JBNN: A Hardware Design for Binarized Neural Networks using Single-Flux-Quantum Circuits

Rongliang Fu, Junying Huang, Haibin Wu, Xiaochun Ye, Dongrui Fan, and Tsung-Yi Ho

Abstract—As a high-performance application of low-temperature superconductivity, superconducting single-flux-quantum (SFQ) circuits have high speed and low-power consumption characteristics, which have recently received extensive attention, especially in the field of neural network inference accelerations. Despite these promising advantages, they are still limited by storage capacity and manufacture reliability, making them unfriendly for feedback loops and very large-scale circuits. The Binarized Neural Network (BNN), with minimal memory requirements and no reliance on multiplication, is undoubtedly an attractive candidate for implementing inference hardware using SFQ circuits. This work presents the first SFQ-based Binarized Neural Network inference accelerator, namely JBNN, with a new representation to binarize weights and activation variables. Every SFQ gate is essentially a pipeline stage, making conventional design methods of the accumulator unsuitable for SFQ circuits. So an SFQ-based accumulative parallel counter using SFQ logic cells including T1, OR, and AND is designed to realize the accumulation, where the data size is reduced to a quarter after passing the XNOR column and the AU layer, largely declining the hardware cost. Our evaluation shows that the proposed design outperforms a cryogenic CMOS-based BNN accelerator design running at 77K by 70.92 times while maintaining 97.89% accuracy on the MNIST benchmark dataset. Without the cooling cost, the power efficiency increases up to 929.18 times.

Index Terms—Superconducting, Single-Flux-Quantum, Accelerator, Binarized Neural Network.

1 Introduction

BINARIZED Neural Networks (BNNs) are the most extreme vision of low-precision neural networks with only binary weights and activations, i.e., +1 and -1. While reducing computation and memory requirements, BNN has recently been closing the accuracy gap and becoming more accurate on larger datasets like ImageNet [1]. Hardware acceleration of BNN, including FPGA- and ASIC-based implementations, has been extensively investigated to achieve high performance and energy efficiency simultaneously [2]–[7]. However, most of these designs are CMOS-based and suffer from a performance limitation as the era of Moore's law draws close.

Currently, we are running out of a convincing option to propel the performance of the computer system further, while maintaining its power budget in the post-Moore era. Therefore, innovative new technologies like quantum, neuromorphic, approximate, and stochastic computing may provide solutions to these challenges. And it is time to actively exploit emerging device technologies with significant potentials and make a serious effort to improve their feasibility by resolving their limitations. Among several candidates, Josephson Junction (JJ) based superconducting single-flux-quantum (SFQ) logic family is a highly promising solution due to their ultra-fast switching speed ($\sim$ 1 ps) and low switching energy ($\sim 10^{-19}$ J/bit), which is six orders of magnitude smaller than semiconductor transistors [8], [9]. In 2004, International Technology Roadmap for Semiconductors (ITRS) listed RSFQ as a promising next-generation integrated circuit technology [10]. Then, the Intelligence Advanced Research Projects Activity (IARPA) proposed the C3 project to develop a superconducting computer in 2013 [11]. Since 2017, superconducting electronic (SCE) technology has been listed in the International Roadmap for Devices and Systems (IRDS), becoming the frontier position and international competition hotspot in the post-Moore era [12]. As a member of superconducting SFQ logic families, it has been demonstrated that a rapid single flux quantum (RSFQ)-based T flip-flop (TFF) can operate at up to 770 GHz at 4.2 K [13]. With this technology, it is feasible to improve the device's clock frequency (and thus performance) by an order of magnitude [14], [15].

Focusing on these high potentials has increased interest in investigating superconducting accelerators in recent years, ranging from attempts to port existing CMOS-based accelerators to superconducting technology with less traditional computing paradigms. Table 1 shows a summary of these accelerators realized by superconducting logic. The application type, improvements in energy efficiency and performance compared to CMOS counterparts, improvements in energy-delay product compared to conventional SFQ, and assumed cooling cost are listed. Tannu et al. [16] developed an reciprocal quantum logic (RQL)-based accelerator for SHA-256 engines commonly used in bitcoin mining applications. Ishida et al. [17] presented an SFQ-based neural processing unit (NPU) design with a similar hardware structure to the TPU [18] core. Other interesting approaches that target superconducting accelerators exploit unconventional computing paradigms that match well with the characteristics of superconductor logic. One such related work is to use the adiabatic quantum-flux-parametron

- *Rongliang Fu and Tsung-Yi Ho are with the Department of Computer Science and Engineering, The Chinese University of Hong Kong, Hong Kong 999077, China.*
E-mail: {rlfu, tyho}@cse.cuhk.edu.hk
- *Junying Huang, Haibin Wu, Xiaochun Ye, and Dongrui Fan are with the SKLP, Institute of Computing Technology, CAS, Beijing 100190, China.*
E-mail: {huangjunying, wuhaibin, yexiaochun, fandr}@ict.ac.cn
- *Corresponding author: huangjunying@ict.ac.cn*

TABLE 1
Brief summary of hardware accelerators using superconducting logic.

Accelerator	SFQ family	Application	Energy-efficiency gains compared to CMOS	Performance gains compared to CMOS	Cooling cost normalized to circuit power
SHA256 [16]	RQL	Bitcoin mining	46x	20%	300x
Neural process unit [17]	ERSFQ	DNN	1.23x	18.7x	400x
SC-based DNN accelerator [19]	AQFP	Handwritten digit classification	6.9×10^4x	NA	NA
Race logic-based accelerator [20], [21]	Asynchronous SFQ	Needleman-Wunsch sequence alignment	NA	32∼37x	NA
		Race trees	NA	35∼40x	NA

(AQFP) logic for stochastic computing (SC)-based deep learning accelerators [19]. Tzimpragos et al. took a different approach, proposed the computational temporal logic concept, and demonstrated superconducting accelerators used for the Needleman-Wunsch sequence alignment algorithm and race trees as a proof of concept [20], [21].

While JJ-based superconducting circuits can provide significant power and performance benefits, this technology faces two critical challenges: limited device density and lack of compact memory technology. The current SFQ technology is roughly equivalent to a “250 nm” equivalent CMOS node [22]. However, existing accelerators mentioned above often require a lot of JJs, on the order of millions. Therefore, it is pretty difficult to manufacture these very large-scale accelerator chips for practical applications when the reliability and yield of superconducting fabrication technology are considered. Moreover, prior work usually adopts a shift-register-based memory array for an SFQ convolutional neural network (CNN) accelerator since it fully utilizes the SFQ gate-level pipelining and does not require complex controls. Unfortunately, the inference throughput of SFQ-based CNN accelerators would be seriously degraded with this on-chip memory structure [23]. Consequently, the use of accelerators built with SFQ technology in the near term is likely to be restricted to application domains that have a high demand for speed but only require a tiny chip with a negligible amount of on-chip memory footprint. The BNN computation commonly used in resource-limited and power-constrained scenarios fits well in such applications, where a bit-wise XNOR operation replaces the hardware-hungry multiplier and the memory storage is drastically reduced.

To realize an efficient BNN accelerator using SFQ logic, several specific challenges need to be overcome. One major challenge is the implementation of the accumulator due to the super-deep pipeline nature of SFQ logic. This is because a feedback loop is needed to implement the accumulation operation, causing significant performance degradation as the next clock pulse should wait for a very long data transfer through the feedback path. Moreover, it is also challenging to implement SFQ-based on-chip memory due to its low driving capability and scalability [17], [23]. In other words, how to implement the BNN architecture that maintains the high-speed advantage of SFQ logic but does not require on-chip storage is another challenge.

This paper is the first to develop an SFQ-based BNN accelerator, namely JBNN, using superconducting technology. We overcome the feedback loop limitation using a pipeline-based neural processing unit without feedback loops, including an accumulative parallel counter (APC) and a comparator. In JBNN, the accumulation results do not need to be stored on-chip, and weights and activations are alternately fed directly to JBNN. While maintaining 97.89% accuracy on the MNIST [24] benchmark dataset, our evaluation shows that JBNN significantly outperforms a cryogenic CMOS design working at 77K by 70.92 times in performance when running BNN workloads. With the cooling cost considered, JBNN's power efficiency is 3.09 times higher than the cryogenic CMOS design. But, with free cooling cost assumed, JBNN's performance per watt becomes significantly higher than the cryogenic design by 929.18 times.

In summary, our work makes the following contributions:

1) To the best of our knowledge, this is the first work to design an SFQ-based purely combinational BNN accelerator architecture with a purely pipeline-based neural processing unit without feedback loops.
2) A new representation is used for the normalization of weights and activations, making the inference process more straightforward and friendly to the hardware implementation of binarized neural networks.
3) Experimental results show that JBNN can deliver extreme performance and power efficiency, outperforming a cryogenic CMOS design running at 77K by 70.92 times and 929.18 times, respectively, while maintaining 97.89% accuracy on the MNIST benchmark dataset.

2 PRELIMINARIES

2.1 Single Flux Quantum logic

The zero resistance phenomenon of superconductivity that the resistance of a substance suddenly disappears under low-temperature conditions, was discovered in 1911 at Leiden by Heike Kamerlingh Onnes and Giles Holst after Onnes was able to liquefy helium in 1908 [25]. Since that, especially the publication of the BCS theory in 1957 [8] which revolutionized the understanding of superconductivity, various studies of superconductivity have emerged. With the discovery of the Josephson tunneling junction by Brian David Josephson [9] in 1962, the modern superconducting logic circuit developed rapidly. A typical Josephson junction structure is shown in Fig. 1(a), consisting of three layers. The upper and lower layers are made of superconducting materials, and the middle layer is a thin insulating layer as the barrier. This structure constitutes a superconductor-insulator-superconductor (SIS) junction, where the superconducting current can tunnel through the potential barrier. There is a phase difference φ between the superconductors on both sides of the barrier. When the current $I = I_c \sin \varphi$

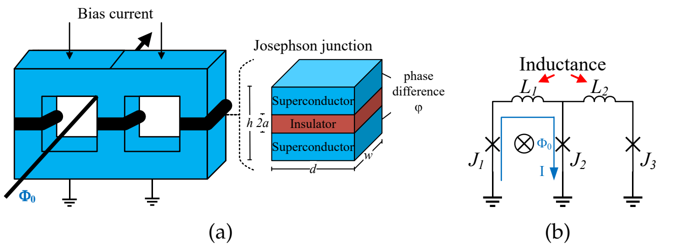

Fig. 1. (a) Superconducting rings with Josephson junctions. (b) Equivalent circuit of (a), where the inductance and cross marks represent the superconductor portion and JJs of each ring, respectively.

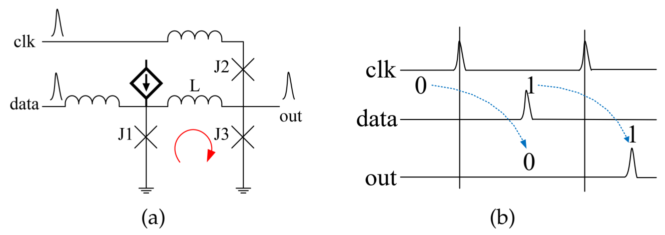

Fig. 2. (a) Simplified JJ-level schematic of DFF; (b) Timing of DFF.

reaches the critical current I_c, the junction can generate an SFQ pulse that is quantized as $\Phi_0 = \frac{h}{2e} \approx 2.07 \times 10^{-15}$Wb. As shown in Fig. 1(b) which is the equivalent circuit of Fig. 1(a), the circle with a cross represents an SFQ stored in the superconducting ring on the left, and superconducting rings in series can form a type of SFQ wiring, namely a Josephson transmission line (JTL).

2.1.1 *Elementary Logic Gates*

In RSFQ technology, binary information is represented as the existence or absence of an SFQ pulse during a specific clock period, respectively. In the following, several typical superconducting RSFQ logic gates are presented.

A. DFF

The RSFQ D flip-flop (DFF) is mainly used to store SFQ pulses, whose simplified JJ-level schematic and timing diagram are shown in Fig. 2. When an SFQ pulse arrives at the input port $data$, if the total current of the input current from the input port $data$ and the bias current exceeds the critical current of $J1$, $J1$ will switch from the superconducting state to the voltage state and generate an SFQ pulse storing in the superconducting ring $J1$-L-$J3$. In this state, if the clock pulse clk arrives and the sum of the current from clk and the current in the superconducting ring $J1$-L-$J3$ exceeds the critical current of $J3$, $J3$ will switch from the superconducting state to the voltage state and generate an SFQ pulse at the output port out, meaning the logic ‘1’.

On the other hand, if there is no SFQ pulse stored in the superconducting loop of the DFF when clk arrives, the current from clk is not large enough to change the state of $J3$. In this case, no SFQ pulse is produced at the output port out, meaning the logic ‘0’.

B. Splitter

Due to the pulse-based representation of signals in RSFQ logic, most gates exhibit a fan-out of one. However, many gates have larger fan-outs in the circuit design. Thus, a new gate called the splitter is introduced in RSFQ logic to solve the multi-fan-out problem. Generally, 1-to-2 and 1-to-3 splitters are noted as SPL and SPL3, respectively, where SPL’s schematic is shown in Fig. 3(a). As illustrated in Fig. 3(b), the gate A can directly connect to gates B and C in CMOS circuits. Whereas in a superconducting RSFQ circuit, a splitter must be inserted at the output port of gate A to drive the other two gates.

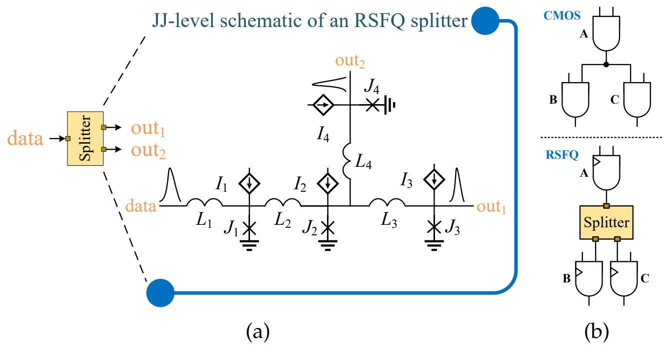

Fig. 3. (a) JJ-level schematic of a 1-to-2 splitter. (b) Example of the splitter insertion.

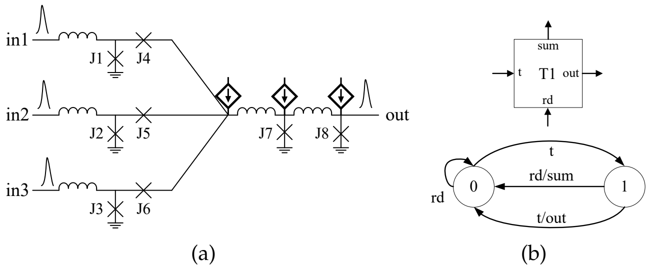

Fig. 4. (a) JJ-level schematic of RSFQ CB3 gate. (b) State transition diagrams and symbol of T1 flip-flop.

C. Confluence Buffer

A confluence buffer (CB) allows the merging of SFQ pulses from two or three different sources. It produces an output SFQ pulse for each incoming pulse from either input. Generally, 2-input and 3-input confluence buffers are noted as CB2 and CB3, respectively. As shown in Fig. 4(a), a CB3 consists of 8 JJs [26]. Both $J1$-$J4$, $J2$-$J5$, and $J3$-$J6$ individually form a buffer structure [27]. Consider an input pulse arriving at terminal $in1$. The input pulse from $in1$ arrives at the input of buffer $J1$-$J4$ and switches $J1$, which transfers the SFQ pulse further along. The resulting signal arrives at the output of buffer $J2$-$J5$ and $J3$-$J6$, switching escape junction $J5$ and $J6$ and preventing this pulse from propagating backward to terminal $in2$ and $in3$. Finally, this pulse switches $J8$, producing the output at terminal out. The CB3’s behavior is symmetric when the input pulse arrives at $in2$ or $in3$. It transfers all pulses from either of its inputs to the output with appropriate delay [28]. Therefore, the confluence buffer operates as an asynchronous OR gate. In this work, the CB3 is used to design a full adder.

D. T1 Flip-Flop

A T1 flip-flop has two inputs, a data input t and a reset input rd, and two outputs, a synchronous output sum and an asynchronous output out. As shown in Fig. 4(b), the state transition diagram has two stable states: ‘1’ and ‘0’,

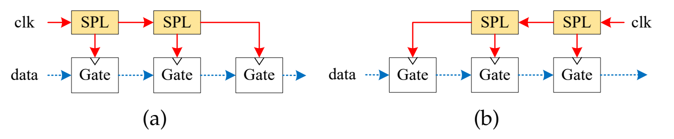

Fig. 5. The clocking schemes for RSFQ circuits: (a) The concurrent-flow clocking; (b) The counter-flow clocking.

characterized by the presence or absence of a magnetic flux quantum. It can be seen that each pulse arriving at t switches the T1 state. When an even-numbered input pulse arrives at t after the last reset, a pulse is generated at its output out. In state '1', a pulse arriving at the input rd resets the T1 to the state '0' and produces an output pulse at the output sum. This work uses the T1 flip-flop to design a full adder.

2.1.2 Clocking in RSFQ

Clocking in superconducting RSFQ logic is much more challenging than CMOS for several reasons [29]. Firstly, due to the peculiar characteristics of RSFQ, most of the logic gates require a clock signal. This feature results in a giant clock distribution network (CDN) because there are many more clock sinks than traditional CMOS circuits, further aggravating the CDN design issues. Secondly, an ultra-high frequency operation in RSFQ circuits requires a reliable ultra-high-speed CDN with controlled clock skew and jitter. Moreover, the methods developed in CMOS are either not applicable to RSFQ or never implemented. Thirdly, the timing uncertainties in RSFQ technology are much more severe due to fabrication process variations, RLC parasitic, bias distribution networks, and thermal fluctuations.

A few commonly-used clocking modes in RSFQ circuits are concurrent-flow clocking, clock-follow-data, and counter-flow clocking [30]. In concurrent-flow clocking, the clock and data flow in the same direction, and the clock arrives before the data, as shown in Fig. 5(a). Characteristically, concurrent-flow clocking could achieve the maximum fastest performance, which is only limited by the intrinsic speed of the gates used in the circuit. In clock-follow-data clocking, the data signal released by the clock from the first cell of the data path arrives at the second cell earlier than the clock. Its minimum clock period is the same as for the concurrent-flow clocking. However, in counter-flow clocking, the clock flows in the opposite direction to the data, resulting in a positive clock skew, as shown in Fig. 5(b). Therefore, concurrent-flow clocking is used in this work targeting high performance.

2.1.3 Path Balancing

Due to the clocking nature of SFQ circuits, they usually need to satisfy the path balancing requirement that all inputs of a clocked SFQ gate have the same logic level to ensure its correct operation. The logic level of gate g in a circuit network N is the length of the longest path in terms of the clocked gate count from any primary input (PI) of N to g. If there is a difference among logic levels of fan-ins of a gate, some DFFs should be inserted into outputs of fan-in gates with minor logic levels [31].

Fig. 6(a) shows an example circuit with two gates, $g1$ and $g2$. The first fan-in ($w1$) of the $g2$ gate has a logic level

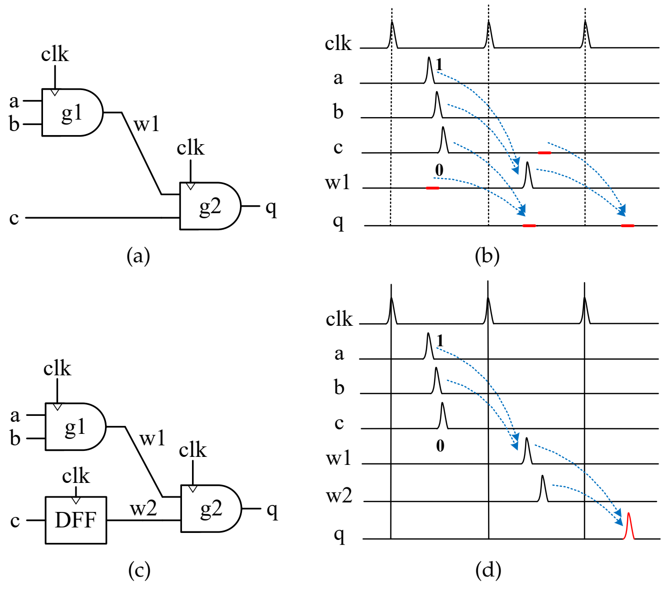

Fig. 6. An example on the necessity of path balancing for function-correct operation in SFQ circuits. (a) and (c) are gate-level schematics with and without path balance, respectively. (b) and (d) are the time diagrams of (a) and (c), respectively.

of 1 and the second fan-in (c) has a logic level of 0. One DFF should be inserted into c, as shown in Fig. 6(c). Without path balancing, correct pulses on c are consumed by this $g2$ gate one clock before the arrival of the corresponding pulses on the input $w1$. Hence, $g2$ can not produce correct output values.

2.1.4 Comparison to CMOS logic

As a well-established, reliable, and reproducible, SFQ-based technology, RSFQ logic [28] has high speed and low power consumption characteristics. Table 2 summarizes the differences between RSFQ logic and CMOS logic in terms of active components, passive components, logic encoding, logic storage, driver strength, clocking, power consumption, and interconnects. It can be seen that SFQ circuits operating at frequencies of tens to hundreds of gigahertz can offer two orders of magnitude speed-up in clock frequency compared to their CMOS counterparts due to the pulse-based logic encoding, the gate-level pipeline structure, the novel clocking, and fast interconnects. SFQ logic utilizes quantized voltage pulses as signals in digital data generation, enabling fast switching. Meanwhile, almost all SFQ logic gate requires a clock signal to release their output, thus SFQ circuits naturally have a gate-level pipeline structure. In this voltage pulse-driven logic, a novel clocking scheme, i.e., concurrent-flow clocking, where the maximum frequency of the SFQ circuit is determined by the timing parameters of the logic cells in the circuit, is employed. It has been shown that this clock method can achieve higher frequencies than zero-skew clocking, widely used in CMOS technology. A detailed description of the superconducting clocking and timing can be found in [30]. Moreover, the voltage pulses in RSFQ circuits can be transmitted by the passive transmission line (PTL) almost losslessly, with the transmission speed at about 1/3 the speed of light.

TABLE 2
Comparison of RSFQ circuits and CMOS circuits.

	RSFQ circuits	CMOS circuits
Active component	JJ	MOSFET
Passive component	Inductance	Capacitance
Logic encoding	SFQ pulse	Voltage level
Logic storage	Flux	Charge
Logic gate	Synchronous	Asynchronous
Drive strength	1	≥ 1
Clocking	Flow clocking	Zero-skew clocking
Frequency	10$\sim$100GHz	$\sim$1GHz
Power consumption	Static power > Dynamic power	Static power $\leq$ Dynamic power
Power supply	Bias current	Voltage
Magnetic interference	Shielding & Moats	/
Interconnect	JTL/PTL	Metal RC line

Another impressive feature of RSFQ logic is that static power dominates the total power consumption. For instance, in RSFQ logic, the dynamic power is around 13 nW per gate compared to 800 nW per gate of static power [32]. This 60x ratio is due to the resistive DC bias network. Fortunately, the static power of RSFQ logic can be eliminated by energy-efficient RSFQ (ERSFQ) logic [33] which is an energy-efficient version of RSFQ logic. It completely excludes the static power dissipation of RSFQ by replacing the bias resistors in RSFQ with JJs with the overhead of doubled dynamic power consumption by JJs. The power of ERSFQ can be written as $P_{total} = 2I_{bias}\Phi_0 f$, where I_{bias} and f are bias current and switching frequency of the circuit, respectively. This leads to switching energy of about 10^{-19} J, which is six orders of magnitude lower than CMOS transistors.

2.2 Binarized Neural Networks

As a promising technique for the inference in neural networks on resource-limited devices, the binarized neural networks considerably save the storage and computation by constraining weights and activations to +1 or -1 [34], [35]. To binarize real-valued variables, there are generally two binarization functions [36]. The first binarization function is deterministic:

$$x^b = \text{Sign}(x) = \begin{cases} 1 & x \geq 0 \\ -1 & x < 0 \end{cases} \tag{1}$$

where x^b is the binarized variable of the real-valued variable x that is from weights or activations. Another attractive binarization function is stochastic:

$$x^b = \text{Sign}(x) = \begin{cases} 1 & \text{with probability } p = \sigma(x) \\ -1 & \text{with probability } 1-p \end{cases} \tag{2}$$

where x^b is the binarized variable and $\sigma(x)$ is:

$$\sigma(x) = \text{clip}\left(\frac{x+1}{2}, 0, 1\right) = \max\left(0, \min(1, \frac{x+1}{2})\right) \tag{3}$$

But the stochastic binarization function requires good quality random numbers generated by the hardware, which has a considerable challenge to implement. So in this work, the Sign binarization function is chosen in the inference process.

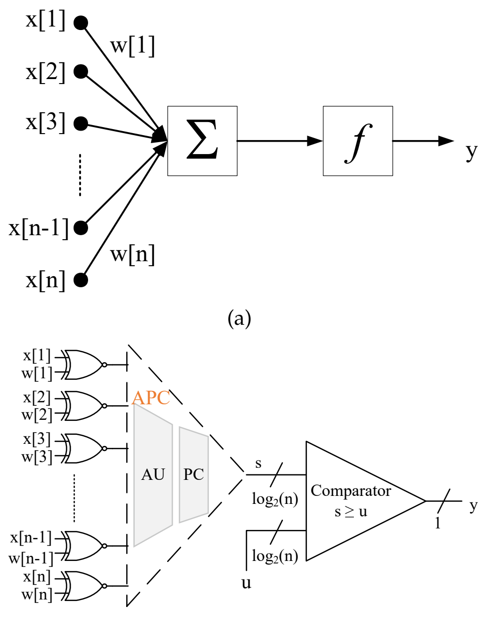

Fig. 7. Schematics of (a) the basic operation and (b) the proposed neural processing unit.

In a full-precision convolutional neural network with the non-linear activation function f, the basic operation, as shown in Fig. 7 (a), can be expressed as

$$y = f\left(\sum_i w_i \cdot x_i\right) \tag{4}$$

By contrast, the neuron activations in a BNN only have the same two possible values, namely -1 or +1, as the weights, except in the first layer. So the products between weights and neuron activations and the sum in Equation 4 can be replaced by the logic XNOR operation and the popcount operation that counts the number of ones in a data vector, respectively. Furthermore, the resulting value can be obtained by comparing it to a trained threshold value μ. Therefore, for a BNN, Equation 4 becomes:

$$y^b = \text{Sign}\left(\underset{i}{\text{popcount}}\left(\text{XNOR}\left(w_i^b, x_i^b\right)\right) - \mu\right) \tag{5}$$

As shown in Algorithm 1, the inference process of the BNN mainly consists of three subprocesses. The input $\boldsymbol{x}$ of the first layer is not binarized, where the activation $\boldsymbol{a}_1^b$ needs to be calculated by the more complex operations than the XNOR and popcount operations:

$$\boldsymbol{a}_1^b = \text{Sign}\left(\sum_i \boldsymbol{w}_1 \cdot \boldsymbol{x} - \boldsymbol{\mu}\right) \tag{6}$$

Besides, the Sign operation is not performed on the output layer. The index of the maximum of its output vector is the predicted value of the whole neural network.

3 SFQ-based BNN Accelerator

As aforesaid, with binarized weights and activations, the dominant computations of a BNN model become binary multiply-accumulate operations, which can be further implemented in a highly hardware-friendly way by simply performing the XNOR and popcount operations. The significant characteristics of SFQ circuits, especially high-speed computation, make them have considerable advantages in computational demanding tasks. Furthermore, In SFQ logic, the XNOR operation can be realized using a superconducting XNOR gate. In this section, an SFQ-based BNN accelerator architecture is proposed, which considers the limitation of SFQ circuits on the storage while combining the high-performance advantages of SFQ circuits and the low storage demanding of BNN.

3.1 Architecture

In original BNNs, weights and activation are restricted to -1 or 1, presented to 2 bits or 1 bit. Generally, with one bit, zero is used to present -1, meaning that extra operations are required to obtain the real value of the accumulation.

In order to simplify the computation process and save the storage for the hardware implementation, the new representation (SN) is used with referring to Stochastic Computing:

$$x^b = \frac{x^{b'}+1}{2}, x^{b'} = \text{Sign}(x) = \begin{cases} 1 & x \geq 0 \\ -1 & x < 0 \end{cases} \tag{7}$$

As shown in Table 3, its SN is 1 when $x \geq 0$ and 0 otherwise. For the multiplication operation, the logical XNOR operation still works. For the addition operation, there are:

$$\sum_{i=0}^{n} x_i^{b'} = 2\sum_{i=1}^{n} x_i^b - n \tag{8}$$

Since almost all gates require a DC clock signal, SFQ circuits naturally have a deep pipeline structure, which increases the difficulty of avoiding RAW hazards and has significant disadvantages for cyclic control circuits. Therefore, the architecture of the neural processing unit is proposed as shown in Fig. 7 (b).

The neural processing unit contains a XNOR column, a n-bit SFQ-based APC and a $\log_2(n)$-bit comparator. In Fig. 7 (b), $\boldsymbol{x}[i]$ and $\boldsymbol{w}[i]$ $(1 \leq i \leq n)$ enter the network alternately. Then the neural processing unit completes the multiplication operation, the accumulation operation, and the Sign operation, whose output is the input of the next layer. The multiplication operation can be achieved by a column of the XNOR gates, the basic logic cell in SFQ circuits. After data goes through the XNOR column, its number reduces by half, considerably shrinking the scale of circuits in hardware implementation.

TABLE 3
-1/1 coding methods and multiplication in SN.

x	>	<	Multiplication	SN	
				Input bitstream	Output
Sign	1	-1	$1 * 1$	$1 \odot 1$	1
Binary	01	11	$1 * (-1), (-1) * 1$	$1 \odot 0, 0 \odot 1$	0
SN	1	0	$(-1) * (-1)$	$0 \odot 0$	1

The popcount function counts the number of ones in the output of the XNOR column to implement the accumulation operation. Assume the input number of bits for APC is N, an input with M bits $(M < N)$ and N_1 ones needs to append additional $\frac{N-M}{2}$ ones to obtain the correct cumulative result sum.

$$sum = 2 * N_1 - M = 2(N_1 + \frac{N-M}{2}) - N \tag{9}$$

Since the data is presented by the SN method, $\boldsymbol{\mu}$ needs to be processed by Equation 10, whose result is compared with the output of APC (explained in the following subsection) to obtain the result of the neural processing unit. $|\boldsymbol{x}|$ is the size of the input $\boldsymbol{x}$ of the current layer. Finally, passing through multiple hidden layers, the output of the output layer can be obtained. The index of the maximum value of the output layer is the predicted output of the neural network for the input. The complete inference process is presented in Algorithm 2.

$$\boldsymbol{u}' = \frac{(\boldsymbol{u} + |\boldsymbol{x}|)}{2} \tag{10}$$

3.2 SFQ-based APC

As a vital component in the neural processing unit, the implementation of the popcount function has a considerable

Algorithm 1: Conventional BNN Inference Process

```
Input: the number of layers $L$, a input vector $\boldsymbol{x}$,
       trained binary weights $\boldsymbol{w}^b$, and a trained
       threshold vector $\boldsymbol{\mu}$
Output: the MLP output $\boldsymbol{a}_L$
1 1. First layer:
2 $\boldsymbol{a}_1^b = \text{Sign}\left(\boldsymbol{w}_1^b \cdot \boldsymbol{x} - \boldsymbol{\mu}_1\right)$
3 2. Remaining hidden layers:
4 for $i$=2 to $L$ do
5     $\boldsymbol{a}_i = \text{popcount}\big(\text{XNOR}\left(\boldsymbol{w}_i^b, \boldsymbol{a}_{i-1}^b\right)\big)$
6     $\boldsymbol{a}_i^b = \text{Sign}\left(\boldsymbol{a}_i - \boldsymbol{\mu}_i\right)$
7 3. Output layer:
8 $\boldsymbol{a}_L = \text{popcount}\big(\text{XNOR}\left(\boldsymbol{w}_L^b, \boldsymbol{a}_{L-1}^b\right)\big) - \boldsymbol{\mu}_L$
9 return $\boldsymbol{a}_L$;
```

Algorithm 2: Proposed BNN Inference Process

```
Input: the number of layers $L$, a input vector $\boldsymbol{x}$,
       trained binary weights $\boldsymbol{w}^b$, and a trained
       threshold vector $\boldsymbol{\mu}$
Output: the MLP output $\boldsymbol{a}_L$
1 $\boldsymbol{x}^b = (\text{Sign}(\boldsymbol{x}) + 1)/2$
2 $\boldsymbol{w}^b = \big(\text{Sign}(\boldsymbol{w}^b) + 1\big)/2$
3 1. First layer:
4 $\boldsymbol{a}_1 = \text{APC}\big(\text{XNOR}\left(\boldsymbol{w}_1^b, \boldsymbol{x}^b\right)\big)$
5 $\boldsymbol{a}_1^b = \big(\text{Sign}\left(\boldsymbol{a}_1 - (\boldsymbol{\mu}_1 + |\boldsymbol{x}^b|)/2\right) + 1\big)/2$
6 2. Remaining hidden layers:
7 for $i$=2 to $L$ do
8     $\boldsymbol{a}_i = \text{APC}\big(\text{XNOR}\left(\boldsymbol{w}_i^b, \boldsymbol{a}_{i-1}^b\right)\big)$
9     $\boldsymbol{a}_i^b = \big(\text{Sign}\left(\boldsymbol{a}_i - (\boldsymbol{\mu}_i + |\boldsymbol{a}_{i-1}^b|)/2\right) + 1\big)/2$
10 3. Output layer:
11 $\boldsymbol{a}_L = \text{APC}\big(\text{XNOR}\left(\boldsymbol{w}_L^b, \boldsymbol{a}_{L-1}^b\right)\big) - \big(\boldsymbol{\mu}_L + |\boldsymbol{a}_{L-1}^b|\big)/2$
12 return $\boldsymbol{a}_L$;
```

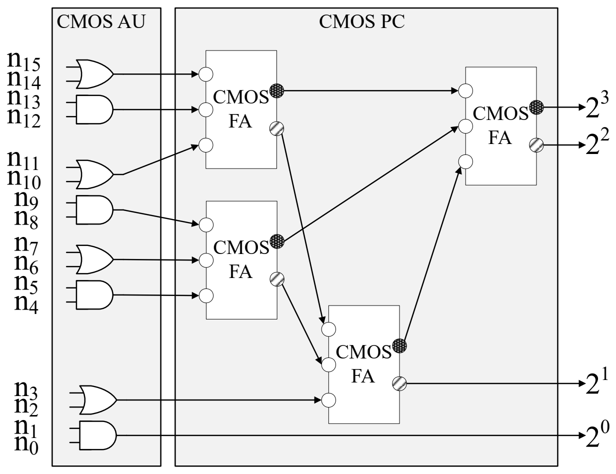

Fig. 8. Example of a 16-4 APC converting a 16-bit stochastic number into a 4-bit binary number.

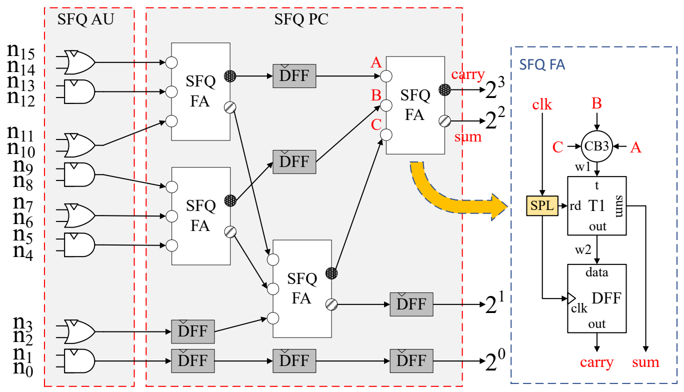

Fig. 9. Path-balanced 16-4 SFQ APC with T1-based 1-bit full adders counting the number of ones in the input bit stream.

impact on the performance of the neural network accelerator. In this section, we focus on the implementation of the popcount function in SFQ logic.

In stochastic computing, the accumulative parallel counter is used to convert a stochastic number to a binary number [37]. Fig. 8 shows an example of APC generating a 4-bit binary number from a 16-bit stochastic number [37]. The parallel counter (PC) in an APC uses some full adders (FAs) to generate $2^0 \sim 2^{m-1}$-weighted bits (a binary number) from a stream of 2^0-weighted input bits (a stochastic number), where m is the number of bits of the binary number. The first layer of the APC in Fig. 8 is an approximation unit (AU) composed of OR-AND gates pairs. Converting a stochastic number into a binary number requires counting the number of ones in the random bit stream. As a result, the SFQ-based APC is proposed to implement the popcount function.

3.2.1 SFQ-based APC design

The proposed APC-based popcount function is implemented using the elementary logic cells in SFQ circuits. Additional DFFs are inserted in the circuit, thereby fulfilling the requirement of path balancing. The path-balanced 16-4 RSFQ APC that counting the number of ones in the input bit stream is shown in Fig. 9. It consists of a superconducting approximation unit (SFQ AU) and a superconducting accurate PC (SFQ PC). The proposed SFQ APC exploits 1-layer AU, which consists of superconducting OR-AND gates pairs, which can halve the data size, further reducing the size of subsequent circuits, as shown in Fig. 9.

The SFQ PC consists of a cascade of full adders. However, adapting the full adder design directly from CMOS logic to SFQ logic using functionally equivalent gates is inefficient in that it involves a significant area overhead (a FA needs at least five gates in CMOS). Therefore, in this paper, referring to the design idea in [30], an SFQ-based specific full adder is implemented where two CB2 in the original full adder are merged into one CB3 gate. Note that, as explained in Section 2.1.3, the SFQ circuits need to be path-balanced. Thus, 7 path-balanced DFFs are inserted into the SFQ PC.

Specifically, the SFQ FA is realized using four gates: a T1 flip flop, a CB3 gate, a DFF, and an SPL gate, as shown in Fig. 9. For clarity reasons, the clock port in the FA's symbol is omitted. Timing corresponding to the correct operation of the SFQ FA in the presence of different input sets, i.e., 111, 101, and 010 are shown in Fig. 10. It can be seen that this FA can output sum and $carry$ every cycle.

3.2.2 Analysis for Pipeline Stage in SFQ APC

In order to clarify how many pipeline stages are required by the proposed SFQ APC, the number of FAs are calculated. Suppose that N-input bits become V-output bits ($N = 2^V$) and $f(V)$ are the number of FAs. Then,

$$f(V) = \frac{N}{2} - V = \frac{N}{2} - log_2 N \tag{11}$$

The design of the SFQ PC is based on a divide-and-conquer strategy: given two PCs each with l inputs, a $(2l+1)$-input PC is obtained using a ripple-carry adder of length $\lfloor log_2 l \rfloor + 1$ [38]. Similarly, two such $(2l+1)$-input PCs can be combined with a $(\lfloor log_2 l \rfloor + 2)$-bit ripple-carry adder to produce a $(4l+3)$-input PC. This procedure is repeated until a PC whose size is greater than or equal to $N-1$ is obtained.

For example, only one full adder is needed when $l = 3$, so the number of pipeline stages is one. When $l = 7$ the required number of ripple-carry adders is $\lfloor log_2 3 \rfloor + 1 = 2$, so a total of 3 stages are required. Similarly, when $l = 15$, a total number of $3 + \lfloor log_2 7 \rfloor + 1 = 6$ stages are required. To generalize, for a l-bit input PC, the required number of stages $s(l)$ is:

$$\begin{aligned} s(l) &= 1 + 2 + 3 + ... + (\lfloor log_2(2^{V-1} - 1) \rfloor + 1) \\ &= 1 + 2 + 3 + ... + \lfloor log_2(l + 1 - 2) \rfloor \end{aligned} \tag{12}$$

Note that $l = 2^V - 1$. In the proposed N-input SFQ APC, $N/2$ superconducting OR-AND gates pairs and $f(V)$ FAs are exploited, where $N = 2^V$. By replacing the term $(l+1)$ in Equation (12) with $N/2$, the number of pipeline stages $S(N)$ in the entire APC can be obtained as follows.

$$\begin{aligned} S(N) &= 1 + 2 + 3 + ... + (\lfloor log_2(N/2 - 2) \rfloor) + 1 \\ &= \frac{\lfloor log_2(N-4) \rfloor * (\lfloor log_2(N-4) \rfloor - 1)}{2} + 1 \end{aligned} \tag{13}$$

The last term in Equation 13 is added due to the 1-layer AU consisting of superconducting OR-AND gates pairs.

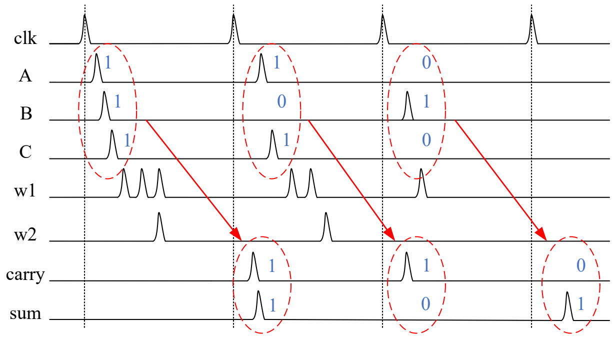

Fig. 10. Timing diagram of the full adder in SFQ logic where three sets of inputs, 111, 101, and 010 and corresponding outputs 11, 10, and 01 are displayed.

3.2.3 Analysis for Area in SFQ APC

The area of an SFQ circuit is denoted by JJ-complexity. The JJ-complexity represents the number of JJs required to design a logic block [16]. In this paper, we use JJ-complexity as a key figure of merit, similar to prior superconducting system designs [39], [40]. Although a logic block consists of logic gates and transmission lines, the JJ-complexity of system-level is evaluated only by computing gate JJ-complexity. Since Passive Transmission Line (PTL) is widely used in the routing of SFQ circuits, whose interconnect JJ-complexity can be negligible.

The gate JJ-complexity JJ_G is derived by adding SFQ AU JJ-complexity JJ_{AU}, the full adder JJ-complexity JJ_{FA}, and path-balanced DFF JJ-complexity JJ_{PB_DFF} as shown in Equation 14.

$$JJ_G = JJ_{AU} + JJ_{FA} + JJ_{PB\_DFF} \tag{14}$$

The computation of JJ_{AU} and JJ_{FA} are trivial given the JJ-complexity of each SFQ gate, as shown in Table 4. As explained in Section 3.2.2, since the design of the SFQ APC is based on a divide-and-conquer strategy, the inserted DFF count $d(l)$ for the l-input SFQ PC can be calculated as

$$\begin{aligned} d(l) &= 2 * d\left(2^{V-1} - 1\right) + s\left(2^{V-1} - 1\right) \\ &\quad + 3 * \left(1 + 2 + 3 + ... + \lfloor log_2(2^{V-1} - 1)\rfloor\right) \end{aligned} \tag{15}$$

After substituting Equation 12 into Equation 15, $d(l)$ is given by

$$\begin{aligned} d(l) &= 2 * d\left((l-1)/2\right) \\ &\quad + (\lfloor log_2(l-3)\rfloor * (\lfloor log_2(l-3)\rfloor - 1))/2 \\ &\quad + 3 * (\lfloor log_2(l-1)\rfloor * (\lfloor log_2(l-1)\rfloor - 1))/2 \end{aligned} \tag{16}$$

Then, the number of DFFs $D(N)$ in the entire APC can be obtained iteratively as follows.

$$\begin{aligned} D(N) &= 2 * D\left(N/2\right) - S\left(N/2\right) + S(N) + \\ &\quad 3 * (\lfloor log_2(N-2)\rfloor - 1) * \lfloor log_2(N-2)\rfloor/2 \end{aligned} \tag{17}$$

Thus, according to the JJ-complexity of each gate, as listed in Table 4, the JJ-complexity JJ_G can be given by:

$$\begin{aligned} JJ_G &= JJ_{AU} + JJ_{FA} + JJ_{PB\_DFF} \\ &= \frac{N}{4} * 27 + \left(\frac{N}{2} - log_2 N\right) * 28 + D(N) * 7 \end{aligned} \tag{18}$$

Fig. 11 shows the layout of the proposed SFQ-based 16-4 APC circuit, which consists of 4 pipeline stages, 7 path-balanced DFFs, and 269 JJs for logic cells.

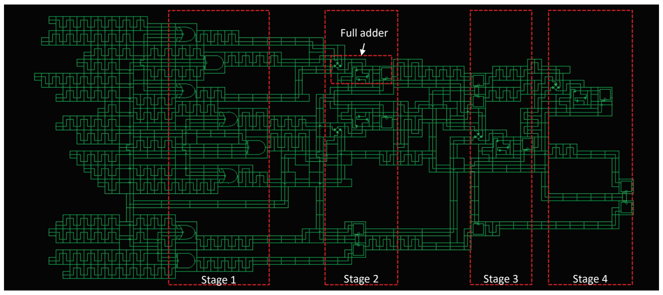

Fig. 11. The layout of the 16-4 SFQ APC design.

4 Experiential Results

The experiment uses the MNIST [24] dataset to evaluate the proposed RSFQ-based BNN architecture in several aspects, including application-level performance and hardware-level performance. The MNIST is an image classification benchmark dataset, consisting of a train set with 60K size and a test set of 10K size, each of which is a 28*28 grayscale image representing a digit ranging between 0 and 9. In this section, the experimental setup is first described, then JBNN is evaluated on performance and power efficiency.

4.1 Experimental setup

As shown in Fig. 12, the structure of the BNN in this work is similar to the MLP proposed by Courbariaux et al. in [34], consisting of 3 hidden layers of 4096 binary units and an L2-SVM output layer of 10 binary units. But in the train process, the parameters γ and β of the batch normalization shown in Equation 19 are not trained and are set to $\gamma = 1$ and $\beta = 0$, respectively. Meanwhile, the output layer is replaced by the softmax function. In the reference process, the weights $\boldsymbol{w}$ and neuron activations $\boldsymbol{x}$ first need to be binarized by Equation 7, making all operations of the whole network bit-wise to avoid the extra computing cost caused by the first layer. After that, the binarized weights and activations are fed into the neural processing unit.

$$y = \frac{x - \mu}{\sqrt{\sigma^2 + \epsilon}}\gamma + \beta \tag{19}$$

TABLE 4
JJ-complexity of SFQ cells.

Cell type	JJ-complexity	Static power @50GHz (nW)	Dynamic power @50GHz (nW)
DFF[1]	7	1815	142
SPL[1]	4	2093	108.6
CB3[2]	8	3789	292.5
T1[3]	9	988	174
AND[1]	15	2805	289
NOT[1]	9	2259	183
XOR[1]	11	2402	232
OR[1]	12	2980	265
XNOR[1]	18	4215	416

[1] The data is from the open-source MIT-LL SFQ library [41].
[2] The static and dynamic power of CB3 [26] is evaluated as 1.5 times that of MERGE [41], respectively.
[3] The static and dynamic power of T1 [42] is calculated according to the calculation method of the library [41].

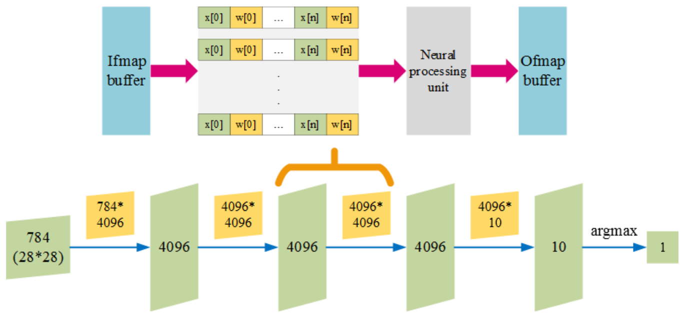

Fig. 12. Overview of the BNN network architecture.

The application-level performance of JBNN is evaluated by testing the accuracy of the proposed BNN inference framework on the MNIST dataset. To comprehensively evaluate the hardware-level performance of JBNN, we designed three different implementations of the BNN inference accelerator for comparison: the CMOS-based synchronous design at 300K, the CMOS-based asynchronous design at 300K, and the cryogenic CMOS-based design at 77K. The three implementations, called $SyncBNN@300K$, $AsyncBNN@300K$, and $CryoBNN@77K$, adopt the same architecture in [43]. This reference architecture is made of a 32×32 processing element (PE) array, composed of a 2 kbits memory array that store weights, 32 XNOR logic gates that perform the XNOR between the 32 bits weights and the 32 bits received data, a 32 bits to 5 bits popcount module compound of basic tree adders. However, unlike PE in [43], where the weights are stored in the spin torque magnetoresistive random access memory (MRAM), we used the static random access memory (SRAM) to store weights.

In pursuit of ultra-high computing speed, the paper focuses on RSFQ logic and ERSFQ logic to realize the hardware implementation of JBNN. As described in Section 2.1.4, the only difference between RSFQ logic and ERSFQ logic is how to supply the DC bias current. It is assumed that the timing parameters and area of ERSFQ logic gates are the same as those of RSFQ logic. Meanwhile, according to the bias current of RSFQ logic design and the power model of ERSFQ [32], we estimate the dynamic and static power of ERSFQ logic gates to be twice that of RSFQ logic and zero, respectively.

4.2 Performance evaluation

This section first presents the accuracy loss obtained by the proposed new binarized representation of weights and activations. Then, hardware-level performance comparisons of JBNN with three different CMOS-based implementations are discussed.

4.2.1 Application-level accuracy

Without retraining on the validation, the test error associated with the best validation error rate after 100 epochs is 1.63%, which is the result of Algorithm 1. Due to the accuracy loss in the binarization of weights and activation variables, the test error using the inference process in Algorithm 2 is 2.11%.

Besides, another case with not considering $\boldsymbol{\mu}$ in the reference process is also evaluated. In this way, the neural processing unit only consists of an XNOR column and an APC whose highest output bit is used as the output of the neural processing unit, further saving circuit cost. In this case, the test errors of the inference process using Algorithm 1 and Algorithm 2 are 9.94% and 10.96%, respectively.

4.2.2 Hardware-level performance

The hardware-level performance of JBNN is compared with three different CMOS-based implementations of the BNN inference accelerator with the same architecture in [43]. These implementations include:

- $SyncBNN@300K$. A digital ASIC version with the square-shaped 32×32 process elements (PEs) array is implemented by standard integrated circuit design tools with the 12 nm TSMC CMOS process. The highest operating frequency reported by the Synopsys Design Compiler tool is 1.2 GHz.
- $AsyncBNN@300K$. Although the synchronous design paradigm is ubiquitous in the electronic design industry and has many advantages, it is undermined by the increasing complexity of VLSI systems and the ever-growing demand for higher performance. However, in an asynchronous circuit, the components evolve autonomously and operate on an "on-demand" basis, indicating a potential for lower power dissipation and better timing performance. Thus, for a fair comparison, we also designed the PE to be asynchronous, where the popcount function is implemented in an asynchronous fashion. The rising edge of the output of the previous popcount bit triggers the clock input of its next popcount bit.
- $CryoBNN@77K$. Cryo-CMOS is the CMOS design that runs at low temperatures (e.g., 77K) for improved performance and power efficiency due to the reduced leakage power and wire resistance at cryogenic temperature. Many proposals have focused on 77K-based cryogenic computing, such as 77K-based DRAM devices [44], processors [45], and quantum control systems [46]. Although some research has been conducted to create a 77K-based CMOS standard cell library [47], the raw data of the library is not available for cryo-CMOS digital circuit design. Thus, to implement the CryoBNN design running at 77K, we first simulated the elementary cells using Hspice and the 16 nm PTM models [48] operating at different temperatures. Then the critical path delay and power consumption for the PE running at 77K are calculated based on the simulated data of the elementary cells.

JBNN consists of 4096 XNOR gates, the APC, and a comparator (CMP for short). Table 5 shows the number of pipeline stages and JJs of these JBNN components. Under this configuration, APC's input is 4096 bits and has 56 pipeline stages. Note that in CMOS-based design, zero skew clocking is used, that all registers update their values every clock cycle, so the number of stages is 1.

The performance is calculated by the frame per second (fps) normalized to the CMOS-based implementations. We estimate fps by multiplying the size of the PE array by the frequency and dividing by the number of total accumulations in the whole network. Since the CMOS-based design

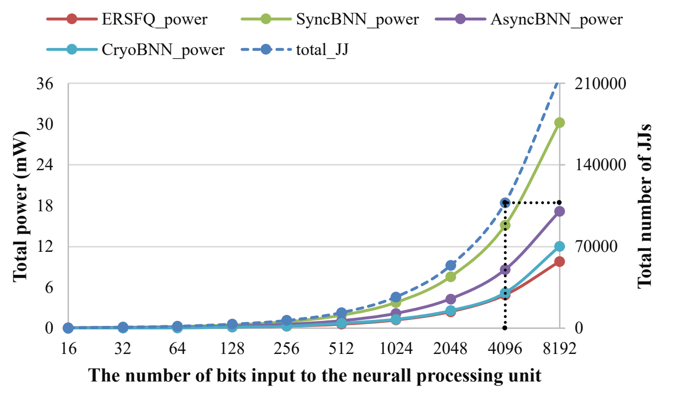

Fig. 13. Total number of JJs and power consumption of the neural processing unit with different input bits.

has a PE array scale of 32 × 32, the images processed per second can be obtained, which is 3.34×10^4. Considering that the utilization rate of PE is usually less than 100%, assuming a 90% PE utilization rate, the effective performance of $SyncBNN$, $AsyncBNN$, and $CryoBNN$ are 3.00×10^4, 3.00×10^4, and 5.61×10^4, respectively. Three hidden layers and one output layer in the network are performed with a JBNN with 8192 inputs (4096 activations, 4096 weights), and it requires 12570 cycles to process one image. Therefore, the number of images processed per second is 3.98×10^6. In other words, the performance of JBNN is 132 times and 70.92 times higher than that of $SyncBNN$ and $CryoBNN$, respectively. This speed-up in performance is mainly thanks to the much higher frequency of the SFQ process and the simple pipeline architecture of JBNN with no feedback loop.

Moreover, our proposed method scales well because the SFQ-based APC used to calculate the popcount is relatively hardware-friendly and area-efficient. For example, as shown in Fig. 13, when the number of bits input to the neural processing unit is increased to 256, the number of JJs required by the proposed method is less than 10,000. Even when the integration scale is limited to 100,000 JJ, the number of input bits can be enlarged to 4096. In addition, since the power of ERSFQ is roughly proportional to the number of JJs, it follows a similar trend as JJ's count, as shown in Fig. 13. We also study the scaling of the power of $SyncBNN$, $AsyncBNN$, and $CryoBNN$ with the number of bits input. Power is found to increase by about two times for every doubling of the PE array size.

TABLE 5
The number of pipeline stages and JJs of JBNN components.

Components / Metrics	APC	CMP	APC+CMP	JBNN	CMOS-based BNN
#stages	56	12	68	69	1
#JJs	140509	1258	141767	215495	/

4.3 Power efficiency evaluation

The performance and power data of $SyncBNN$ and $AsyncBNN$ are reported by Synopsys Design Compiler, whereas the data of $CryoBNN$ is calculated by applying the delay and power data of 77K-based elementary cells to the synthesized netlist of $SyncBNN$. Two different SFQ device technologies, RSFQ and ERSFQ technology are employed to evaluate JBNN's power-efficiency. The power consumption and performance per Watt (i.e., power-efficiency) for JBNN and the CMOS-based designs are shown in Table 6.

With RSFQ device technology, JBNN consumes 60.57 mW, and ERSFQ-JBNN consumes only 9.78 mW because there is no static power consumption in ERSFQ technology. Due to the simple computational logic in BNN, $CryoBNN$ dissipates the minimal power of 12.03 mW in its operation. Assuming a 9.65x cooling overhead for 77K as in [45], its power efficiency (fps per Watt) is 4.38×10^5. Our evaluation observed a 929.18x and 233x improvement in power-efficiency with ERSFQ technology over $CryoBNN$ and $AsyncBNN$, respectively, providing free cooling. If 300x cooling cost is considered, the normalized power efficiency of ERSFQ-JBNN becomes 1.35×10^6, which is 3.09x higher than $CryoBNN$. The energy-efficient version of the SFQ technology, such as ERSFQ circuits, is more suitable for achieving high energy efficiency since they eliminate the static power consumption and retain the high-frequency characteristics of the RSFQ circuits.

5 Conclusion

Superconducting technology is a highly promising solution in post-Moore's era. However, the potential of SFQ computing has not yet been fully realized because of the fabrication, timing, and memory limitations. This paper addresses the challenges as follows. First, a new representation is proposed for the binarization of weights and activation variables. With it, in the inference process of BNN, the first layer can also utilize binary operations with binarized inputs while maintaining acceptable accuracy, about 97.89%.

TABLE 6
Power efficiency comparison between the CMOS-based BNN implementations and JBNN using RSFQ and ERSFQ logic.

Metrics / Networks	Cooling	Frequency (GHz)	Static Power (mW)	Dynamic Power (mW)	Total Power (mW)	Power Efficiency (fps/W)
RSFQ-JBNN	0	50	55.67	4.89	60.57	6.57×10^7
RSFQ-JBNN	300x	50	55.67	4.89	60.57	2.18×10^5
ERSFQ-JBNN	0	50	0	9.78	9.78	4.07×10^8
ERSFQ-JBNN	300x	50	0	9.78	9.78	1.35×10^6
SyncBNN@300K	0	1.2	125.00×10^{-6}	30.24	30.24	9.94×10^5
AsyncBNN@300K	0	1.2	884.43×10^{-6}	17.19	17.19	1.75×10^6
CryoBNN@77K	9.65x	2.24	26.42×10^{-6}	12.03	12.03	4.38×10^5

Next, the hardware design of the binarized neural network is first proposed using superconducting single flux quantum logic, called JBNN, which is an area-efficient and purely pipeline-based neural processing unit without feedback loops. In JBNN, an SFQ-based accumulative parallel counter is designed, where the full adder is realized by the T1 cell, an SFQ elementary logic cell. Our evaluation shows that the proposed design outperforms a cryogenic CMOS-based BNN accelerator design running at 77K by 70.92 times in performance when running BNN workloads. With the cooling cost considered, JBNN's power efficiency is 3.09x higher than the cryogenic BNN design. However, with free cooling cost assumed, JBNN's performance per watt becomes significantly higher than the cryogenic design by 929.18 times.

Acknowledgments

The layout of the 16-4 SFQ APC circuit is depicted with the support of the SIMIT-Nb03 standard cell library from the Shanghai Institute of Microsystem and Information Technology, CAS. The research work described in this paper was conducted in the Intelligent Design Automation Lab funded by The Hong Kong Jockey Club Charities Trust, and also supported in part by the Strategic Priority Research Program of the Chinese Academy of Sciences under Grant XDA18000000, the National Natural Science Foundation of China under Grant 61732018, 61872335, and the Youth Innovation Promotion Association CAS.

References

[1] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[2] Mohammad Ghasemzadeh, Mohammad Samragh, and Farinaz Koushanfar. ReBNet: Residual binarized neural network. In *2018 IEEE 26th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM)*, pages 57–64. IEEE, 2018.

[3] Ritchie Zhao, Weinan Song, Wentao Zhang, Tianwei Xing, Jeng-Hau Lin, Mani Srivastava, Rajesh Gupta, and Zhiru Zhang. Accelerating binarized convolutional neural networks with software-programmable FPGAs. In *Proceedings of the 2017 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, FPGA '17, pages 15–24, New York, NY, USA, 2017. Association for Computing Machinery.

[4] Peng Guo, Hong Ma, Ruizhi Chen, Pin Li, Shaolin Xie, and Donglin Wang. FBNA: A fully binarized neural network accelerator. In *2018 28th International Conference on Field Programmable Logic and Applications (FPL)*, pages 51–513, 2018.

[5] Yaman Umuroglu, Nicholas J Fraser, Giulio Gambardella, Michaela Blott, Philip Leong, Magnus Jahre, and Kees Vissers. Finn: A framework for fast, scalable binarized neural network inference. In *Proceedings of the 2017 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, pages 65–74, 2017.

[6] Hiroki Nakahara, Tomoya Fujii, and Shimpei Sato. A fully connected layer elimination for a binarizec convolutional neural network on an FPGA. In *2017 27th International Conference on Field Programmable Logic and Applications (FPL)*, pages 1–4. IEEE, 2017.

[7] Daniel Bankman, Lita Yang, Bert Moons, Marian Verhelst, and Boris Murmann. An always-on 3.8μJ/86% CIFAR-10 mixed-signal binary CNN processor with all memory on chip in 28-nm CMOS. *IEEE Journal of Solid-State Circuits*, 54(1):158–172, 2018.

[8] T Van Duzer, CW Turner, DG McDonald, and Alan F Clark. Principles of superconductive devices and circuits. *Physics Today*, 35(2):80, 1982.

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

[10] Arnold H. Silver. Superconductor technology for high-end computing system issues and technology roadmap. In *Proceedings of the 2005 ACM/IEEE Conference on Supercomputing*, SC '05, page 64, USA, 2005. IEEE Computer Society.

[11] IARPA - C3. https://www.iarpa.gov/research-programs/c3. Accessed: July 10, 2022.

[12] International roadmap for devices and systems (IRDS). https://irds.ieee.org/. Accessed: July 10, 2022.

[13] W. Chen, A.V. Rylyakov, V. Patel, J.E. Lukens, and K.K. Likharev. Rapid single flux quantum T-flip flop operating up to 770 GHz. *IEEE Transactions on Applied Superconductivity*, 9(2):3212–3215, 1999.

[14] Ikki Nagaoka, Masamitsu Tanaka, Koji Inoue, and Akira Fujimaki. 29.3 A 48GHz 5.6mW gate-level-pipelined multiplier using single-flux quantum logic. In *2019 IEEE International Solid- State Circuits Conference - (ISSCC)*, pages 460–462, 2019.

[15] Ikki Nagaoka, Masamitsu Tanaka, Kyosuke Sano, Taro Yamashita, Akira Fujimaki, and Koji 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 *2019 IEEE International Superconductive Electronics Conference (ISEC)*, pages 1–3. IEEE, 2019.

[16] Swamit S Tannu, Poulami Das, Michael L Lewis, Robert Krick, Douglas M Carmean, and Moinuddin K Qureshi. A case for superconducting accelerators. In *Proceedings of the 16th ACM International Conference on Computing Frontiers*, pages 67–75, 2019.

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

[18] Norman P Jouppi, Cliff Young, Nishant Patil, David Patterson, Gaurav Agrawal, Raminder Bajwa, Sarah Bates, Suresh Bhatia, Nan Boden, Al Borchers, et al. In-datacenter performance analysis of a tensor processing unit. In *Proceedings of the 44th annual international symposium on computer architecture*, pages 1–12, 2017.

[19] Ruizhe Cai, Ao Ren, Olivia Chen, Ning Liu, Caiwen Ding, Xuehai Qian, Jie Han, Wenhuo Luo, Nobuyuki Yoshikawa, and Yanzhi Wang. A stochastic-computing based deep learning framework using adiabatic quantum-flux-parametron superconducting technology. In *2019 ACM/IEEE 46th Annual International Symposium on Computer Architecture (ISCA)*, pages 567–578. IEEE, 2019.

[20] Georgios Tzimpragos, Dilip Vasudevan, Nestan Tsiskaridze, George Michelogiannakis, Advait Madhavan, Jennifer Volk, John Shalf, and Timothy Sherwood. *A Computational Temporal Logic for Superconducting Accelerators*, pages 435–448. Association for Computing Machinery, New York, NY, USA, 2020.

[21] Georgios Tzimpragos, Jennifer Volk, Dilip Vasudevan, Nestan Tsiskaridze, George Michelogiannakis, Advait Madhavan, John Shalf, and Timothy Sherwood. Temporal computing with superconductors. *IEEE Micro*, 41(3):71–79, 2021.

[22] Haipeng Zha, Naveen Kumar Katam, Massoud Pedram, and Murali Annavaram. HiPerRF: A dual-bit dense storage SFQ register file. In *2022 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, pages 415–428. IEEE, 2022.

[23] Farzaneh Zokaee and Lei Jiang. SMART: A heterogeneous scratchpad memory architecture for superconductor SFQ-based systolic CNN accelerators. In *MICRO-54: 54th Annual IEEE/ACM International Symposium on Microarchitecture*, pages 912–924, 2021.

[24] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[25] Heike Kamerlingh Onnes. Investigations into the properties of substances at low temperatures, which have led, amongst other things, to the preparation of liquid helium. *Nobel lecture*, 4:306–336, 1913.

[26] T. Hirose, T. Asai, and Y. Amemiya. Pulsed neural networks consisting of single-flux-quantum spiking neurons. *Physica C: Superconductivity and its Applications*, 463-465:1072–1075, 2007.

[27] Gleb Krylov and Eby G Friedman. *Single Flux Quantum Integrated Circuit Design*. Springer Cham, 2021.

[28] K.K. Likharev and V.K. Semenov. RSFQ logic/memory family: a new josephson-junction technology for sub-terahertz-clock-frequency digital systems. *IEEE Transactions on Applied Superconductivity*, 1(1):3–28, 1991.

[29] Ramy N. Tadros. *Clocking Solutions for SFQ Circuits*. PhD thesis, 2019.

[30] Kris Gaj, Eby G. Friedman, and Marc J. Feldman. Timing of multi-gigahertz rapid single flux quantum digital circuits. *Journal of VLSI signal processing systems for signal, image and video technology*, 16:247–276, 1997.

[31] Ghasem Pasandi, Alireza Shafaei, and Massoud Pedram. SFQmap: A technology mapping tool for single flux quantum logic circuits. In *2018 IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 1–5, 2018.

[32] Oleg A. Mukhanov. Energy-efficient single flux quantum technology. *IEEE Transactions on Applied Superconductivity*, 21(3):760–769, 2011.

[33] DE Kirichenko, Saad Sarwana, and AF Kirichenko. Zero static power dissipation biasing of RSFQ circuits. *IEEE Transactions on Applied Superconductivity*, 21(3):776–779, 2011.

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

[35] Mohammad Rastegari, Vicente Ordonez, Joseph Redmon, and Ali Farhadi. XNOR-Net: Imagenet classification using binary convolutional neural networks, 2016.

[36] Matthieu Courbariaux, Yoshua Bengio, and Jean-Pierre David. BinaryConnect: Training deep neural networks with binary weights during propagations. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'15, pages 3123–3131, Cambridge, MA, USA, 2015. MIT Press.

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

[38] B. Parhami and Chi-Hsiang Yeh. Accumulative parallel counters. In *Conference Record of The Twenty-Ninth Asilomar Conference on Signals, Systems and Computers*, volume 2, pages 966–970, 1995.

[39] Mikhail Dorojevets, Zuoting Chen, Christopher L Ayala, and Artur K Kasperek. Towards 32-bit energy-efficient superconductor rql processors: The cell-level design and analysis of key processing and on-chip storage units. *IEEE Transactions on Applied Superconductivity*, 25(3):1–8, 2014.

[40] Mikhail Dorojevets and Zuoting Chen. Fast pipelined storage for high-performance energy-efficient computing with superconductor technology. In *2015 12th International Conference & Expo on Emerging Technologies for a Smarter World*, pages 1–6. IEEE, 2015.

[41] L. Schindler. RSFQ cell library. https://github.com/sunmagnetics/RSFQlib. Version: 2.1.1, Release date: 24 August 2021.

[42] S.V. Polonsky, V.K. Semenov, and A.F. Kirichenko. Single flux, quantum B flip-flop and its possible applications. *IEEE Transactions on Applied Superconductivity*, 4(1):9–18, 1994.

[43] Tifenn Hirtzlin, Bogdan Penkovsky, Marc Bocquet, Jacques-Olivier Klein, Jean-Michel Portal, and Damien Querlioz. Stochastic computing for hardware implementation of binarized neural networks. *IEEE Access*, 7:76394–76403, 2019.

[44] Dongmoon Min, Ilkwon Byun, Gyu-Hyeon Lee, Seongmin Na, and Jangwoo Kim. CryoCache: A fast, large, and cost-effective cache architecture for cryogenic computing. In *Proceedings of the 25th International Conference on Architectural Support for Programming Languages and Operating Systems*, pages 449–464, 2020.

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

[46] Jongseok Park, Sushil Subramanian, Lester Lampert, Todor Mladenov, Ilya Klotchkov, Dileep J Kurian, Esdras Juarez-Hernandez, Brando Perez Esparza, Sirisha Rani Kale, Asma Beevi KT, et al. A fully integrated Cryo-CMOS SoC for state manipulation, readout, and high-speed gate pulsing of spin qubits. *IEEE Journal of Solid-State Circuits*, 56(11):3289–3306, 2021.

[47] Hongliang Zhao and Xinghui Liu. Modeling of a standard 0.35 μm CMOS technology operating from 77 k to 300 k. *Cryogenics*, 59:49–59, 2014.

[48] Predictive technology model. https://ptm.asu.edu/. Accessed: September 5, 2022.

Rongliang Fu received his BS degree in software engineering from the Northwestern Polytechnical University, Xi'an, China, in 2018 and his MS degree in computer science and technology from the University of Chinese Academy of Sciences, Beijing, China, in 2021. His research interests include electronic design automation and computer architecture.

Junying Huang received her Ph.D. degree in microelectronics and solid-state electronics from the University of Chinese Academy of Sciences in 2016. Currently she is an associate professor with the Department of High-throughput Computer Research Center, Institute of Computing Technology, Chinese Academy of Sciences. Her research interests include superconductive RSFQ logic, computer architecture, electronic design automation, and hardware security.

Haibin Wu received the MS degree in computer science and technology from the University of Chinese Academy of Sciences, Beijing, China. His research interests include hardware accelerators for neural networks and computer architecture.

Xiaochun Ye received the Ph.D. degree in computer architecture from the Institute of Computing Technology, Chinese Academy of Sciences (CAS), in 2010. Currently he is a Professor Researcher, director of the High-Throughput Computer Research Center in Institute of Computing Technology, CAS. His main research interests include many-core processor structure and software simulation technology.

Dongrui Fan received the Ph.D. degree in computer architecture from the Institute of Computing Technology (ICT), Chinese Academy of Sciences (CAS), in 2005. Currently he is a Professor researcher in the ICT, CAS. He is also a senior member of the China Computer Federation (CCF) and a Distinguished Researcher of the Chinese Academy of Sciences. His main research direction includes designs for many-core processors, high-throughput processors, system design, and data flow computing.

Tsung-Yi Ho received his Ph.D. in Electrical Engineering from National Taiwan University in 2005. Currently he is the professor of the Department of Computer Science and Engineering at The Chinese University of Hong Kong, and also serves as the VP Technical Activities of IEEE CEDA and the Executive Committee of ASP-DAC and ICCAD. His research interests include several areas of computing and emerging technologies, especially in design automation of microfluidic biochips. He is an ACM Distinguished Member and a Global STEM Scholar of Hong Kong.