SupeRBNN: Randomized Binary Neural Network Using Adiabatic Superconductor Josephson Devices

Zhengang Li
Northeastern University
Boston, USA
li.zhen@northeastern.edu

Geng Yuan
University of Georgia
Athens, USA
geng.yuan@uga.edu

Tomoharu Yamauchi
Tokyo City University
Tokyo, Japan
g2281461@tcu.ac.jp

Zabihi Masoud
Northeastern University
Boston, USA
m.zabihi@northeastern.edu

Yanyue Xie
Northeastern University
Boston, USA
xie.yany@northeastern.edu

Peiyan Dong
Northeastern University
Boston, USA
dong.pe@northeastern.edu

Xulong Tang
University of Pittsburgh
Pittsburgh, USA
tax6@pitt.edu

Nobuyuki Yoshikawa
Yokohama National University
Yokohama, Japan
nyoshi@ynu.ac.jp

Devesh Tiwari
Northeastern University
Boston, USA
d.tiwari@northeastern.edu

Yanzhi Wang
Northeastern University
Boston, USA
yanz.wang@northeastern.edu

Olivia Chen
Tokyo City University
Tokyo, Japan
olivia.chen@ieee.org

ABSTRACT

Adiabatic Quantum-Flux-Parametron (AQFP) is a superconducting logic with extremely high energy efficiency. By employing the distinct polarity of current to denote logic '0' and '1', AQFP devices serve as excellent carriers for binary neural network (BNN) computations. Although recent research has made initial strides toward developing an AQFP-based BNN accelerator, several critical challenges remain, preventing the design from being a comprehensive solution. In this paper, we propose SupeRBNN, an AQFP-based randomized BNN acceleration framework that leverages software-hardware co-optimization to eventually make the AQFP devices a feasible solution for BNN acceleration. Specifically, we investigate the randomized behavior of the AQFP devices and analyze the impact of crossbar size on current attenuation, subsequently formulating the current amplitude into the values suitable for use in BNN computation. To tackle the accumulation problem and improve overall hardware performance, we propose a stochastic computing-based accumulation module and a clocking scheme adjustment-based circuit optimization method. To effectively train the BNN models that are compatible with the distinctive characteristics of AQFP devices, we further propose a novel randomized BNN training solution that utilizes algorithm-hardware co-optimization, enabling simultaneous optimization of hardware configurations. In addition, we propose implementing batch normalization matching and the weight rectified clamp method to further improve the overall performance. We validate our SupeRBNN framework across various datasets and network architectures, comparing it with implementations based on different technologies, including CMOS, ReRAM, and superconducting RSFQ/ERSFQ. Experimental results demonstrate that our design achieves an energy efficiency of approximately 7.8×10^4 times higher than that of the ReRAM-based BNN framework while maintaining a similar level of model accuracy. Furthermore, when compared with superconductor-based counterparts, our framework demonstrates at least two orders of magnitude higher energy efficiency.

CCS CONCEPTS

• **Hardware → Emerging technologies**; • **Computing methodologies → Machine learning**; • **Computer systems organization → Architectures**.

KEYWORDS

Deep Learning, Superconducting, AQFP, BNN, Stochastic Computing

ACM Reference Format:
Zhengang Li, Geng Yuan, Tomoharu Yamauchi, Zabihi Masoud, Yanyue Xie, Peiyan Dong, Xulong Tang, Nobuyuki Yoshikawa, Devesh Tiwari, Yanzhi Wang, and Olivia Chen. 2023. SupeRBNN: Randomized Binary Neural Network Using Adiabatic Superconductor Josephson Devices. In *56th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO '23), October 28–November 01, 2023, Toronto, ON, Canada.* ACM, New York, NY, USA, 15 pages. https://doi.org/10.1145/3613424.3623771

MICRO '23, October 28–November 01, 2023, Toronto, ON, Canada

ACM ISBN 979-8-4007-0329-4/23/10…$15.00
https://doi.org/10.1145/3613424.3623771

1 INTRODUCTION

In recent years, deep learning and deep neural networks (DNNs) have become the core enabler of a broad spectrum of artificial intelligence (AI) applications such as image recognition [22], natural language processing [23], and autonomous driving [6]. However, the high computation and storage demands of DNN executions are still an essential challenge for the democratization of AI.

A significant amount of effort has been dedicated to reducing network energy consumption at the algorithmic level. Recent studies have proposed Binary Neural Networks (BNNs) as a solution [19, 20, 47, 58], which have a 32× smaller memory footprint than conventional DNNs that use 32-bit floating-point precision, despite having a similar network structure. Additionally, BNNs can avoid the tremendous floating-point multiply-accumulation (MAC) operations in conventional DNN models by employing bit-wise exclusive-NOR and popcount logic. In recent years, advancements in BNN training techniques and network structure designs have led to significant improvements in network accuracy [9, 75], making BNN a promising candidate for energy-efficient-oriented designs.

In addition to algorithmic optimizations, significant advancements have been achieved in the hardware domain, with superconducting electronics (SCE) being a prime example. Superconducting logic families, leveraging magnetic flux quantization and quantum interference in Josephson-junction (JJ)-based superconductor loops, have emerged as promising candidates for future computing. The IEEE International Roadmap on Devices and Systems (IRDS) has recognized SCE as one of the top-level roadmaps since 2018 [21, 33]. Among various superconducting logic families, Adiabatic Quantum-Flux-Parametron (AQFP) logic stands out for its exceptional energy efficiency. In 2019, researchers experimentally demonstrated a 1.4 zJ energy dissipation for each operation in AQFP at the device level [67]. On the circuit level, authors in [15] have shown that, compared to state-of-the-art CMOS technology, AQFP can achieve an energy-efficiency gain in the range of $10^4 \sim 10^5$.In addition, research on AQFP design automation has been conducted worldwide, aiming to achieve system-level AQFP circuit design and implementation [11, 12, 60, 71]. Thanks to advancements in the EDA environment for AQFP VLSI design, several successful AQFP chips have been demonstrated [3, 70, 76].

Diverging from previous neural network acceleration efforts focused on RSFQ superconducting logic, such as superNPU [37] and JBNN [27], recent research [77] has recognized the immense potential of integrating BNN with AQFP technology to achieve exceptionally efficient DNN accelerator design, marking an initial endeavor in this direction. In [77], a crossbar synapse array architecture using AQFP devices is proposed, which is a prototype module that intends to efficiently compute the vector-matrix multiplications required for the MAC operation in BNNs. However, this is far from a complete solution to make the AQFP devices feasibly used for BNN acceleration. There are still several critical challenges that need to be addressed.

First of all, the utilization of AQFP devices for constructing crossbar arrays poses a challenge regarding their *randomized behavior issue* (**Challenge #1**). Specifically, when building an AQFP-based crossbar, the accumulated current in the analog domain on each crossbar column will suffer from the *current attenuation* caused by the increasing superconductive inductance as the crossbar size increases. With an attenuated input current, the AQFP buffer may not be able to precisely detect the direction of the input current, resulting in randomized outputs (more details in Section 4.2). Such randomness will introduce inaccuracy in BNN computation, resulting in significant discrepancies between the BNN model that is trained on software and its actual behavior during hardware implementation. Consequently, the accuracy of the network may degrade substantially.

Moreover, due to the current attenuation issue and immature manufacturing technology, the AQFP-based crossbar has limited scalability (**Challenge #2**). This indicates that the size of the crossbar array cannot be arbitrarily large. Multiple crossbar arrays must be employed to accommodate all the weights of a BNN layer or a convolutional filter. However, this will raise another problem: how to effectively accumulate the intermediate results from the corresponding crossbar columns between multiple crossbars (**Challenge #3**). This is not a trivial task since the binary intermediate results are used. Inappropriately addressing this problem can lead to significant accuracy degradation. Last but not least, the hardware configurations such as crossbar size and the threshold current of the AQFP buffer also need to be optimized to deliver the best accuracy while considering the hardware performance (**Challenge #4**).

Due to these critical challenges, we would like to ask whether this is yet another crippled design that has to end with compromised accuracy or hardware performance? Fortunately, the answer is *no*. To overcome these four challenges, we first investigate the randomized behavior of AQFP devices. Then, we propose an AQFP randomized behavior-aware BNN training paradigm, which incorporates the randomized behavior of AQFP buffer by formulating the binarization process of the output feature maps in a probabilistic manner according to the amplitude of the crossbar's output current (**Contribution #1**). We also incorporate the weight-rectified clamp method to help improve randomized BNN training accuracy. After that, we simulate the impact of current attenuation in terms of different crossbar sizes and formulate the current amplitude into the mathematical value used in BNN computation. By doing this, the gap between the BNN model that is trained on software and the model's actual behavior in hardware implementation can be mitigated (**Contribution #2**).

Intriguingly, we find that the unique randomness behavior of AQFP devices is inherently compatible with the stochastic computing (SC) technique. Therefore, to solve the problem of accumulating the intermediate results from the corresponding crossbar columns between multiple crossbars, we propose a novel and efficient SC-based accumulation module circuit to add up the intermediate result as well as improve the model accuracy impacted by the randomized behavior (**Contribution #3**). Since the randomized behavior that appears in the AQPF buffer's output is constrained and dependent on the input current amplitude, it can be seamlessly converted to a stochastic number (SN) via a specific observation window with minimal hardware overhead. Due to the significant influence of hardware configurations on the model accuracy, we propose a comprehensive software-hardware co-optimization. This helps optimize the hardware configurations of AQFP-based BNN accelerator design, such as crossbar synapse array size, stochastic

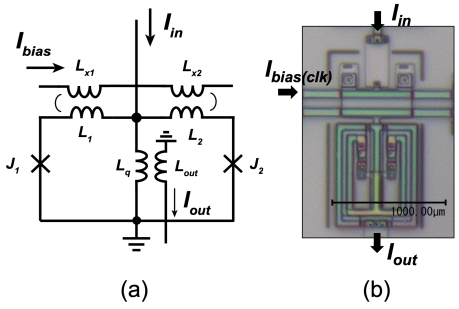

Figure 1: Adiabatic Quantum-Flux-Parametron Logic. (a) Schematic of an AQFP buffer. (b) Microphotograph of the fabricated AQFP buffer using 4-layer niobium process [51].

computing bit-stream length, and "gray-zone" width of AQFP buffer by comprehensively considering power consumption, energy efficiency, and hardware computing error (**Contribution #4**). Besides that, we introduce a batch normalization (BN) matching method to address the floating-point computation problem induced by BN layer with no additional peripheral circuits. And a clocking scheme adjustment-based circuit optimization is also proposed to improve the hardware performance (**Contribution #5**).

To validate the effectiveness of SupeRBNN, a series of detailed comparative experiments are provided. We analyze the accuracy distribution according to multiple hardware configurations and the sensitivity of the relationship between SC bit-stream length with model accuracy. We also compare our method with multiple representative technologies, including CMOS, ReRAM, and RSFQ/ERSFQ on MNIST and CIFAR-10 datasets. SupeRBNN achieves about 7.8×10^4 times higher energy efficiency with a similar model accuracy level compared with the representative ReRAM-based BNN framework on CIFAR-10 dataset.

2 BACKGROUND AND RELATED WORK

2.1 Model Quantization and Binary Neural Network

Model quantization is a crucial technique for DNN inference acceleration. It maps the 32-bit floating-point weight and activation values in a DNN model using fewer bits representation. Existing model quantization research can be categorized according to quantization schemes. Binary neural network (BNN) [19, 20, 47, 58] and ternary neural network (TNN) [32, 81] use extremely low precision for DNN models, and low-bit-width fixed-point neural network [17, 80] quantizes models with the same interval between each quantization level. Among them, with weights constrained to $\{-1, 1\}$, multiplications of BNN can be replaced by additions/subtractions. Additions/subtractions can also be eliminated using XNOR and AND operations if activations are quantized to binary as well. This can significantly reduce operations and simplify hardware implementation, which is ideal for low-power consumption scenarios.

As a pioneer work, Courbariaux et al. [20] first binarized both weights and activations with the sign function. To overcome the almost everywhere zero gradients in the sign function, they incorporated the STE [8] as an approximation to enable the gradient back-propagation. However, the limited representational ability of BNNs leads to a significant drop in accuracy. To mitigate the accuracy drop, XNOR-Net [58] introduces scaling factors obtained from the L_1-norm of the weights or activations to reduce the quantization error. Then, the rotated binary neural network (RBNN) [46] explores and reduces the quantization error by considering the influence of the angular bias between the binarized weight vector and its full-precision version. Later works propose new gradient estimation functions and binarization-friendly network architectures to promote the BNN performance [30, 43, 47, 48, 57, 75, 78].

2.2 AQFP Superconducting Logic

AQFP originates from quantum-flux-parametron (QFP) logic, one among many superconducting logic families, which was first proposed in 1985 [50]. Authors in [66] proposed an adiabatic version of QFP to obtain extremely low energy dissipation by re-parameterizing the device to allow QFP gates to be operated at an adiabatic mode, resulting in roughly 5∼6 orders lower energy dissipation than its CMOS counterpart.

Like many other superconductor-based logic families, AQFP also employs the Josephson Junction (JJ) as the basic switching element to obtain the state transition for logic encoding. The most basic structure of AQFP circuits is the AQFP buffer, which consists of a double-Josephson-Junction SQUID (J_1, J_2) [18], as shown in Figure 1. A minimalist approach has been proposed to create an AQFP cell library containing essential logic gates (e.g., INVERTER, AND, OR, and MAJORITY gates) built from AQFP buffers for digital circuit design [69]. Utilizing different directions (positive and negative) of output current pulses (I_{out}) to represent distinct logic states (0 or 1), the accumulation operation for outputs from various AQFP gates can be efficiently achieved through a straightforward current summation in the analog domain. Moreover, when keeping a high excitation current to an AQFP buffer, the logic state stored in the AQFP buffer can be retained, making it possible to be used as a single-bit memory cell for storing the 1-bit BNN weights. These characteristics render AQFP well-suited for addressing MAC operations in BNNs using a crossbar-based in-memory computing architecture.

However, due to the principle of AQFP buffer, the output current is sensitive to the direction of the input current. When the amplitude of input current is very small, which falls in the "grayzone" ΔI_{in} [25] of an AQFP buffer, the stochastic switching behavior (caused by the thermal or quantum fluctuation) exists in an AQFP buffer will make it hard to detect the direction of the input current, resulting in a randomized output with a probability related to input current, i.e., $0 < P(I_{in}) < 1$. This unique property is a double-edged sword: it introduces inaccuracy but also enables compatibility with stochastic computing.

Diverging from the previous neural network acceleration works [37] targeting RSFQ superconducting logic, recent research [77] proposes a crossbar synapse array architecture designed for implementing BNN models tailored for AQFP logic. However, unresolved issues like current attenuation, limited scalability, and the randomized behavior of AQFP buffers still hinder the true implementation of AQFP-based crossbar array architecture. Our proposed framework addresses and resolves these challenges, making it a feasible solution.

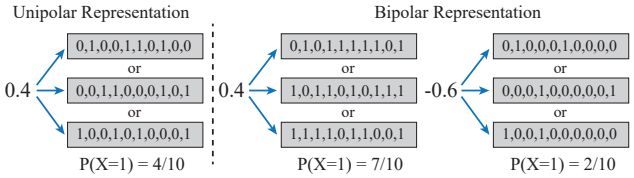

Figure 2: Examples of the unipolar and bipolar representations of stochastic numbers.

2.3 Stochastic Computing

Stochastic computing (SC) is a paradigm that represents a number, named stochastic number (SN), by counting the number of ones in a bitstream. For example, the bitstream 0100110100, containing four ones in a 10-bit stream, represents a real number $x = P_X = 4/10 = 0.4$. (Here we use X to represent the stochastic bitstream, whereas x represents the real value associated with X.) In the bit-stream, each bit is independent and identically distributed (i.i.d.). In addition to the above unipolar encoding format, SC can also represent numbers in the range of [-1, 1] using the bipolar encoding format. Concretely, a real number x is processed by $P(X = 1) = (x+1)/2$. Hence, 0.4 can be represented by 1011011101, as $P(X = 1) = (0.4 + 1)/2 = 7/10$. -0.6 can be represented by 0100100000, with $P(X = 1) = (-0.6 + 1)/2 = 2/10$. Figure 2 shows the examples of different SN representation format.

Recent work SC-AQFP [13] develops AQFP-based DNN acceleration framework trying to use stochastic computing to realize the whole DNN implementation. But it can only work on a very small network for simple tasks (e.g., MNIST) without complex layers (e.g., batch normalization) and requires a pretty large bit-stream length (i.e., 256~2048). Compared with SC-AQFP, our proposed SupeRBNN contributes a new computational paradigm, where stochastic computing is used as a component for the accumulation of intermediate results, which can work on larger DNN and requires smaller bit-stream length (i.e., 16~32).

3 CHALLENGES AND MOTIVATIONS

As mentioned in the introduction, the characteristics of the AQFP buffer well match the needs of computation in BNN models. It is appealing to design the ultra energy-efficient AQFP-based BNN accelerator. Recent work [77] proposes an AQFP-based crossbar synapse array architecture targeting BNN model implementation. This architecture pre-stores BNN weights and deploys XNOR macro inside logic-in-memory cells, which can achieve energy-efficient in-memory computing theoretically. But the randomized behavior of AQFP buffers, current attenuation within the crossbar, the limited hardware scalability, and the hardware configuration problem makes it hard to realize a practical deployment.

Randomized Behavior of AQFP Buffer: Because of the thermal noise and/or quantum fluctuation impact, the output of AQFP buffer presents randomized switching behavior, especially when input current amplitude falls in a certain range, known as a finite "gray-zone" ΔI_{in} [25], in which the AQFP buffer may not be able to precisely detect the direction of the input current. Such a phenomenon introduces in-accuracy in BNN computation and may lead to a degraded network accuracy eventually.

To handle this problem, we first investigate the randomized behavior and simulate this phenomenon within our research scope (4.2K), then incorporate it into our proposed AQFP-aware BNN training algorithm. (Section 4.2 and Section 5.1)

Previous ReRAM and PCM-based work [38] also consider randomness on devices. There are two types of noises considered. Programming Noise and Draft Noise. People usually add a random variable to the original weights to mimic the potential noise/imprecision when mapping the model on different products/hardware, and make a trained model have overall better performance/robustness on different products/hardware. These noises are deterministic after a model is mapped to a specific hardware. They are not data-dependent. On the contrary, the randomized behavior in AQFP devices is data-dependent, which depends on both I_in and hardware configuration for each computation. Therefore, we need to analyze the probability of the intermediate results and incorporate this randomized behavior inside the training algorithm.

Crossbar Current Attenuation and Scalability Problem: When building an AQFP-based crossbar, the accumulated current in the analog domain on each crossbar column will suffer from the current attenuation caused by the increasing superconductive inductance. The relationship between accumulated current amplitude with the mathematical value (the latent activation value in BNN) varies in terms of the crossbar size which increases the randomized behavior in the value domain because the attenuated input current amplitude is more likely to fall into the "gray-zone" of the AQFP buffer. As a result, such randomness in the value domain is intensified when the crossbar size becomes larger. Since excessive current attenuation results in completely randomized output, the AQFP crossbar scalability is limited and it is not able to accommodate all the weights from a BNN layer or a convolutional filter. To overcome the limited scalability of AQFP crossbar, we use multiple crossbars to accumulate the intermediate result of each BNN filter. To mitigate the impact of the current attenuation on BNN computation, we investigate the impact of crossbar size, formulate the current amplitude into the value used in BNN computation, and incorporate the factor of current attenuation into the AQFP-aware BNN training (Section 4.2).

Accumulation of Intermediate Result Problem: In the design of [77], AQFP buffer is used as the neuron circuit of the crossbar (Section 4.1), which functions both a sign operator and an analog-digital-converter (ADC), directly outputting the 1-bit binarized results. This architecture is ultra-energy-efficient but requires one column of the crossbar to contain a whole filter computation in BNN. But as mentioned above, the limited scalability of AQFP crossbar may not satisfy the demand of BNN model and we need to use multiple crossbars to do the intermediate results' accumulation.

To handle the problem of intermediate results' accumulation as well as preserve the accuracy impacted by the AQFP buffer randomized behavior, we design a novel SC-based accumulation module circuit as the output peripheral circuit to add up the intermediate results from each crossbar and convert the stochastic numbers back to 1-bit value as the input of the next layer (Section 4.3).

Hardware Configuration Problem: In general, the crossbar accelerator designs prefer a larger crossbar size and coarse-grained computations to ensure a higher computation throughput and energy efficiency [2, 45, 64]. The AQFP-based design becomes more

complex since the hardware configurations, such as the crossbar size, the "gray-zone" width, the threshold current of the AQFP comparator (illustrated in Section 4.2), and the bit-stream length of SNs not only affect the energy efficiency but also affect the randomized behavior as well as the model accuracy. We need to make optimization to deliver the best accuracy while considering the hardware constraints. Therefore, we propose a comprehensive algorithm-hardware co-optimization for both randomized BNN training and hardware configurations (Section 5.4). To fully leverage the potential of AQFP devices, we also introduce a batch normalization matching method to address the floating-point computation problem induced by BN layer with no additional peripheral circuits (Section 5.4).

4 HARDWARE DESIGN OF AQFP-BASED RANDOMIZED BNN ACCELERATOR

In this section, we first revisit the AQFP-based crossbar synapse array and the corresponding neuron circuit design proposed in [77] (Section 4.1). Then, we make a comprehensive analysis of the randomized behavior of AQFP buffer and crossbar current attenuation and propose our novel designs. In Section 4.2, we explore the impact of crossbar size on the current attenuation and analysis the randomized behavior of AQFP buffer and formulate the current amplitude into the value used in BNN computation. In Section 4.3, we propose a stochastic computing-based accumulation module to accumulate the intermediate computation results from the crossbar columns by considering the randomized outputs from the neuron circuits.

4.1 AQFP-based Crossbar Synapse Array Design for BNN

Although BNNs employ binary weights and activations, they still suffer from significant data movement between the memory and computing units in conventional Von Neumann architectures. This data movement can lead to performance bottlenecks and increased energy consumption. Considering the nature of the AQFP buffer that can be used as a single-bit memory cell and its output current can be easily accumulated in the analog domain, the logic-in-memory (LiM) array-based architecture is used to perform BNNs, employing the in-memory/near-memory computing concept. Figure 3 illustrates the circuit architecture of the AQFP BNNs. The binarized weights are pre-stored in the 1-bit AQFP LiM cells and multiplied by the in-cell XNOR macro. The output of each LiM cell is the multiplication result of the input activation a_i in the i_{th} row of the crossbar and the corresponding pre-stored weight $w_{i,j}$ in the i_{th} row and j_{th} column of the synapse array, and the multiplication result is represneted as $a_i \odot w_{i,j}$. Being different from conventional popcount-based accumulation in BNNs, the crossbar adopts an analog summation approach to add up all the outputs directly since the logic '1' and '0' in AQFP are represented by positive and negative current pulses. The accumulation result represented by the current sum-up of each column in the illustrated synapse array will be sent to the neuron circuits.

As shown in Figure 1, a basic AQFP gate consists of two inductor-Josephson-junction loops L_1-J_1 and L_2-J_2, and the output logic state is denoted by the positive or negative current flowing through the

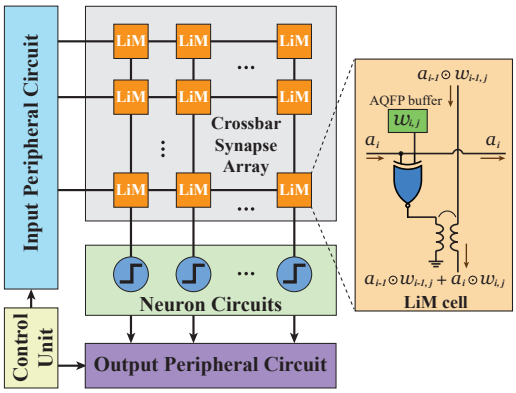

Figure 3: AQFP-based crossbar synapse array circuit architecture.

output inductor L_{out}. Therefore, an AQFP buffer can also serve as a current sensor since AQPF buffers can detect directions/signs of the input current and convert them into '0's or '1's. This unique characteristic makes AQFP naturally able to be used as the analog-digital-converters (ADCs) in BNN since the BNN also requires 1-bit representation for the intermediate computation results such as the output feature maps.

Therefore, the neuron circuit can be simply built by using the AQFP buffers. For a specific column of the crossbar, the output currents of each LiM cell are merged by the magnetic coupling and obtain the accumulated current. Then, depending on the direction of the accumulated current, an AQFP buffer serves as both a sign function and an ADC to binarize and covert the accumulated current into logic state '0' or '1'.

4.2 Randomized Behavior of AQFP buffer and Crossbar Current Attenuation Analysis

Ideally, the neuron circuit in the BNN desires to generate deterministic results to ensure accurate computation. The randomized behavior of AQFP-based neuron circuits may introduce computation inaccuracy, leading to a potential accuracy drop eventually. Therefore, we need an effective way to quantify the randomized behavior, so that we can integrate it into the BNN training process. And with such a randomized-aware trained BNN, the accuracy can be significantly preserved. Moreover, understanding the relationship between the crossbar size and randomized behavior can also help us select appropriate hardware configurations for the implementation.

As we mentioned earlier, the AQFP buffer can serve as a current sensor to detect directions/signs of the input current. However, randomized switching behavior exists in an AQFP comparator when input current amplitude falls in a certain range, known as a finite "gray-zone" ΔI_{in} [25], resulting in a finite output probability $0 < P(I_{in}) < 1$, introduced by the thermal or quantum fluctuation. Quantitative research [73] on the quantum fluctuation effect on Josephson device shows that ΔI_{in} grows at high temperatures due to thermal noise, whereas at $T \rightarrow 0$, it saturates due to quantum fluctuations. Within our research scope (4.2K), we only consider thermal fluctuations as noise sources. Figure 4 shows the output

probability of '1' corresponding to a given input current amplitude in micro-ampere level, where the boundary of randomized switching is around ±2 μA.

The probability of output of a forward current from AQFP buffer can be formulated as:

$$P\left(I_{in}\right) = 0.5 + 0.5\,\mathrm{erf}\left(\sqrt{\pi}\frac{\left(I_{in} - I_{th}\right)}{\Delta I_{in}}\right), \tag{1}$$

where I_{in} is the input current amplitude of the AQFP buffer, which is accumulated through the whole column in the crossbar synapse array. ΔI_{in} means the width of the "gray-zone". I_{th} means the current threshold which can be adjusted manually, and erf $(\cdot)$ is the error function.

To better explore the impact of this randomized behavior on the BNN and quantify it, we conduct an analysis of the crossbar current attenuation.

For the input of the crossbar synapse array, we use $+70\mu A$ and $-70\mu A$ to present value of +1 and −1, respectively. Since the current is added together (merged) in an analog manner via superconductive inductance, the merged current amplitude inevitably attenuates as more inputs in the merging circuits bring larger inductance. As shown in Fig. 5, we measure the degree of current attenuation under different crossbar synapse array sizes. According to the rationale of the current attenuation, it is reasonable to see the amplitude of the output current decrease as the crossbar size increases. Then, we generate a corresponding mathematical fitting curve of it. The curve can be expressed in the form of:

$$I_1(C_s) = A \cdot C_s^{-B}, \tag{2}$$

where, I_1 is the output current amplitude representing the value of 1, C_s is the size of crossbar synapse array, A and B are positive constants of fitting parameters.

In consequence, the current amplitude representing the logic state of '1' in the neural network varies according to the size of the crossbar synapse array. We need to figure out the relationship between the output current amplitude with the presented value and convert the current amplitude to the specific value in the intermediate feature map of the neural network.

To this end, we can convert the probability Equation (1) into the DNN value version:

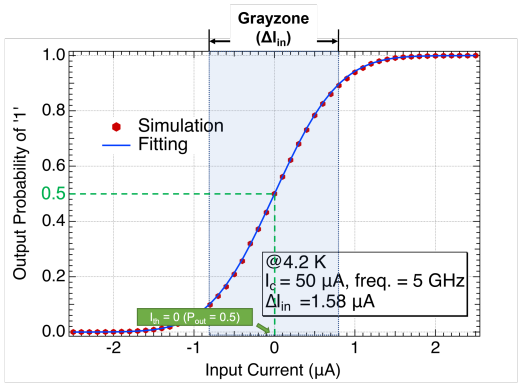

Figure 4: The relationship between output probability of "1" with input current on AQFP buffer.

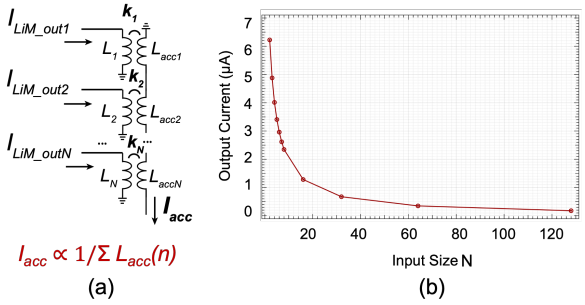

Figure 5: (a) Schematic of analog accumulation circuit. (b) Current Attenuation Curve. The relationship between output current with crossbar synapse array size.

$$P_v\left(V_{in}\right) = 0.5 + 0.5\,\mathrm{erf}\left(\sqrt{\pi}\frac{\left(V_{in} - V_{th}\right)}{\Delta V_{in}(C_s)}\right), \tag{3}$$

where, V_{in} is the mathematical value converted from the input current of AQFP buffer, V_{th} and $\Delta V_{in}(C_s)$ are the counterpart of I_{th} and ΔI_{in}, respectively. $\Delta V_{in}(C_s)$ can be presented as follows:

$$\Delta V_{in}(C_s) = \Delta I_{in}/I_1(C_s). \tag{4}$$

With the DNN value version of the probability expression, we make it possible to consider the AQFP randomized behavior in the BNN training process.

4.3 Stochastic Computing-based Accumulation Module Design

Though the randomized behavior of AQFP buffer is not an ideal property for the neuron circuit design, it also makes the AQFP buffer inherently compatible with the stochastic computing (SC) technique. In SC, the stochastic number (SN) is used to represent the value of a number, which consists of a time-independent bit sequence (as introduced in Section 2.3). Since the randomized behavior that appears in the AQPF buffer presents an output probability dependence on the input current amplitude, which can provide a sufficient level of SNs through a certain observation window with almost no hardware overhead.

For example, as shown in Figure 6 (a), for each clock cycle/phase, an AQFP buffer in the neuron circuit will generate a 1-bit output with the probability of '1' or '0' depending on the accumulated current from the corresponding crossbar column. Using the 1-bit output directly carries a higher risk of being affected by the randomized behavior of the AQFP buffer. But, if we allow a longer observation window for the output of the neuron circuit while keeping the input of the crossbar unchanged, we can obtain an output bit-stream, which is naturally a stochastic number, thanks to the true randomness property of the AQFP buffer [29, 68].

Limited by the crossbar current attenuation property and the hardware manufacture constraints, the crossbar size cannot be arbitrarily large. Therefore, multiple crossbars are needed to accommodate all the weights of the same BNN layer. To make the convolution computation correct, it is required to accumulate the SNs from the same column of different crossbars. And we propose our SC-based accumulation module for the inter-crossbar accumulation. As shown in Figure 6 (b), we choose to use the approximate

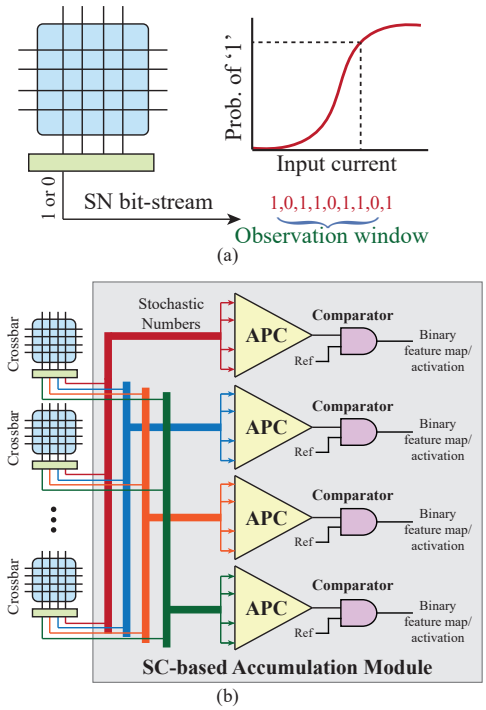

Figure 6: (a) Convert intermediate results to stochastic number through a certrain observation window. (b) Architecture design of stochastic computing-based accumulation module.

parallel counters (APCs) [41] to perform the SN accumulation between different crossbars. The APC counts the number of 1s in the inputs and represents the result with a binary number. This method consumes fewer logic gates compared with the conventional accumulative parallel counter [41, 53]. A comparator is followed by the APC to perform as a step function to generate 1-bit feature map/activation of BNN. Note that all the logic cells/circuits, such as APCs and comparators are designed based on the AQFP standard cell library consisting of all the AQFP logic gates including AND, OR, buffer, inverter, majority, splitter and read-out interfaces. The binary feature maps and activations are represented by positive and negative currents so that they can be directly used as the input for the crossbars for the next level computation.

In general, the larger SN length will result in a higher accuracy of SC, but at the cost of longer computation clock cycles/phases. In our work, we also make the SC bit-stream length one of the dimensions in our algorithm-hardware co-optimization (more details in Section 5.4.2). By incorporating SC and using our SC-based accumulation module, the possible accuracy loss introduced by the AQFP neuron circuit can be efficiently and effectively resolved.

To have a better understanding of where each BNN computational block is implemented, we show the overall matching graph as Fig. 7. The weight matrices in BNN blocks are separated and pre-stored in each AQFP crossbar. The batch normalization is directly converted and matched into the neuron circuits after each crossbar without additional cost (refer to Section 5.2). SC-based accumulation module is used to collect the output of each neuron circuit and generate the intermediate result of each BNN block. The

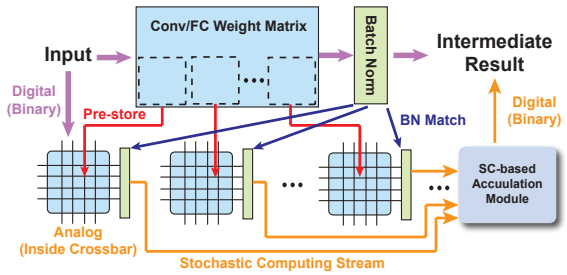

Figure 7: Computation matching from software model (upside) to hardware architecture (downside).

data representations are marked for the corresponding data flows, e.g., Analog, stochastic computing stream, and digital.

4.4 Clocking Scheme Adjustment-based Circuit Optimization

In AQFP, all logic gates are synchronized by a multi-phase clock, facilitating data propagation between adjacent logic stages during a sufficient overlapping window of their respective clock phases. This distinctive characteristic necessitates a minimum of a 3-phase clock system. Current AQFP designs commonly employ a 4-phase clocking system, as it simplifies the testing process: a 4-phase clock can be easily generated using a 2-phase ac with a 90-degree phase difference in conjunction with a dc offset. Due to the synchronization nature of AQFP, numerous buffers must be inserted to ensure that all logic paths are balanced, preventing possible data propagation failure caused by the non-overlap of adjacent logic stages in a typical 4-phase clocking scheme. However, increasing the clock phase for the computing part can significantly reduce the buffers required for path-balancing [61]. This is because the clock phase overlap exists not only in adjacent logic stages but also in non-adjacent stages. Our simulations indicate that the total Josephson junction (JJ) count can be reduced by at least 20.8% and 27.3%, assuming 8-phase and 16-phase clocking, respectively. On the other hand, the buffer-chain-based memory (BCM) employed in this study is achieved in a fully balanced structure without any inserted buffer, and the clock is independent of the computing part. Thus, we propose an alternative approach that involves reducing the number of clock phases in the memory design from the original 4 phases to 3 phases, resulting in a 20% reduction in the total JJ count of the memory component. These simulations demonstrate the significant potential for clock phase adjustment-based component circuit optimization in enhancing the performance and efficiency of AQFP-based computing systems.

5 ALGORITHM AND HARDWARE CO-OPTIMIZATION

5.1 Randomized-aware Binary Neural Network Training

As the existence of the randomized behavior of AQFP buffer, training a BNN normally will lead to a significant performance mismatch between the pure software results and actual implementation on hardware, resulting in severe accuracy degradation. To

mitigate this issue, it is desirable to make the training process of BNN randomized-aware.

Given a DNN, for ease of representation, we simply denote its per-layer real-valued weights as w_r and the inputs as a_r. Then, the convolutional result can be expressed as:

$$Y = \text{CONV}\left(a_r, w_r\right), \tag{5}$$

where CONV $(\cdot)$ represents the standard convolution. For simplicity, we omit the effect of stochastic computing and non-linear operation in this subsection.

Binarized quantization aims to quantizes weight $\beth_r$ and activation $\dashv_r$ to binarized levels, i.e., $\{+1, -1\}$. Following XNORNet [58], given x_r, the corresponding binarized value x_b can be achieved by the sign function:

$$x_b = \text{sign}\left(x_r\right) = \begin{cases} +1, & \text{if } x_r \geq 0, \\ -1, & \text{otherwise}, \end{cases} \tag{6}$$

Taking the AQFP randomized behavior into consideration, each activation value is generated with the value probability function. Different from the conventional BNN quantization, the randomized activation a_b can be presented as:

$$a_b = \text{sign}\left(a_r\right) = \begin{cases} +1, & \text{with probability } P_v\left(a_r\right), \\ -1, & \text{with probability } 1 - P_v\left(a_r\right), \end{cases} \tag{7}$$

To mitigate the large quantization error in DNN binarization, XNOR-Net [58] applies two scaling factors for the quantized weights w_b and activations a_b, respectively. Since weight and activation are multiplied in convolution layers, we can simplify these two scaling factors as one parameter, denoted as α. Then, the binary convolution operation can be formulated as:

$$Y_b = \text{BCONV}\left(a_b, w_b\right) \odot \alpha, \tag{8}$$

where BCONV $(\cdot)$ denotes the binary convolution which includes bit-wise operations XNOR. $\odot$ represents the element-wise multiplication. Here α is set to be a learnable vector that contains independent values for each output channel.

For BNN training, the forward-propagation is expressed by Equation (8) with the binarized values w_b and a_b, while the real-valued w_r and a_r are updated during the back-propagation. However, the gradient of the sign function is an impulse function that breaks the transitivity of the derivative. The back-propagation can not be processed directly. Following STE [8], we compute the approximate gradient of the loss function L, as following:

$$\frac{\partial L}{\partial w_r} = \frac{\partial L}{\partial w_b} \cdot \frac{\partial w_b}{\partial w_r} \approx \frac{\partial L}{\partial w_b}, \tag{9}$$

For the gradient of the activations, since the AQFP probability function has already turned the sign function into the error function, we can leverage this characteristic to achieve the back-propagation instead of using piece-wise polynomial function [49]. Using the expected value of a_b as the approximation, we can have the AQFP randomized-aware back-propagation as follows:

$$\frac{\partial L}{\partial a_r} = \frac{\partial L}{\partial a_b} \cdot \frac{\partial a_b}{\partial a_r} \approx \frac{\partial L}{\partial a_b} \cdot \frac{\partial \mathbb{E}\left(a_b\right)}{\partial a_r}, \tag{10}$$

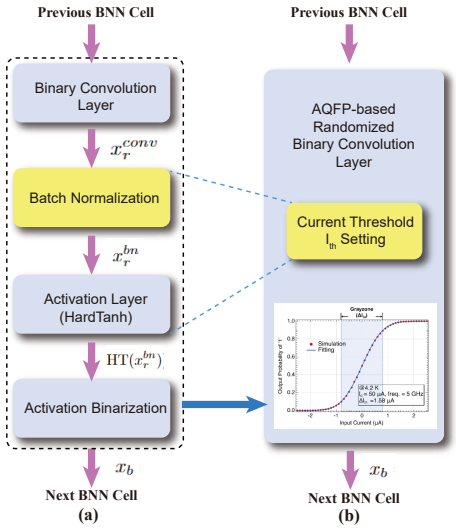

Figure 8: BNN cell architecture. (a) Basic BNN convolution cell, (b) converted AQFP-based randomized BNN convolution cell.

where $\mathbb{E}\left(a_b\right) = \text{erf}\left(\sqrt{\pi}\frac{(a_r - V_{th})}{\Delta V_{in}(C_s)}\right)$

In this way, we implement both forward-propagation and backward-propagation, which achieves the AQFP randomized-aware training.

5.2 Batch Normalization Matching

Batch Normalization (BN) [36] is a DNN layer that normalizes the activation values in the mini-batch during training. Many neural networks use the BN since it is important in stabilizing and accelerating the training process. But BN brings additional floating-point computation in the inference period which causes inefficiency in the BNN implementation on AQFP devices. In this section, we propose an AQFP-aware BN matching technique.

BN can be described by the following equation:

$$Y = \gamma \frac{X - \mu}{\sqrt{\sigma^2 + \epsilon}} + \beta \tag{11}$$

where X and Y are the input and output of BN, and γ, β, μ, and σ stand for weight, bias, mean, and standard deviation, respectively. ϵ is a small constant value to prevent the potential zero in the denominator. γ and β are updated through back-propagation in the training process. μ and σ are updated using a moving average during training but fixed in inference. Note that BN in the inference process becomes a linear transformation, which makes it possible to convert BN into a simple addition operation in BNN and match crossbar synapse array.

As shown in Fig. 8 (a), we use a common BNN cell as an example. The data get through the convolution layer followed by a BN layer and an activation layer (HardTanh), then input into the binarization layer before getting into the next BNN cell. The transferred values before the binarization layer are deformed back to floating-point values x_r^{bn} due to the existence of BN. Given the output values x_r^{conv} for binary convolution layer, the x_r^{bn} for BN layer, and the scaling factor α, the BN can be rewritten as:

$$x_r^{bn} = \gamma \frac{x_r^{conv} \cdot \alpha - \mu}{\sqrt{\sigma^2 + \epsilon}} + \beta \tag{12}$$

The output x_b of the BNN cell can be indicated as:

$$x_b = \text{sign}(\text{HT}(x_r^{bn})) = \begin{cases} +1, & \text{if } x_r \geq 0, \\ -1, & \text{otherwise,} \end{cases} \tag{13}$$

where HT means the activation function HardTanh.

Combined with equation (12) and AQFP probability function, the whole cell can be merged as:

When $\gamma > 0$:

$$x_b = \begin{cases} +1, \text{with probability } P_v(D), \\ -1, \text{with probability } 1 - P_v(D), \end{cases} \tag{14}$$

When $\gamma < 0$:

$$x_b = \begin{cases} +1, \text{with probability } 1 - P_v(D), \\ -1, \text{with probability } P_v(D), \end{cases} \tag{15}$$

where $D = x_r^{conv} + \frac{\beta\sqrt{\sigma^2+\epsilon}}{\gamma \cdot \alpha} - \frac{\mu}{\alpha}$.

Thus, we can achieve a similar activation format as equation (7) by leveraging the current threshold mentioned in Equation (1) with the setting:

$$I_{th} = \left(-\frac{\beta\sqrt{\sigma^2 + \epsilon}}{\gamma \cdot \alpha} + \frac{\mu}{\alpha} \right) \cdot I_1(C_s). \tag{16}$$

As shown in Fig. 8 (b), by adjusting the hardware configuration I_{th} in the AQFP probability function, the whole cell is converted into one randomized binary convolution layer without additional peripheral circuits. If the computation needs to be separated into multiple crossbars with stochastic computing as shown in Fig. 6 (b), we can divide I_{th} evenly and assign them to the corresponding crossbar.

5.3 Weight Rectified Clamp Method

As pointed out in [5, 75], the real-valued weights w_r of a quantized network roughly follow the zero-mean Laplace distribution due to their quantization in the forward propagation. Most weights are gathered around the distribution peak, while many outliers fall into the two tails, far away from the peak. These outliers adversely affect the training of a BNN and slow down the convergence when training BNNs. It is because though the magnitudes of the weights are updated during the back-propagation by gradient descent, the chances of changing their signs are extremely small, which limits the representational ability of BNNs [75].

To revive these outlier weights and promote the BNN training performance, we apply weight rectified clamp method following ReCU [75]:

$$\text{ReCU}(w_r) = \max\left(\min\left(w_r, Q_{(\tau)}\right), Q_{(1-\tau)}\right), \tag{17}$$

where $Q(\tau)$ and $Q(1-\tau)$ denote the τ quantile and $(1-\tau)$ quantile of Weight, respectively.

As proved in ReCU [75], this technique can move the outlier weights towards the distribution peak to increase the probability of changing their signs, which decreases the quantization error, as well as promotes the representational ability of BNNs.

5.4 Hardware Configuration Optimizations

In this section, we optimize the hardware configurations of AQFP-based randomized BNN accelerator design, including crossbar synapse array size, stochastic computing bit-stream length, and "gray-zone" width ΔI_{in} by comprehensively considering power consumption, energy efficiency, and hardware computing error. The computing error mainly comes from two aspects: the average mismatch error AME comes from the output expectation bias of AQFP buffer (see Section 5.4.2); and the stochastic computing error including SN quantization error and random fluctuation [4, 56]. For simplicity, we omit the mathematical deduction of the latter error and use a series of accuracy comparative experiments to directly analyze it (see Section 5.4.1).

5.4.1 Stochastic Computing Bit-stream Length Optimization. We take advantage of the probabilistic behavior that appears in the AQFP buffer to achieve stochastic computing among multiple crossbar synapse arrays. In this process, the stochastic computing bit-stream length is a critical configuration that has a close relationship with model accuracy, inference latency, and power consumption. Generally, a large bit-stream-length leads to better accuracy but suffers from longer inference latency and more power consumption.

To choose a proper bit-stream length, we conduct a series of experiments to explore its behavior on model accuracy. Our general observation is that, as the SC bit-stream length increases (from 1), the model accuracy is improved significantly at the beginning but the accuracy stabilizes after the bit-stream length reaches 16~32. Therefore, using a bit-stream length longer than 32 will not have considerable gain in accuracy. Compared with pure stochastic computing work which generally requires a pretty large bit-stream length, e.g., 512, 1024, to maintain the stability of computation, SupeRBNN reaps benefit from the low demand of bit-stream length, thus achieving fast computation speed. Detailed results can be found in Section 6.3.

5.4.2 Optimization for Width of Gray-zone ΔI_{in} and Crossbar Size.

Generally, given a crossbar size C_s, for the stochastic computing of bipolar signals, the information carried in a stochastic stream of bits X is $x = (2P(X=1) - 1) \cdot C_s = 2P(X) \cdot C_s - \cdot C_s$, where X is the stochastic bitstream, and x represents the real value associated with X ($-C_s \leq x \leq +C_s$). Since the AQFP buffer is used to generate the stochastic number with the probability $P(X=1) = P_v(x)$. the expected value of the carried information $y = (2P_v(x) \cdot C_s - C_s) = \text{erf}\left(\sqrt{\pi}\frac{(x-V_{th})}{\Delta V_{in}(C_s)}\right) \cdot C_s$ does not exactly match the real value x. The nonlinear probability function of AQFP buffer causes an expectation mismatch, which impacts the robustness and accuracy of the model. We show more comparison results in Section 6.

Considering the activation value distribution, the average mismatch error AME can be defined as:

$$\text{AME} = \frac{1}{C_s} \int_{-C_s}^{+C_s} f(x|C_s)\,(x-y)^2 \, \mathrm{d}x, \tag{18}$$

where $f(x|C_s)$ is the probability density function of AQFP-buffer input value x. Early works [5, 79] have shown that the real-valued weight and activation for a quantized model roughly follow a Gaussian distribution. Thus, $f(x)$ can also be approximated as a Gaussian distribution related to C_s, i.e., $f(x|C_s) \sim N(C_s\mu, C_s\sigma^2)$.

We optimize the related hardware configuration ΔI_{in} and crossbar size C_s by minimizing AME. Since C_s is highly related to hardware performance, we first constraint C_s to a range that meets the energy efficiency demand, then adjust both C_s and ΔI_{in} to find the local optimal solution within it. Related comparison experiments are incorporated in Section 6.

6 EXPERIMENTAL RESULTS

In this section, we present optimizations of AQFP hyper-parameters along with comparison results with respect to model accuracy, power consumption, energy efficiency, etc. Finally, we perform thorough optimizations on the overall SupeRBNN to construct the AQFP-aware randomized BNN on both MNIST and CIFAR-10 datasets compared with multiple representative works based on different techniques, including CMOS-based DDN [16] and SyncBNN [27], ReRAM-based IMB [40], RSFQ/ERSFQ-based JBNN [27], and AQFP-based pure stochastic computing work SC-AQFP [13].

6.1 Experiment Setup

AQFP hardware implementation is achieved using a semi-automated design approach that targets the AIST 4-layer 10 kA/cm^2 niobium process (HSTP) [51]. Analog cells and circuits, such as AQFP neurons and merging circuits (analog accumulation), are manually designed at the Josephson-junction (JJ) level. This process takes into account device characteristics and is optimized with superconductor inductance extraction tools. In contrast, logic cells and circuits, such as logic-in-memory cells, stochastic accumulators (APCs), and comparators, are designed using the AQFP standard cell library. This library consists of all AQFP logic gates, including AND, OR, buffer, inverter, majority, splitter, and read-out interfaces. Figure 9 displays the microphotograph of a fabricated 8×8 AQFP crossbar block. The clock/excitation used to drive the entire circuit is a 4-phase sinusoidal current, achieving a 5 GHz clock rate and a 50 ps stage-to-stage delay. By introducing a delay-line (microstripline) based clocking scheme [31], the overall latency is further reduced. This approach effectively increases the total clock phases to 40 by delaying the sinusoidal current by 5 ps between each adjacent logic stage. Circuit-level verification is conducted using a modified version of the Josephson simulator Jsim [24], which accounts for thermal noise. The fabricated 8×8 AQFP crossbar block is further validated at 4.2 K inside a liquid helium Dewar, interfaced with a customized cryogenic probe, as illustrated in the right of Figure 9, which shows the block diagram of the tested system. The setup includes a chip bonded to a ceramic substrate and housed in a cryogenic probe for testing at 4.2 K. Waveform generators provide data and sinusoidal inputs, while DC voltage sources support a 4-phase clocking scheme. Low-noise differential amplifiers amplify output signals for oscilloscope analysis. A 4-layer shield made by Permalloy effectively blocks the external magnetic field. A remote host manages all input-output operations for automated data handling. Except for the chip and probe, all equipment is at

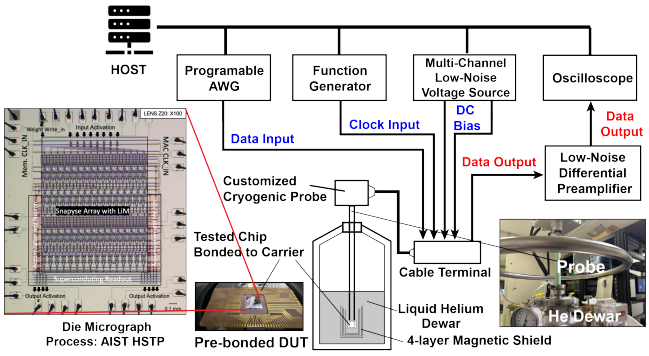

Figure 9: Module validation setup. Left: Die micrograph of a prototype 8×8 crossbar. Right: Block diagram of the tested system.

room temperature. Only low-speed module functionality (100 kHz) has been assessed, with high-speed tests planned.

For the thorough optimizations on CIFAR-10 dataset, SupeRBNN trains from scratch and takes 600 epochs to perform the whole AQFP-aware randomized BNN training with a batch size of 256. The learning rate is initialized as 0.1 and decays with a cosine annealing schedule. SGD [59] is used as the optimizer in the training process. Additional training optimizations, such as warmup and weight rectified clamp method are performed during the training. The number of warmup epochs is 5. And we follow the work [75] to initialize the rectified clamp parameter τ as 0.85, then increase it gradually to the maximum of 0.99 during training.

6.2 Hardware Results of the Proposed AQFP-based Crossbar Synapse Array

Since crossbar synapse array size is a crucial hardware configuration that is highly related to energy efficiency, we first explore the relationship between them.

As shown in Table 1, we present the hardware results of our proposed AQFP-based crossbar synapse array, including the latency, number of JJs, and energy dissipation (per clock cycle) for one crossbar synapse array of different sizes. As the crossbar area increases, all three hardware benchmarks increase but with different growth trends. Given a number of JJs, we can get a range of crossbar sizes that meet our energy efficiency requirement.

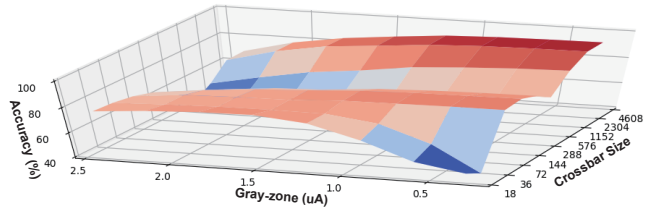

Figure 10: Accuracy distribution in two dimensions of Gray-zone ΔI_{in} and crossbar size. The stochastic bit-stream length used here is 1.

Table 1: Circuit latency, JJ count, and energy dissipation under different crossbar sizes.

Crossbar Area	Latency (ps)	#JJs	Energy Dissipation (aJ)
4×4	60	384	1.92
8×8	120	1152	5.76
16×16	240	3840	19.20
18×18	270	4752	23.76
36×36	540	17280	86.4
72×72	1080	65664	328.32
144×144	2160	255744	1278.72

6.3 Sensitivity Analysis on Relationship between SC Bit-stream Length and Accuracy

To choose a proper bit-stream length, we conduct a series of experiments to explore its behavior on model accuracy. As shown in Fig 11, we use VGG-small training on CIFAR-10 as an example, four different crossbar sizes are incorporated in the comparison. We observe that, as the SC bit-stream length increases, the model accuracy improves a lot at the beginning, but maintains a stable value after the SC bit-stream length reaches 16~32. Keep increasing the SC bit-stream length over 32 will not have considerable accuracy improvements but will result in a longer computing time.

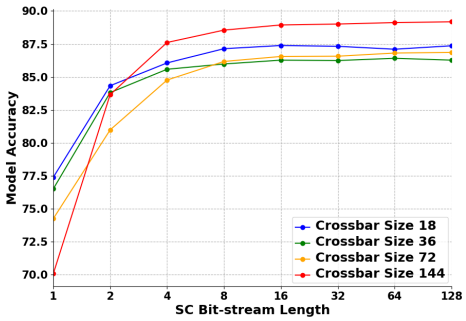

Figure 11: Relationship between SC bit-stream length with model accuracy. VGG-small trained on CIFAR-10 with four different crossbar sizes are deployed in the comparison. The ΔI_{in} is set to be 2.4 μA in this experiment.

6.4 The Overall Comparison among Different ΔI_{in} Crossbar Size Configurations

Here, we do a series of experiments to prove our methodology. VGG-small is used to be trained on CIFAR-10 dataset for these experiments.

Using bit-stream length as 1 for the example, the overall accuracy comparison among different ΔI_{in} and crossbar size is shown in Figure 10, where the x-axis, y-axis, and z-axis represent the values of ΔI_{in}, crossbar size C_s and model accuracy, respectively. As we can see, the accuracy distribution represents a close relationship to both ΔI_{in} and C_s. The growth trend between accuracy with one of the configurations changes largely when another one is modified. This behavior brings in multiple accuracy peaks within the whole accuracy distribution, which matches what we predicted in Section 5.4. Using hardware benchmarks, e.g., energy consumption, efficiency, to constraint crossbar size, a comprehensive optimization can be conducted as mentioned in Section 5.4 within the target distribution area to find the local optimal solution.

6.5 Device Level Comparison with Cryogenic CMOS Technique

In addition to superconducting devices, there has been a notable investigation into cryogenic devices based on CMOS technology, which presents itself as a viable alternative solution. These Cryo-CMOS devices offer the potential to enhance the energy efficiency of computer systems by capitalizing on diminished leakage currents and wire latency. A variety of endeavors have been undertaken in the realm of cryogenic CMOS-based research to bolster the overall performance metrics of hardware infrastructure [1, 7, 52, 55, 62, 63].

In the modern landscape of cryogenic computing, a prevalent objective encompasses the attainment of two distinct low-temperature thresholds, 77K and 4K, achieved by applying Liquid Nitrogen (LN) and Liquid Helium (LHe), respectively. Unlike superconducting computation that thrives at 4K level temperature, 77K temperature is more actively considered for cryogenic CMOS-based design to save the cooling consumption. According to [1, 7, 52, 55, 62, 63], for 77 K, the cooling consumption is approximately 9.65 times the device consumption, and the 77K Cryo-CMOS can achieve about 1.5 times the energy efficiency of the conventional room temperature CMOS.

According to our device level simulation, we observe that lower frequency can generally achieve higher energy efficiency. To make a comprehensive comparison, we test our AQFP-based device under different frequencies from 0.1GHz to 10.0GHz. CMOS-BNN [42] under 1.4MHz and 622MHz, HERMES [39] under 1GHz, CryoBNN [27] under 2.24GHz, and their corresponding Cryo-CMOS counterparts are incorporated in the comparison as shown in Fig, 12. We consider both the energy efficiency with/without cooling consumption in Cryo-CMOS and our AQFP framework. As illustrated in Fig. 12, in contrast to Cryo-CMOS, our approach consistently attains approximately four orders of magnitude superior energy efficiency when solely accounting for device consumption, and achieves a notable enhancement of two to three orders of magnitude in energy efficiency when factoring in cooling consumption.

6.6 Optimization Result

As shown in Table 2, SupeRBNN optimizes the model accuracy under given energy efficiency constraints. For CIFAR-10, we provide the result compared with DDN [16], CMOS-BNN [42], IMB [40] and STT-BNN [54]. DDN is a representative CMOS-based digital accelerator. CMOS-BNN is a BNN accelerator based on 10-nm FinFET CMOS under low frequency, 13MHz (thus has a higher energy efficiency). IMB uses resistive memory crossbar Array (RCA) with RRAM architecture to implement BNN computation. STT-BNN combines spin-transfer torque magnetoresistive Random Access Memory (MRAM) with BNN to improve energy efficiency. Besides these works, recent work [10] explore the low temperature CMOS (77K), which may potentially achieve 1.5 times overall better energy efficiency compared with the conventional room temperature CMOS. For full-precision VGG-small model, DDN can achieve 92.5% top-1 accuracy with 0.45 TOPS/W energy efficiency. Our

Table 2: Model accuracy on Cifar-10 dataset under different energy efficiency constraints. CMOS-BNN [42] has a lower frequency of 13MHz and thus has a relatively higher energy efficiency compared with other CMOS-based work.

Design	Scheme	Accuracy	Energy Efficiency W/O Cooling (TOPS/W)	Energy Efficiency W/ Cooling (TOPS/W)	Power (mW)	Throughput (images/ms)
DDN (VGG-Small) [16]	Full-precision	92.5	0.28	-	-	-
IMB [40]	Binary	87.7	82.6	-	12.5	1.3
STT-BNN [54]	Binary	80.1	311	-	-	-
CMOS-BNN [42]	Binary	92.0	617	-	-	-
Ours (VGG-Small)	Binary	91.7	1.9×10^5	4.8×10^2	6.2×10^{-3}	2.0
Ours (VGG-Small)	Binary	90.6	3.8×10^5	9.5×10^2	6.3×10^{-3}	3.9
Ours (VGG-Small)	Binary	89.2	1.5×10^6	3.8×10^3	6.4×10^{-3}	15.2
Ours (VGG-Small)	Binary	87.4	6.8×10^6	1.7×10^4	7.6×10^{-3}	47.4
Ours (ResNet-18)	Binary	92.2	1.9×10^5	4.8×10^2	6.2×10^{-3}	2.2

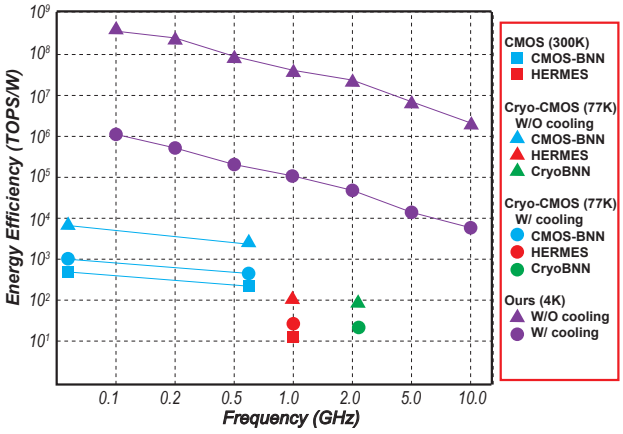

Figure 12: Comparison with room/low-temperature CMOS techniques according to energy efficiency and frequency. Among them, Cryo-CMOS counterpart results of CMOS-BNN [42] and HERMES [39] are based on estimation, Cryogenic result of CryoBNN is from [27].

Table 3: Comparison with RSFQ-JBNN, ERSFQ-JBNN, CMOS-based SyncBNN, SC-AQFP, and our implementation (MLP) on MNIST Dataset.

Design	Accuracy	Energy Efficiency (TOPS/W) W/O Cooling	W/ Cooling
SyncBNN [27]	98.4	36.6	36.6
RSFQ [27]	97.9	2.4×10^3	8.1
ERSFQ [27]	97.9	1.5×10^4	50.0
SC-AQFP [13]	96.9	9.8×10^3	24.5
Ours	98.1	1.5×10^6	3.8×10^3

SupeRBNN achieves 4.2×10^5 times better energy efficiency with a similar level of accuracy (92.2%) on the same model. Compared with IMB with BNN model, SupeRBNN has a much higher frequency 5GHz, 7.8×10^4 times of higher energy efficiency with similar model accuracy. When we loosen the efficiency constraint, SupeRBNN can achieve 91.7% and 92.2% top-1 model accuracy on VGG-small and ResNet-18, respectively, with the energy efficiency of 1.9×10^5 TOPS/W. The cooling cost for typical superconducting digital circuits is about 400× the chip power dissipation [34]. Even considering cryo energy, SupeRBNN still shows 205.8× higher energy efficiency compared to IMB under the same level of accuracy.

To compare with JBNN [27] and SC-AQFP [13], we test our approach on MNIST dataset. As shown in Table 3, SyncBNN, RSFQ, and ERSFQ are CMOS-based, RSFQ-based, and ERSFQ-based BNN accelerators in JBNN paper [27]. SC-AQFP is the AQFP-based pure stochastic computing accelerator. We use the same model architecture (MLP) as shown in JBNN [27]. With the same BNN model, our approach consistently achieves two to four orders of magnitude better energy efficiency compared with CMOS, RSQ, and ERSFQ-based accelerators with similar accuracy. Compared with SC-AQFP, which processes pure stochastic computing on AQFP devices, our approach achieves 153× better energy efficiency for both cooling/non-cooling situations with 2% better top-1 accuracy.

7 DISCUSSION

In addition to AI-focused accelerators, the proposed AQFP technology can also be employed for conventional general-purpose computing to cater to a variety of application scenarios. The AQFP technology boasts a standard cell library designed for different manufacturing processes, such as Japan AIST 4-layer process HSTP [51] and US MIT-LL 8-layer Nb process SFQ5ee [72], presenting a rich assortment of over 80 cells [26]. This includes 3- and 5-input logic gates, signal-driving boosters, and refined interfaces across various superconducting logic families.

The development of a comprehensive EDA toolchain—from logic synthesis to placement and routing—is specifically tailored for this standard cell library. Digital modelling and a synthesis flow for cell-based AQFP structural circuit generation were proposed in 2017, which can be seen as the earliest attempt towards AQFP design automation [74]. This synthesis flow is further tailored to support more AQFP features by different research groups in [28, 35, 71]. For the development of placement and routing, T. Tanaka et al. proposed a framework using a genetic algorithm (GA) for placement and a left-edge channel routing scheme in 2019 [70], whereas Y. Chang et al. proposed another framework adapting a learning-based placer to minimize the runtime overhead in 2020 [14]. H. Li et al. have developed a different tool using a negotiation-based A* router, targeting processes allowing multiple routable metal layers [44]. System-level performance analysis has been successfully conducted using the

aforementioned EDA framework [15]. These results ensure that AQFP technology is compatible with conventional logic and memory design, including but not limited to microprocessor, register file and random-access memory. Moreover, by amplifying superconducting signals to voltage levels, specially designed on-chip interfaces between AQFP and conventional CMOS technologies have been implemented and demonstrated [65]. This paves the way for the system to be employed in broader applications, including supercomputing, cloud computing, and secure computing.

8 CONCLUSION

In this paper, we first make the analysis of the randomized behavior of AQFP buffer and current attenuation feature of AQFP, then propose the randomized-aware BNN training algorithm effectively integrating the randomized behavior into the BNN training process. To solve the intermediate results accumulation problem as well as preserve the model accuracy, we inspiringly convert the randomized output of the neuron circuit to the stochastic computing domain and propose a novel stochastic computing-based accumulation module. Finally, we propose an algorithm-hardware co-optimization method and batch normalization matching to close the gap between software with hardware. The clocking scheme adjustment-based circuit optimization is also applied to improve the overall performance. Based on our algorithm-hardware co-optimization the hardware configurations of our AQFP-based randomized BNN accelerator, including crossbar synapse array size, stochastic computing bitstream length, and "gray-zone" width of AQFP buffer by comprehensively considering power consumption, energy efficiency, and hardware computing error, are jointly optimized.

ACKNOWLEDGMENTS

This study was supported by JST PRESTO Program (Grant No. JPMJPR19M7), FOREST Program (Grant Number JPMJFR226W, Japan), and the NSF Expedition program CCF-2124453, NSF CCF-2008514.

REFERENCES

[1] Shamiul Alam, Md Shafayat Hossain, Srivatsa Rangachar Srinivasa, and Ahmedullah Aziz. 2023. Cryogenic memory technologies. *Nature Electronics* 6, 3 (2023), 185–198.

[2] Aayush Ankit, Izzat El Hajj, Sai Rahul Chalamalasetti, Geoffrey Ndu, Martin Foltin, R Stanley Williams, Paolo Faraboschi, Wen-mei W Hwu, John Paul Strachan, and Kaushik Roy. 2019. PUMA: A programmable ultra-efficient memristor-based accelerator for machine learning inference. In *Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems*. 715–731.

[3] Christopher L. Ayala, Tomoyuki Tanaka, Ro Saito, Mai Nozoe, Naoki Takeuchi, and Nobuyuki Yoshikawa. 2021. MANA: A Monolithic Adiabatic iNtegration Architecture Microprocessor Using 1.4-zJ/op Unshunted Superconductor Josephson Junction Devices. *IEEE Journal of Solid-State Circuits* 56, 4 (2021), 1152–1165. https://doi.org/10.1109/JSSC.2020.3041338

[4] Timothy J Baker and John P Hayes. 2020. Bayesian accuracy analysis of stochastic circuits. In *2020 IEEE/ACM International Conference On Computer Aided Design (ICCAD)*. IEEE, 1–9.

[5] Ron Banner, Yury Nahshan, and Daniel Soudry. 2019. Post training 4-bit quantization of convolutional networks for rapid-deployment. *Advances in Neural Information Processing Systems* 32 (2019).

[6] Michael G Bechtel, Elise McEllhiney, Minje Kim, and Heechul Yun. 2018. Deeppicar: A low-cost deep neural network-based autonomous car. In *2018 IEEE 24th International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA)*.

[7] Arnout Beckers, Farzan Jazaeri, Andrea Ruffino, Claudio Bruschini, Andrea Baschirotto, and Christian Enz. 2017. Cryogenic characterization of 28 nm bulk CMOS technology for quantum computing. In *2017 47th European Solid-State Device Research Conference (ESSDERC)*. IEEE, 62–65.

[8] Yoshua Bengio, Nicholas Léonard, and Aaron Courville. 2013. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432* (2013).

[9] Adrian Bulat, Brais Martinez, and Georgios Tzimiropoulos. 2020. Bats: Binary architecture search. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXIII*. Springer, 309–325.

[10] Ilkwon Byun, Dongmoon Min, Gyu-hyeon Lee, Seongmin Na, and Jangwoo Kim. 2020. CryoCore: A fast and dense processor architecture for cryogenic computing. In *2020 ACM/IEEE 47th Annual International Symposium on Computer Architecture (ISCA)*. IEEE, 335–348.

[11] Ruizhe Cai, Olivia Chen, Ao Ren, Ning Liu, Caiwen Ding, Nobuyuki Yoshikawa, and Yanzhi Wang. 2019. A Majority Logic Synthesis Framework for Adiabatic Quantum-Flux-Parametron Superconducting Circuits. In *Proceedings of the 2019 on Great Lakes Symposium on VLSI* (Tysons Corner, VA, USA) *(GLSVLSI '19)*. Association for Computing Machinery, New York, NY, USA, 189–194. https://doi.org/10.1145/3299874.3317980

[12] Ruizhe Cai, Olivia Chen, Ao Ren, Ning Liu, Nobuyuki Yoshikawa, and Yanzhi Wang. 2019. A Buffer and Splitter Insertion Framework for Adiabatic Quantum-Flux-Parametron Superconducting Circuits. In *2019 IEEE 37th International Conference on Computer Design (ICCD)*. 429–436. https://doi.org/10.1109/ICCD46524.2019.00067

[13] Ruizhe Cai, Ao Ren, Olivia Chen, Ning Liu, Caiwen Ding, Xuehai Qian, Jie Han, Wenhui Luo, Nobuyuki Yoshikawa, and Yanzhi Wang. 2019. A stochastic-computing based deep learning framework using adiabatic quantum-flux-parametron superconducting technology. In *Proceedings of the 46th International Symposium on Computer Architecture*. 567–578.

[14] Yi-Chen Chang, Hongjia Li, Olivia Chen, Yanzhi Wang, Nobuyuki Yoshikawa, and Tsung-Yi Ho. 2020. ASAP: An Analytical Strategy for AQFP Placement. In *2020 IEEE/ACM International Conference On Computer Aided Design (ICCAD)*. 1–7.

[15] Olivia Chen, Ruizhe Cai, Yanzhi Wang, Fei Ke, Taiki Yamae, Ro Saito, Naoki Takeuchi, and Nobuyuki Yoshikawa. 2019. Adiabatic quantum-flux-parametron: Towards building extremely energy-efficient circuits and systems. *Scientific reports* 9, 1 (2019), 1–10.

[16] Yunji Chen, Tao Luo, Shaoli Liu, Shijin Zhang, Liqiang He, Jia Wang, Ling Li, Tianshi Chen, Zhiwei Xu, Ninghui Sun, et al. 2014. Dadiannao: A machine-learning supercomputer. In *2014 47th Annual IEEE/ACM International Symposium on Microarchitecture*. IEEE, 609–622.

[17] Jungwook Choi, Zhuo Wang, et al. 2018. Pact: Parameterized clipping activation for quantized neural networks. *arXiv:1805.06085* (2018).

[18] John Clarke and Alex I Braginski. 2006. *The SQUID handbook: Applications of SQUIDs and SQUID systems*. John Wiley & Sons.

[19] Matthieu Courbariaux, Yoshua Bengio, and Jean-Pierre David. 2015. Binaryconnect: Training deep neural networks with binary weights during propagations. In *NeurIPS*.

[20] Matthieu Courbariaux, Itay Hubara, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. 2016. Binarized neural networks: Training deep neural networks with weights and activations constrained to+ 1 or-1. *arXiv preprint arXiv:1602.02830* (2016).

[21] Erik Debenedictis. 2018. *IEEE Superconducting and Quantum Information Activities*. Technical Report. Sandia National Lab.(SNL-NM), Albuquerque, NM (United States).

[22] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. 2009. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*. Ieee, 248–255.

[23] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*.

[24] Emerson S. Fang. 1989. A Josephson integrated circuit simulator (JSIM) for superconductive electronics application.

[25] T.V. Filippov, Y.A. Polyakov, V.K. Semenov, and K.K. Likharev. 1995. Signal resolution of RSFQ comparators. *IEEE Transactions on Applied Superconductivity* 5, 2 (1995), 2240–2243. https://doi.org/10.1109/77.403031

[26] Coenrad J. Fourie, Kyle Jackman, Johannes Delport, Lieze Schindler, Tessa Hall, Pascal Febvre, Lucas Iwanikow, Olivia Chen, Christopher L. Ayala, Nobuyuki Yoshikawa, Mark Law, Thomas A. Weingartner, Yanzhi Wang, Peter Beerel, Sandeep Gupta, Haipeng Zha, Sasan Razmkhah, Mustafa Altay Karamuftuoglu, and Massoud Pedram. 2023. Results from the ColdFlux Superconductor Integrated Circuit Design Tool Project. *IEEE Transactions on Applied Superconductivity* (2023), 1–27. https://doi.org/10.1109/TASC.2023.3306381

[27] Rongliang Fu, Junying Huang, Haibin Wu, Xiaochun Ye, Dongrui Fan, and Tsung-Yi Ho. 2022. JBNN: A Hardware Design for Binarized Neural Networks Using Single-Flux-Quantum Circuits. *IEEE Trans. Comput.* 71, 12 (2022), 3203–3214.

[28] Rongliang Fu, Mengmeng Wang, Yirong Kan, Nobuyuki Yoshikawa, Tsung-Yi Ho, and Olivia Chen. 2023. A Global Optimization Algorithm for Buffer and Splitter Insertion in Adiabatic Quantum-Flux-Parametron Circuits. In *Proceedings of the 28th Asia and South Pacific Design Automation Conference* (Tokyo, Japan) *(ASPDAC*

'23). Association for Computing Machinery, New York, NY, USA, 769–774. https://doi.org/10.1145/3566097.3567936

[29] James E Gentle. 2003. *Random number generation and Monte Carlo methods*. Vol. 381. Springer.

[30] Ruihao Gong, Xianglong Liu, Shenghu Jiang, Tianxiang Li, Peng Hu, Jiazhen Lin, Fengwei Yu, and Junjie Yan. 2019. Differentiable soft quantization: Bridging full-precision and low-bit neural networks. In *ICCV*. 4852–4861.

[31] Yuxing He, Naoki Takeuchi, and Nobuyuki Yoshikawa. 2020. Low-latency power-dividing clocking scheme for adiabatic quantum-flux-parametron logic. *Applied Physics Letters* 116, 18 (2020), 182602. https://doi.org/10.1063/5.0005612 arXiv:https://doi.org/10.1063/5.0005612

[32] Zhezhi He and Deliang Fan. 2019. Simultaneously optimizing weight and quantizer of ternary neural network using truncated gaussian approximation. In *CVPR*.

[33] D Scott Holmes. 2021. Cryogenic Electronics And Quantum Information Processing. In *2021 IEEE International Roadmap for Devices and Systems Outbriefs*. IEEE, 1–93.

[34] D Scott Holmes, Andrew L Ripple, and Marc A Manheimer. 2013. Energy-efficient superconducting computing—Power budgets and requirements. *IEEE Transactions on Applied Superconductivity* 23, 3 (2013), 1701610–1701610.

[35] Chao-Yuan Huang, Yi-Chen Chang, Ming-Jer Tsai, and Tsung-Yi Ho. 2021. An Optimal Algorithm for Splitter and Buffer Insertion in Adiabatic Quantum-Flux-Parametron Circuits. In *2021 IEEE/ACM International Conference On Computer Aided Design (ICCAD)*. 1–8. https://doi.org/10.1109/ICCAD51958.2021.9643456

[36] Sergey Ioffe and Christian Szegedy. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*. PMLR, 448–456.

[37] Koki Ishida, Ilkwon Byun, Ikki Nagaoka, Kosuke Fukumitsu, Masamitsu Tanaka, Satoshi Kawakami, Teruo Tanimoto, Takatsugu Ono, Jangwoo Kim, and Koji Inoue. 2020. SuperNPU: An extremely fast neural processing unit using superconducting logic devices. In *2020 53rd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*. IEEE, 58–72.

[38] Sanjay Kariyappa, Hsinyu Tsai, Katie Spoon, Stefano Ambrogio, Pritish Narayanan, Charles Mackin, An Chen, Moinuddin Qureshi, and Geoffrey W Burr. 2021. Noise-resilient DNN: Tolerating noise in PCM-based AI accelerators via noise-aware training. *IEEE Transactions on Electron Devices* 68, 9 (2021), 4356–4362.

[39] Riduan Khaddam-Aljameh, Milos Stanisavljevic, J Fornt Mas, Geethan Karunaratne, Matthias Braendli, Femg Liu, Abhairaj Singh, Silvia M Müller, Urs Egger, Anastasios Petropoulos, et al. 2021. HERMES Core–A 14nm CMOS and PCM-based In-Memory Compute Core using an array of 300ps/LSB Linearized CCO-based ADCs and local digital processing. In *2021 Symposium on VLSI Circuits*. IEEE, 1–2.

[40] Hyungjun Kim, Yulhwa Kim, and Jae-Joon Kim. 2019. In-memory batch-normalization for resistive memory based binary neural network hardware. In *Proceedings of the 24th Asia and South Pacific Design Automation Conference*. 645–650.

[41] Kyounghoon Kim, Jongeun Lee, and Kiyoung Choi. 2015. Approximate de-randomizer for stochastic circuits. In *2015 International SoC Design Conference (ISOCC)*. IEEE, 123–124.

[42] Phil C Knag, Gregory K Chen, H Ekin Sumbul, Raghavan Kumar, Steven K Hsu, Amit Agarwal, Monodeep Kar, Seongjong Kim, Mark A Anders, Himanshu Kaul, et al. 2020. A 617-TOPS/W all-digital binary neural network accelerator in 10-nm FinFET CMOS. *IEEE journal of solid-state circuits* 56, 4 (2020), 1082–1092.

[43] Cong Leng, Zesheng Dou, Hao Li, Shenghuo Zhu, and Rong Jin. 2018. Extremely low bit neural network: Squeeze the last bit out with admm. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

[44] Hongjia Li, Mengshu Sun, Tianyun Zhang, Olivia Chen, Nobuyuki Yoshikawa, Bei Yu, Yanzhi Wang, and Yibo Lin. 2021. Towards AQFP-Capable Physical Design Automation. In *2021 Design, Automation Test in Europe Conference Exhibition (DATE)*. 954–959. https://doi.org/10.23919/DATE51398.2021.9474259

[45] Weitao Li, Pengfei Xu, Yang Zhao, Haitong Li, Yuan Xie, and Yingyan Lin. 2020. TIMELY: Pushing Data Movements and Interfaces in PIM Accelerators Towards Local and in Time Domain. *arXiv preprint arXiv:2005.01206* (2020).

[46] Mingbao Lin, Rongrong Ji, Zihan Xu, Baochang Zhang, Yan Wang, Yongjian Wu, Feiyue Huang, and Chia-Wen Lin. 2020. Rotated binary neural network. *Advances in neural information processing systems* 33 (2020), 7474–7485.

[47] Xiaofan Lin, Cong Zhao, and Wei Pan. 2017. Towards accurate binary convolutional neural network. *Advances in neural information processing systems* 30 (2017).

[48] Zechun Liu, Zhiqiang Shen, Marios Savvides, and Kwang-Ting Cheng. 2020. Reactnet: Towards precise binary neural network with generalized activation functions. In *European conference on computer vision*. Springer, 143–159.

[49] Zechun Liu, Baoyuan Wu, Wenhan Luo, Xin Yang, Wei Liu, and Kwang-Ting Cheng. 2018. Bi-real net: Enhancing the performance of 1-bit cnns with improved representational capability and advanced training algorithm. In *Proceedings of the European conference on computer vision (ECCV)*. 722–737.

[50] K. Loe and E. Goto. 1985. Analysis of flux input and output Josephson pair device. *IEEE Transactions on Magnetics* 21, 2 (1985), 884–887. https://doi.org/10.1109/TMAG.1985.1063734

[51] S. Nagasawa, K. Hinode, T. Satoh, H. Akaike, Y. Kitagawa, and M. Hidaka. 2005. Reliability evaluation of Nb 10 kA/cm2 fabrication process for large-scale SFQ circuits. *Physica C: Superconductivity and its Applications* 426-431 (2005), 1525 – 1532. https://doi.org/10.1016/j.physc.2005.03.058 Proceedings of the 17th International Symposium on Superconductivity (ISS 2004).

[52] SS Teja Nibhanupudi, Siddhartha Raman Sundara Raman, Mikaël Cassé, Louis Hutin, and Jaydeep P Kulkarni. 2021. Ultra-low-voltage UTBB-SOI-based, pseudo-static storage circuits for cryogenic CMOS applications. *IEEE Journal on Exploratory Solid-State Computational Devices and Circuits* 7, 2 (2021), 201–208.

[53] Behraoz Parhami and Chi-Hsiang Yeh. 1995. Accumulative parallel counters. In *Conference Record of The Twenty-Ninth Asilomar Conference on Signals, Systems and Computers*, Vol. 2. IEEE, 966–970.

[54] Thi-Nhan Pham, Quang-Kien Trinh, Ik-Joon Chang, and Massimo Alioto. 2022. STT-BNN: A Novel STT-MRAM In-Memory Computing Macro for Binary Neural Networks. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems* 12, 2 (2022), 569–579. https://doi.org/10.1109/JETCAS.2022.3169759

[55] Divya Prasad, Manoj Vangala, Mudit Bhargava, Arnout Beckers, Alexander Grill, Davide Tierno, Krishnendra Nathella, Thanusree Achuthan, David Pietromonaco, James Myers, et al. 2022. Cryo-computing for infrastructure applications: A technology-to-microarchitecture co-optimization study. In *2022 International Electron Devices Meeting (IEDM)*. IEEE, 23–5.

[56] Weikang Qian, Xin Li, Marc D Riedel, Kia Bazargan, and David J Lilja. 2010. An architecture for fault-tolerant computation with stochastic logic. *IEEE transactions on computers* 60, 1 (2010), 93–105.

[57] Haotong Qin, Ruihao Gong, Xianglong Liu, Mingzhu Shen, Ziran Wei, Fengwei Yu, and Jingkuan Song. 2020. Forward and backward information retention for accurate binary neural networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2250–2259.

[58] Mohammad Rastegari, Vicente Ordonez, Joseph Redmon, and Ali Farhadi. 2016. Xnor-net: Imagenet classification using binary convolutional neural networks. In *European conference on computer vision*. Springer, 525–542.

[59] Herbert E. Robbins. 2007. A Stochastic Approximation Method. *Annals of Mathematical Statistics* 22 (2007), 400–407.

[60] Ro Saito, Christopher L. Ayala, Olivia Chen, Tomoyuki Tanaka, Tomohiro Tamura, and Nobuyuki Yoshikawa. 2021. Logic Synthesis of Sequential Logic Circuits for Adiabatic Quantum-Flux-Parametron Logic. *IEEE Transactions on Applied Superconductivity* 31, 5 (2021), 1–5. https://doi.org/10.1109/TASC.2021.3061636

[61] Ro Saito, Christopher L. Ayala, and Nobuyuki Yoshikawa. 2021. Buffer Reduction Via N-Phase Clocking in Adiabatic Quantum-Flux-Parametron Benchmark Circuits. *IEEE Transactions on Applied Superconductivity* 31, 6 (2021), 1–8. https://doi.org/10.1109/TASC.2021.3073837

[62] Rakshith Saligram, Suman Datta, and Arijit Raychowdhury. 2021. CryoMem: A 4K-300K 1.3 GHz eDRAM macro with hybrid 2T-gain-cell in a 28nm logic process for cryogenic applications. In *2021 IEEE Custom Integrated Circuits Conference (CICC)*. IEEE, 1–2.

[63] Rakshith Saligram, Divya Prasad, David Pietromonaco, Arijit Raychowdhury, and Brian Cline. 2021. A 64-bit arm CPU at cryogenic temperatures: Design technology co-optimization for power and performance. In *2021 IEEE Custom Integrated Circuits Conference (CICC)*. IEEE, 1–2.

[64] Ali Shafiee, Anirban Nag, Naveen Muralimanohar, Rajeev Balasubramonian, John Paul Strachan, Miao Hu, R Stanley Williams, and Vivek Srikumar. 2016. ISAAC: A convolutional neural network accelerator with in-situ analog arithmetic in crossbars. *ACM SIGARCH Computer Architecture News* (2016).

[65] M. Suzuki, M. Maezawa, H. Takato, H. Nakagawa, F. Hirayama, S. Kiryu, M. Aoyagi, T. Sekigawa, and A. Shoji. 1999. An interface circuit for a Josephson-CMOS hybrid digital system. *IEEE Transactions on Applied Superconductivity* 9, 2 (1999), 3314–3317. https://doi.org/10.1109/77.783738

[66] Naoki Takeuchi, Dan Ozawa, Yuki Yamanashi, and Nobuyuki Yoshikawa. 2013. An adiabatic quantum flux parametron as an ultra-low-power logic device. *Superconductor Science and Technology* 26, 3 (2013), 035010.

[67] Naoki Takeuchi, Taiki Yamae, Christopher L. Ayala, Hideo Suzuki, and Nobuyuki Yoshikawa. 2019. An adiabatic superconductor 8-bit adder with 24kBT energy dissipation per junction. *Applied Physics Letters* 114, 4 (01 2019). https://doi.org/10.1063/1.5080753 arXiv:https://pubs.aip.org/aip/apl/article-pdf/doi/10.1063/1.5080753/13036671/042602_1_online.pdf 042602.

[68] Naoki Takeuchi, Yuki Yamanashi, and Nobuyuki Yoshikawa. 2013. Measurement of 10 zJ energy dissipation of adiabatic quantum-flux-parametron logic using a superconducting resonator. *Applied Physics Letters* 102, 5 (2013), 052602.

[69] Naoki Takeuchi, Yuki Yamanashi, and Nobuyuki Yoshikawa. 2015. Adiabatic quantum-flux-parametron cell library adopting minimalist design. *Journal of Applied Physics* 117, 17 (5 2015), 173912.

[70] Tomoyuki Tanaka, Christopher L. Ayala, Qiuyun Xu, Ro Saito, and Nobuyuki Yoshikawa. 2019. Fabrication of Adiabatic Quantum-Flux-Parametron Integrated Circuits Using an Automatic Placement Tool Based on Genetic Algorithms. *IEEE Transactions on Applied Superconductivity* 29, 5 (2019), 1–6. https://doi.org/10.

1109/TASC.2019.2900220

[71] Eleonora Testa, Siang-Yun Lee, Heinz Riener, and Giovanni De Micheli. 2021. Algebraic and Boolean Optimization Methods for AQFP Superconducting Circuits. In *2021 26th Asia and South Pacific Design Automation Conference (ASP-DAC)*. 779–785.

[72] Sergey K. Tolpygo, Vladimir Bolkhovsky, Terence J. Weir, Leonard M. Johnson, Mark A. Gouker, and William D. Oliver. 2015. Fabrication Process and Properties of Fully-Planarized Deep-Submicron Nb/Al–Nb Josephson Junctions for VLSI Circuits. *IEEE Transactions on Applied Superconductivity* 25, 3 (2015), 1–12. https://doi.org/10.1109/TASC.2014.2374836

[73] Thomas J. Walls, Timur V. Filippov, and Konstantin K. Likharev. 2002. Quantum Fluctuations in Josephson Junction Comparators. *Physical Review Letters* 89, 21 (Nov. 2002), 217004. https://doi.org/10.1103/PhysRevLett.89.217004

[74] Qiuyun Xu, Christopher L. Ayala, Naoki Takeuchi, Yuki Murai, Yuki Yamanashi, and Nobuyuki Yoshikawa. 2017. Synthesis Flow for Cell-Based Adiabatic Quantum-Flux-Parametron Structural Circuit Generation With HDL Back-End Verification. *IEEE Transactions on Applied Superconductivity* 27, 4 (2017), 1–5. https://doi.org/10.1109/TASC.2017.2662017

[75] Zihan Xu, Mingbao Lin, Jianzhuang Liu, Jie Chen, Ling Shao, Yue Gao, Yonghong Tian, and Rongrong Ji. 2021. Recu: Reviving the dead weights in binary neural networks. In *Proceedings of the IEEE/CVF international conference on computer vision*. 5198–5208.

[76] Taiki Yamae, Naoki Takeuchi, and Nobuyuki Yoshikawa. 2019. A reversible full adder using adiabatic superconductor logic. *Superconductor Science and Technology* 32, 3 (jan 2019), 035005. https://doi.org/10.1088/1361-6668/aaf8c9

[77] Tomoharu Yamauchi, Hao San, Nobuyuki Yoshikawa, and Olivia Chen. 2023. Design and Implementation of Energy-Efficient Binary Neural Networks Using Adiabatic Quantum-Flux-Parametron Logic. *IEEE Transactions on Applied Superconductivity* 33, 5 (2023), 1–5. https://doi.org/10.1109/TASC.2023.3243180

[78] Jiwei Yang, Xu Shen, Jun Xing, Xinmei Tian, Houqiang Li, Bing Deng, Jianqiang Huang, and Xian-sheng Hua. 2019. Quantization networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 7308–7316.

[79] Kai Zhong, Tianchen Zhao, Xuefei Ning, Shulin Zeng, Kaiyuan Guo, Yu Wang, and Huazhong Yang. 2020. Towards lower bit multiplication for convolutional neural network training. *arXiv preprint arXiv:2006.02804* 3, 4 (2020).

[80] Shuchang Zhou, Yuxin Wu, et al. 2016. Dorefa-net: Training low bitwidth convolutional neural networks with low bitwidth gradients. *arXiv:1606.06160* (2016).

[81] Chenzhuo Zhu, Song Han, Huizi Mao, and William J Dally. 2017. Trained ternary quantization. In *ICLR*.