SMART: A Heterogeneous Scratchpad Memory Architecture for Superconductor SFQ-based Systolic CNN Accelerators

Farzaneh Zokaee
fzokaee@iu.edu
Indiana University
Bloomington, USA

Lei Jiang
jiang60@iu.edu
Indiana University
Bloomington, USA

ABSTRACT

Ultra-fast & low-power superconductor single-flux-quantum (SFQ)-based CNN systolic accelerators are built to enhance the CNN inference throughput. However, shift-register (SHIFT)-based scratchpad memory (SPM) arrays prevent a SFQ CNN accelerator from exceeding 40% of its peak throughput, due to the lack of random access capability. This paper first documents our study of a variety of cryogenic memory technologies, including Vortex Transition Memory (VTM), Josephson-CMOS SRAM, MRAM, and Superconducting Nanowire Memory, during which we found that none of the aforementioned technologies made a SFQ CNN accelerator achieve high throughput, small area, and low power simultaneously. Second, we present a heterogeneous SPM architecture, SMART, composed of SHIFT arrays and a random access array to improve the inference throughput of a SFQ CNN systolic accelerator. Third, we propose a fast, low-power and dense pipelined random access CMOS-SFQ array by building SFQ passive-transmission-line-based H-Trees that connect CMOS sub-banks. Finally, we create an ILP-based compiler to deploy CNN models on SMART. Experimental results show that, with the same chip area overhead, compared to the latest SHIFT-based SFQ CNN accelerator, SMART improves the inference throughput by 3.9× (2.2×), and reduces the inference energy by 86% (71%) when inferring a single image (a batch of images).

CCS CONCEPTS

• **Hardware → Quantum technologies**; **Static memory**; **Logic circuits**; **Memory and dense storage**.

KEYWORDS

scratchpad memory, single-flux-quantum, CNN accelerator

ACM Reference Format:
Farzaneh Zokaee and Lei Jiang. 2021. SMART: A Heterogeneous Scratchpad Memory Architecture for Superconductor SFQ-based Systolic CNN Accelerators. In *MICRO-54: 54th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO '21), October 18–22, 2021, Virtual Event, Greece.* ACM, New York, NY, USA, 13 pages. https://doi.org/10.1145/3466752.3480041

MICRO '21, October 18–22, 2021, Virtual Event, Greece

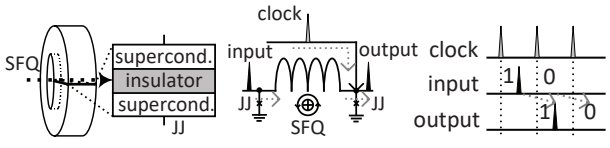
ACM ISBN 978-1-4503-8557-2/21/10…$15.00
https://doi.org/10.1145/3466752.3480041

1 INTRODUCTION

Deep learning has been the dominant approach to solving a wide variety of problems such as computer vision [24], natural language processing, and recommender systems. However, an inference of convolutional neural networks (CNNs) requires a multitude of computing-intensive convolutions. For instance, an AlexNet inference [24] costs 1.5 billion multiply-accumulate (MAC) operations involving 61 million parameters. As the era of Moore's law draws to a close, recent work [17] builds a systolic CNN accelerator, SuperNPU, to process CNN inferences by superconductor SFQ logic. The SFQ technology [30, 57] enables a low-level voltage impulse-driven switching, so that SFQ-based designs can achieve extremely high frequency (e.g., ~ 70 GHz) but consume only tiny energy (e.g., 10^{-19} J per switching). SuperNPU [17] is designed to run at 52 GHz by consuming only 1.9 W power. Compared to the state-of-the-art (SOTA) CMOS TPU [21], SuperNPU improves the batch inference throughput of various CNNs by 23×.

Unfortunately, the inference throughput of SFQ-based systolic CNN accelerators is seriously limited by their on-chip scratchpad memory (SPM) *arrays.* SFQ logic gates can naturally implement the gate-level pipelining, i.e., a clock pulse triggers a SFQ gate to transfer the stored SFQ to its adjacent gates. By a pulse-driven clock, SFQ circuits flow many data pulses through one wire simultaneously to achieve high operating frequency. However, SFQ-based decoders cost significant hardware overhead [36, 37], because the maximal fan-out of a SFQ gate is only 2 [40]. Therefore, it is economical and convenient to implement shift-register-based memory (SHIFT) arrays comprising only serially-connected delay-flip-flops for a SFQ systolic CNN accelerator, since SHIFT fully utilizes the SFQ gate-level pipelining and does not require complex controls. However, SHIFT makes the SOTA SFQ systolic CNN accelerator SuperNPU [17] achieve only 40% of its maximal inference throughput when processing a large batch of images, due to the lack of random access capability. Moreover, SuperNPU can only reach 16% of its peak inference throughput when inferring a single image. Nowadays most clients are sensitive to the end-to-end latency of cloud-based services. It is more likely for data centers [13] to process CNN inferences with only small batch sizes, e.g., one image, simply because they are required to respond the clients rapidly and have no time to form a large batch.

It is difficult to construct a fast, dense, and power-efficient on-chip SPM architecture with random access capability for SFQ CNN accelerators by prior cryogenic memory technologies. SFQ logic works only at the 4K cryogenic temperature, so the SPM of a SFQ-based CNN accelerator has to use cryogenic memory technologies that can maintain their functionality and reliability at 4K. SOTA



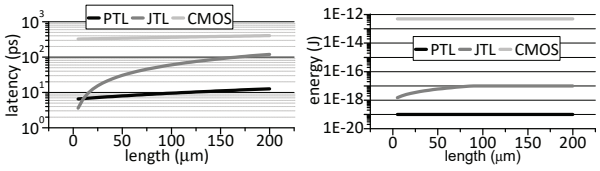
(a) A SFQ ring. (b) A SFQ DFF. (c) DFF operations.

Figure 1: Josephson Junction and SFQ Delay-Flip-Flop.

cryogenic memory technologies include Vortex Transition Memory (VTM) [44, 46], Josephson-CMOS SRAM [11, 37, 48], Magnetic Memory (MRAM) [38], and Superconducting Nanowire Memory (SNM) [3, 61]. First, prior cryogenic memory technologies use SFQ-based decoders, thereby suffering from large hardware overhead, due to the fan-out limitation of SFQ gates. Second, the scalability of VTM is poor, although accessing a VTM array costs only 0.1 ns. A VTM cell [44] is composed of four Josephson Junctions (JJs) and occupies 99 μm^2 at the 600 $\mu A/\mu m^2$ technology. A large capacity VTM-based SPM requires prohibitively large chip area. Third, Josephson-CMOS SRAM, MRAM, and SNM have too long access latency to match the ultra-high operating frequency of a SFQ CNN accelerator. For instance, accessing a 28 MB SRAM array at 4K requires 2~4 ns, while writing a MRAM or SNM cell costs >2 ns. Such long access latency seriously deteriorates the inference throughput of a SFQ CNN accelerator.

In this paper, we propose a novel heterogeneous **S**cratchpad **M**emory **AR**chi**T**ecture, **SMART**, for SFQ systolic CNN accelerators to improve their inference throughput. Our contributions are summarized as follows.

- **A comparison of cryogenic memory technologies**: We compared a variety of SFQ-compatible cryogenic memory technologies including VTM, Josephson-CMOS SRAM, MRAM, and SNM on the SOTA SFQ systolic CNN accelerator, SuperNPU. We found that no prior cryogenic memory technology can support SuperNPU to obtain high inference throughput, low power consumption, and small hardware overhead at the same time.
- **A heterogeneous SPM architecture**: We present a heterogeneous SPM architecture that combines SHIFT arrays and a random-access-memory (RANDOM) array to support ultra-fast sequential accesses and fast random accesses. A SFQ CNN accelerator can store its sequentially accessed data in SHIFT arrays and randomly accessed data in the RANDOM array separately.
- **A pipelined CMOS-SFQ RANDOM array**: We propose a dense CMOS-SFQ RANDOM array for SMART to achieve fast and power-efficient random accesses. We built a pipelined SFQ-based H-Tree by SFQ passive transmission lines (PTLs) to decrease the access latency and energy consumption. Our pipelined CMOS-SFQ array uses SFQ-based H-Trees to connect CMOS sub-banks, each of which consists of SRAM cells and CMOS peripherals, e.g., row decoders, column multiplexers, and sense amplifiers.
- **An ILP-based compiler**: We formulated the allocation and prefetching of input, weight, output, and PSum data to SMART as an integer-linear-programming (ILP) problem. Our ILP-based compiler makes near-optimal schedules for various CNN models on a SFQ systolic CNN accelerator with SMART.
- **Inference throughput and throughput per Watt**: We evaluated and compared SMART to the SOTA SFQ systolic CNN accelerator, SuperNPU. Under the same area constraint, compared to SuperNPU, SMART improves the inference throughput by 3.9× (2.2×), and reduces the inference energy by 86% (71%) when inferring a single image (a batch of images).



(a) Latency. (b) Energy.

Figure 2: A comparison between SFQ and CMOS wires.

The paper is organized as: SFQ logic and cryogenic memories are introduced in Section 2. Section 3 describes design motivation. SMART is proposed in Section 4. We present experiment methodology and results in Section 5 and Section 6 respectively. Related work is presented in Section 7, followed by our conclusion in Section 8.

2 BACKGROUND

2.1 SFQ Technology

Josephson Junction. Superconductor SFQ logic [26, 49] is one of the most promising emerging technologies for ultra-fast and low-power computing at cryogenic temperatures. A basic element of SFQ technology, i.e., a superconductor ring [26], is shown in Figure 1(a). Instead of voltage levels in CMOS logic, SFQ circuits use the existence of a single magnetic flux quantum (SFQ) in the superconductor ring to represent "1" or "0". A superconductor ring stores and transfers the SFQ by Josephson junctions (JJs) [50, 51], each of which consists of a thin insulator sandwiched by two superconductors. A JJ can reliably operate at ~ 70 GHz. Each JJ switching costs only ~ 10^{-19} J.

SFQ Delay-Flip-Flop. To explain the working mechanism of SFQ logic, we use a SFQ-based delay-flip-flop (DFF) as an example because of its simple structure, i.e., it consists of only a single superconductor ring and a clock line. As Figure 1(b) shows, an input pulse makes the current flowing through the left JJ higher than its *critical current* I_c. And then, the left JJ produces a voltage pulse, which is stored in the ring as a SFQ. When a clock pulse arrives, the right JJ is activated, and the SFQ in the ring is outputted as a voltage pulse. A SFQ DFF passes a "1" as the existence of the stored SFQ between two clock pulses, as shown in Figure 1(c). In contrast, if there is no input pulse during a clock period, no voltage pulse ("0") is produced on the output. Several chips [33, 34] composed of SFQ logic units and memories are fabricated and demonstrated at tens of GHz.

SFQ Interconnect. SFQ logic components are connected by active Josephson transmission lines (JTLs) and passive transmission lines (PTLs) [43]. As Figure 2(a) shows, compared to a CMOS wire, JTL and PTL enjoy two orders of magnitude shorter latency, since they have no DC resistance [18, 19]. A PTL requires a much smaller delay than a JTL, particularly when the length is large. Furthermore, the energy comparison between CMOS and SFQ interconnects is shown in Figure 2(b). The energy of a CMOS wire is roughly six orders of magnitude greater than the energy dissipated by a PTL. To implement a long line, a JTL consumes 100× more energy than a PTL.

SFQ Fan-out. Unlike CMOS logic, each SFQ gate can drive only one other node [22, 40], due to the use of SFQ pulses. That is to say,

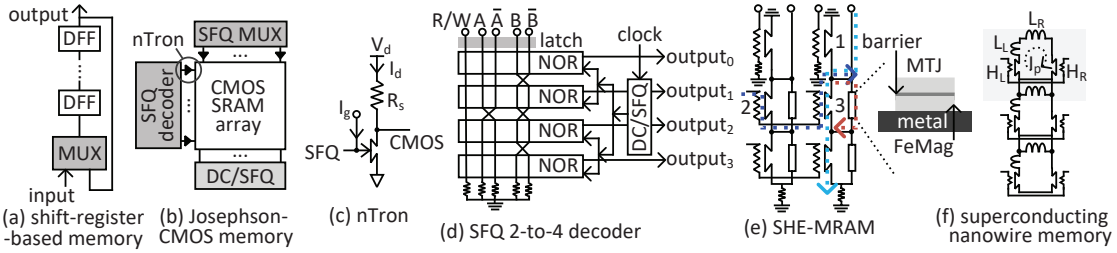


Figure 3: Various cryogenic memory technologies and their components.

the fan-out of a SFQ gate is only one. If a gate needs to have >1 fan-out, a SFQ splitter is required to be inserted at the output of the gate to enable a fan-out of two. To support additional fan-outs, a binary tree of SFQ splitters can be used. Because of the fan-out constraint, it is expensive to implement peripherals of a memory array by SFQ logic. For instance, a SFQ 4-to-16 decoder fabricated by the NEC Nb standard process occupies 885 μm×350 μm [35], i.e., $77K\,\mathcal{F}^2$, where we define $\mathcal{F}$ as the diameter of a JJ. However, we synthesized a 28 nm CMOS 4-to-16 decoder occupying only 18.7 μm^2, i.e., $23K\,F^2$, where F is the technology node size, i.e., 28 nm.

CMOS Compatibility. Superconducting SFQ technology is CMOS compatible [41]. A CMOS SRAM array and SFQ peripherals have been successfully fabricated on the same wafer [11]. CMOS circuits optimized for cryogenic temperatures are first fabricated on a wafer. SFQ logic can subsequently be fabricated on the same wafer using standard SFQ process technology [11].

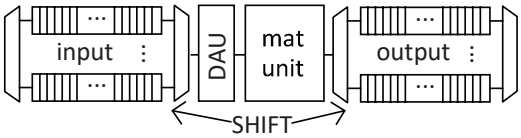


Figure 4: SuperNPU: a SFQ-based systolic CNN accelerator (DAU: data alignment unit).

2.2 SuperNPU and SHIFT

To accelerate deep learning inferences, a recent work [17] proposes a SFQ systolic CNN accelerator, SuperNPU, as shown in Figure 4. Due to the gate-level pipelining and the pulse-driven clocking, it would be easy to implement systolic and pipelined matrix multiplication units that can operate at 52.6 GHz with low power consumption by SFQ logic. Instead of power-hungry hardware-managed caches [1], SuperNPU uses only SHIFT [17] as its on-chip SPM arrays to store input, weight, output, and PSum data. As Figure 3(a) shows, SHIFT comprises serially connected DFFs and a feedback loop. As Table 1 describes, due to its simple structure, SHIFT can achieve ultra-short access latency, high density, and low power consumption. An access to a SHIFT cell requires only 0.02 ns and consumes only 0.1 fJ. A SHIFT cell occupies only $39\,\mathcal{F}^2$, where $\mathcal{F}$ is the diameter of a JJ. However, SHIFT arrays seriously limit the inference throughput of SuperNPU, i.e., sequentially accessing CNN data makes SuperNPU achieve only 40% of its peak inference throughput even when processing a batch of images.

2.3 Cryogenic Memory

Though SFQ-based computing logic units [10, 15, 23, 39, 47] achieve ultra-high operating frequency and low power consumption, it is challenging to implement low-power and dense random-access-memory (RANDOM) arrays that can match the speed of superconducting computing at 4K. There are several types of cryogenic memory technologies that can serve as on-chip SPM for a SFQ systolic CNN accelerator.

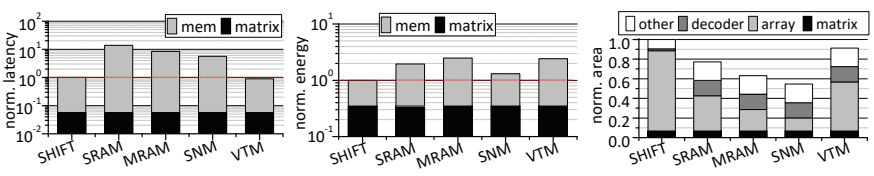
Vortex Transition Memory (VTM). JJ-based Vortex Transition Memory (VTM) [44, 46] has been demonstrated at the scale of 512-byte. However, VTM suffers from poor scalability. As Table 1 shows, each VTM cell [44] consists of four JJs and eight inductors, thereby occupying a cell size of $203\,\mathcal{F}^2$. A VTM cell must use large superconductor rings. It is difficult to create a VTM cell in a smaller size even with self-shunted JJs. As a result, a recent VTM array demonstration [44] achieves only 0.9 Mbit/cm^2 functional density. Accessing a VTM array typically costs 0.1 ns [44, 46].

Table 1: The comparison between cryogenic memories.

Features	SHIFT	VTM	SRAM	MRAM	SNM
Read Latency (ns)	0.02	0.1	2 ~ 4	0.1	0.1
Write Latency (ns)	0.02	0.1	2 ~ 4	2	3
Cell Size	$39\mathcal{F}^2$	$203\mathcal{F}^2$	$146F^2$	$89\mathcal{F}^2$	$54\mathcal{F}^2$
Read Energy	$0.1fJ$	$0.1pJ$	$0.1pJ$	$1pJ$	$10fJ$
Write Energy	$0.1fJ$	$0.1pJ$	$0.1pJ$	$8pJ$	$10fJ$
Leakage Power	no	tiny	medium	tiny	tiny
Random Access	no	yes	yes	yes	yes

Josephson-CMOS SRAM. Due to the SFQ CMOS compatibility, prior work [11, 37, 48, 54] builds a Josephson-CMOS memory array that connects a SFQ decoder and a SFQ multiplexer to a SRAM array via nTrons [60], as shown in Figure 3(b). These works [11, 37, 48, 54] have demonstrated that SRAM can reliably operate at 4K but with faster speed and lower power consumption compared to the room temperature. As Figure 3(c) highlights, nTron is a superconducting device whose superconductivity can be switched by the injection of hot quasiparticles generated at the gate. SFQ circuits can use nTrons to access CMOS components at 10 GHz [60]. Therefore, it is more practical to implement large and reliable cryogenic memory arrays by Josephson-CMOS SRAM, due to the maturity of CMOS SRAM technology. However, it is important to note that SRAM is slow, e.g., accessing a 28 MB SRAM array typically costs 2~4 ns, as shown in Table 1. Moreover, a SFQ-based decoder [37] costs significant hardware overhead. Due to the fan-out limitation, as shown in Figure 3(d), a SFQ-based N-to-2^N decoder requires at least $O(2^N)$ SFQ splitters to distribute its clock pulses. A SFQ decoder [35] is larger than its CMOS counterpart by multiple times, even if JJ can be scaled to the same size of a transistor.

Magnetic Memory (MRAM). To build a fast, dense, and power-efficient cryogenic memory array, recent work [38] suggests a spin hall effect (SHE) magnetic RAM (MRAM) array, as shown in Figure 3(e). A SHE-MRAM cell consists of a SHE magnetic tunnel



(a) Latency. (b) Energy. (c) Area (@28nm).

Figure 5: The comparison of SuperNPU with various cryogenic-memory-technology-based SPM when inferring AlexNet (mem: SPM; matrix: matrix unit).

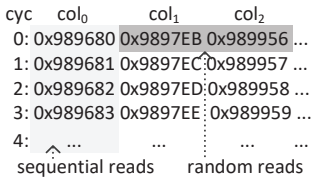


Figure 6: Memory accesses of SuperNPU (cyc: cycle; col: PE array column).

junction (MTJ) and a superconducting heater-cryotron (hTron) bit-select element. A SHE-MTJ consists of a MTJ sitting on a metallic spin hall channel, while a hTron, which is a variant of the nTron, can be driven by SFQ logic and thus supports sufficient current to switch the SHE-MTJ. A SHE-MRAM cell is 89 $\mathcal{F}^2$, as shown in Table 1. Besides SFQ decoders and multiplexers, a SHE-MTJ array is connected to row and column driving hTrons. To write a cell, the SFQ multiplexer sends a triggering pulse to each corresponding column hTron. The bias current (1) flows through all hTrons in the column, which are superconducting. A row hTron is triggered by the SFQ decoder and sends its bias current to all bit-select hTrons in that row (2). For a hTron which receives both the current from the column driver and the current from the row driver, a writing pulse is generated to the SHE-MTJ channel to change the state of the MTJ (3). The switching of a SHE-MRAM typically costs 2 ns [38]. The reading process is similar to that of writing, except that the reading current is much smaller.

Superconducting Nanowire Memory (SNM). A Superconducting Nanowire Memory (SNM) [3, 61] can be also used to build a cryogenic memory array. As Figure 3(f) shows, each SNM cell has two hTrons, such that the right hTron has a larger switching current and larger inductance than the left hTron. The two hTrons are connected serially so that both hTrons are modulated by the same current. The cell has four connections arranged in two electrically isolated pairs, wherein one is the access port, while the other is the select port. As Table 1 shows, a SNM cell is only 54 $\mathcal{F}^2$. To write a cell, a bias current is applied to the column, and flows through all the cells within the column, but its amplitude is too small to alter the state of any cells. A row enabling current is applied to the row. This weakens the channels of the hTrons within the row, thereby allowing the write bias to cause the selected cells to switch. A write operation spends 3 ns [3, 61]. Each read is destructive. After each read, a write operation is required to restore the data.

3 MOTIVATION

In this section, we present the design motivation by comparing the inference latency, energy consumption, and area overhead of SuperNPU [17] with SPMs made by various cryogenic memory technologies. SuperNPU has two 24 MB SHIFT-based SPMs for inputs and outputs/PSums, respectively. We used other cryogenic memory technologies that support random accesses to build a 64-bank 12 MB input SPM, a 256-bank 16 MB output/PSum SPM, and a 64 KB weight SPM for SuperNPU. We evaluated SuperNPU for one-image inferences, thus SPMs with such capacities are large enough for each layer of AlexNet without generating thrashing traffic to DRAM. The configuration of SuperNPU is shown in Section 5.

Inference Latency. As Figure 5(a) shows, SuperNPU using SHIFT spends a huge portion of inference latency in sequentially searching the input and PSum data. If SuperNPU SPMs support random accesses, the inference latency can be reduced. However, since Jose-phson-CMOS SRAM, VTM, MRAM, and SNM have much longer read and write latencies, no prior cryogenic memory technology can significantly reduce the inference latency. The write latencies of SRAM, MRAM, and SNM are >2 ns, they prolong the inference latency of SuperNPU by at least 5×. Only VTM decreases the inference latency of SuperNPU by 11% over SHIFT, since the latency saving introduced by its random access capability is larger than the slowdown caused by its prolonged access latency. If there were a random access array with 0.02 ns latency, SuperNPU would have eliminated memory access stalls. Such fast random access arrays can reduce the inference latency of SuperNPU by 94%.

Inference Energy. The energy comparison of various types of on-chip SPM arrays is shown in Figure 5(b). Since all the other cryogenic memory technologies have larger read and write energy than SHIFT, they enlarge the energy of an AlexNet inference by 30%∼2.5× over SHIFT. Although CMOS SRAM dissipates large leakage power at room temperatures, the cryogenic temperatures substantially reduce leakage by >90% [28]. As a result, the large write energy makes cryogenic SHE-MRAM consume even more energy than Josephson-CMOS SRAM.

Area Overhead. The area comparison between various types of on-chip SPM arrays is highlighted in Figure 5(c). SuperNPU [17] assumes JJs can be scaled to 28 nm. We adopted the same assumption for SHIFT-, MRAM-, SNM-, and VTM-based SPM arrays. We also assumed SRAM arrays are fabricated at 28 nm. The SHIFT SPMs of SuperNPU have few SFQ decoders and multiplexers to select banks, each of which is a long lane of SHIFT memory cells. Although the capacity of MRAM-, SNM-, and VTM-based SPM arrays is 58% of that of SHIFT, they can reduce from 8% to 45% of the area. This is because they use more SFQ peripherals and have larger cells, which are demonstrated in Table 1. Particularly, SFQ-based decoders cost 16%∼ 28% of the area in non-SHIFT arrays. Due to the fact that Josephson-CMOS SRAM has the second largest cell size, compared to SHIFT, the Josephson-CMOS SRAM array with a 58% capacity reduces the area by only 22%.

Drawbacks of Prior Cryogenic Memories. Compared to the perfect pipeline without memory stall, the SHIFT-based SPMs prolong the inference latency of SuperNPU by 17×, due to the fact that it only supports sequential reads. As the memory traces in Figure 6 show, when SuperNPU reads weights, it has both sequential and random reads. Although SHIFT-based SPM can efficiently process sequential reads, it also has to move many unnecessary bits to support random accesses. Josephson-CMOS-SRAM-, MRAM-, SNM-, and VTM-based SPM arrays can perform random accesses, but they cannot achieve reasonable latency reduction, since their

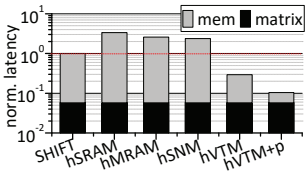


Figure 7: The inference latency comparison of a heterogeneous SPM.

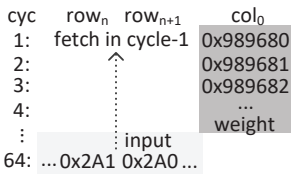


Figure 8: SMART prefetching (cyc: cycle; row: PE array row; col: PE array column).

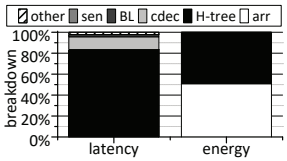


Figure 9: The latency & energy of CMOS H-Trees in 28 MB Josephson-CMOS array with 256 banks.

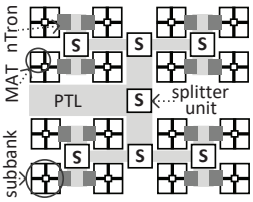


Figure 10: A CMOS-SFQ array.

read or/and write latency are too long. MRAM and SNM are bottlenecked by their write latency and energy. Despite that VTM has the shortest access latency among prior cryogenic memory technologies, it is still not fast enough to make an observable latency reduction. Furthermore, the large VTM cell size significantly enlarges the array area. Thus, although the SFQ peripherals of Josephson-CMOS SRAM are very fast, CMOS H-Trees [28] inside SRAM arrays greatly degrade the access latency and energy. The area efficiency of Josephson-CMOS-SRAM-, MRAM-, SNM-, and VTM-based SPM arrays are limited by SFQ peripherals. In summary, no prior cryogenic memory technology is a good candidate to implement on-chip SPMs for SuperNPU.

4 SMART

In this section, we propose a heterogeneous SPM architecture, SMART, in order to reduce the inference latency of a SFQ systolic CNN accelerator. SMART is composed of SHIFT arrays performing sequential accesses and a random-access-memory (RANDOM) array supporting random accesses. We further present a fast RANDOM array, i.e., a pipelined SFQ-CMOS array, for SMART to minimize the inference latency, energy and hardware area. A pipelined SFQ-CMOS array uses SFQ PTLs and splitter units to implement H-trees connecting CMOS sub-banks to achieve small access latency and energy. At last, we propose an ILP-based compiler to deploy various CNN models on SMART.

4.1 A Heterogeneous SPM Architecture

We present a heterogeneous SPM architecture consisting of SHIFT arrays and a RANDOM array for a SFQ systolic CNN accelerator. For each convolutional layer, SHIFT arrays store all data receiving sequential accesses, while the RANDOM array is used to support random accesses during an inference. There are two challenges we face when trying to use this heterogeneous SPM architecture to effectively reduce the inference latency of the SFQ systolic accelerator. First, though SHIFT arrays process sequential accesses well, the inference latency of the accelerator is still heavily influenced by the access latency of the RANDOM array. However, it is difficult to build a fast, dense, and power-efficient RANDOM array by prior cryogenic memory technologies. Second, there is no compilation technique that can deploy a CNN and enable prefetching on the heterogeneous SPM architecture. Although data allocation to SPMs has been heavily studied before, prior work [8, 27, 45, 53, 55] focuses only on general-purpose applications running on CPUs.

We elaborate the two challenges in applying heterogeneous SPMs on SuperNPU in Figure 7, where we assume a perfect data allocation for both sequentially accessed data and randomly accessed data. We consider three 32 KB SHIFT arrays for inputs, outputs & PSums, and weights as their SPMs, respectively. All CNN data share a 28 MB 256-bank RANDOM array in the heterogeneous SPM architecture. The RANDOM array can be built by Josephson-CMOS-SRAM, MRAM, SNM, or VTM. We call these heterogeneous SPM schemes hSRAM, hMRAM, hSNM, and hVTM in Figure 7. Compared to SHIFT, hSRAM, hMRAM, and hSNM prolong the inference latency by 3.36×, 2.59×, and 2.38×, respectively. hVTM reduces the inference latency by 70% over SHIFT, due to its short access latency. We find that the RANDOM array access latency in SMART heavily influences the inference latency of the accelerator. This is because for a weight-stationary systolic CNN accelerator, most accesses to input, and output & PSum data are random. The systolic accelerator maintains an iterative computing flow, where weights are first deployed on the matrix unit, inputs are fetched to start a systolic computation, and then the next iteration continues, as shown in Figure 8. Considering the fact that there is no dependency between inputs and weights, if the prefetching of inputs to its SPM is enabled, we can start the systolic computation earlier. As Figure 7 shows, the prefetching (hVTM+p) further reduces the inference latency by 64.4% over hVTM. However, no prior SPM management technique supports prefetching for an accelerator.

4.2 A Pipelined CMOS-SFQ Array

4.2.1 The limitations imposed by CMOS H-trees. In an array, both the address and data of a memory request are routed by H-Trees [31], which make the memory request consistent in its access to all MATs. A memory array has two separate H-Trees including a request network and a reply network. Data and addresses are transferred from the edge of the array to MATs by the request network, while data are sent out from MATs by the reply network. Both the request and reply H-Trees are composed of two parts including a network connecting the array edge to the bank edge, and a network connecting the bank edge to MATs.

The Josephson-CMOS array access latency can be divided into SFQ decoder delay, CMOS H-Tree delay, CMOS decoder delay, CMOS wordline delay, CMOS bitline delay, CMOS sense amplifier delay, and SFQ DC/SFQ delay. Throughout the components, the CMOS H-tree dominates the latency and energy consumption of a large Josephson-CMOS SRAM array at 4K. As Figure 9 shows, the H-tree costs 84% of the access latency, and 49% of the access energy in a 256-bank 28 MB Josephson-CMOS SRAM array. Particularly, in the sub-10*nm* regime, the resistance of copper wires [5] exponentially increases as the process technology scales. Therefore, the latency and energy consumption of H-trees will become more significant in Josephson-CMOS arrays at future process nodes.

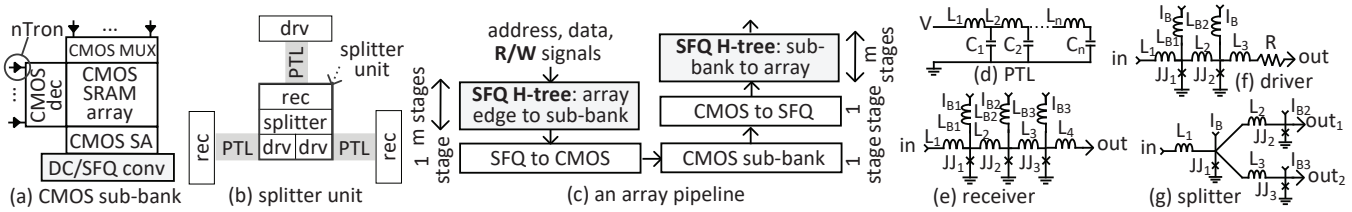
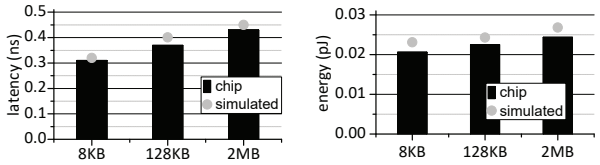


Figure 11: The components and pipeline of a pipelined CMOS-SFQ array.



(a) Latency. (b) Energy.

Figure 12: The validation of a CMOS sub-bank.

4.2.2 A Pipelined CMOS-SFQ Array. **Overall Architecture**. We propose a pipelined CMOS-SFQ array as shown in Figure 10 to reduce the access latency and energy at 4K. Our pipelined CMOS-SFQ array consists of only CMOS sub-banks connected by SFQ H-Trees. The design philosophy of our CMOS-SFQ array is different from Josephson-CMOS SRAM [11, 37, 48]. To avoid the large hardware overhead of SFQ decoders, we use SRAM cells and CMOS peripherals including row decoders, column multiplexers, and sense amplifiers. We use PTL lines and SFQ-based peripherals including splitters, drivers, receivers, and nTrons to build SFQ H-Trees. The major components of our pipelined CMOS-SFQ array can be summarized as follows.

- **CMOS Sub-bank**: As Figure 11(a) shows, CMOS sub-banks of a pipelined CMOS-SFQ array are constructed by SRAM cells and CMOS peripherals including CMOS row decoders, column multiplexers, and sense amplifiers. To drive the row decoders and column multiplexers, we use nTron devices to convert the SFQ memory requests to electrical signals for a CMOS sub-bank. After a CMOS sub-bank makes the data ready, we also use level-driven DC/SFQ converters [48] to transform the data in sense amplifiers into SFQ pulses.
- **SFQ H-Tree**: We use PTL lines to replace all CMOS (e.g., copper) lines in a pipelined CMOS-SFQ array. Due to the fan-out limitation of SFQ logic, we add a splitter unit to each position where the fan-out needs to be increased. The details of a splitter unit can be viewed in Figure 11(b). In order to pass a SFQ pulse via a PTL line, we need a driver at the source end and a receiver at the destination end of the PTL line. A splitter unit consists of a receiver at the input end, two drivers at the two output ends, and a splitter connecting them together.

Pipeline. We propose a multi-stage pipeline architecture for our CMOS-SFQ array in Figure 11(c). To communicate with the SFQ systolic matrix unit, request SFQ H-trees transfer each memory request to a sub-bank from the array edge. nTrons are used to convert the SFQ request to electrical signals that can drive CMOS arrays to fetch (write) the data from (to) the CMOS sub-bank. If the request is a read, level-driven DC/SFQ converters are adopted to convert the electrical signals of the reading data back to SFQ pulses. Finally, the SFQ data pulses are returned to the systolic matrix unit via reply SFQ H-trees. Since splitter units in SFQ H-Trees naturally have gate-level pipelining, multiple memory requests can be transferred simultaneously in the same H-Tree. If we can guarantee all requests go to different sub-banks, a CMOS-SFQ array can process these requests in a pipelined way. To decide the frequency of the pipeline, we identified the operations of nTrons (SFQ to CMOS), CMOS sub-banks, and level-driven DC/SFQ converters as the bottlenecks. Both a nTron and a level-driven DC/SFQ converter [48] can complete a conversion around 0.1 ns. We can limit the latency of each sub-bank within ~0.1 ns by adjusting the number of MATs inside a sub-bank. Then, a H-Tree operation can be broken into multiple pipeline stages by inserting SFQ repeaters, each of which is composed of a driver and a receiver, so that each pipeline stage of H-tree can also fit into ~0.1 ns. The detailed pipeline design space exploration is shown in Section 4.2.4. Since all memory accesses of a systolic CNN accelerator can be known before executions, it is possible to read (write) a line from (to) a pipelined SFQ-CMOS array every ~0.1 ns via data allocation and prefetching.

4.2.3 Modeling and Validation. **Modeling a CMOS Sub-bank at 4K**. We adopted the cryogenic memory model, CryoRAM [25] to model a CMOS SRAM sub-bank. CryoRAM includes a validated cryogenic MOSFET model cryo-pgen, and a CACTI-based cryogenic memory model cryo-mem. Cryo-pgen can derive a variety of MOSFET characteristics at only 77K. We modified cryo-pgen to model MOSFET at 4K by adjusting three fabrication-related and temperature-dependent MOSFET variables including carrier mobility, carrier's saturation velocity, and threshold voltage based on recent cryogenic MOSFET data [2, 12]. Then, we plugged the 4K MOSFET parameters generated by cryo-pgen into cryo-mem to study the access latency and energy of a CMOS array at 4K.

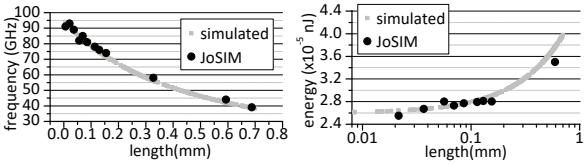
Validating the 4K CMOS Sub-bank Model. We validated the access latency and energy of a CMOS array at 4K generated by cryo-mem against a published 4K SRAM array demonstration [48] fabricated at 0.18 μm. As Figure 12 shows, the 4K SRAM demonstration has three configurations: an 8 KB sub-bank consisting of eight MATs, a 128 KB sub-bank containing 32 MATs, and a 2 MB sub-bank comprised of 128 MATs. The latency values simulated by our modified cryo-mem are larger than those of the 4K SRAM chip by 3%~8% as shown in Figure 12(a), since we applied conservative cryogenic MOSFET parameters to cryo-mem. Our conservative cryogenic MOSFET parameters also make the energy values of our modified cryo-mem larger than those of the 4K SRAM chip by 8%~12%.

Modeling a SFQ H-Tree at 4K. The components of a SFQ H-Tree include the follows.

- **PTL**: We used micro-strip PTLs [20], due to its small size, better scalability and simplicity of geometry. A micro-strip PTL can be represented as a lossless distributed LC network shown in Figure 11(d). The inductance per unit length of a micro-strip PTL

Table 2: The latency and power of SFQ H-Trees.

Component	Latency (ps)	Leakage Power (μW)	Dynamic Power (nW)
Splitter	7	0	0.15
Driver	3.5	0.874	0.181
Receiver	5.25	0	0.275
nTron	103.02	8.8	13



(a) Latency. (b) Energy.

Figure 13: The validation of our SFQ H-Tree model.

(L) [29] is composed of the magnetic inductance introduced by magnetic fluxes within a superconductive line, and the kinetic inductance caused by the motion of paired electrons. L can be calculated as:

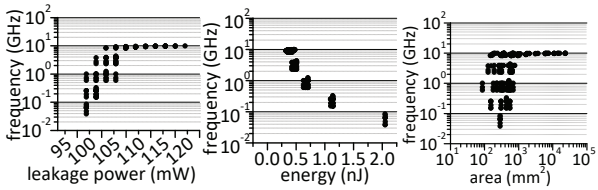
$$L = \frac{\mu_0 h}{Kw}\left[1 + \frac{\lambda_1}{h}\coth\left(\frac{t_1}{\lambda_1}\right) + \frac{\lambda_2}{h}\coth\left(\frac{t_2}{\lambda_2}\right)\right] \quad (1)$$

where w is the line width; t_1 means the thickness of the PTL; t_2 is the thickness of the ground plane of the PTL; K indicates the fringing field factor; h is the thickness of dielectric; λ_1 and λ_2 denote penetration depths of the micro-strip and the ground plane, respectively.

$$C = \frac{\epsilon_r \epsilon_0 w}{h} \quad (2) \qquad Z = \sqrt{\frac{L}{C}} \quad (3) \qquad T = N\sqrt{L \times C} \quad (4)$$

The capacitance per unit length of a micro-strip PTL (C) can be calculated by Equation 2, where w and h are defined in Equation 1; ϵ_r is the dielectric constant of the insulation between the line and ground plane layer; and ϵ_0 is the permittivity of free space. As Equation 3 shows, the impedance of a micro-strip PTL can be derived from the inductance and capacitance per unit. The delay of a micro-strip PTL is a function of total LC, and increases linearly with the line length as shown in Equation 4, where N is the number of LC sections in the micro-strip PTL.

- **Splitter**: Due to the fan-out limitation, a splitter [40] is the core of a splitter unit used to transform a pulse to two pulses, each of which can be sent in one direction of a cross-point in the H-Tree. The structure of a splitter is shown in Figure 11(g), where a SFQ pulse is converted into two flux quanta. A splitter consists of three inductors and three JJs. The latency, and dynamic power of a splitter are shown in Table 2.
- **Driver & Receiver**: As Figure 11(b) shows, a SFQ pulse is sent to a PTL by a driver [43] and received by a receiver [43]. A PTL driver in Figure 11(f) consists of a 2-stage JTL cascaded with a resistance. The JTL acts as both a buffer and a SFQ pulse reconstruction device. A receiver composed of a 3-stage JTL is exhibited in Figure 11(e). The resonance frequency f of a PTL with a driver and a receiver is defined as $f = \frac{1}{2T+t_0}$, where T is the PTL delay, another T avoids the resonance, and t_0 is the delay of a driver and a receiver [6]. The operating frequency of a PTL can be set to at most 90% of f [32]. Otherwise, the resonance effect on the PTL may cause timing jitters and errors. In order to increase the frequency of a PTL, we need to insert more repeaters, each of which consists of a driver and a receiver. Therefore, a long PTL can be partitioned into shorter segments. Inserting repeaters into a PTL increases not only the resonance frequency, but also the power consumption of the PTL. The bias currents and resistors in the bias network of a driver increase the static power, while more JJs introduced by repeater insertion also increase the dynamic power. The area overhead of repeater insertion is proportional to the number of JJs.



(a) Latency. (b) Energy. (c) Area.

Figure 14: The design space exploration.

Validating the 4K SFQ H-Tree Model. We implemented our pipelined SFQ H-Trees (Equation 1~Equation 4) in the CACTI-based cryogenic memory model cryo-mem [25]. We mainly focus on validating the new modules added to cryo-mem including PTL lines and splitter units, each of which consists of a driver, a receiver, and a splitter. Thus, we used a splitter unit shown in Figure 11(b) with various PTL lengths to perform the validation. We measured the latency and energy of passing a SFQ pulse from the top driver to the bottom right receiver, since the two bottom receivers are the same. We ran the superconductor SPICE simulator, JoSIM [7], to validate the results of pipelined CMOS-SFQ arrays generated by our modified cryo-mem. We assumed Hypres ERSFQ $1.0\mu m$ technology [56] to validate the splitter unit. Figure 13(a) exhibits the latency comparison of a splitter unit with various PTL lengths between our model and JoSIM, while their energy correlation is described in Figure 13(b). Compared to the JoSIM HSPICE results, the latency values of a driver and a receiver estimated by our SFQ H-Tree model have ±6% deviations, particularly when the PTL length is $<0.2mm$. The energy values of a SFQ H-Tree predicted by our model are also close to the JoSIM results with ±11% errors.

4.2.4 Pipeline Design Space Exploration. The design space exploration of our pipelined SFQ-CMOS array is exhibited in Figure 14. The bottleneck of the entire pipeline of our SFQ-CMOS array lies in the stage of nTrons, whose latency is 103.02 ps, since we cannot further break the latency into multiple pipeline stages. Therefore, the maximal frequency of our pipelined SFQ-CMOS array is 9.6 GHz. To achieve the maximal pipeline frequency, we adjusted the size of CMOS sub-banks and the frequency of SFQ H-Trees. By reducing the size of CMOS sub-banks, the access latency to sub-banks is reduced to fit into one pipeline stage, since bitlines and wordlines in each MAT become shorter. However, the leakage power and area overhead of a sub-bank increased substantially, since more CMOS peripherals were added into each sub-bank. On the other hand, we inserted drivers and receivers to break a H-Tree into more pipeline stages, each of which has the latency of 103.02 ps. As a result, both the area overhead and access energy of a pipelined SFQ-CMOS array increase.

Table 3: The notations of the ILP formulation.

Notation	Description
$\mathcal{M}$	Memory object: weight (α), input (β), output (γ), PSum (δ)
i	The i_{th} edge in the DAG
ls	SPM access: load ($\mathcal{L}$), and store ($\mathcal{S}$)
st	The status of $\mathcal{M}$: in a SHIFT array (H), in a RANDOM array (R), accesses between H and R (HR), accesses between H and DRAM (HD), accesses between R and DRAM (RD)

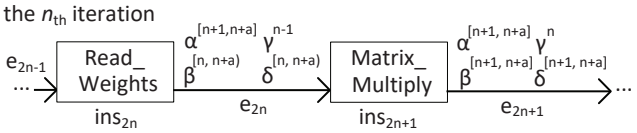


Figure 15: The DAG of a convolutional layer.

4.3 A Compiler for Heterogeneous SPMs

We built a novel compiler to allocate and prefetch memory objects onto SMART composed of SHIFT arrays and a RANDOM array for a SFQ systolic CNN accelerator by integer linear programming (ILP). No prior SPM management technique has the ability to schedule or prefetch memory requests for a systolic CNN accelerator, since prior work [8, 27, 45, 53, 55] focuses on general-purpose applications with multiple basic blocks, each of which is an instruction sequence with no branches in except to the entry and no branches out except at the exit. A convolutional layer is a 6-nested loop [59] belonging to a basic block. Our ILP-based compiler aims to allocate and prefetch memory objects at the instruction level without modifying the computing flow of a systolic CNN accelerator. Instead of 1-byte data, we set the granularity of allocation to memory objects, each of which is a multi-byte data block with consecutive addresses, to capture the temporal and spatial locality. Unlike prior SPM management schemes [8, 27, 45, 53, 55], which assume a memory object is alive throughout the whole basic block, we performed lifespan analysis of each memory object on the directed acyclic graph (DAG) of each convolutional layer to see how many iterations a memory object can live. Our compiler makes the near-optimal memory object allocation and prefetching to SMART on edges of the DAG of a convolutional layer. We designed our ILP-based technique for SMART consisting of private SHIFT arrays for inputs, weights, and PSums/outputs, and a shared RANDOM array for all, to enable data movements between SHIFT and RANDOM arrays, and to decide the schedule of a convolutional layer.

Memory Object: We considered weights (α), inputs (β), outputs (γ), and PSum (δ) results that need to be accumulated as candidates for SPM allocation. An ideal memory trace including all read and write accesses can be generated by the accelerator simulator SCALE-SIM [42] by assuming that there is no delay caused by SPMs and DRAM. To capture fine-grained spatial and temporal locality, we grouped consecutive memory addresses across different processing elements (PEs) or consecutive cycles into one memory object $\mathcal{M}$. A memory object can be a weight filter kernel, a part of the input map, or an output channel.

Lifespan Analysis: We performed the lifespan analysis of memory objects at the instruction level on the DAG of a convolutional layer, as shown in Figure 15. Unlike prior SPM management schemes [8, 27, 45, 53, 55] compiling complex general-purpose applications on a CPU, our compiler focuses on each convolutional layer, which contains only one basic block. To maintain the original computing flow of the systolic CNN accelerator, a convolutional layer is first unrolled and compiled into a DAG. Each node in the DAG is an instruction of the systolic CNN accelerator, e.g., Google TPU [21], which has several types of CISC instructions as follows.

- *Read_Weights*: Sending weights to the Matrix Unit.
- *Matrix_Multiply*: Making the Matrix Unit perform a matrix multiply from the SPMs into accumulators.
- *Activate*: Performing activations and poolings.
- *Write(Read)_Host_Memory*: Writing (Reading) data from SPMs (the CPU memory) to the CPU memory (SPMs).

An edge between two instructions indicates that the destination node has data dependency on the source node. We annotated each edge with its related memory objects. For instance, at e_{2n-1}, i.e., the last edge of the $(n-1)_{th}$ iteration of the layer, the weight objects (α^n) for the next (n_{th}) iteration have to be fetched.

Prefetching. Unlike prior SPM schemes [8, 27, 45, 53, 55], we enable the data fetching of memory objects that will be used in next several iterations by prolonging the lifespan of each memory object. For example, in Figure 15, for the first edge e_{2n} of the n_{th} iteration, besides writing the output objects of the previous $(n-1)_{th}$ iteration (γ^{n-1}), our compiler reads the weight objects $\alpha^{[n+1,n+a]}$ for next a iterations, the input objects $\beta^{[n,n+a)}$ for current and next $(a-1)$ iterations, and the PSum objects $\delta^{[n,n+a)}$ for current and next $(a-1)$ iterations. The allocation and schedule results achieved by our ILP-based compiler are only "near"-optimal, since we do not exhaustively search the best value of a. Instead, we set a fixed value for a.

ILP Variable: We define binary variables of the ILP formulas to attain the near-optimal scheme on a SFQ systolic CNN accelerator with SMART. As Table 3 shows, these variables can be summarized as $\mathcal{M}_{ls}^{i,st}$, where $\mathcal{M}$ can be α, β, γ, or δ; ls can be $\mathcal{L}$ or $\mathcal{S}$; and st can be H, R, HR, HD, and RD. For instance, if an input memory object is allocated to the SHIFT array on the i_{th} edge of the DAG, we have $\beta^{i,H}=1$ and $\beta^{i,R}=0$. Setting a binary variable of SPM access to 1 indicates a load or store is enabled. For example, $\beta_{\mathcal{L}}^{i,HD}=1$ denotes loading the input memory object from the DRAM to the RANDOM SPM on the i_{th} edge of the DAG.

ILP Objective Function: The objective function is to obtain the shortest execution time of each convolutional layer on a systolic CNN accelerator with heterogeneous SPM architecture. The objective function is summarized as

$$\begin{aligned}\max \sum_i \sum_{\mathcal{M}\in\{\alpha,\beta,\gamma,\delta\}} \{&T_s^H\times\mathcal{M}^{i,H}+T_s^R\times\mathcal{M}^{i,R}\\&-T_r^{HD}\times\mathcal{M}_{\mathcal{L}}^{i,HD}-T_r^{RD}\times\mathcal{M}_{\mathcal{L}}^{i,RD}\\&-T_r^{HR}\times\mathcal{M}_{\mathcal{L}}^{i,HR}-T_w^{HD}\times\mathcal{M}_{\mathcal{S}}^{i,HD}\\&-T_w^{RD}\times\mathcal{M}_{\mathcal{S}}^{i,RD}-T_w^{HR}\times\mathcal{M}_{\mathcal{S}}^{i,HR}\}\end{aligned}\tag{5}$$

where T_s^H (T_s^R) is the reduced latency if a memory object is allocated to a SHIFT (RANDOM) array instead of the DRAM. T_r^{HD} / T_r^{RD} / T_r^{HR} is the latency of reading a memory object from DRAM / DRAM / a RANDOM array and writing it to a SHIFT / RANDOM / SHIFT array. T_w^{HD} / T_w^{RD} / T_w^{HR} is the latency of writing a memory

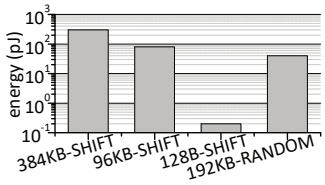


Figure 16: The energy.

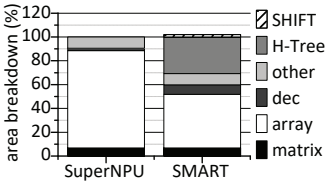


Figure 17: The area.

object back to DRAM / DRAM / a RANDOM array from a SHIFT / RANDOM / SHIFT array.

ILP Constraints: We use the following ILP constraints to guarantee the correctness of the final SPM allocation and schedule of a convolutional layer.

- **DAG and lifespan**: The scheduling and prefetching result has to match the lifespan analysis of memory objects, and the data dependency of the DAG.
- **Consistency of SPM accesses**: The consistency of SPM accesses is enforced by

$$\begin{aligned} &\forall i < j, \quad \mathcal{M}^{j,H} - \mathcal{M}^{j,HD}_{\mathcal{L}} - \mathcal{M}^{j,HR}_{\mathcal{L}} - \mathcal{M}^{i,H} = 0 \\ &\forall i < j, \quad \mathcal{M}^{j,R} - \mathcal{M}^{j,RD}_{\mathcal{L}} - \mathcal{M}^{i,R} = 0 \\ &\forall i < j, \quad \mathcal{M}^{j,HR}_{\mathcal{L}} - \mathcal{M}^{i,R} \leq 0 \end{aligned} \tag{6}$$

  If we allocate a memory object to a SHIFT array on an edge e_j, as displayed in the first line of Equation 6, this memory object should be either allocated in the same array on a prior edge e_i ($i < j$) or loaded to this SPM on edge e_j. The second line guarantees the consistency of SPM accesses in the RANDOM array. The last line enforces the memory object should be already allocated to the RANDOM array on edge e_i, if it is loaded to a SHIFT array on edge e_j from this RANDOM array.
- **SPM size**: The aggregate size of all memory objects allocated to the same array cannot exceed the array size.
- **SPM bandwidth**: The total read (write) bandwidth of a SPM cannot exceed its maximal read (write) bandwidth.
- **Sub-bank**: If two requests are scheduled to the same sub-bank at the same time, they are processed sequentially.

4.4 Design Overhead

The Heterogeneous SPM. SuperNPU [17] has a 24 MB 64-bank input SHIFT buffer, a 24 MB 256-bank output/PSum SHIFT buffer, and a 128 KB weight SHIFT buffer. In contrast, SMART has three 256-bank 32 KB SHIFT arrays for inputs, outputs/PSums, and weights, respectively. It also has a 256-bank 28 MB SFQ-CMOS SRAM array that can be operated at 9.7 GHz for all data.

- *Latency*: The access latency of a SHIFT array is 0.02 ns, while a SFQ-CMOS bank can read or write 1-byte data each 0.11 ns.
- *Leakage*: A SHIFT array has no leakage, but the leakage power consumption of the pipelined SFQ-CMOS SRAM array is 102 mW.
- *Dynamic energy*: As Figure 16 shows, compared to a 384KB or 96KB bank of SuperNPU, the SHIFT arrays of SMART move only 128 DFFs per access, thereby reducing the access energy by 99%. The access to the SFQ-CMOS array of SMART costs only 50% of the dynamic energy of accessing the 96KB bank SuperNPU, due to low-power SFQ H-Trees.
- *Area*: Compared to SuperNPU, SMART reduces the SPM capacity by 41%. But it has more CMOS sub-banks and more repeaters in SFQ H-Trees to achieve 9.7 GHz. As Figure 17 shows, SMART increases the area by 3%, when we assume SFQ JJs and CMOS transistors can be scaled to 28nm [17].

Table 4: The baseline configuration.

Name	Description
TPU	0.7GHz; 45 TMAC/s peak perf.; PE array size 256 × 256; input, weight, and output: 24 MB; PSum: 4 MB
SuperNPU	52.6GHz; 842 TMAC/s peak perf.; PE array size 64 × 256; input: 64-bank, 24 MB; output/PSum: 256-bank, 24 MB; weight: 128 KB, 0.02 ns
SMART	52.6GHz; 842 TMAC/s peak perf.; PE array size 64 × 256; three 32 KB SHIFT arrays for inputs, outputs/PSums, and weights: 256-bank, 0.02 ns; a 28 MB SFQ- CMOS array: 256-bank, 0.11 ns

The ILP-based Compiler. We used SCALE-SIM [42] to extract the DAGs of each CNN model, and identify memory objects. We adopted the Gurobi ILP solver [14] to solve our ILP equations. For each of our CNN models (shown in Section 5), the ILP solver can find a solution within one hour.

5 EXPERIMENTAL METHODOLOGY

Simulation. We used SCALE-SIM [42] to model SMART, and our baselines including CMOS-based Google TPU [21] and superconducting SFQ-based SuperNPU [17]. SCALE-SIM supports cycle-acc-urate performance simulations of a systolic CNN accelerator running inferences. The configurations of SMART and our baselines are shown in Table 4. We set the memory bandwidth of TPU, SuperNPU, and SMART to 300 GB/s. The average power consumption of TPU is 40W [21], while the power consumption of SuperNPU fabricated by the Hypres ERSFQ 1.0μm technology [56] is only 1.9W. We assume all components of SMART are also fabricated by the same ERSFQ 1.0μm technology. The cooling cost of SuperNPU and SMART at 4K is 400× [16] of their power consumption.

CNN Models. We selected six CNN models that have different characteristics, e.g., computational intensity, network topology and on-chip memory bandwidth needs. We ran single-image and batch-based inferences on baselines. The batch size setting is the same as [17]. For TPU and SMART, in a batch, AlexNet has 22 images, while VGG16 has 3 images. All the other models have 20 images in a batch. For SuperNPU, since it has larger SPMs, except VGG16 having 7 images in a batch, all the other models have 30 images in each batch.

Cryogenic Memory Modeling. The details of SFQ-CMOS array modeling can be found in Section 4.2.2. We modified the cryogenic memory model cryo-mem [25] to derive the access latency, energy consumption and area of VTM, MRAM, SNM arrays with the memory parameters in Table 1. We validated the simulated results of cryo-mem on VTM, MRAM, SNM arrays against their published array demonstrations [3, 38, 44] respectively. We observed at most a 14% error between the cryo-mem simulated data and the fabricated array. Compared to the large performance and energy degradation caused by VTM, MRAM, SNM arrays, the errors of cryo-mem are not significant.

Schemes. Besides our baseline TPU, we implemented and compared the following schemes:

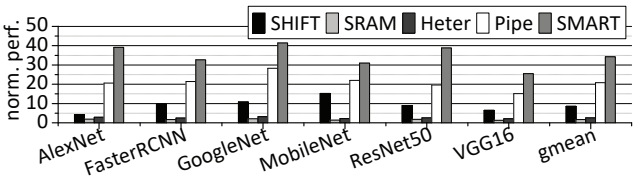


Figure 18: The single-image speedup (norm. to TPU).

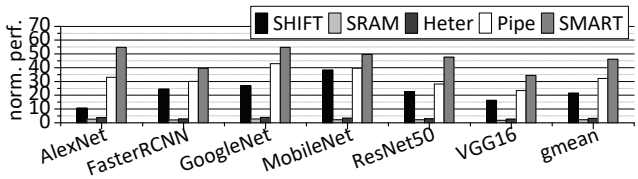


Figure 19: The batch speedup (norm. to TPU).

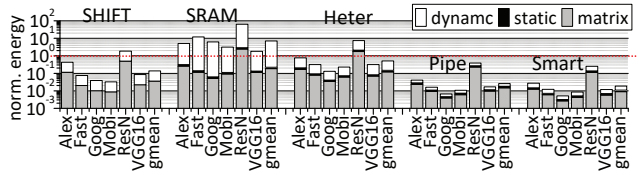


Figure 20: The single image energy reduction (norm. to TPU; matrix: matrix unit energy; dynamic: SPM dynamic energy; and static: SPM static energy).

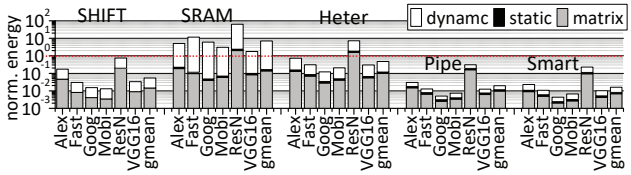


Figure 21: The batch energy reduction (norm. to TPU; matrix: matrix unit energy; dynamic: SPM dynamic energy; and static: SPM static energy).

- *SuperNPU*: The configuration of SuperNPU [17] is shown in Table 4.
- *SRAM*: SuperNPU replaces all SHIFT arrays by Josephson-CMOS SRAM arrays with the same capacity of TPU.
- *Heter*: Three 32 KB SHIFT arrays are added to the SRAM scheme. We assume an ideal SPM allocation, where the sequentially accessed data are always allocated in SHIFT arrays while the randomly accessed data are always allocated in the SRAM arrays.
- *Pipe*: Pipe replaces all Josephson-CMOS SRAM arrays of the Heter scheme by a 28 MB pipelined SFQ-CMOS SRAM array.
- *SMART*: Our ILP-base compiler is used by the Pipe scheme. The prefetching iteration number a is set to 3.

6 RESULTS AND ANALYSIS

6.1 Inferring a Single Image

Performance. The performance improvement achieved by SMART inferring a single image is shown in Figure 18. The performance is measured by the throughput (i.e., TMAC/s) normalized to that of the TPU. Average customers are sensitive to the latency of their cloud-based machine learning services. Therefore, the performance of a single image inference becomes more critical, because TPUs in the cloud have no time to form a large image batch. For one-image inferences, SuperNPU improves the inference throughput by only 8.6× over TPU, although the operating frequency of SuperNPU is 75× higher than that of TPU. Compared to SuperNPU, Josephson-CMOS SRAM arrays actually decrease the inference throughput. This is because the benefit brought by the random access capability of Josephson-CMOS SRAM is offset by its slow access speed. Even if we add a small SHIFT array to each heterogeneous SPM, we cannot win back the performance loss. Heter still obtains lower inference throughput than SuperNPU. On the contrary, our pipe-lined SFQ-CMOS array (Pipe) greatly improves the inference throughput, on average, by 2.4× over SuperNPU, due to its ultra-fast random access ability. Our ILP compiler (SMART) further increases the inference throughput improvement to 3.9× over SuperNPU, since it enables the prefetching of input, weight, and PSum data of a model.

Energy Consumption. The energy comparison between various schemes when inferring a single image is shown in Figure 20. Since SuperNPU is fabricated by the ERSFQ technology, it has no leakage power. We consider the cooling cost of each scheme at 4K as 400× [16] of the power consumption of that scheme. Since, on average, SuperNPU improves the performance per Watt by 23% over TPU [17], it consumes more energy on large CNN models, e.g., ResNet50 when considering the cooling overhead. SRAM and Heter tend to increase the inference energy when inferring a single image, because they obtain only longer inference latency and spend larger power in their Josephson-CMOS SRAM arrays. Our pipelined SFQ-CMOS array (Pipe) reduces the power consumption of RANDOM arrays by replacing CMOS H-Trees with SFQ H-Trees. Moreover, Pipe also shortens the inference latency over SuperNPU. As a result, Pipe reduces the inference energy by 81%. SMART decreases the inference energy by 86% over SuperNPU by further reducing the inference latency. On average, SMART uses only 1.9% of the inference energy of TPU when inferring the same image. For SMART, 48% of its energy is consumed by the matrix units, while 42% of its energy is the dynamic energy of the heterogeneous SPM.

6.2 Inferring a Batch of Images

Performance. The performance improvement achieved by SMART inferring a batch of images is shown in Figure 19. The inference performance of a batch of images shares the same trend as that of a single image. Compared to the single image case, SuperNPU inferring a batch of images improves the inference throughput by 2.5×. In contrast, SMART processing a batch of images improves the inference throughput by only 34.5% over the single image case of SMART. This is because SuperNPU has larger on-chip space to store more images, i.e., SuperNPU has 48 MB SPM arrays, while SMART has only a 28 MB on-chip RANDOM array. On average, when processing a batch of images, SMART improves the inference throughput over SuperNPU by 2.2×.

Energy Consumption. The energy reduction of SMART inferring a batch of images is shown in Figure 21. We also consider the cooling cost in the comparison. The inference energy of a batch shares the same trend as that of a single image. On average, SMART reduces the inference energy by 71% over SuperNPU, and uses only 1.6% of the inference energy of TPU when processing a batch of images. In SMART, 42.3% of its energy consumption is the energy of the matrix units, while 48.9% of the energy is the dynamic energy of its heterogeneous SPM arrays.

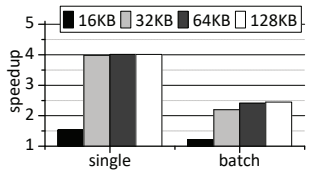


Figure 22: SHIFT capacity (norm. to SuperNPU).

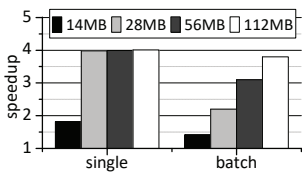


Figure 23: RAND. capacity (norm. to SuperNPU).

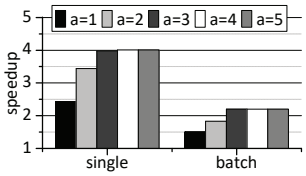


Figure 24: Prefetch. iter. # (norm. to SuperNPU).

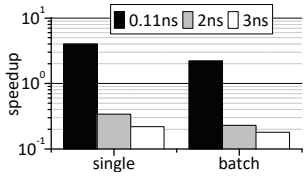


Figure 25: RAND. W lat-ency (norm. to SuperNPU).

6.3 Sensitivity Study

SHIFT array capacity. The sensitivity study on the capacity of SHIFT arrays in SMART is shown in Figure 22. The input, output/PSum, and weight data have three SHIFT arrays with the capacity of X, where X can be 16 KB, 32 KB, 64 KB, and 128 KB. Compared to 32 KB, the larger capacity of SHIFT arrays cannot help single-image inferences, and only slightly improve the inference throughput on a batch of images by 11%. On the contrary, three 16 KB SHIFT arrays greatly increase the swapping traffic between SHIFT arrays and the RANDOM array, thereby decreasing the inference throughput of a single image and a batch of images by 61% and 45%, respectively.

RANDOM array capacity. The sensitivity study on the RANDOM array capacity in SMART is shown in Figure 23. Though the input, output/PSum, and weight data have three SHIFT arrays respectively, they share the same RANDOM array. We tried different capacities of the RANDOM array in the figure. Compared to 28 MB, further increasing the RANDOM array capacity does not improve the single-image inference throughput. However, a 56 MB (112 MB) RANDOM array improves the inference throughput of a batch by 41% (73%). On the other hand, a smaller RANDOM array hurts the inference throughput of both a single image and a batch of images.

Prefetching iteration number. The sensitivity study on the prefetching iteration number of SMART is shown in Figure 24. Our ILP compiler achieves only near-optimal results, since we did not exhaustively explore the optimal prefetching iteration number. We set the prefetching iteration number $a = 3$. $a = 1$ indicates there is no prefetching. A smaller a substantially decreases the throughput of both single-image and batch inferences. On the other hand, a larger a (e.g., $a = 4$) does not obviously improve the inference throughput of six CNN models we selected.

Write latency. The sensitivity study on the write latency of the RANDOM array in SMART is shown in Figure 25. Since MRAM and SNM have smaller cell sizes than SRAM, if JJs can be scaled to the same size of a transistor, it is possible to use them to build a much denser RANDOM array. However, their write latency is much longer. We explore different values of the write latency of the RANDOM array in the figure. A longer write operation significantly decreases the throughput of both single-image and batch inferences, since the outputs of a layer are the inputs of the next layer. Therefore, these high-density cryogenic memory technologies may not be ideal candidates to implement the RANDOM array due to their slow writes.

7 RELATED WORK

SFQ Accelerators. As we are approaching the end of Moore's Law, several ambitious designs for superconducting ALUs [9, 23] and microprocessors [58] have been presented to demonstrate the capability of SFQ computing. For domain-specific computing, besides SFQ CNN systolic accelerators, a SFQ stochastic-computing-based deep learning accelerator [4] also demonstrates ultra-high inference throughput. Moreover, a SFQ-based temporal logic accelerator [52] is built to significantly boost the throughput of genome alignment. A SFQ-based SHA-256 accelerator [49] is designed to maximize the processing throughput of cryptographic hash functions. These superconducting designs primarily depend on simplified architectures, bit-serial processing, and shift registers. However, the use of SFQ shift registers is not a viable solution for more complex accelerator designs.

Cryogenic Memories and Caches. Recent work adopts the 77K cryogenic temperature to improve the performance and energy consumption of off-chip DRAM main memories [25] and on-chip SRAM caches [28]. However, these studies investigate only how the main memory and cache architectures are influenced by the 77K temperature when running general-purpose applications on CPUs. No prior work designs an on-chip SPM architecture for SFQ systolic CNN accelerators at the 4K temperature.

8 CONCLUSION

In this paper, we propose a heterogeneous SPM architecture, SMART, consisting of SHIFT arrays and a RANDOM array for SFQ deep learning accelerators to maximize their inference throughput. However, we found that no existing memory technology can serve as the RANDOM array of SMART to obtain high inference throughput, small chip area, and low power consumption at the same time. We built a fast, dense and power-efficient pipelined CMOS-SFQ array that supports random accesses in SMART. We also created an ILP-based SPM allocation and prefetching technique to minimize the inference latency on SMART. Experimental results show that, with the same area overhead, compared to the prior SHIFT-based SFQ CNN accelerator, SMART improves the inference throughput by 3.9× (2.2×), and reduces the inference energy by 86% (71%) when inferring a single image (a batch of images).

ACKNOWLEDGMENTS

The authors would like to thank the anonymous reviewers for their valuable comments and helpful suggestions. This work was partially supported by the National Science Foundation (NSF) through awards CCF-1908992, CCF-1909509, and CCF-2105972.

REFERENCES

[1] Rajeshwari Banakar, Stefan Steinke, Bo-Sik Lee, Mahesh Balakrishnan, and Peter Marwedel. 2002. Scratchpad memory: A design alternative for cache on-chip memory in embedded systems. In *IEEE International Symposium on Hardware/Software Codesign*. 73–78.

[2] A. Beckers, F. Jazaeri, A. Grill, S. Narasimhamoorthy, B. Parvais, and C. Enz. 2020. Physical Model of Low-Temperature to Cryogenic Threshold Voltage in MOSFETs. *IEEE Journal of the Electron Devices Society* 8 (2020), 780–788. https://doi.org/10.1109/JEDS.2020.2989629

[3] Brenden A Butters, Reza Baghdadi, Murat Onen, Emily A Toomey, Owen Medeiros, and Karl K Berggren. 2021. A scalable superconducting nanowire memory cell and preliminary array test. *Superconductor Science and Technology* 34, 3 (2021), 035003.

[4] 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 *ACM/IEEE 46th Annual International Symposium on Computer Architecture (ISCA)*. 567–578.

[5] Xiangyu Chen, Jiale Liang, and H.-S. Philip Wong. 2012. Interconnect Scaling into the Sub-10nm Regime. In *ACM International Workshop on System Level Interconnect Prediction*. 2.

[6] B. B. Chonigman, A. Shukla, M. Habib, V. Gupta, A. Talalaevskii, A. Sahu, D. Kirichenko, A. Inamdar, and D. Gupta. 2021. Optimization of Passive Transmission Lines for Single Flux Quantum Circuits. *IEEE Transactions on Applied Superconductivity* (2021), 1–1. https://doi.org/10.1109/TASC.2021.3062589

[7] Johannes Delport. 2018. JoSIM - Superconductor SPICE Simulator. https://github.com/JoeyDelp/JoSIM.

[8] Jean-Francois Deverge and Isabelle Puaut. 2007. WCET-directed dynamic scratchpad memory allocation of data. In *IEEE Euromicro Conference on Real-Time Systems*. IEEE, 179–190.

[9] T Filippov, M Dorojevets, A Sahu, A Kirichenko, C Ayala, and O Mukhanov. 2011. 8-bit asynchronous wave-pipelined RSFQ arithmetic-logic unit. *IEEE transactions on applied superconductivity* 21, 3 (2011), 847–851.

[10] Timur V Filippov, Anubhav Sahu, Alex F Kirichenko, Igor V Vernik, Mikhail Dorojevets, Christopher L Ayala, and Oleg A Mukhanov. 2012. 20 GHz operation of an asynchronous wave-pipelined RSFQ arithmetic-logic unit. *Physics Procedia* 36 (2012), 59–65.

[11] U. Ghoshal, D. Hebert, and T. Van Duzer. 1993. Josephson-CMOS memories. In *IEEE International Solid-State Circuits Conference Digest of Technical Papers*. 54–55. https://doi.org/10.1109/ISSCC.1993.280086

[12] Alexander Grill, E Bury, Jakob Michl, S Tyaginov, D Linten, Tibor Grasser, Bertrand Parvais, Ben Kaczer, Michael Waltl, and I Radu. 2020. Reliability and variability of advanced CMOS devices at cryogenic temperatures. In *IEEE International Reliability Physics Symposium (IRPS)*. IEEE, 1–6.

[13] Udit Gupta, Carole-Jean Wu, Xiaodong Wang, Maxim Naumov, Brandon Reagen, David Brooks, Bradford Cottel, Kim Hazelwood, Mark Hempstead, Bill Jia, et al. 2020. The architectural implications of facebook's dnn-based personalized recommendation. In *IEEE International Symposium on High Performance Computer Architecture (HPCA)*. 488–501.

[14] LLC Gurobi Optimization. 2021. Gurobi Optimizer Reference Manual. http://www.gurobi.com

[15] H. Hara, K. Obata, H. Park, Y. Yamanashi, K. Taketomi, N. Yoshikawa, M. Tanaka, A. Fujimaki, N. Takagi, K. Takagi, and S. Nagasawa. 2009. Design, Implementation and On-Chip High-Speed Test of SFQ Half-Precision Floating-Point Multiplier. *IEEE Transactions on Applied Superconductivity* 19, 3 (2009), 657–660. https://doi.org/10.1109/TASC.2009.2018039

[16] D. S. Holmes, A. L. Ripple, and M. A. Manheimer. 2013. Energy-Efficient Superconducting Computing—Power Budgets and Requirements. *IEEE Transactions on Applied Superconductivity* 23, 3 (2013), 1701610–1701610. https://doi.org/10.1109/TASC.2013.2244634

[17] 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 *IEEE/ACM International Symposium on Microarchitecture (MICRO)*. 58–72.

[18] Tahereh Jabbari and Eby G. Friedman. 2020. Global Interconnects in VLSI Complexity Single Flux Quantum Systems. In *the Workshop on System-Level Interconnect: Problems and Pathfinding*. Article 4, 7 pages.

[19] Tahereh Jabbari, Gleb Krylov, Stephen Whiteley, Jamil Kawa, and Eby G Friedman. 2020. Repeater Insertion in SFQ Interconnect. *IEEE Transactions on Applied Superconductivity* 30, 8 (2020), 1–8.

[20] T. Jabbari, G. Krylov, S. Whiteley, J. Kawa, and E. G. Friedman. 2020. Repeater Insertion in SFQ Interconnect. *IEEE Transactions on Applied Superconductivity* 30, 8 (2020), 1–8. https://doi.org/10.1109/TASC.2020.3000982

[21] Norman P Jouppi, Cliff Young, Nishant Patil, David Patterson, Gaurav Agrawal, Raminder Bajwa, Sarah Bates, Suresh Bhatia, Nan Boden, Al Borchers, et al. 2017. In-datacenter performance analysis of a tensor processing unit. In *IEEE/ACM International Symposium on Computer Architecture*. 1–12.

[22] Yoshio Kameda, Shinichi Yorozu, and Yoshihito Hashimoto. 2006. Automatic single-flux-quantum (SFQ) logic synthesis method for top-down circuit design. In *Journal of Physics: Conference Series*, Vol. 43. 287.

[23] Alex F Kirichenko, Igor V Vernik, Michael Y Kamkar, Jason Walter, Maximilian Miller, Lucian Remus Albu, and Oleg A Mukhanov. 2019. ERSFQ 8-bit parallel arithmetic logic unit. *IEEE Transactions on Applied Superconductivity* 29, 5 (2019), 1–7.

[24] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. ImageNet Classification with Deep Convolutional Neural Networks. In *Advances in Neural Information Processing Systems*, Vol. 25. Curran Associates, Inc.

[25] Gyu-hyeon Lee, Dongmoon Min, Ilkwon Byun, and Jangwoo Kim. 2019. Cryogenic Computer Architecture Modeling with Memory-Side Case Studies. In *ACM/IEEE International Symposium on Computer Architecture*. 774–787.

[26] K. K. Likharev and V. K. Semenov. 1991. RSFQ logic/memory family: a new Josephson-junction technology for sub-terahertz-clock-frequency digital systems. *IEEE Transactions on Applied Superconductivity* 1, 1 (1991), 3–28. https://doi.org/10.1109/77.80745

[27] Y. Liu and W. Zhang. 2012. Exploiting multi-level scratchpad memories for time-predictable multicore computing. In *IEEE International Conference on Computer Design (ICCD)*. 61–66.

[28] Dongmoon Min, Ilkwon Byun, Gyu-Hyeon Lee, Seongmin Na, and Jangwoo Kim. 2020. Cryocache: A fast, large, and cost-effective cache architecture for cryogenic computing. In *ACM International Conference on Architectural Support for Programming Languages and Operating Systems*. 449–464.

[29] Hamid Reza Mohebbi and A Hamed Majedi. 2009. CAD model for circuit parameters of superconducting-based hybrid planar transmission lines. *Superconductor Science and Technology* 22, 12 (2009), 125028.

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

[31] N. Muralimanohar, R. Balasubramonian, and N. Jouppi. 2007. Optimizing NUCA Organizations and Wiring Alternatives for Large Caches with CACTI 6.0. In *IEEE/ACM International Symposium on Microarchitecture (MICRO 2007)*. 3–14. https://doi.org/10.1109/MICRO.2007.33

[32] NA N Joukov, DE Kirichenko, AYu Kidiyarova-Shevchenko, and M Yu Kupriyanov. 2000. Matching of Rapid Single Flux Quantum Digital Circuits and Superconductive Microstrip Lines. *IEEE Transactions on Applied Superconductivity* 167 (2000), 745–748.

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

[34] Ikki Nagaoka, Masamitsu Tanaka, Kyosuke Sano, Taro Yamashita, Akira Fujimaki, and Koji Inoue. 2019. Demonstration of an Energy-Efficient, Gate-Level-Pipelined 100 TOPS/W Arithmetic Logic Unit Based on Low-Voltage Rapid Single-Flux-Quantum Logic. In *IEEE International Superconductive Electronics Conference (ISEC)*. 1–3.

[35] Shuichi Nagasawa, Haruhiro Hasegawa, Tatsunori Hashimoto, Hideo Suzuki, Kazunori Miyahara, and Youichi Enomoto. 1999. Design of a 16 kbit superconducting latching/SFQ hybrid RAM. *Superconductor Science and Technology* 12, 11 (1999), 933.

[36] Shuichi Nagasawa, Haruhiro Hasegawa, Tatsunori Hashimoto, Hideo Suzuki, Kazunori Miyahara, and Youichi Enomoto. 2000. Superconducting SFQ-NOR Decoder. In *Advances in Superconductivity XII*. Springer, 1093–1095.

[37] Shuichi Nagasawa, Haruhiro Hasegawa, Tatsunori Hashimoto, Hideo Suzuki, Kazunori Miyahara, and Youichi Enomoto. 2001. Superconducting latching/SFQ hybrid RAM. *IEEE transactions on applied superconductivity* 11, 1 (2001), 533–536.

[38] Minh-Hai Nguyen, Guilhem J Ribeill, Martin V Gustafsson, Shengjie Shi, Sriharsha V Aradhya, Andrew P Wagner, Leonardo M Ranzani, Lijun Zhu, Reza Baghdadi, Brenden Butters, et al. 2020. Cryogenic memory architecture integrating spin Hall effect based magnetic memory and superconductive cryotron devices. *Scientific reports* 10, 1 (2020), 1–11.

[39] Tomohiro Ono, Hideo Suzuki, Yuki Yamanashi, and Nobuyuki Yoshikawa. 2017. Design and implementation of an SFQ-based single-chip FFT processor. *IEEE Transactions on Applied Superconductivity* 27, 4 (2017), 1–5.

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

[41] A Potts, GJ Parker, JJ Baumberg, and PAJ de Groot. 2001. CMOS compatible fabrication methods for submicron Josephson junction qubits. *IEE Proceedings-Science, Measurement and Technology* 148, 5 (2001), 225–228.

[42] Ananda Samajdar, Yuhao Zhu, Paul Whatmough, Matthew Mattina, and Tushar Krishna. 2018. Scale-Sim: Systolic CNN Accelerator Simulator. *arXiv preprint arXiv:1811.02883* (2018).

[43] Lieze Schindler, Paul le Roux, and Coenrad J Fourie. 2020. Impedance matching of passive transmission line receivers to improve reflections between RSFQ logic cells. *IEEE Transactions on Applied Superconductivity* 30, 2 (2020), 1–7.

[44] Vasili K Semenov, Yuri A Polyakov, and Sergey K Tolpygo. 2019. Very large scale integration of Josephson-junction-based superconductor random access memories. *IEEE Transactions on Applied Superconductivity* 29, 5 (2019), 1–9.

[45] Vivy Suhendra, Tulika Mitra, Abhik Roychoudhury, and Ting Chen. 2005. WCET centric data allocation to scratchpad memory. In *IEEE International Real-Time Systems Symposium (RTSS'05)*.

[46] Shuichi Tahara, Ichiro Ishida, Yumi Ajisawa, and Yoshifusa Wada. 1989. Experimental vortex transitional nondestructive read-out Josephson memory cell.

Journal of applied physics 65, 2 (1989), 851–856.

[47] M. Tanaka, F. Matsuzaki, T. Kondo, N. Nakajima, Y. Yamanashi, A. Fujimaki, H. Hayakawa, N. Yoshikawa, H. Terai, and S. Yorozu. 2004. A single-flux-quantum logic prototype microprocessor. In *IEEE International Solid-State Circuits Conference*. 298–529 Vol.1. https://doi.org/10.1109/ISSCC.2004.1332714

[48] Masamitsu Tanaka, Masato Suzuki, Gen Konno, Yuki Ito, Akira Fujimaki, and Nobuyuki Yoshikawa. 2016. Josephson-CMOS hybrid memory with nanocryotrons. *IEEE Transactions on Applied Superconductivity* 27, 4 (2016), 1–4.

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

[50] S. K. Tolpygo, V. Bolkhovsky, D. E. Oates, R. Rastogi, S. Zarr, A. L. Day, T. J. Weir, A. Wynn, and L. M. Johnson. 2018. Superconductor Electronics Fabrication Process with MoNx Kinetic Inductors and Self-Shunted Josephson Junctions. *IEEE Transactions on Applied Superconductivity* 28, 4 (2018), 1–12.

[51] S. K. Tolpygo, V. Bolkhovsky, S. Zarr, T. J. Weir, A. Wynn, A. L. Day, L. M. Johnson, and M. A. Gouker. 2017. Properties of Unshunted and Resistively Shunted Nb/AlOx-Al/Nb Josephson Junctions With Critical Current Densities From 0.1 to 1 mA/μm^2. *IEEE Transactions on Applied Superconductivity* 27, 4 (2017), 1–15. https://doi.org/10.1109/TASC.2017.2667403

[52] Georgios Tzimpragos, Dilip Vasudevan, Nestan Tsiskaridze, George Michelogiannakis, Advait Madhavan, Jennifer Volk, John Shalf, and Timothy Sherwood. 2020. A Computational Temporal Logic for Superconducting Accelerators. In *ACM International Conference on Architectural Support for Programming Languages and Operating Systems*. 435–448.

[53] Sumesh Udayakumaran, Angel Dominguez, and Rajeev Barua. 2006. Dynamic allocation for scratch-pad memory using compile-time decisions. *ACM Transactions on Embedded Computing Systems (TECS)* 5, 2 (2006), 472–511.

[54] T. Van Duzer, L. Zheng, S. R. Whiteley, H. Kim, J. Kim, X. Meng, and T. Ortlepp. 2013. 64-kb Hybrid Josephson-CMOS 4 Kelvin RAM With 400 ps Access Time and 12 mW Read Power. *IEEE Transactions on Applied Superconductivity* 23, 3 (2013), 1700504–1700504. https://doi.org/10.1109/TASC.2012.2230294

[55] M. Verma and P. Marwedel. 2006. Overlay techniques for scratchpad memories in low power embedded processors. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 14, 8 (2006), 802–815.

[56] D. T. Yohannes, R. T. Hunt, J. A. Vivalda, D. Amparo, A. Cohen, I. V. Vernik, and A. F. Kirichenko. 2015. Planarized, Extendible, Multilayer Fabrication Process for Superconducting Electronics. *IEEE Transactions on Applied Superconductivity* 25, 3 (2015), 1–5. https://doi.org/10.1109/TASC.2014.2365562

[57] S Yorozu, Y Kameda, H Terai, A Fujimaki, T Yamada, and S Tahara. 2002. A single flux quantum standard logic cell library. *Physica C: Superconductivity* 378 (2002), 1471–1474.

[58] Nobuyuki Yoshikawa, F Matsuzaki, N Nakajima, K Fujiwara, K Yoda, and K Kawasaki. 2003. Design and component test of a tiny processor based on the SFQ technology. *IEEE transactions on applied superconductivity* 13, 2 (2003), 441–445.

[59] Chen Zhang, Peng Li, Guangyu Sun, Yijin Guan, Bingjun Xiao, and Jason Cong. 2015. Optimizing FPGA-Based Accelerator Design for Deep Convolutional Neural Networks. In *ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*. 161–170.

[60] Qing-Yuan Zhao, Adam N McCaughan, Andrew E Dane, Karl K Berggren, and Thomas Ortlepp. 2017. A nanocryotron comparator can connect single-flux-quantum circuits to conventional electronics. *Superconductor Science and Technology* 30, 4 (2017), 044002.

[61] Qing-Yuan Zhao, Emily A Toomey, Brenden A Butters, Adam N McCaughan, Andrew E Dane, Sae-Woo Nam, and Karl K Berggren. 2018. A compact superconducting nanowire memory element operated by nanowire cryotrons. *Superconductor Science and Technology* 31, 3 (2018), 035009.