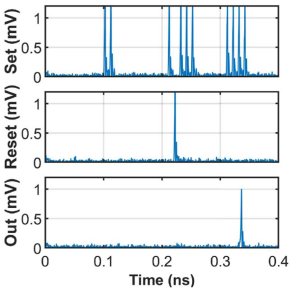

Fig. 5. Simulations result of a TU with two RTFF, which means the threshold value of four.

will enter the superconductor loop and delete the stored current. The circuit has 11 JJs, and its critical margin is measured as 55% by using qCS (quantum Cell Studio) [31] with the optimization techniques in [32]. In simulation and optimization, our chosen tools include JSIM [33], a well-established simulator, and qCS, an optimizer tool that utilizes JSIM as its simulation engine. The latter has been developed as part of the ColdFlux/SuperTools project [34]. The block diagram of the TU and its cascading structure is given in Fig. 3. The reset signal is the same for all TUs. TU was implemented for threshold values of 2 and 4. Simulation results for these values are shown in Figs. 4 and 5, respectively.

From simulation results, we observe that a neuron with a threshold value of '2' fires only if the input equals two or more. Similarly, a neuron with a threshold of '4' fires when the input equals or exceeds four. The reset signal puts the circuits in the ground state.

B. Threshold Adjustment Unit

TAU is responsible for initializing the threshold level by loading a predetermined number of pulses in TU. While the TU hardware determines the maximum threshold, TAU allows

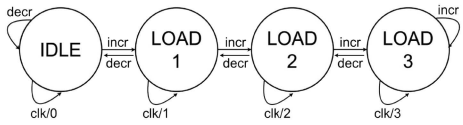

Fig. 6. State Machine of TAU. When the circuit is in an Idle state, the clock signal generates no output, whereas the decrement signal maintains the idle state. Each increment signal advances the machine to a higher state; progressively more SFQ pulses are generated until the last state is reached.

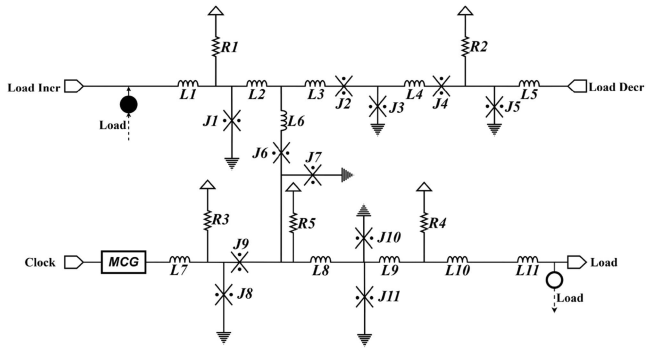

Fig. 7. Schematic of TAU. The schematic of the memory unit is given here. The complete circuit mechanism, including MCG (Multiple Clock Generator) and Local Feedback Wiring, is available in [35]. (L1 = 1.89 pH, L2 = 7.36 pH, L3 = 2.14 pH, L4 = 2.33 pH, L5 = 4.10 pH, L6 = 3.91 pH, L7 =1.73 pH, L8 = 0.66 pH, L9 = 1.71 pH, L10 = 0.45 pH, L11 = 1.84 pH, J1 = 382 μA, J2 = 311 μA, J3 = 365 μA, J4 = 341 μA, J5 = 274 μA, J6 = 158 μA, J7 = 256 μA, J8 = 230 μA, J9= 263 μA, J10 = 372 μA, J11 = 270 μA, R1 = 5.42 Ω, R2 = 7.59 Ω, R3 = 10.34 Ω, R4 = 8.67 Ω, R5 = 9.17 Ω.).

the neuron to have any threshold between 0 and the TU limit. Therefore, this essential component necessitates the storage of multiple SFQ pulses to facilitate the initial data-loading process into the thresholding unit. Furthermore, it requires the data to be provided in a sequential manner. To address these requirements, we employ a multifluxon non-destructive read-out (M-NDRO) circuit.

The proposed M-NDRO unit in [35] can store up to 3 SFQ pulses, offering dedicated increment, decrement, clock, and output ports. Notably, the data stored within the M-NDRO cell remains undisturbed until the increment or decrement signal arrives. The state machine of the TAU is given in Fig. 6. It has four states whereby the increment and decrement signals cause state transitions. When the clock signal arrives, the state machine generates an SFQ pulse in LOAD 1, 2 SFQ pulses in LOAD 2, and 3 SFQ pulses in LOAD 3. Fig. 7 demonstrates the memory unit used in TAU. The memory unit has the storage loop of JJ1-L2-L6-JJ6-JJ7. The loop storage increases with the pulses from the load increment port and decreases with the pulses from the decrement port. The mechanism of the M-NDRO and all circuit components (Local Feedback Wiring, Multiple Clock Generator Unit, etc.) were discussed in detail in [35]. The critical margin is measured as 20% (±10%) by qCS. The limiting part of the TAU is the storage loop, which needs to store up to 3 SFQ pulses. Therefore, the margin is limited to the loop's L and I_c parameters.

Since the circuit has a non-destructive structure, for each threshold value one load operation is sufficient. This attribute

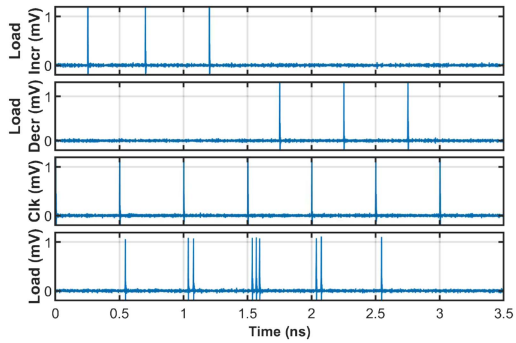

Fig. 8. Simulation result of a TAU demonstrates the increment and decrement function. With the Incr signal, the number of SFQ pulses at each clock increases. The Decr signal will reduce the generated pulses at clock signal arrival.

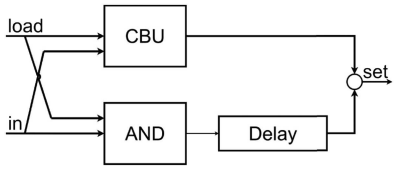

Fig. 9. Arbiter block diagram shows the lossless CBU, which merges the Load and input and sends them to the set signal. When two pulses arrive in a short timing window, the CBU generates just one pulse; however, in that case, asynchronous AND will generate a pulse and apply it to the output.

ensures data preservation for each received clock signal, enhancing the efficiency and reliability of the storage mechanism. The circuit loads the initializing data sequentially, which is required by TU. The simulation result of TAU is shown in Fig. 8. The maximum storage capability of TAU limits the data load. Here, since the M-NDRO can store up to 3 SFQ pulses, the maximum value of the load data is 3. The maximum storage value can be increased by using multiple parallel M-NDRO blocks if needed.

C. Arbiter Circuit

We have designed a novel Arbiter circuit to mitigate the risk of data loss upon precise timing specifications. The block diagram of this circuit is shown in Fig. 9. In this configuration, when both the load data and the input data arrive within a specific time window, the CBU generates one pulse, while concurrently, the asynchronous AND cell yields the other pulse. Subsequently, these two pulses can be combined with the delayed version of the AND cell's output, generating the set data for the TAU. The delay value depends on the maximum number of input pulses and may be adjusted for the input window. This order of the pulses and data flow safeguards against potential data loss, enhancing the reliability and robustness of the system.

IV. Simulation and Results

Circuit simulations were performed under different scenarios to observe the circuit's functionality. In the first simulation reported in Fig. 10, the threshold value is changed from four to three and then switched back to four again. There is no load value in the first two clock cycles, so the threshold value is set

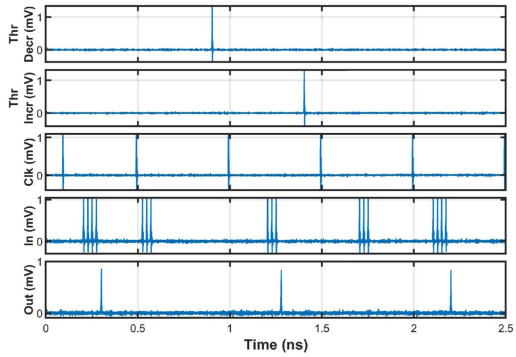

Fig. 10. Simulation result for threshold values four and three. The default value is four. With an incoming Incr pulse, the threshold value becomes three, and with the Decr pulse, the threshold value goes back to four.

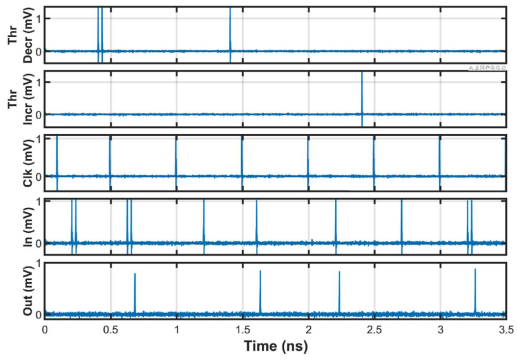

Fig. 11. Simulation result for threshold values two and one.

to its maximum value of four. Therefore, after four SFQ pulses in the first cycle, the neuron fires and generates an SFQ pulse. In the second clock cycle, three SFQ pulses arrive at the TU but cannot exceed the threshold value, so there is no output. Before the third clock cycle, the increment signal generates a load pulse, dropping the threshold to three. Therefore, three input pulses cause an output in the neuron. In the next cycle, the decrement signal arrives and eliminates the load, bringing the threshold value back to four again. In the second simulation shown in Fig. 11, the threshold change is observed from two to one and back to two again. At the first clock cycle, the threshold value of the circuit has the maximum threshold value of four. Hence, giving two SFQ pulses from the input port doesn't trigger the neuron. Before the 2nd clock cycle, two SFQ pulses were given to the increment port to generate two load pulses. After the clock signal, it loads the data and sets the threshold value as two. Because of this, after the second clock signal, an SFQ pulse is generated. At the 4th clock, another increment signal adjusts the neuron's threshold again. It changes the TAU state to LOAD3 and inserts 3 SFQ pulses into the TU. Therefore, the threshold value is set to 1 at this point. One SFQ pulse is enough to trigger the neuron. When the decrement signal arrives, it changes the TAU state from LOAD3 to LOAD2 and again sets the threshold as two. As seen from the simulations, the proposed circuit supports changing the neuron's threshold value repeatedly. In this case, each threshold change takes 40 ps. The simulations adjust the threshold multiple times to observe the

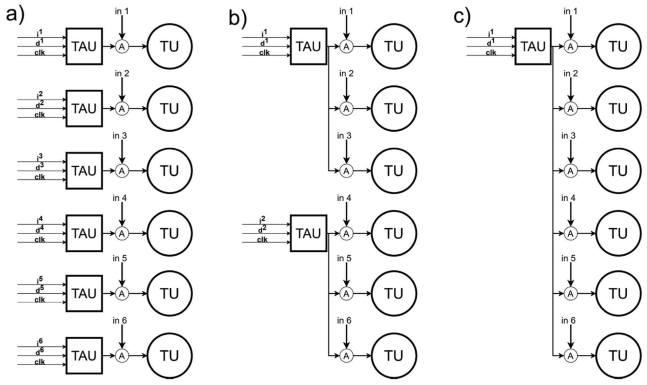

Fig. 12. TAU implementation scenarios. a) Individual TAU for each neuron: increases the flexibility but bulky, b) One TAU block for each kernel size of 3: decreases TAU count and area compared with a, c) One TAU block for each layer: decreases the area, can be used for reconfiguring layers on hardware for inference implementations.

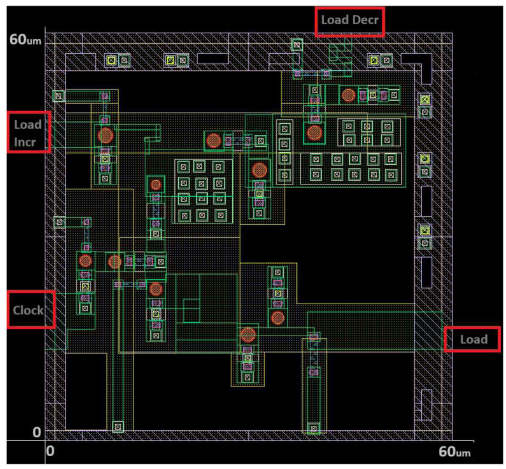

Fig. 13. Multi flux storage Unit Layout of TAU (Multiflux-NDRO). The cell size is $60 \times 60\ \mu m^2$ and is compatible with our existing cell library.

TABLE I
COMPARISON OF MNIST DATASET MAPPING WITH DIFFERENT LAYER THRESHOLDS

Network	Thresholds	Accuracy (%)
96 96 48 10	2 2 2 2	96.231
	4 2 2 2	96.351
	4 4 2 2	96.071
128 96 96 10	2 2 2 2	96.411
	4 2 2 2	97.071
	4 4 2 2	95.951

different scenarios. However, only one initial threshold should be enough for the inference neural network implementations.

Three different usages of TAU unit is given in Fig. 12.

Enabling the trainability of individual neurons requires separate TAU, TU, and Arbiter units for each neuron. This will increase the cost of hardware and in bigger networks will result in more resources than necessary. Instead of tuning threshold values for individual neurons, we suggest modifying thresholds for network layers as in Fig. 12(c) or specific kernels/ blocks shown in Fig. 12(b), collectively. Establishing threshold domains for specific neurons would require only one TAU for a group of neurons. Additionally, employing one threshold unit per layer offers the advantage of flexible threshold reconfiguration at the layer level that can be realized inference due to high speed. This approach will reduce hardware overhead and the accuracy gain from this method is comparable with modifying individual neurons. The inference change of thresholds can also be utilized for resource reuse in large networks [36].

To demonstrate the applicability of the proposed method, we chose the MNIST dataset and applied one TAU per layer. We implemented an inference SNN with three hidden layers using snnTorch [12]. To study the effect, we chose two structures with different fanins, changed the threshold in each layer, and calculated the accuracy. As a result, we observed notable variations in the network performance shown in Table I. In addition to the practical advantages of reducing hardware overhead and enhancing flexibility, these findings also underscore the role of neuron thresholds at each layer and suggest that the refinement and optimization of neuron thresholds in neuromorphic architectures improves accuracy.

Furthermore, the standard cells, including wiring, logic, and memory cells, are already fabricated and tested. Currently, the layout design of Multiflux-NDRO is completed using the MIT LL SFQ5ee process. The layout of the M-NDRO storage part is given in Fig. 13. After verifying the TAU layout, we are planning to fabricate and test the complete on-chip trainable circuit individually and in a small network, where we can test a small dataset, e.g., Iris, using adjustable thresholds [14].

V. CONCLUSION

We presented an on-chip trainable neuron design for spiking neural networks, where the threshold values of the neuron circuit can be increased or decreased at inference, depending on the specific network or its applications. The different units' parameters needed for realizing the circuit with comprehensive simulation results were provided. The threshold adjustment time is 40 ps, and the overall circuit's margins after optimization is $\pm 10\%$. The circuit is scalable and can support adjusting the threshold of a single neuron or multiple neurons at the same time, which increasing the accuracy of the system.

REFERENCES

[1] S. Furber, "Large-scale neuromorphic computing systems," *J. Neural Eng.*, vol. 13, no. 5, Aug. 2016, Art. no. 051001. [Online]. Available: https://dx.doi.org/10.1088/1741-2560/13/5/051001

[2] W. Maass, "Networks of spiking neurons: The third generation of neural network models," *Neural Netw.*, vol. 10, no. 9, pp. 1659–1671, 1997. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0893608097000117

[3] K. Roy, A. R. Jaiswal, and P. Panda, "Towards spike-based machine intelligence with neuromorphic computing," *Nature*, vol. 575, pp. 607–617, 2019. [Online]. Available: https://api.semanticscholar.org/CorpusID:208329736

[4] K. K. Likharev and V. K. Semenov, "RSFQ logic/memory family: A new Josephson-junction technology for sub-terahertz-clock-frequency digital systems," *IEEE Trans. Appl. Supercond.*, vol. 1, no. 1, pp. 3–28, Mar. 1991.

[5] S. Razmkhah and P. Febvre, "Superconducting quantum electronics," in *Beyond-CMOS: State of the Art and Trends*. Hoboken, NJ, USA: Wiley, Jul. 21, 2023, ch. 8, pp. 295–391, doi: 10.1002/9781394228713.ch8.

[6] J.-Q. Yang et al., "Neuromorphic engineering: From biological to spike-based hardware nervous systems," *Adv. Mater.*, vol. 32, 2020, Art. no. 2003610. [Online]. Available: https://api.semanticscholar.org/CorpusID:226287531

[7] H. Hendy and C. E. Merkel, "Review of spike-based neuromorphic computing for brain-inspired vision: Biology, algorithms, and hardware," *J. Electron. Imag.*, vol. 31, 2022, Art. no. 010901. [Online]. Available: https://api.semanticscholar.org/CorpusID:246467172

[8] C.-K. Lin et al., "Programming spiking neural networks on Intel's Loihi," *Computer*, vol. 51, no. 3, pp. 52–61, Mar. 2018.

[9] G. Haessig, A. Cassidy, R. Alvarez, R. Benosman, and G. Orchard, "Spiking optical flow for event-based sensors using IBM's TrueNorth neurosynaptic system," *IEEE Trans. Biomed. Circuits Syst.*, vol. 12, no. 4, pp. 860–870, Aug. 2018.

[10] S. G. Hu, G. C. Qiao, T. P. Chen, Q. Yu, Y. Liu, and L. M. Rong, "Quantized STDP-based online-learning spiking neural network," *Neural Comput. Appl.*, vol. 33, no. 19, pp. 12317–12332, Oct. 2021, doi: 10.1007/s00521-021-05832-y.

[11] Y. Wu, L. Deng, G. Li, J. Zhu, Y. Xie, and L. Shi, "Direct training for spiking neural networks: Faster, larger, better," in *Proc. 33rd AAAI Conf. Artif. Intell., 31st Innov. Appl. Artif. Intell. Conf., 9th AAAI Symp. Educ. Adv. Artif. Intell.*, 2019, Art. no. 162, doi: 10.1609/aaai.v33i01.33011311.

[12] J. K. Eshraghian et al., "Training spiking neural networks using lessons from deep learning," in *Proc. IEEE*, vol. 111, no. 9, pp. 1016–1054, Sep. 2023.

[13] K. Sidiropoulou, E. Pissadaki, and P. Poirazi, "Inside the brain of a neuron," *EMBO Rep.*, vol. 7, pp. 886–892, 2006.

[14] A. Bozbey, M. A. Karamuftuoglu, S. Razmkhah, and M. Ozbayoglu, "Single flux quantum based ultrahigh speed spiking neuromorphic processor architecture," 2020, *arXiv:1812.10354*. [Online]. Available: https://arxiv.org/abs/1812.10354

[15] M. L. Schneider et al., "Energy-efficient single-flux-quantum based neuromorphic computing," in *Proc. IEEE Int. Conf. Rebooting Comput.*, 2017, pp. 1–4.

[16] T. J. Strain, L. J. McDaid, T. M. McGinnity, L. P. Maguire, and H. M. Sayers, "An STDP training algorithm for a spiking neural network with dynamic threshold neurons," *Int. J. Neural Syst.*, vol. 20, no. 06, pp. 463–480, 2010, [Online]. Available: https://doi.org/10.1142/S0129065710002553

[17] D. Horn and M. Usher, "Neural networks with dynamical thresholds," *Phys. Rev. A*, vol. 40, pp. 1036–1044, Jul. 1989. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevA.40.1036

[18] X. Xie, S. Wen, Z. Zeng, and T. Huang, "Memristor-based circuit implementation of pulse-coupled neural network with dynamical threshold generators," *Neurocomputing*, vol. 284, pp. 10–16, 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0925231218300419

[19] P. Diehl and M. Cook, "Unsupervised learning of digit recognition using spike-timing-dependent plasticity," *Front. Comput. Neurosci.*, vol. 9, 2015, Art. no. 99. [Online]. Available: https://www.frontiersin.org/articles/10.3389/fncom.2015.00099

[20] X. Zhong and H. Pan, "A spike neural network model for lateral suppression of spike-timing-dependent plasticity with adaptive threshold," *Appl. Sci.*, vol. 12, no. 12, 2022, Art. no. 5980. [Online]. Available: https://www.mdpi.com/2076-3417/12/12/5980

[21] A. Shaban, S. Bezugam, and D. M. Suri, "An adaptive threshold neuron for recurrent spiking neural networks with nanodevice hardware implementation," *Nature Commun.*, vol. 12, Jul. 2021, Art. no. 4234.

[22] Y. Chen, Y. Mai, R. Feng, and J. Xiao, "An adaptive threshold mechanism for accurate and efficient deep spiking convolutional neural networks," *Neurocomputing*, vol. 469, pp. 189–197, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S092523122101568X

[23] P. Crotty, K. Segall, and D. Schult, "Biologically realistic behaviors from a superconducting neuron model," *IEEE Trans. Appl. Supercond.*, vol. 33, no. 4, Jun. 2023, Art. no. 1800806.

[24] M. L. Schneider et al., "Ultralow power artificial synapses using nanotextured magnetic Josephson junctions," *Sci. Adv.*, vol. 4, no. 1, 2018, Art. no. e1701329. [Online]. Available: https://www.science.org/doi/abs/10.1126/sciadv.1701329

[25] E. Toomey, K. Segall, M. Castellani, M. Colangelo, N. Lynch, and K. K. Berggren, "Superconducting nanowire spiking element for neural networks," *Nano Lett.*, vol. 20, no. 11, pp. 8059–8066, 2020, doi: 10.1021/acs.nanolett.0c03057.

[26] T. Hirose, T. Asai, and Y. Amemiya, "Spiking neuron devices consisting of single-flux-quantum circuits," *Physica C: Supercond. Appl.*, vol. 445–448, pp. 1020–1023, Oct. 2006.

[27] M. A. Karamuftuoglu, A. Bozbey, and S. Razmkhah, "JJ-Soma: Toward a spiking neuromorphic processor architecture," *IEEE Trans. Appl. Supercond.*, vol. 33, no. 8, Nov. 2023, Art. no. 1400607.

[28] M. L. Schneider and K. Segall, "Fan-out and fan-in properties of superconducting neuromorphic circuits," *J. Appl. Phys.*, vol. 128, Dec. 2020, Art. no. 214903.

[29] E. Toomey, K. Segall, and K. K. Berggren, "Design of a power efficient artificial neuron using superconducting nanowires," *Front. Neurosci.*, vol. 13, 2019, Art. no. 933. [Online]. Available: https://www.frontiersin.org/articles/10.3389/fnins.2019.00933

[30] A. E. Schegolev, N. V. Klenov, G. I. Gubochkin, M. Y. Kupriyanov, and I. I. Soloviev, "Bio-inspired design of superconducting spiking neuron and synapse," *Nanomaterials*, vol. 13, no. 14, 2023, Art. no. 2101. [Online]. Available: https://www.mdpi.com/2079-4991/13/14/2101

[31] M. A. Karamuftuoglu, H. Cong, and M. Pedram, "qCS: Quantum cell studio standalone software tool," 2023. [Online]. Available: https://github.com/Karamuft/qCS

[32] M. A. Karamuftuoglu, S. N. Shahsavani, and M. Pedram, "Margin and yield optimization of single flux quantum logic cells using swarm optimization techniques," *IEEE Trans. Appl. Supercond.*, vol. 33, no. 1, Jan. 2023, Art. no. 1300110.

[33] E. S. Fang, "A Josephson integrated circuit simulator (JSIM) for superconductive electronics application," in *Proc. Extended Abstr. Int. Supercond. Electron. Conf.*, 1989.

[34] C. J. Fourie et al., "Results from the ColdFlux superconductor integrated circuit design tool project," *IEEE Trans. Appl. Supercond.*, vol. 33, no. 8, Nov. 2023, Art. no. 1304926.

[35] B. Z. Ucpinar, Y. Kopur, M. A. Karamuftuoglu, S. Razmkhah, and M. Pedram, "Design of a superconducting multiflux non-destructive readout memory unit," 2023, *arXiv:2309.14613*. [Online]. Available: https://arxiv.org/abs/2309.14613

[36] C. Zou et al., "A scatter-and-gather spiking convolutional neural network on a reconfigurable neuromorphic hardware," *Front. Neurosci.*, vol. 15, 2021, Art. no. 694170.