Design of Discrete Hopfield Neural Network Using a Single Flux Quantum Circuit

H. He, Y. Yamanashi, Member, IEEE, N. Yoshikawa, Senior Member, IEEE

Abstract—The superconductor single flux quantum (SFQ) logic family has been recognized as a promising candidate to resolve the energy consumption crisis in the post-Moore era, owing to its high switching speed and low power consumption. In the field of machine learning, where technology and computational requirements are growing rapidly (e.g., image recognition and natural language processing), there is great potential for the implementation of SFQ circuits. In this study, we investigate and implement a discrete Hopfield neural network (DHNN) using SFQ circuits. A DHNN is a binary neural network with less information than a standard full precision neural network; it also provides a higher processing speed. It is mainly used for pattern recognition and recovery. We designed the DHNN circuit with two patterns, each with eight elements. The circuit operates at the clock frequency of more than 50 GHz.

Index Terms—Hopfield neural networks, Discrete Hopfield neural network (DHNN), Single flux quantum (SFQ), Neural computation, Simulation.

I. Introduction

As a type of machine learning, neural networks have achieved impressive success in a wide range of fields, including image recognition/classification, language translation, and speech recognition [1]. A majority of the current neural networks are based on software implementations that utilize the conventional complementary metal oxide semiconductor (CMOS) processor. Therefore, their performance is limited by the inherent serial nature of the software algorithm that requires a large amount of computational power. In this era, we no longer have compelling options to improve the performance of computer systems while maintaining their power and temperature budgets. The development of computationally efficient, low-energy consumption hardware (accelerator) for neural computation has been a long-standing and challenging topic. Therefore, this is the right time to exploit emerging device technologies with great potential and apply them to practical applications.

Among the several candidates for neural network accelerators, the superconductor single flux quantum (SFQ) logic family [2,3] is a promising solution owing to its ultra-fast speed and low power consumption.

Manuscript receipt and acceptance dates will be inserted here. This work was supported in part by JSPS KAKENHI under Grant JP19H01945 and JP 19H05614. *(Corresponding author: Yuki Yamanashi.)*

H. He, Y. Yamanashi, and N. Yoshikawa are with Department of Electrical and Computer Engineering, Yokohama National University, Yokohama, Kanagawa 240-8501, Japan (e-mail: yamanashi-yuki-kr@ynu.ac.jp)

SFQ technology implements a low-level voltage pulse-driven switching that allows extremely fast switching (~10^{-12} s) and low energy consumption (~10^{-19} J per switching). Therefore, using this technology, the clock frequency of the device can be increased by one order of magnitude (i.e., tens of GHz [4, 5]) compared to that of the conventional CMOS circuit. We believe that using the ultra-low power consumption and high-frequency switching characteristics of the SFQ circuits can execute a more efficient accelerator for neural computation (especially for matrix computation).

Numerous studies have used SFQ circuits to implement neural computation. For example, at the circuit element level, magnetic Josephson junction (MJJ)-based synaptic circuits have been proposed [6]. Hardware and algorithm optimization of ANN implementation proposed SC-based DNN using AQFP can achieve up to 6.8×10^4 times higher energy efficiency compared to CMOS-based implementation [7]. At the computer architecture level, SuperNPU, an SFQ-based neural processing unit (NPU) architecture, has been investigated [8]. However, although the former study has demonstrated the low energy characteristics, implementing large-scale circuits has not been reported. The latter study illustrates the potential of SFQ circuits that rivals CMOS circuits in neural computation point paths; however, the computational volume and circuit area are still considered major drawbacks.

In addition to employing hardware design efforts to improve the computational efficiency of neural computing circuits, optimizing the neural computing algorithms at the software (algorithm) level is an extremely important approach. There have been extensive studies on neural network compression and acceleration, such as quantization. In this study, we selected the most representative binary neural network, wherein both the

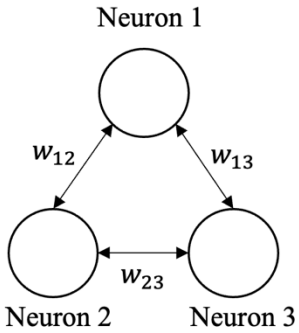

Fig. 1. Hopfield neural network with three neurons.

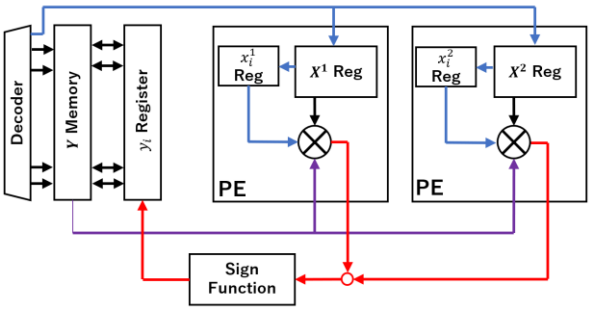

Fig. 2. Overview of the designed DHNN system.

weights and activations are represented by one bit [9]. Binary neural networks utilize approximately 1/32 of the computational units and memory capacity used by a full precision neural network that uses floating-point numbers. Therefore, we designed a binary neural network, the discrete Hopfield neural network (DHNN) [10,11] using SFQ logic circuits. In addition, we combined the use of a systolic processing array that has the potential to scale up to large-scale networks. We designed a DHNN circuit with two patterns, each with eight pixels.

II. Discrete Hopfield Neural Network

In 1982, John Hopfield, together with David Tank, proposed an asynchronous, fully connected neural network model called the Hopfield neural network (HNN). Fig. 1 illustrates a three-neuron HNN. The n-neuron HNN consists of n fully connected neurons with no self-connections. The HNN has two update methods: synchronous and asynchronous. The synchronous update method updates all the weights stored. In contrast, the asynchronous method updates one weight randomly at a time. Neurons in the HNN are also available in both continuous-valued representation (analog value) and discrete-valued representation (binary value). Experimental results have confirmed that the performance of a discrete HNN (DHNN), using asynchronous updates and discrete-valued representations, is superior to that of other HNNs [12,13].

The DHNN contains two information sets, a storage pattern for storing information—that is, the state of the neuron—and a retrieve pattern for recovery. In the n-neuron DHNN, each neuron carries binary information of the storage pattern that is connected to every other neuron through a symmetric weight matrix **W**; **W** indicates the connection between the neurons. Each neuron is connected to all other n-1 neurons except itself. The weights w_{ij} are the interconnections between the i-th and j-th neurons. The network has symmetric weights. The absence of self-connection avoids permanent feedback of its own value.

Storage patterns X^u are n-dimension vectors $\mathbf{X^u} = \{x_1^u, \dots, x_n^u\}$, where $x_i^u \in \{-1, 1\}$. Retrieve patterns are binary vectors of n dimensions $\mathbf{Y} = \{y_1, \dots, y_n\}, y_i \in \{-1, 1\}$. **W** represents the product between each neuron of the storage pattern, that is, the connection between each neuron. The *i*-th row and *j*-th column element of the *u*-th weight matrix **W** is expressed as

TABLE I
Truth Table for Computing h_i

x_i	$Y \cdot X$	h_i
0	0	1
0	1	0
1	0	0
1	1	1

$$W_{ij}^u = x_i^u * x_j^u. \quad (1)$$

If there are many storage patterns, then the total weight is the sum of the weights calculated for each storage pattern, and it is represented as

$$\mathbf{W_i} = \mathbf{W}^1 + \mathbf{W}^2 + \cdots + \mathbf{W}^n. \quad (2)$$

The retrieve pattern updates the state by interacting with the weights based on Hebb's rule [14]. The update steps are as follows:

1. Randomly select an element of retrieve pattern y_i
2. Compute the inner product h_i of the retrieve pattern and the i-th row of the weight W_i

$$h_i = \mathbf{Y} * \mathbf{W_i} \quad (3)$$

3. Calculate the sign function sign() of h_i and update y_i to y_i'; y_i' is updated to 1 if $h_i \geq 0$; else, it is updated to −1

$$y_i' = \text{sign}(h_i) \quad (4)$$

4. Repeat steps 1–3 until the retrieve pattern converges to a stable value

$$\forall y_i' = \forall y_i \quad (5)$$

III. Designed DHNN System

Fig. 2 displays an overview of the designed DHNN system that primarily consists of the following blocks:

- A decoder composed of a non-destructive read-out flip-flop with a complementary output (NDROC) tree
- **Y** memory that stores the retrieve pattern **Y**
- y_i register
- Processing element (PE) to compute $\mathbf{Y} * \mathbf{W_i}$ for generating h_i
- A sign function circuit composed of a bit count circuit consisting of toggle flip-flops (TFFs) connected in series to calculate the sign of h_i

The steps to update y_i once using the DHNN system are as follows:

- The decoder is used to read out y_i from the **Y** memory, save y_i to the y register, and reset y_i
- The partial product of h_i is calculated using multiple PEs and input into the sign function to obtain the result

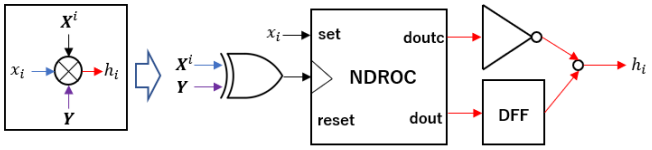

Fig. 3. Gates used to calculate h_i in PE.

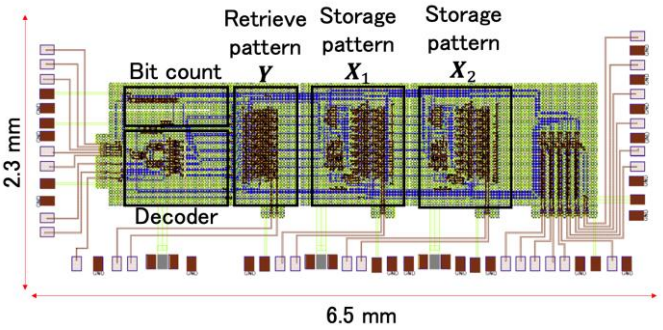

Fig. 4. Layout of the designed DHNN system.

- This result is input back into the **Y** register
- If the result is obtained as 1, y_i is updated to 1; otherwise, y_i keeps the state reset in step 1, wherein it is updated to 0 (this system uses 0 to represent −1)

In Section 2, we introduced the algorithm that determines the updated h_i, the inner product of **Y**, and the *i*-th column of **W**. The *n*-th storage pattern requires at least n^2 clock signals and n gates to operate. In addition, the circuit requires a control circuit for the readout and an $m \times n^2$ weight matrix memory for the m storage patterns. We optimized the algorithm to calculate h_i efficiently using the following equation:

$$h_i = Y * W_i = Y * (x_i * \{x_1, \ldots, x_n\}) = x_i(Y * X). \quad (6)$$

Using Eq. 6, instead of the weight matrices, each storage pattern can be saved to calculate h_i. Therefore, the number of registers used for calculating h_i is reduced from $m \times n^2$ to $m \times n$. The drawback of Eq. (6) is that calculating h_i requires x_i to be copied n times to compute with $Y * X$; this is computed conventionally by a clock generator, thereby causing a large delay. However, this drawback can be resolved by employing an NDROC gate. An NDROC is a non-destructive read-out flip-flop with a complementary output gate. By inputting x_i to the set port of the NDROC, h_i can be computed by inputting $Y * X$ into the clk port. The truth table for computing h_i is displayed in Table I, and the designed circuit is illustrated in Fig. 3.

After optimizing the algorithm to compute h_i, the circuit area of the DHNN system is proportional to the number of elements in the storage patterns (i.e., the number of neurons), which, in turn, is proportional to the memory size of the retrieve patterns. Each retrieve pattern occupies 1/5 of the circuit area of the DHNN system, and the circuit area of the system increases by 1/5 for each additional retrieve pattern. The layout of the designed DHNN system is illustrated in Fig. 4. The designed DHNN system has 2 storage patterns, and each pattern has eight elements (4×2 matrix). We designed the test circuit assuming the use of the AIST 10 kA/cm^2 Nb advanced process 2 (AIST-ADP2) [15] and the CONNECT cell library for the AIST-ADP2

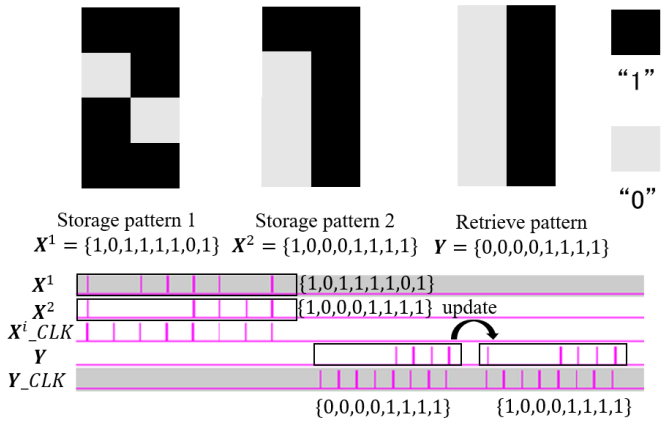

Fig. 5 Simulation result of the designed DHNN circuit. The retrieve pattern Y is converged to the storage pattern 2.

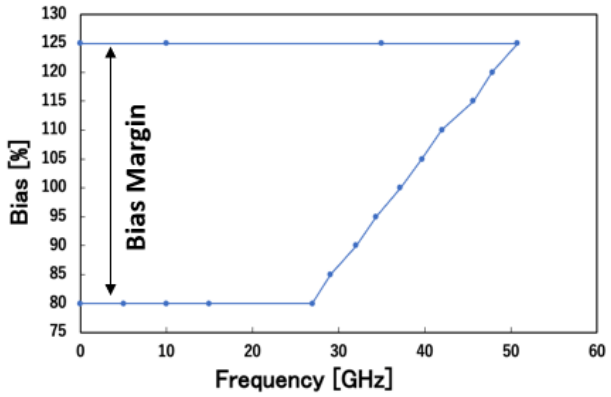

Fig. 6. Characteristics of the bias margin and frequency. The bias is normalized by 2.5 mV, which is the designed bias voltage of the cell library we used [17, 18].

[16, 17]. The area of the circuit is 6.5 mm × 2.3 mm. The number of Josephson junctions in the designed DHNN system is 5426.

We used NC-Verilog to verify the operation of the designed DHNN circuit. One example of the simulation results is presented in Fig. 5. The first element of the retrieve pattern Y is different from that of the storage pattern X^2. The first element y^1 is selected for updating, and after one update step, the retrieve pattern is updated to storage pattern X^2. Fig. 6 illustrates the simulated dependence of the bias margin on the clock frequency. The designed circuit was numerically simulated at up to 50 GHz clock frequencies at the bias voltage of 3.125 mV.

We estimated the dynamic and static power consumption of the designed SFQ DHNN system. Dynamic dissipation is calculated as $P_D = \alpha n I_b \Phi_0 f$, where α is the switching probability of the Josephson junction (JJ) when the clock is input (we assumed α = 0.2), *n* is the number of JJs in the circuit, I_b is the average dc bias current for one JJ (we use I_b ~ 100 μA), Φ_0 is the flux quantum, and f is the clock frequency. Static power dissipation is calculated as $P_S = I_{b\_total} V_b$, where I_{b_total} and

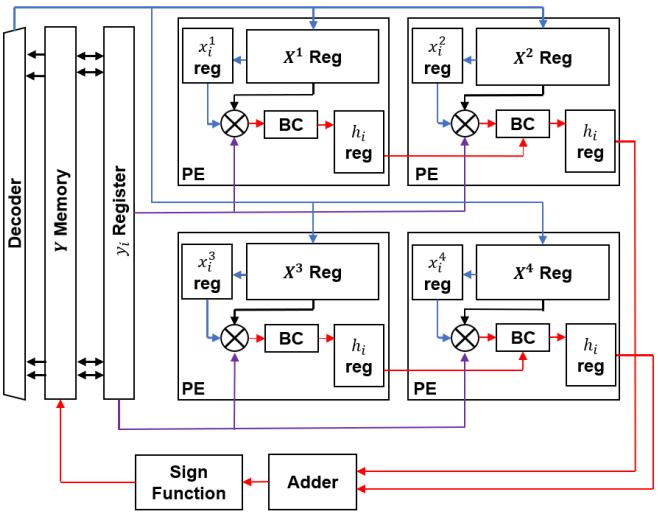

Fig. 7. Improved DHNN system.

V_b are the total bias current and the dc supply voltage (V_b of 2.5 mV is used in our cell library [16, 17]), respectively. Dynamic power dissipation at the frequency of 50 GHz and static power dissipation of the whole designed circuit is 11.2 μW and 1.46 mW, respectively.

According to the digital simulation results, computation time of the whole system for two storage patterns with 8 pixels is 746.1 ps. The time to calculate the h_i per storage pattern is 238.0 ps and the time to calculate each pixel is 29.8 ps. Computation time of the whole system is proportional to the number of pixels and storage patterns.

The designed DHNN circuit could not operate h_i in parallel, because the h_i values are calculated by the PEs sequentially. To resolve this, the bit count circuit used a trigger flip-flop (TFF) cell (T1 cell) that can accumulate the binary stream of information and read and store the state. The result of the accumulation is passed to the register and the next PE; as a result, h_i is calculated in parallel, as illustrated in Fig. 7. The new architecture reduces latency by a factor of number of storage pattern over the original architecture.

IV. Conclusion

We investigated the hardware architecture for the SFQ DHNN system that realizes designing the circuit with a small footprint. We added registers to PE to parallelize the algorithm. Digital circuit simulations indicates that the retrieve pattern is converged to the storage pattern. This means the designed SFQ DHNN system can be applied to image recognition. We designed the DHNN circuit with two storage patterns and eight elements. The circuit area was 6.5 mm × 2.3 mm and the circuit was numerically simulated at up to 50 GHz clock frequencies, assuming the use of the AIST-ADP2.

References

[1] D.A. et al., "Deep Speech 2: End-to-End Speech Recognition in English and Mandarin," *Proc. 33rd Int'l Conf. Machine Learning*, 2016, pp. 173–182.

[2] 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.

[3] D. E. Kirichenko, S. Sarwana and A. F. Kirichenko, "Zero static power dissipation biasing of RSFQ circuits, " *IEEE Trans. Appl. Supercond.*, vol. 21, no. 3, pp. 776–779, Jun.2011.

[4] I. Nagaoka, M. Tanaka, K. Inoue, and A. Fujimaki, "A 48GHz 5.6mW gate-level-pipelined multiplier using single-flux quantum logic," in I*EEE International Solid- State Circuits Conference (ISSCC) 2019*, pp. 460–462.

[5] I. Nagaoka, M. Tanaka, K. Sano, T. Yamashita, A. Fujimaki, and K. Inoue, "Demonstration of an energy-efficient, gate-level-pipelined 100 TOPS/W arithmetic logic unit based on low-voltage rapid single-flux-quantum logic," in *IEEE International Superconductive Electronics Conference (ISEC) 2019*.

[6] Schneider, M.L., Donnelly, C.A., Haygood, I.W. et al. "Synaptic weighting in single flux quantum neuromorphic computing," *Sci Rep.*, vol. 10, 934, Jan. 2020.

[7] R. Cai, A. Ren, O. Chen, N. Liu, C. Ding, X. Qian, J. Han, W. Luo, N. Yoshikawa, and Y. Wang, "A stochastic-computing based deep learning framework using adiabatic quantum-flux-parametron superconducting technology," In *Proceedings of the 46th International Symposium on Computer Architecture (ISCA '19)*. New York, NY, USA, pp. 567–578, 2019.

[8] K. Ishida et al., "SuperNPU: An Extremely Fast Neural Processing Unit Using Superconducting Logic Devices," in *53rd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO) 2020*, pp. 58-72, doi: 10.1109/MICRO50266.2020.00018.

[9] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi, "Xnor-net: Imagenet classification using binary convolutional neural networks," in *European conference on computer vision*, pp. 525–542. 2016.

[10] J. J. Hopfield and D. W. Tank, "Computing with neural circuits- A model." *Science*, vol. 233, no. 4764, pp. 625–633, Aug. 1986.

[11] D. W. Tank and J. J. Hopfield, "Collective computation in neuronlike circuits," *Scientific American*, vol. 257, no. 6 , pp. 104–114, Dec. 1987.

[12] N. Singh and A. Kapoor, "Cloud Hopfield neural network: Analysis and simulation," in *International Conference on Advances in Computing, Communications and Informatics (ICACCI) 2015*, pp. 203-209, doi: 10.1109/ICACCI.2015.7275610.

[13] N. Soni, N. Singh, A. Kapoor and E. K. Sharma, "Face recognition using cloud Hopfield neural network," in *International Conference on Wireless Communications, Signal Processing and Networking (WiSPNET) 2016*, pp. 416-419, doi: 10.1109/WiSPNET.2016.7566167.

[14] S. Haykin, "Neural Network: A comprehensive foundation," Prentice Hall ,1998.

[15] S. Nagasawa, K. Hinode, T. Satoh, M. Hidaka, H. Akaike, A. Fujimaki, N. Yoshikawa, K. Takagi, and N. Takagi, "Nb 9-layer fabrication process for superconducting large-scale SFQ circuits and its process evaluation," *IEICE Trans. Electron.*, vol. E97-C, no. 3, pp. 132–140, Mar. 2014

[16] Yorozu S, Kameda Y, Terai H, Fujimaki A, Yamada T and Tahara, "A single flux quantum standard logic cell library," *Physica C*, Vol. 378–381, no. 2, pp. 1471–1474, Oct. 2002

[17] H. Akaike et al., "Design of single flux quantum cells for a 10-Nb-layer process," *Physica C*, vol. 469, no. 15–20, pp. 1670–1673, Oct. 2009.

[18] Y. Yamanashi et al., "100 GHz Demonstrations Based on the Single-Flux Quantum Cell Library for the 10 kA/cm^2 Nb Multi-Layer Process," *IEICE Trans. Electron.*, vol. E93-C, no. 4, pp. 440–444, Apr. 2010.