Superconducting Computing with Alternating Logic Elements

Georgios Tzimpragos
UC Santa Barbara
gtzimpragos@cs.ucsb.edu

Jennifer Volk
UC Santa Barbara
jevolk@ucsb.edu

Alex Wynn
MIT Lincoln Laboratory
alexander.wynn@ll.mit.edu

James E. Smith
University of Wisconsin-Madison (Emeritus)
jes@ece.wisc.edu

Timothy Sherwood
UC Santa Barbara
sherwood@cs.ucsb.edu

***Abstract*—Although superconducting single flux quantum (SFQ) technologies offer the potential for low-latency operation with energy dissipation of the order of attojoules per gate, their inherently pulse-driven nature and stateful cells have led to designs in which every logic gate is clocked. This means that clocked buffers must be added to equalize logic path lengths, and every gate becomes a pipeline stage. We propose a different approach, where gates are clock-free and synchronous designs have a conventional look-and-feel. Despite being clock-free, however, the gates are state machines by nature. To properly manage these state machines, the logical clock cycle is composed of two synchronous alternating phases: the first of which implements the desired function, and the second of which returns the state machines to the ground state. Moreover, to address the challenges associated with the asynchronous implementation of Boolean NOT operations in pulse-based systems, values are represented as unordered binary codes – in particular, dual-rail codes. With unordered codes, AND and OR operations are functionally complete.**

We demonstrate that our new approach, xSFQ, with its dual-rail construction and alternating clock phases, along with "double-pumped" logical latches and a timing optimization through latch decomposition, is capable of implementing arbitrary digital designs without gate-level pipelining and the overheads that come with it. We evaluate energy-delay tradeoffs enabled by this approach through a mix of detailed analog circuit modeling, pulse-level discrete-event simulation, and high-level pipeline efficiency analysis. The resulting systems are shown to deliver energy-delay product (EDP) gains over conventional SFQ even with pipeline hazard ratios (HR) below 1%. For hazard ratios equal to 15% and 20% and a design resembling a RISC-V RV32I core (excluding the cost of interlock logic), xSFQ achieves 22x and 31x EDP savings, respectively.

***Index Terms*—superconductor electronics, alternating logic, unordered codes, pipelining, xSFQ**

I. Introduction

Beginning in 1983 when Josephson junctions (JJs) were first fabricated with currently used materials [11], a continuous and extended effort has carried the superconducting field from the development of Rapid Single Flux Quantum (RSFQ) logic in 1985 [25] to chips with nearly a million JJs [35]. With the realization of self-shunted JJs in 2017 [43], chips with 10s of millions of JJs are now in sight. However, many technical hurdles remain before large scale computations can enjoy the benefits of superconducting materials.

Although some of the hurdles are inherent to the fabrication process, others are due to a mismatch between traditional architectural abstractions and fundamental characteristics of superconductor circuits. In particular, JJs naturally communicate via impulses, each of which consists of a single flux quantum (SFQ). Performing Boolean operations by arranging for two or more nearly instantaneous pulses to always arrive simultaneously at JJ logic gates is clearly not realistic, so methods developed for transistor logic gates, where values are represented as voltage levels, do not readily carry over.

An approach commonly used in SFQ logic systems[1] is to consider the presence of a pulse during a given time interval as a logical 1 and the lack of a pulse as a logical 0. This convention, however, requires two things of SFQ circuits: first, all logic gates must agree on a prescribed time interval for evaluation; second, they must be able to "remember" whether or not a pulse has arrived during the interval. The second requirement fits nicely with the inherently stateful nature of superconducting cells composed of Superconducting Quantum Interference Devices (SQUIDs, or loops formed by two JJs and one inductor), which hold SFQ pulses. On the other hand, the first requirement is more difficult to fulfill and is traditionally met through extremely fine-grained clocking.

Because SFQ cells hold state (in the form of stored flux), conventional SFQ logic gates are typically synchronous (indivisibly conjoined with a 1 bit register) and their value is only known once the gate's clock pulse arrives (see Figure 1). For this reason, not only does the clock provide the time interval necessary for defining logic values, but each clock pulse also serves to reset, or "relax", a gate's state back to ground, so it is ready to receive new inputs in the next cycle. While this method has enabled a variety of working designs [1], [39], it requires that a clock signal be delivered to every logic gate, which makes circuits highly sensitive to

This material is based upon work supported by the National Science Foundation under Grants No. 1763699, 1730309, 1717779, 1563935.

[1]In this paper, SFQ logic systems are used as an umbrella term to include a superset of the various RSFQ-derived logic families.

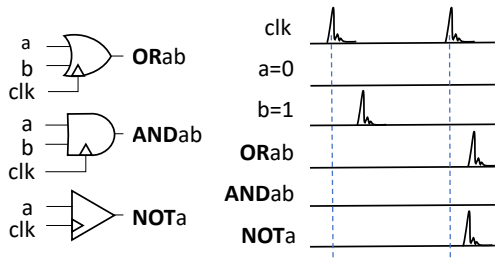

Fig. 1. Boolean (synchronous) SFQ cells. If a pulse appears in a *clk* interval is understood as a logical 1; otherwise, it is a logical 0. The arrival of a *clk* signal releases the content of the cell and resets it.

timing variability (skews) and requires that clocked buffers be inserted as pads to equalize logic path lengths. This increases overall complexity, reduces energy efficiency, and fundamentally constrains architectural choices.

In this paper, we propose and evaluate an alternative solution to the problem of constructing Boolean logic from stateful superconducting elements communicating via impulses. Through the careful co-design of logical value encoding, SFQ circuit elements, and architecture, we are able to (a) maintain well understood and composable Boolean logic abstractions, while (b) avoiding clocks at every logic gate and the insertion of delay pads. In doing so, we (c) ensure stateful elements always return to the ground state, which (d) allows computer architecture considerations, such as pipeline depths, to drive design decisions rather than circuit-level requirements. Finally, we (e) provide physically realizable, low-energy, and low-delay implementations that are fully compatible with existing SFQ design processes. We demonstrate that these properties are achieved through four core contributions.

First, we show that AND and OR gates can be envisioned as pulse-based LASTARRIVAL (LA) and FIRSTARRIVAL (FA) operations. Although LA and FA cells are asynchronous (and hence clock-free), they remain stateful. Correct operation depends on guaranteeing that the asynchronous cells are always returned to their initial state prior to the start of the next round of computation. This can be achieved by dividing a logical clock cycle into two alternating synchronous phases. The *excite* phase operates on pulse-coded logic values and the *relax* phase propagates the values needed to return the asynchronous cell back to its ground state. This is essentially an implementation of alternating logic [31], in which logic state machines always return to the correct initial state at the end of each two-phase logical cycle.

Second, although LA and FA cells implement AND and OR gates, AND and OR alone are not functionally complete. To establish functional completeness, a NOT function is needed. Here, we build on the theory of unordered binary codes [5], [36]. In an unordered binary code – one in which no codeword covers another – any Boolean function can be implemented using only AND and OR gates. The classic dual-rail (DR) approach is a special case; in contrast to existing DR-based SFQ approaches [7], [10], [23], [28], which rely on synchronous logic gates and use complementary data signals to generate the required clock or control signals, in xSFQ, DR codes are used only for guaranteeing functional completeness.

Third, we develop new and practical circuit realizations of LA and FA cells. Existing SFQ implementations do not support the two-phase alternating convention and rely on external signals or thermal/power-cycling to reset [45], [46]. Resetting via thermal/power-cycling is impractical due to high power dissipation and long thermal time constants, and providing an external reset signal to every logic gate introduces scaling problems.

Fourth, we present alternation-aware registers along with a new pipeline balancing technique capable of hiding much of the performance overhead inherent in a two-phase approach. In order to make sequential networks amenable to alternation, each logical "flip flop" in the design is implemented with a coupled pair of DESTRUCTIVE READ OUT (DRO) cells. The DRO cells are then distributed along a combinational logic pathway in a manner that is analogous to traditional circuit retiming [24].

To evaluate the functionality and performance of our logic system, dubbed *xSFQ*, we construct detailed SPICE-level models of the proposed cells, perform discrete-event simulations of more complex superconducting alternating systems, extend CMOS-oriented analytical power-performance models to fit superconducting technology, and demonstrate the benefits of xSFQ over conventional SFQ-based sytems through an energy-delay product (EDP) analysis. We find that for a design resembling a RISC-V RV32I core [2] (consisting of 10,000 two-input logic gates and a critical path of 150 gates) and a 10% pipeline hazard ratio (HR), xSFQ achieves $14\times$ EDP reduction excluding the overhead of interlock and flushing circuitry. These gains increase super linearly with the length of the critical path, the number of synchronous buffers required to equalize uneven datapaths, and the ratio of pipeline hazards. For example, for 15% and 20% HRs, $22\times$ and $31\times$ EDP saving are observed, respectively, excluding the overhead of interlock logic.

II. COMPUTING WITH SUPERCONDUCTORS

A. Fundamental concepts

Superconductor electronics are defined by three basic features: (a) the absence of resistance in static circuits at superconducting temperatures, (b) the Josephson effect – which governs the fundamental switching element in superconductor circuits, the JJ – and (c) the propagation of single flux quanta[2] (pulses produced by JJs of the order of mV in amplitude and a few ps in width) instead of static voltage levels as in CMOS. As a two-terminal device, the JJ does not switch in the same way as three-terminal CMOS transistors. Normally, current flows through the JJ with no impediment, like a zero-resistance wire; however, at a threshold called the "critical current", the resistively-shunted JJ blocks off current flow for

[2]One magnetic flux quantum is 2.07×10^{-15} Wb or $2.07\,\mathrm{mV} \times \mathrm{ps}$ in units more familiar to computer architects.

a short time as it switches, thereby creating an SFQ pulse on the JJ output. Each pulse is a short burst of magnetic energy observed through a change in voltage.

B. Opportunities

One of the most compelling arguments for the use of superconductor electronics is their potential to significantly surpass end-of-roadmap CMOS circuits based on energy-delay product, even when the overhead due to cooling is considered [16]. SFQ circuits can achieve $10 - 100$ times higher clock frequencies than CMOS, and the switching energy of an individual JJ is $\sim 10^{-19}$J. Energy-efficient versions of the SFQ technology, such as ERSFQ [21], also eliminate static power dissipation without sacrificing speed or circuit-level equivalence to the more traditional RSFQ, and a recently-proposed AC/SFQ powering scheme reduces bias requirements by locally storing small currents from rectified AC voltage to power SFQ gates [34]. Furthermore, because SFQ pulses travel along resistance-free wires with no RC charge process involved, transmission requires little or no energy [37]. From an architectural standpoint, these facts, along with the stateful nature of superconducting elementary cells, give designers the opportunity to explore a fundamentally different trade-off space (e.g., negligible or no overhead of pipeline registers and extremely energy-efficient interconnects), reevaluate existing architectural solutions, and exploit the unique characteristics of superconductor electronics for the development of innovative computing machines.

Besides speeding up and reducing the power consumption of CMOS circuits for classical applications, superconductor electronics open pathways for scaling up quantum computers. In particular, the ability of superconducting circuits to operate at very high speed enables the fast processing of qubit output data for error correction and generation of control signals – expanding the simple quantum volume (number of computational qubits of a machine multiplied by the number of gates expected to perform without error) of near-term quantum machines by several orders of magnitude [15]. Moreover, their low power overhead and cryogenic operating temperature allow them to reside next to the quantum processor, thus eliminating the control cables that leave the cryogenic environment and introduce thermal noise.

C. Challenges

Despite their advantages, superconductor electronics pose a number of challenges. One of the most profound is the pulsed-based nature of computation. Pulses cannot be sampled like voltage levels because they do not coincide with picosecond precision. Thus, methods developed for transistors, and other latching circuits, do not carry over easily.

Moreover, an SFQ pulse is fundamentally discrete. Because of this quantization, a fan-out that produces two pulses from a single pulse requires an active component called a SPLITTER. As a consequence, signals with significant fan-out inflate circuit size considerably. To make matters worse, the relatively high process variability in superconductor electronics [42] can skew signals significantly across a fan-out tree, leading to synchronization problems, additional logic overheads, and reduced operating speeds.

Finally, the lack of a reliable and high-capacity random access memory operating at 4.2 degrees Kelvin, the default temperature for superconductor electronics, imposes its own distinctive limitations. Recent studies indicate that cold memories built from CMOS DRAM operating at 77 degrees Kelvin offer a promising solution in the near future [47], while advances in fabrication technologies are encouraging for competitive JJ memories in the longer term [33]. In both cases, however, the expected gap between the access latency of memory and the high operational speed of SFQ logic circuits introduces challenging microarchitectural problems.

D. Current Status

Superconducting technologies have been studied for several decades. ALU designs [39], [40] and microprocessors [1], [3], [8], [48] have been presented in an effort to capitalize on the promise of superconductors. The majority of these implementations are based on simplified architectures, bit-serial processing, and shift-register-based on-chip memories. For example, the modern CORE e4 [1] is an 8 bit-serial RSFQ microprocessor that contains 4 general-purpose registers, can execute 20 different instruction types, and achieves up to 333 million instructions per second (MIPS) while dissipating an estimated 2.03 mW of power. To the best of our knowledge, this estimate does not include the cost of cooling, which increases power requirements by approximately two orders of magnitude according to Carnot's thermodynamic efficiency theorem.

More recently, there has been increasing interest in the exploration of superconducting accelerators for emerging applications. For example, Tannu et al. [41] developed a Reciprocal Quantum Logic (RQL)-based design for SHA-256 (cryptographic hashing) engines [26]. To maximize their gains, the authors focused on the optimization of adder circuits, which are the most critical components of the SHA engine, and proposed a fault-tolerant architecture that allows JJ critical currents to be lowered from 38 μA to 10 μA. The reported results indicate $46\times$ energy efficiency gains and 20% performance gains compared to CMOS. Ishida et al. [19] presented an ERSFQ-based neural processing unit. Based on the reported results, the performance per Watt of the proposed superconducting design is $490\times$ and $1.23\times$ higher than a Tensor Processing Unit (TPU)-like [20] CMOS implementation, without and with the cost of cooling accounted for, respectively.

Other interesting approaches that target accelerators rely on the exploitation of less traditional computing paradigms that match well with the characteristics of superconductor electronics. For example, Cai et al. [6] presented a deep learning framework based on stochastic computing that uses Adiabatic Quantum-Flux-Parametron (AQFP) technology. According to their simulation results, the proposed deep neural network (DNN) design can achieve up to 6.9×10^4 times higher energy efficiency than CMOS – to the best of our knowledge, this

again excludes the cost of cooling. Tzimpragos et al. [45], [46] took a much different approach and proposed a computational temporal logic, relying solely on asynchronous SFQ gates. In particular, the authors claimed that the natural language for expressing computations in a pulse-based system is one that precisely describes the temporal relationships between these pulses. To support their arguments, they provided SFQ-based designs of DNA sequencing and decision tree accelerators, achieving $\sim 40\times$ lower latency compared to CMOS.

These ideas and related designs pave a promising path forward. Nevertheless, their shortcomings and underlying assumptions cannot be ignored. The computational limits and efficiency of stochastic computing and temporal logic for general-purpose tasks have yet to be explored, especially for superconducting systems. The remaining accelerator architectures support only dataflow processing without complex control flows, while the demonstrated microprocessors (and other designs) are not scalable. For example, several existing benchmark SFQ circuits require a significant number of DRO cells for padding uneven datapaths. According to published results for the ISCAS85 benchmark circuits [12], the number of required DRO cells for padding exceeds the number of logic gates by more than $2.5\times$ on average (ranging from $1.5\times$ to $5\times$) [30]. Furthermore, the only realistic way to avoid extremely high Logical Cycles per Instruction (LCPIs), at least in the case of microprocessors, is to apply fine grained temporal multithreading [18] where, ideally, the number of threads is as large as the number of gates on the critical path. Additionally, the throughput commonly reported in superconducting papers refers to a theoretical peak rate based solely on the frequency of individually clocked gates (e.g., delays due to pipeline stalls are excluded). Every gate is essentially a pipeline stage, and extremely deep pipelines put more pressure on the already challenging superconductor memory system.

III. xSFQ Logic Design

At the physical level, conventional CMOS and superconducting technologies are radically different. Yet, it is obviously desirable if the tools, techniques, and computer architecture concepts developed for conventional computer systems can be applied to superconducting designs. We strive to achieve this at the logic level of abstraction. In other words, we aim to first perform digital design using what appear to be ordinary logic gates and then map them in a straightforward way to superconducting cells.

The differences between designing with conventional and superconducting technologies, however, introduce an entirely different design space with respect to trade-offs and constraints. As discussed above, one of the most important of these involves cell fan-outs. Superconducting cells by themselves are limited to driving one cell. Fan-outs of two or more require SPLITTERs, which are limited to a fan-out of two. Hence, large fan-outs must be constructed as binary trees of SPLITTERs. This means that clock, global reset, and other high fan-out signals are relatively expensive compared to CMOS.

Regarding fan-in, although it is possible to design superconducting cells with more than two data inputs, they are not commonly used because they are significantly more complex and mostly developed in an *ad hoc* way. Hence, for our purposes, it is assumed that superconducting cells are limited to two inputs. Of course, in the logic domain, the designer may use multi-fan-in gates, with the awareness that these will be expanded into expensive multi-cell superconducting circuits.

A. Clocking Discipline

In keeping with the above design philosophy, we propose and develop a clocking discipline that is aligned with conventional synchronous CMOS; e.g., ranks of clocked storage elements separated by unclocked combinational networks. This is in sharp contrast with existing superconducting methods where, typically, every gate is clocked.

In the logic domain, these basic clocked storage elements operate as D FLIP-FLOPS (DFFs) that translate to DRO cells in the xSFQ domain. Figure 2 provides a Mealy machine-based representation of a DRO cell. Recalling that a logical clock cycle consists of two synchronous phases, at the logic network level, a synchronizing phase pulse causes the stored signal pulses to be released from the DRO cells. Then, signal pulses propagate through a forward flow network of asynchronous xSFQ cells with pulses eventually arriving at downstream DRO cell inputs. The subsequent phase releases them from that stage as the process continues. As with conventional synchronous logic design, the phase period must be long enough to allow for propagation along the longest signal path through the xSFQ combinational network (and respect setup time requirements).

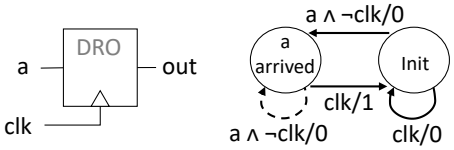

Fig. 2. Symbol and Mealy machine representation of a DESTRUCTIVE READ OUT (DRO) cell.

As part of the logical signaling discipline, a given data wire transmits at most one pulse during a given (synchronous) phase. The time of the pulse within the phase period does not affect its logical interpretation – only its presence or absence matters. Hence, the designer should worry only about the timing constraints set by the lengths of signal paths, which must fit within a clock phase, and not timing relationships across paths.

B. FA & LA Cells Specification

According to their semantics [45], [46], FA and LA cells implement asynchronous (unclocked) state machines that operate on pulsed inputs and produce pulsed outputs. As its name indicates, an FA cell emits an output pulse in response to the first input pulse that arrives at either of its inputs. Any later input pulse does not affect the FA cell output. An LA cell

emits an output pulse only if both inputs receive pulses. The output pulse occurs in response to the second input pulse.

Figure 3 illustrates the state transitions of these cells as they occur during the time frame of a (clock) phase (although the cells are not clocked themselves).

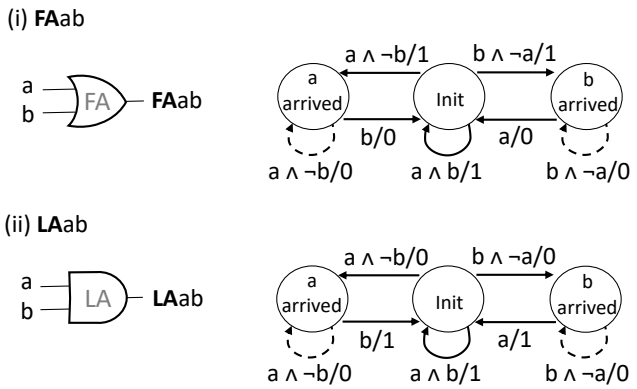

Fig. 3. Symbol and Mealy machine representation of (i) FA and (ii) LA cells.

When used in this way, we observe that the FA cell implements an OR function with respect to pulse signals and the LA cell implements an AND function. For proper operation of both cell types, the initial state must be *Init* and each input line must be restricted to one pulse per phase. The arrival of more than one pulse indicates a violation of the signaling convention.

C. Two-phase Alternating Encoding

Based on the shown Mealy machines alone, there is no guarantee that the initial state will be restored at the end of a phase. For example, if the FA cell observes only a single input pulse during a given phase, it will be in either the *a arrived* or *b arrived* state at the end of the phase. To force FA cells to transition back to the *Init* state, the obvious solution is to fan out an explicit global reset signal to all FA cells. The same holds true for LA cells. But, as noted above, this requires a binary tree of active SPLITTERs and will be very expensive.

We propose a novel approach that accomplishes a state reset by using only the functional input wires. There are no explicit reset wires, and thus no need for a reset distribution network. This approach does so by providing all mechanisms to reset the gates within the logic encoding itself. This is the motivation for dividing a *logical cycle* into a pair of *physical cycles*, or synchronous *phases*, where an *excite* phase is followed by a *relax* phase. In the excite phase, the pulsed inputs are given their logically equivalent values. During the relax phase, the pulsed inputs are given their complement values – Figure 4. This guarantees that each FA and LA cell always receives exactly one pulse at each input port throughout the two-phase logical cycle and returns its initial state. We define this as an *alternating encoding*.

Figure 5 presents all possible alternating input pulse sequences for FA and LA cells. Both the excite and relax phases are shown for all legal input combinations. In every case, if

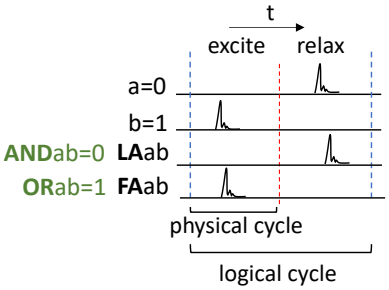

Fig. 4. The value of an alternating binary variable appears during the excite phase and is followed by its complemented value during the relax phase. Each phase corresponds to a physical cycle (no two-phase clocking is happening). An excite-relax pair forms an xSFQ logical cycle.

the initial state is *Init* and if the input pulses alternate between the excite and relax phases, then (a) the output during the relax phase is opposite the output for the excite phase, and (b) the final state is always *Init*. We define this as the *alternating signal property*.

	excite					relax				
state	**inputs**		**FA**ab	**LA**ab	**state**	**inputs**		**FA**ab	**LA**ab	**state**
	a	b				a	b			
Init	0	0	0	0	Init	1	1	1	1	Init
Init	0	1	1	0	b arrived	1	0	0	1	Init
Init	1	0	1	0	a arrived	0	1	0	1	Init
Init	1	1	1	1	Init	0	0	0	0	Init

Fig. 5. Alternating input pulse sequences for FA and LA cells.

Importantly, the alternating signal property holds not only for individual cells, as just shown exhaustively, but also for any network *composed* of these cells.

Theorem 1: Any feed-forward network composed of FA and LA cells will have the alternating signal property.

Proof sketch. An individual FA or LA cell is alternating, and is a network of size depth 1. If two networks having the alternating signal property connect to the inputs of an FA or LA cell, then the FA or LA cell must observe alternating inputs, so its output satisfies the alternating signal property with respect to the network inputs. By induction, the overall network is therefore alternating [3].

Hence, by using alternating signal inputs (which consume two synchronous phases), a state reset is achieved without an explicit reset signal. Clearly the two phase clocking system requires additional time compared to a single phase system, but note that an explicit reset signal would also consume additional time; perhaps as much as a full clock phase. Also observe that the total number of pulses over an excite-relax pair is always constant, which may be useful for the detection of erroneous operation [31].

[3] SPLITTERs are used for fan-out, and thus are not considered logic elements.

D. Unordered Codes and Functional Completeness

As just described, an FA cell implements a logical OR gate that operates on pulses, and an LA cell implements a logical AND gate. Nevertheless, a NOT gate is missing. Implementing a NOT gate is typically problematic with pulse signaling because it implies knowledge that a pulse will not occur during the leading excite phase, but a circuit cannot wait until this phase is over to make this determination, and it cannot look into the future. However, if input and output signals are constrained to be members of an unordered code, then functional completeness is achievable with AND and OR gates alone.

Definition 3.1: A vector $X = x_1x_2...x_n$ is said to *cover* another vector $Y = y_1y_2,...y_n$ (denoted $X \geq Y$) if for all $i \in n$, $y_i = 1$ implies $x_i = 1$. Vectors X and Y are unordered if $X \not\geq Y$ and $Y \not\geq X$.

For example, if $X_1 = [0011]$ and $Y_1 = [0111]$, Y_1 covers X_1. On the other hand, the vectors $X_2 = [1001]$ and $Y_2 = [1010]$ are unordered.

Definition 3.2: A binary code is unordered if no two members of the code are ordered.

Well-known examples of unordered codes include one-hot codes, dual-rail (DR) codes, Berger codes [4], bi-quinary codes (the IBM 650 was a bi-quinary coded decimal computer [44]), and various *ad hoc* codes [5], [36].

Theorem 2: Any Boolean function whose domain consists of an unordered code can be implemented using only AND and OR gates.

Proof sketch. By construction – the complement of any bit can be formed as an AND/OR function of the other bits. This is done by first selecting all codewords for which the subject bit is a 0, and then forming a set of minterms corresponding to the 1 bits in the selected codewords. Summing the minterms yields the complement of the subject bit.

Example: Consider the 2-out-of-4 code (each tuple consists of 4 bits and has exactly 2 logical 1s), $[x_1x_2x_3x_4] = \{[1100], [1010], [1001], [0110], [0101], [0011]\}$. The value of $\overline{x_1} = x_2x_3 + x_2x_4 + x_3x_4$.

In practice, arbitrary unordered codes are difficult to work with. Although the complement of any bit can be formed via an AND/OR function of the other bits, the AND/OR network can become quite large. Even for the relatively small 2-out-of-4 code presented above, the complement of any bit requires 3 AND gates and 2 OR gates (fan-in = 2).

Generally speaking, the more efficient the code, the more difficult it is to form complements. If efficiency is defined as the total number of information bits per number of code bits, the most efficient unordered code is k-out-of-$2k$. For larger values of k, k-out-of-$2k$ codes are cumbersome for implementing arbitrary functions. However, the special case k=1 yields the DR code. Although the DR code is relatively inefficient, it is particularly simple and easy to work with. Figure 6 illustrates a straightforward mapping from an arbitrary logic network consisting of AND, OR, and NOT gates

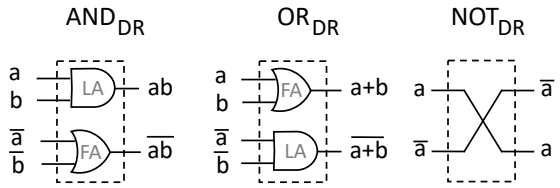

Fig. 6. Dual-rail (DR) implementations of AND, OR, NOT functions. Each AND and OR gate consists of one FA-LA pair, just with different wiring. The NOT gate has no overhead.

to a DR implementation using only ANDs and ORs. The complement of any bit is immediately available at zero circuit cost.

Corollary 2.1: Any Boolean function having DR inputs can be implemented using only AND and OR gates.

In this paper, DR logic forms the backbone of the design methodology, with other unordered codes being used in cases where they lead to fewer gates or are otherwise advantageous. For example, 1-out-of-n codes are natural for holding decoded values. So, one might simply maintain some values in decoded form. Translating between a DR code and a 1-out-of-n code can be done using n gates (AND gates for DR to 1-out-of-n; OR gates for 1-out-of-n to DR).

Note that the rote mapping between AND/OR/NOT Boolean functions and DR equivalents shown in Figure 6 is not required. In some cases, a cheaper DR design may be achieved by other means.

Example: If one begins with the "standard" full adder composed of 9 NAND gates, the straightforward DR translation consumes 18 FA/LA cells. However, a 14 cell implementation is possible [14] if freed from using only AND and OR DR pairs (see Figure 7). We note that inputs and outputs are still in DR format and that each logical AND and OR gate shown can be implemented with LA and FA cells, as already discussed.

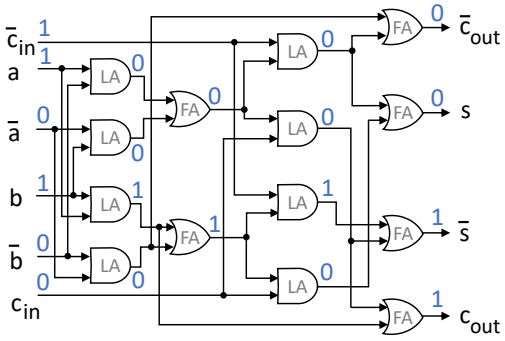

Fig. 7. Dual-rail full adder composed of 14 FA/LA cells [14]. Example where $a = 1$, $b = 1$, and $c_{in} = 0$ is shown.

IV. Realizing xSFQ in Hardware

In Section III, we showed that LA and FA cells realize AND and OR gates; thus, they are the primary building blocks of combinational xSFQ circuits. Their superconducting implementation has previously been explored [45], [46]. However,

prior designs require either the use of external signals or thermal/power adjustments to reset. On the one hand, the use of external signals leads to significant overheads. On the other hand, thermal/power-cycling is slow and forces a "complete" system reset, in which useful information may be lost. In this paper, we propose alternative circuit implementations that not only comply with alternating logic but also are more compact and outperform their counterparts.

Experimental setup. For the development and evaluation of our analog models, we use Cadence's Spectre simulator and superconductor device model files for the SFQ5ee 10 kA/cm^2 process [42].

A. FA & LA *Cells Implementation*

Our goal is to design circuits that satisfy functional correctness and: (a) evaluate without the need for a clock signal, (b) return to the ground state at the end of a logical cycle without the need for explicit wired reset signals, (c) provide a practical way to reinitialize in case an error occurs due to faulty hardware (or for any other reason), (d) achieve relatively similar propagation delays on datapaths that are at least as short as their SFQ AND/OR counterparts, and (e) minimize the JJ count, which affects area and defines the power dissipation of the circuit.

For the design of the LA circuit, shown in Figure 8 (i), we use as a reference the clockless dynamic SFQ AND gate originally developed by Rylov [32]. As already discussed, superconducting cell states arise from the storage of flux within SQUIDs (loops formed by two JJs, one on each side of an inductor). Here, such loops are formed around the input inductors L0 and L1, the output JJs from the prior stage, and J4. The output JJs from the prior stage are not shown, but are built into every gate with an output wire. In the top loop, L0 holds input *a* until input *b* arrives; likewise, in the bottom loop, L1 holds *b* until *a* arrives. Both inputs must arrive for the circuit to emit an output pulse and properly reset. In xSFQ, one pulse will arrive at each input port during a two-phase logical cycle, and thus the LA cell will transition from the *Init* state to either the *a arrived* or *b arrived* state and back to *Init* in one full cycle. If, for some reason, the second input does not arrive before the end of the logical cycle, the stored flux is removed by the J0-J2-R0 or J1-J3-R1 loop using a technique referred to here as *bleed-out* because it drains the flux quantum over time. The bleed-out rate of the cell depends to some extent on the amount of serial resistance. In the provided example, the bleed-out window is set to 29 ps, for which we used an R0 value of 0.67 Ω. However, when R0 is set to 0.70 Ω, the bleed-out window increases to 42 ps. Note that in all shown cases but the last, a and b input pulses arrive within a 29 ps offset (which is the cell's bleed-out window); thus, there is an output pulse. In the last case, the offset is 58 ps and no output pulse appears.

Figure 8 (ii) shows the schematic of the FA cell. It is implemented with a modification of an inverted C-element, originally presented by SUNY researchers [25]. Each input has a SQUID that stores the opposite input's flux until both

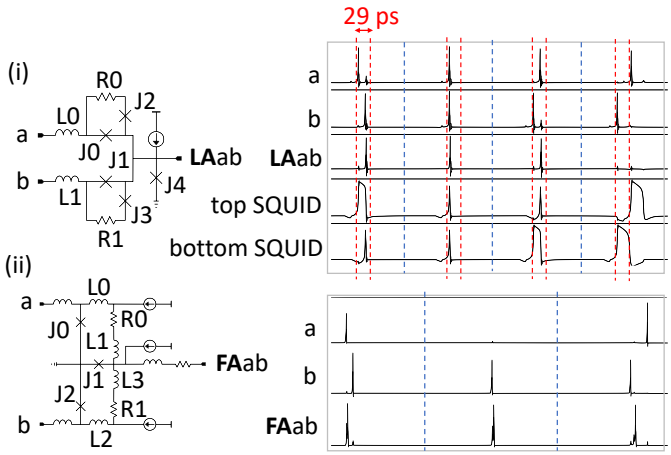

Fig. 8. Panel (i): LA cell schematic and waveform. The simulations cover both cases where input pulses arrive within and outside the bleed-out window boundaries. The results shown provide evidence that the cell satisfies the alternating logic requirements (used for normal operation) and supports "bleed-out" (used to recover from a faulty operation). For this example, the bleed-out rate, which is how quickly the SQUIDs can drain their respective fluxes, is 29 ps. Panel (ii): FA cell schematic and waveform. In the first and third cases, both input pulses arrive. The cell fires upon the arrival of the first input pulse, transitions to the *a arrived* or *b arrived* state, and waits for the second input to return to the *Init* state. In the second case, only one of the input pulses arrives (faulty operation). The bleed-out feature allows the gate to return back to the *Init* state.

signals have arrived – that is, when input b (a) arrives, it is both propagated to J4, generating an output pulse, and to SQUID J0-L0-L1-J1 (J2-L2-L3-J1), which stores the flux until a (b) arrives and cancels it out. Similar to the LA cell that bleeds out the flux, this cell also bleeds out through small serial resistors R0 and R1. We add a current source to the central node to mitigate bias current redistribution, and remove several redundant JJs on the input and output wires to reduce propagation delay.

Note that even in the rare case where two input pulses arrive "simultaneously", both LA and FA cells behave as expected (there are no race conditions in the asynchronous machines). Also, because clocking is not part of their semantics, the cells are delay-insensitive. Additionally, their bleed-out feature is used for resetting only in the case of faulty operation. Thus, when designing an xSFQ system, the bleed-out window of LA/FA cells should be configured to the length of a logical cycle (two synchronous phases).

Implementation results are summarized in Table I. To estimate the energy consumption of the designs, we assume that all JJs switch over a logical cycle. The switching energy of a single JJ is 2×10^{-19}J. Compared to prior LA and FA implementations [45], [46], the presented LA cell uses 1 fewer JJ, supports bleed-out, and does not introduce any latency overhead. The proposed FA cell uses 7 fewer JJs, does not require an explicit wired reset signal, supports bleed-out, and achieves 30% lower latency. The bias margins for both elements, according to our experiments, are -30% to +30%.

Compared to more traditional SFQ implementations of Boolean AND and OR gates (Table II), the xSFQ-based

TABLE I
AREA, LATENCY, AND ENERGY ESTIMATES OF LA AND FA CELLS IMPLEMENTED IN THE SFQ5EE 10 KA/CM^2 PROCESS.

Element	Area (#JJs)	Latency (ps)	Energy (aJ)
LA	5	8.0	1.0
FA	3	9.0	0.6

implementations require at least 30% fewer JJs and 55% less energy than their counterparts (even when LA and FA cells are used in pairs). Regarding latency, although 50% longer delays may be observed in single 2-input gates (due to the two-phase nature of the proposed DR alternating logic scheme), for composite functions, xSFQ designs still deliver performance gains. These gains are expected to grow with the size of the design, as the timing overhead incurred by fine-grained clocking is no longer a consideration.

TABLE II
AREA, LATENCY, AND ENERGY ESTIMATES OF SYNCHRONOUS SFQ AND AND OR CIRCUITS IMPLEMENTED IN THE SFQ5EE 10 KA/CM^2 PROCESS.

Element	Area (#JJs)	Latency (ps)	Energy (aJ)
sync. AND	11	9.2	2.2
sync. OR	12	8.0	2.4

To provide further evidence of xSFQ circuit efficiency, we use an 8-bit ALU as a reference example. An SFQ implementation [29] consists of 9 pipeline stages (fixed and equal to the number of gates on the critical path), requires 4,908 JJs, and achieves 120 TOPS/W in Low-Voltage RSFQ (1.4 POPS/W if only the switching power is considered, $\sim$ 22 μW). In xSFQ, pipeline depth is configurable. For a purely combinational design, our estimates are: 2,800 JJs, 1.7 μW (switching power), and 1.8 POPS/W. Note that in the above analysis, we assume that all pipeline stages in SFQ will always be busy.

B. Storage Elements

The above-described FA and LA cells, along with the alternating DR encoding introduced by xSFQ, are sufficient for the implementation of any combinational logic block. However, for the realization of real-world computing systems, xSFQ-amenable storage elements are still needed. As already discussed, in xSFQ, each logical cycle consists of a pair of physical cycles (excite-relax phases). Thus, an xSFQ storage element must be able to generate the time-offset complement with respect to the primary excite phase.

A simple method of implementing an xSFQ latch is to use a *pair* of synchronous DRO cells. Figure 9 illustrates such a component for the case of DR codes. In particular, the shown circuit latches the a or $\overline{a}$ when it arrives on the excite phase and releases it at the start of the next excite phase. Likewise, the data latched in the relax phase are written out on the next relax phase. Implementation results for DRO and SPLITTER cells are in Table III.

Note that although xSFQ requires data in a DR alternating format, traditional binary storage is still possible with the use of binary-to-alternating-DR (BAC) and alternating-DR-to-binary (ABC) converters. Presenting optimized BAC and ABC designs is, however, beyond the scope of this paper and will be covered in future work.

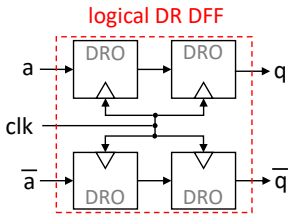

Fig. 9. xSFQ DR latch built from DROs. To fan-out the clock signal, we use three SPLITTERS (appearing as black dots).

TABLE III
AREA, LATENCY, AND ENERGY ESTIMATES OF DRO AND SPLITTER CIRCUITS IMPLEMENTED IN THE SFQ5EE 10 KA/CM^2 PROCESS.

Element	Area (#JJs)	Latency (ps)	Energy (aJ)
DRO	6	5.1	1.2
Splitter	3	4.3	0.6

V. REBALANCING EXCITE AND RELAX PHASES

With asynchronous combinational logic elements and synchronous storage elements available, one can perform digital design in xSFQ just as in CMOS without being constrained by pipelining with gate-level granularity. Figure 10 (i) illustrates a pipelined xSFQ circuit, where a block of combinational logic is surrounded by clock-synchronous register blocks.

Each logical DFF is implemented in xSFQ as a "double-pumped" latch (Figure 9) and the combinational logic consists of interconnected LA and FA cells. Although this structure is fully functional, if we strip each logical DFF down to the very basic digital design equivalents, the resulting system is unsatisfying from an architectural standpoint: it is a completely unbalanced pipeline, because no computation is done between two successive clock-synchronous DRO cells.

Observe that even though each pair of DRO cells is part of the same logical DFF, the DRO cells can be split and *redistributed* in a balanced way through retiming [24]. Figure 10 (ii) depicts a rebalanced version of the circuit shown in Figure 10 (i). As the DRO cells are pushed through the fabric of combinational logic, the excite and relax phases become balanced. In the ideal case, where the DRO cell propagation delay is zero and the combinational logic is perfectly balanced, retiming can completely hide the overhead associated with the relax phase – Figure 10 (iii).

Functional evaluation. In Section IV, we presented SPICE-level models and simulation results for all xSFQ logic elements. For the evaluation of more complex systems, we develop a discrete-event simulation framework, released as an open-source tool on GitHub[4]. To effectively express the behavior of the primary elements, we opt for a lightweight,

[4]https://github.com/UCSBarchlab/PyLSE

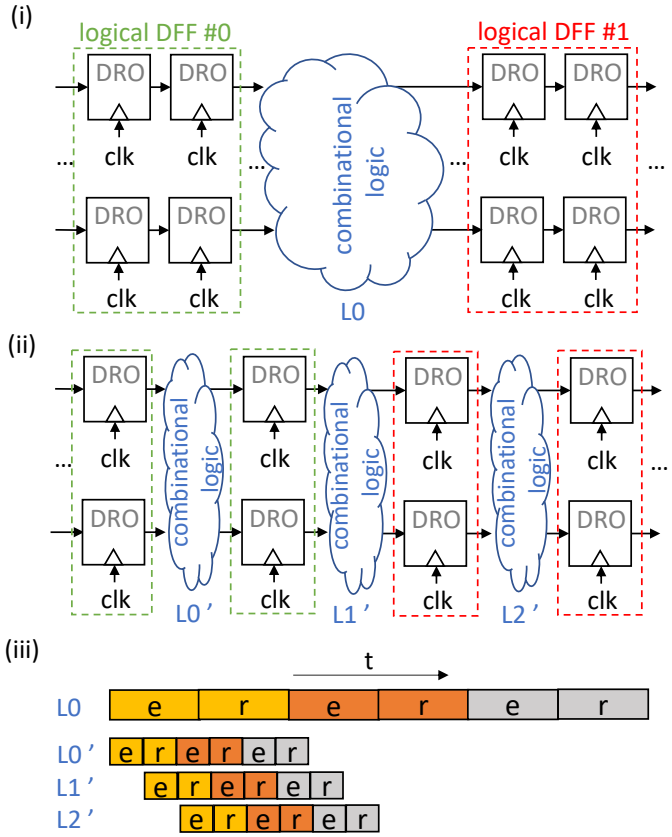

Fig. 10. Panel (i): Unbalanced xSFQ pipeline. Panel (ii): Distributing DRO cells in a balanced way through combinational logic blocks via retiming; L0 ≡ L0’+L1’+L2’. Panel (iii): Under the assumptions of zero DRO propagation delay and perfectly balanced retiming, 100% of the delay overhead introduced by alternating encoding can be hidden. We use letter e to represent excite phases and letter r for the relax phases. An excite-relax pair forms one logical cycle.

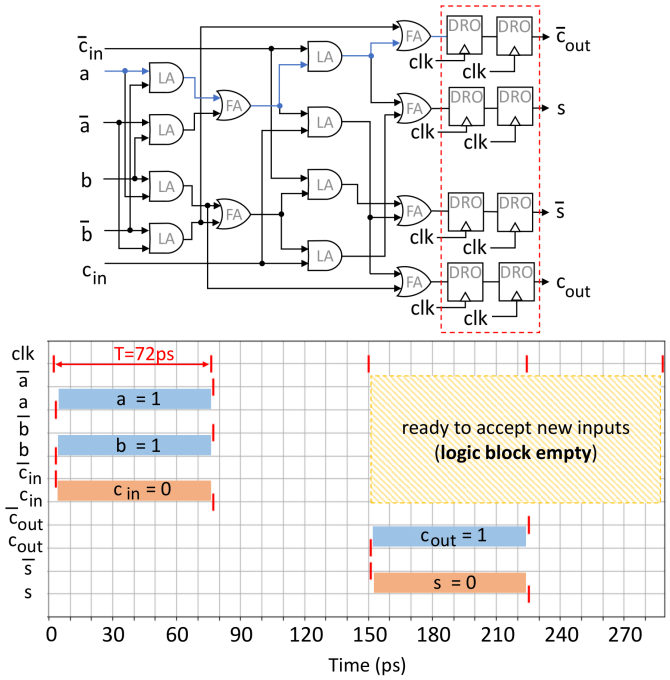

Fig. 11. Simulation of the DR full adder design shown in Figure 7. Vertical red lines (in the simulation graph) represent pulses. The minimum duration (T) of a physical cycle (phase) is defined by the delay of the longest signal path through the xSFQ combinational network – colored blue. For easier reading, we also illustrate the logical values of the circuit’s input and output variables (we probe output signals after they exit the last rank of output DRO cells).

object-oriented state machine-based implementation. The state machine corresponding to each cell contains all legal state transitions and allows for the easy diagnosis of fan-out violations or logical faults (e.g., fewer or more than the number of expected pulses appearing on a line). Moreover, each element object can store the number of JJs required for its physical implementation and its propagation delay, which facilitates area estimation, timing analysis, and a more realistic simulation. Events are discrete variables, not continuous ones, and simulation is based on the event-oriented paradigm, in which all pending events are first stored as a set and then inspected based on their scheduled event times.

To provide evidence for the functional correctness of the proposed retiming methodology, we apply it to the full adder design shown in Figure 7. Figure 11 provides simulation results for an unbalanced circuit – a DR full adder implemented as a combinational circuit and followed by a rank of logical DFFs. To make the design even more realistic, we assume that the inputs and outputs of each cell are buffered by Josephson Transmission Lines (JTLs), which are not computationally necessary but commonly used to improve flux transmission between SFQ logic cells. According to our SPICE simulations, the propagation delay of a JTL is 5.7 ps. The DRO cell setup time requirement – 2.3 ps – is also taken into account. Under these assumptions, the longest estimated propagation delay, and thus the shortest duration of a physical cycle (phase), is 72 ps.

The rebalanced full adder design is in Figure 12. In this case, the critical path consists of 2 xSFQ cells instead of 4. The longest propagation delay is 44.4 ps (the duration T of a physical cycle/phase is set to 45 ps in our simulation) and the number of logical cycles required to complete computation remains the same.

VI. Energy-Delay Product Analysis on Pipelined Superconductor Microprocessors.

Finding the optimal pipeline depth for a microprocessor is probably one of the most well-studied problems in computer microarchitecture [9], [13], [17], [22], [38]. To the first order, pipelining can offer a speed up of N, when N pipeline stages are used. However, this improvement comes at the expense of dynamic power. Thus, performance and power act in opposition.

Unlike CMOS design, architects have minimal control of the pipeline structure in prior conventional SFQ-based superconductor microprocessors [1], [18]. As already discussed, conventional superconducting Boolean gates are synchronous and act as independent pipeline stages – with the attendant benefits and problems. Considering individually clocked gates

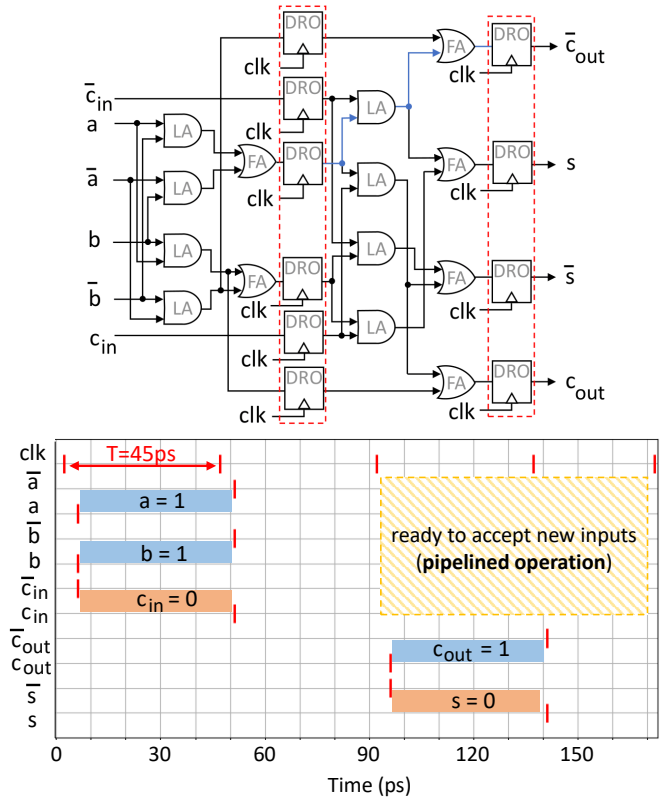

Fig. 12. Rebalanced DR full adder design and simulation. Vertical red lines (in the simulation graph) represent pulses. The minimum duration (T) of a physical cycle (phase) is defined by the delay of the longest signal path through the xSFQ combinational network – colored blue. For easier reading, we also illustrate the logical values of the circuit's input and output variables (we probe output signals after they exit the rank of output DRO cells, similar to Figure 11).

in conventional SFQ and the complexity of modern microarchitectures, we expect the number of pipeline stages in a prior conventional superconductor microprocessor to be in the order of hundreds. For example, our synthesis results of a single-cycle RV32I [2] indicate that the number of two-input gates on its critical path is approximately 150. Another noteworthy problem with the conventional approach is that the number of stages is not known until the synthesis process completes.

In contrast, the proposed xSFQ does not impose such constraints. In this section, we first revisit the CMOS performance models presented by Hartstein and Puzak [13], modify them where needed, then describe the area and energy consumption of xSFQ and conventional SFQ designs as functions of the pipeline depth, and, finally, conduct an energy-delay product (EDP) comparison. To the best of our knowledge, this is the first study on optimal pipeline depth for a superconductor microprocessor.

A. Performance Model

Following Hartstein and Puzak [13], the basic performance metric is Time per Instruction (TPI), which is the inverse of the (Million) Instructions per Second metric. TPI can be thought of as the sum of the time that a microprocessor is busy doing some useful work (T_{BZ}) and the time that it is stalled because of pipeline hazards (T_{NBZ}) divided by the total number of program instructions (N_I). To calculate the number of Logical Cycles per Instruction (LCPI) [5], which is another useful performance metric, we divide the sum of T_{BZ} and T_{NBZ} by T_{BZ}. Expressions for T_{BZ}, T_{NBZ}, TPI, and LCPI (for a scalar machine) are in Table IV.

We follow the original Hartstein and Puzak notation where possible. In particular, variable t_p represents the total logic delay of the microprocessor, t_s the physical cycle time, p the number of architectural pipeline stages, t_o the latch delay overhead, γ the weighted average of the fraction of the pipeline stalled by hazards ($\gamma \in [0, 1]$), and N_H the number of pipeline hazards.

In the case of xSFQ, p can take values ranging from 1 to $N_{lg\_cp}/2$, where N_{lg_cp} is the number of gates on the critical path. Exceeding this upper bound is not useful, as rebalancing no longer applies. In the t_s, T_{BZ}, and T_{NBZ} expressions, we divide/multiply by a factor of 2 to account for the effects of rebalancing. In conventional SFQ, t_s is equal to the longest propagation delay among the available synchronous SFQ gates, given that p is no longer a free variable. Note that the given equations do not capture timing skews or the propagation delay of interconnect lines.

B. Area Model

Regarding area, we assume that the majority of hardware resources is associated with logic gates (LA, FA, and conventional SFQ Boolean cells), latches (DRO cells), and SPLITTERs. We categorize these cells as follows: LA and FA cells are considered asynchronous, and conventional SFQ Boolean cells and DRO cells are considered synchronous. As for SPLITTERs, we count only those that are part of the clock distribution network, which dominates the total SPLITTER count. Table IV provides a description of our analytical area models (in terms of gate count).

As discussed in Section II, in conventional SFQ, DRO cells are needed for padding unequal datapaths. We assume that their number N_{dro} increases linearly with the number of logic gates (N_{lg}); o_{dp} is the growth rate associated with the padding overhead. The number of logic cells (N_{lc}) is equal to the number of logic gates (in SFQ), and the number of SPLITTERs on the clock line (N_{splt}) matches the total number of synchronous cells; $N_{dro} + N_{lg}$ in this case.

To estimate N_{dro} in an xSFQ design, the methodology introduced by Srinivasan et al. [38] is followed. N_L is the number of logical latches for a single-stage pipeline, η the latch growth exponent, and a factor of 2 is used to compensate for the additional cost of the xSFQ "double pumped" architectural latches. Variable o_{dr} denotes the overhead introduced by DR codes. More specifically, if LA and FA cells are used in pairs, $o_{dr} = 2$; otherwise, $o_{dr} = 1$. This overhead is accounted for in the logic cells estimation, as well ($N_{lc} = N_{lg} o_{dr}$). Moreover,

[5]We use LCPI instead of Cycles per Instruction (CPI) because each xSFQ logical cycle consists of two physical cycles.

in xSFQ, the only synchronous components are DRO cells, and thus $N_{splt} = N_{dro}$.

TABLE IV
PERFORMANCE, AREA, AND ENERGY MODELS FOR XSFQ AND CONVENTIONAL SFQ.

	xSFQ	**conv. SFQ**
$p \in$	$[1, N_{lg\_cp}/2]$	$[N_{lg\_cp}]$
$\delta \in$	$[1]$	$[0, 1]$
Performance		
$t_s =$	$t_o + t_p/(2p)$	t_o
$T_{BZ} =$	$2t_s N_I$	$t_s N_I$
$T_{NBZ} =$	$\gamma N_H(t_o(2p) + t_p)$	$\gamma N_H t_o p$
$TPI =$	$(T_{BZ} + T_{NBZ})/N_I$	$(T_{BZ} + T_{NBZ})/N_I$
$LCPI =$	$1 + T_{NBZ}/T_{BZ}$	$1 + T_{NBZ}/T_{BZ}$
Area		
$N_{dro} =$	$2N_{LP}{}^{\eta} o_{dr}$	$N_{lg} o_{dp}$
$N_{lc} =$	$N_{lg} o_{dr}$	N_{lg}
$N_{splt} =$	N_{dro}	$N_{dro} + N_{lg}$
Energy		
$E_{ac} =$	$(N_{lc} e_{lc} \delta) LCPI$	0
$E_{sc} =$	$(N_{dro} e_{dro} \delta) LCPI$	$(N_{dro} e_{dro} \delta + N_{lc} e_{lc} \delta) LCPI$
$E_{clk} =$	$(2N_{splt} e_{splt}) LCPI$	$(N_{splt} e_{splt}) LCPI$
$EPI =$	$E_{ac} + E_{sc} + E_{clk}$	$E_{ac} + E_{sc} + E_{clk}$

C. Energy Model

The energy metric used is Energy per Instruction (EPI), which is the sum of dynamic energy consumed by asynchronous cells (E_{ac}), synchronous cells (E_{sc}), and the clocking distribution network (E_{clk}) for the execution of a single instruction [6]. Expressions for E_{ac}, E_{sc}, and E_{clk} are in Table IV, and are functions of the following parameters: LCPI, N_{lc}, N_{dro}, e_{lc} (average energy dissipation of a logic cell), e_{dro} (energy dissipation of a DRO cell), e_{splt} (energy dissipation of a SPLITTER), and δ (switching activity factor).

In conventional SFQ, no LA and FA cells are used, $E_{ac} = 0$. However, the cost of logic gates is accounted for, along with the cost of DRO cells, in the E_{sc} expression. In xSFQ, the energy consumed by LA and FA cells is part of E_{ac}. The factor of 2 appearing in the E_{clk} expression compensates for the additional delay caused by the relax phase.

[6]As already discussed, recent energy-efficient SFQ logics have zero static power dissipation without sacrificing speed or compatibility with existing fabrication processes.

D. EDP Comparison

To quantify the gains of xSFQ over conventional SFQ technologies, we perform various simulations with N_H/N_I, γ, N_{lg_cp}, and o_{dp} as free variables. The assumptions are: (a) all gate inputs and outputs are buffered by JTLs, (b) the clock distribution network has zero skew and can be wave-pipelined (common practice in conventional SFQ designs), (c) xSFQ FA and LA cells are always used in pairs (the most conservative case), (d) excite and relax phases are balanced, and (e) $\eta = 1.3$, similar to Hartstein and Puzak [13]. We note that our models do not include the circuitry overhead required for pipeline interlock and flushing. Moreover, $N_{lg} = 10,000$, $N_{lg\_cp} = 150$ (based on synthesis results of a RISC-V RV32I core), and $\gamma = 0.8$, if not stated differently. The 2.8 ps and 2.3 ps setup time requirements of conventional SFQ Boolean and DRO cells are also considered.

An EDP versus pipeline depth comparison between xSFQ and conventional SFQ for various pipeline hazard rates is in Figure 13. In the case where $N_H = 0$, conventional SFQ achieves better results than xSFQ. This is expected, as this scenario favors very deep pipelines. LCPI is 1, each xSFQ logical cycle consumes two physical cycles, the minimum physical cycle time (t_s) is shorter for SFQ than xSFQ, and a gate-level pipelined xSFQ design requires more synchronous cells than its SFQ equivalent. However, as the number of hazards increases to more realistic values [27] and LCPI becomes greater than 1, xSFQ gains surpass conventional SFQ. Table V provides more detailed results.

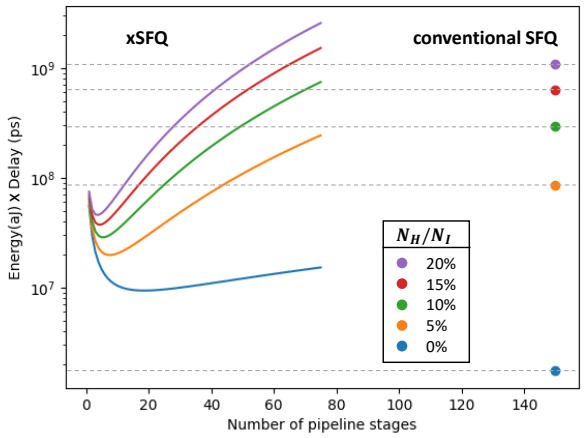

Fig. 13. EDP versus pipeline depth comparison. $N_H/N_I \in [0, 0.2]$, $\gamma = 0.8$, $o_{dp} = 2.5$, $N_{lg} = 10,000$, $N_{lg\_cp} = 150$, $\eta = 1.3$. The y-axis is on a logarithmic scale.

In all three cases – $\gamma = 0.8, \gamma = 0.9, \gamma = 1.0$ – xSFQ performs better than conventional SFQ in terms of EDP and EDDP (energy-delay^2 product) for non-zero pipeline hazard rates. More specifically, the gains increase super linearly with N_H/N_I, while the optimal design point shifts to shorter pipelines. For example, for $\gamma = 0.8$ and $N_H/N_I = 5\%$, the optimum number of pipeline stages p is 8 if optimized for

EDP, and 13 if optimized for EDDP. For $N_H/N_I = 10\%$ and $N_H/N_I = 20\%$, p becomes 5 and 4 for maximum EDP gains, and 9 and 6 for maximum EDDP gains. In a similar vein, increases in γ lead to higher EDP and EDDP gains and shorter pipelines. Moreover, xSFQ achieves $4\times$ lower EPI than conventional SFQ even in the case of hazard-free execution. An increase of one order of magnitude is observed for $N_H/N_I = 5\%$ and two orders of magnitude for $N_H/N_I = 20\%$.

TABLE V
EDP, EDDP, AND EPI GAINS ACHIEVED BY XSFQ OVER CONVENTIONAL SFQ.

$\gamma = 0.8$

N_H/N_I	EDP	EDDP	EPI
0%	0.2x (p=19)	0.05x (p=61)	4x
5%	4.3x (p=8)	1.6x (p=13)	27x
10%	10.2x (p=5)	4.5x (p= 9)	48x
15%	16.8x (p=4)	8.1x (p=7)	67.6x
20%	23.7x (p=4)	12.2x (p=6)	86x

$\gamma = 0.9$

N_H/N_I	EDP	EDDP	EPI
0%	0.2x (p=19)	0.05x (p=61)	4x
5%	5x (p=7)	1.9x (p=13)	29.6x
10%	11.8x (p=5)	5.3x (p=9)	53x
15%	19.4x (p=4)	9.6x (p=7)	74.6x
20%	27.1x (p=3)	14.3x (p=6)	94.5x

$\gamma = 1.0$

N_H/N_I	EDP	EDDP	EPI
0%	0.2x (p=19)	0.05x (p=61)	4x
5%	5.7x (p=7)	2.2x (p=12)	32.2x
10%	13.5x (p=5)	6.2x (p=8)	58x
15%	22x (p=4)	11.1x (p=6)	81.4x
20%	30.8x (p=3)	16.5x (p=5)	103x

An EPI breakdown of EDP-optimized xSFQ and conventional SFQ designs is provided in Table VI ($\gamma = 0.8$). As pipeline hazards ratio increases, shorter pipelines are preferable, which leads to less energy consumed by synchronous elements and clock distribution network (in the case of xSFQ), despite the increase in LCPI. In conventional SFQ, where pipeline depth is not configurable, the energy consumed by these components increases significantly.

TABLE VI
EPI BREAKDOWN OF EDP-OPTIMIZED XSFQ AND CONVENTIONAL SFQ DESIGNS.

N_H/N_I	E_{ac} (fJ)		E_{sc} (fJ)		E_{clk} (fJ)	
	xSFQ	SFQ	xSFQ	SFQ	xSFQ	SFQ
0%	24	0	15	50	27	49
5%	32	0	7	352	11	343
10%	34	0	4	653	7	637
15%	36	0	3	954	5	931
20%	39	0	3	1,256	6	1,225

Finally, Figure 14 plots EDP graphs for various N_{lg_cp} and o_{dp} values ($\gamma = 0.8$). The crossover point for all cases is below a 2% hazard rate and shifts to the left as N_{lg_cp} and o_{dp} increase. More specifically, EDP is more sensitive to the number of gates on the critical path than the overhead for the padding of uneven paths. For example, the crossover point for $N_{lg\_cp} = 100$ and $o_{dp} = 2.5$ is at $N_H/N_I = 2\%$, for $N_{lg\_cp} = 200$ and $o_{dp} = 2.5$ at $N_H/N_I = 1\%$, and for $N_{lg\_cp} = 100$ and $o_{dp} = 5.0$ at $N_H/N_I = 1.5\%$. For a 10% hazard rate, the EDP gains achieved by xSFQ over conventional SFQ for these three cases are $6.1\times$, $14.6\times$, and $8.5\times$, respectively, excluding the cost of interlock logic.

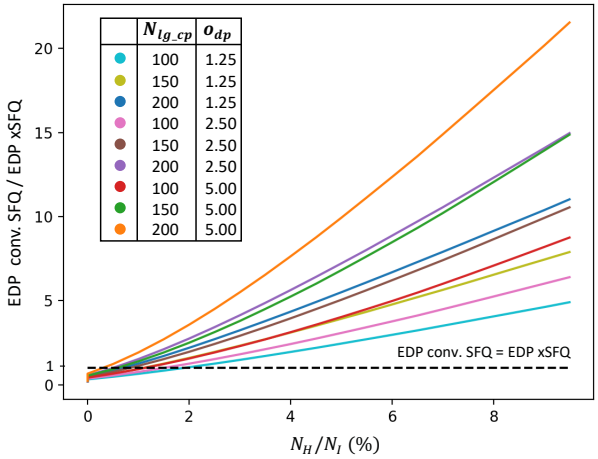

Fig. 14. EDP conventional SFQ/xSFQ comparison for N_H/N_I ranging from 0% to 10%, N_{lg_cp} from 100 to 200, and o_{dp} from 1.25 to 5.0.

VII. CONCLUSIONS

Superconducting technology is an increasingly promising candidate for energy-efficient computing, especially for large-scale high-throughput or low-latency applications having low memory requirements and involving communications-intensive processing. A major design constraint is that the pulse-driven nature of SFQ logic means that each logic element must "remember" when a pulse has come through so that it can be combined with future pulses. In order to implement traditional logic functions, each state machine must be returned to a ground state before new inputs can be applied. Rather than rely on a clock being delivered to each and every gate in the system as in conventional SFQ (which inflates circuit size, increases dynamic power dissipation, and forces the use of unnecessarily deep pipelines), we propose an alternating, unordered encoding (typically DR) that is functionally complete and allows for the design of completely clock-free combinational logic elements. In this new logic scheme, dubbed xSFQ, we rely on each pulsed data line to alternately "excite" and "relax" each logic element. Clocking is then used only for the synchronization of storage elements (e.g., latches), similar to CMOS, and sequential network designs have a conventional look-and-feel. To verify our hypothesis, we design analog circuit models for all proposed logic elements and validate their operation in a superconducting-aware version of SPICE.

We also build a discrete-event simulation framework to aid in the evaluation of more complex systems and use it to demonstrate the effectiveness of the presented optimizations, as well as the versatility of xSFQ.

Like any new logic family there are relative advantages and disadvantages compared to prior approaches. However, we find through a detailed analysis of energy-delay product for pipelined designs that xSFQ is superior to conventional SFQ in nearly all use-cases. The exceptions require hundreds of pipeline stages with 99% stall-free operation. If even a few levels of logic are desirable between (architectural) pipeline stages, xSFQ is superior in terms of EDP, EDDP, and EPI. For example, for a design resembling a RISC-V RV32I core and a 20% pipeline hazards, xSFQ achieves $31\times$ EDP, $17\times$ EDDP, and $103\times$ EPI gains compared to conventional SFQ. We expect these gains to be even more significant if the overhead of interlock logic is accounted for.

The provision of xSFQ's dual-rail construction, along with its alternating periods of excitation and relaxation, introduces new opportunities for phase rebalancing optimizations. Circuit- and gate-level enhancements that further exploit this logical framework are likely possible. Moreover, the freedom from excessively deep pipelines unlocks new architectural opportunities and makes the design process more familiar to those coming from more traditional digital design backgrounds. Finally, the presented analytical power-performance models allow for exploring the impact of pipeline depth on the efficiency of superconductor microprocessors and other non-streaming applications, as well as opening pathways for identifying the technology's key architectural challenges. The designs and simulation infrastructure used in this work can be found on our GitHub repository.

Acknowledgment

This material is based upon work supported by the National Science Foundation under Grants No. 1763699, 1730309, 1717779, 1563935. The authors would also like to thank Michael Christensen for his assistance with PyLSE, Dmitri Strukov for his support, and Sergey Rylov, Vasili Semenov, Konstantin Likharev, George Michelogiannakis, and the anonymous reviewers for their helpful comments.

References

[1] Y. Ando, R. Sato, M. Tanaka, K. Takagi, N. Takagi, and A. Fujimaki, "Design and demonstration of an 8-bit bit-serial rsfq microprocessor: Core e4," *IEEE Transactions on Applied Superconductivity*, vol. 26, no. 5, pp. 1–5, Aug 2016.

[2] K. Asanović and D. A. Patterson, "Instruction sets should be free: The case for risc-v," *EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2014-146*, 2014.

[3] C. L. Ayala, T. Tanaka, R. Saito, M. Nozoe, N. Takeuchi, and N. Yoshikawa, "Mana: A monolithic adiabatic integration architecture microprocessor using 1.4-zj/op unshunted superconductor josephson junction devices," *IEEE Journal of Solid-State Circuits*, vol. 56, no. 4, pp. 1152–1165, 2021.

[4] J. M. Berger, "A note on error detection codes for asymmetric channels," *Information and control*, vol. 4, no. 1, pp. 68–73, 1961.

[5] B. Bose, "On unordered codes," *IEEE Transactions on Computers*, no. 2, pp. 125–131, 1991.

[6] R. Cai, A. Ren, O. Chen, N. Liu, C. Ding, X. Qian, J. Han, W. Luo, N. Yoshikawa, and Y. Wang, "A stochastic-computing based deep learning framework using adiabatic quantum-flux-parametron superconducting technology," in *Proceedings of the 46th International Symposium on Computer Architecture*, ser. ISCA '19. New York, NY, USA: Association for Computing Machinery, 2019, p. 567–578. [Online]. Available: https://doi.org/10.1145/3307650.3322270

[7] Z. J. Deng, N. Yoshikawa, S. R. Whiteley, and T. Van Duzer, "Self-timing and vector processing in rsfq digital circuit technology," *IEEE Transactions on Applied Superconductivity*, vol. 9, no. 1, pp. 7–17, 1999.

[8] M. Dorojevets, P. Bunyk, and D. Zinoviev, "Flux chip: design of a 20-ghz 16-bit ultrapipelined rsfq processor prototype based on 1.75-/spl mu/m lts technology," *IEEE Transactions on Applied Superconductivity*, vol. 11, no. 1, pp. 326–332, March 2001.

[9] P. K. Dubey and M. J. Flynn, "Optimal pipelining," *Journal of Parallel and Distributed Computing*, vol. 8, no. 1, pp. 10–19, 1990.

[10] H. R. Gerber, C. J. Fourie, and W. J. Perold, "Rsfq-asynchronous timing (rsfq-at): a new design methodology for implementation in cad automation," *IEEE Transactions on Applied Superconductivity*, vol. 15, no. 2, pp. 272–275, 2005.

[11] M. Gurvitch, M. A. Washington, and H. A. Huggins, "High quality refractory josephson tunnel junctions utilizing thin aluminum layers," *Applied Physics Letters*, vol. 42, no. 5, p. 472–474, Mar 1983. [Online]. Available: https://aip.scitation.org/doi/10.1063/1.93974

[12] M. C. Hansen, H. Yalcin, and J. P. Hayes, "Unveiling the iscas-85 benchmarks: a case study in reverse engineering," *IEEE Design Test of Computers*, vol. 16, no. 3, pp. 72–80, 1999.

[13] A. Hartstein and T. R. Puzak, "Optimum power/performance pipeline depth," in *Proceedings. 36th Annual IEEE/ACM International Symposium on Microarchitecture, 2003. MICRO-36.* IEEE, 2003, pp. 117–125.

[14] D. S.-M. Ho, "The study of a totally self-checking adder." ILLINOIS UNIV URBANA COORDINATED SCIENCE LAB, Tech. Rep., 1972.

[15] A. Holmes, M. R. Jokar, G. Pasandi, Y. Ding, M. Pedram, and F. T. Chong, "Nisq+: Boosting quantum computing power by approximating quantum error correction," in *2020 ACM/IEEE 47th Annual International Symposium on Computer Architecture (ISCA)*, 2020, pp. 556–569.

[16] D. S. Holmes, A. M. Kadin, and M. W. Johnson, "Superconducting computing in large-scale hybrid systems," *Computer*, vol. 48, no. 12, pp. 34–42, Dec 2015.

[17] M. S. Hrishikesh, N. P. Jouppi, K. I. Farkas, D. Burger, S. W. Keckler, and P. Shivakumar, "The optimal logic depth per pipeline stage is 6 to 8 fo4 inverter delays," in *Proceedings 29th Annual International Symposium on Computer Architecture*, 2002, pp. 14–24.

[18] K. Ishida, M. Tanaka, I. Nagaoka, T. Ono, S. Kawakami, T. Tanimoto, A. Fujimaki, and K. Inoue, "32 ghz 6.5 mw gate-level-pipelined 4-bit processor using superconductor single-flux-quantum logic," in *2020 IEEE Symposium on VLSI Circuits*, 2020, pp. 1–2.

[19] K. Ishida, I. Byun, I. Nagaoka, K. Fukumitsu, M. Tanaka, S. Kawakami, T. Tanimoto, T. Ono, J. Kim, and K. Inoue, "Supernpu: An extremely fast neural processing unit using superconducting logic devices," in *Proceedings. 53th Annual IEEE/ACM International Symposium on Microarchitecture, 2020. MICRO-53.*, 2020.

[20] N. P. Jouppi, C. Young, N. Patil, D. Patterson, G. Agrawal, R. Bajwa, S. Bates, S. Bhatia, N. Boden, A. Borchers, R. Boyle, P.-l. Cantin, C. Chao, C. Clark, J. Coriell, M. Daley, M. Dau, J. Dean, B. Gelb, T. V. Ghaemmaghami, R. Gottipati, W. Gulland, R. Hagmann, C. R. Ho, D. Hogberg, J. Hu, R. Hundt, D. Hurt, J. Ibarz, A. Jaffey, A. Jaworski, A. Kaplan, H. Khaitan, D. Killebrew, A. Koch, N. Kumar, S. Lacy, J. Laudon, J. Law, D. Le, C. Leary, Z. Liu, K. Lucke, A. Lundin, G. MacKean, A. Maggiore, M. Mahony, K. Miller, R. Nagarajan, R. Narayanaswami, R. Ni, K. Nix, T. Norrie, M. Omernick, N. Penukonda, A. Phelps, J. Ross, M. Ross, A. Salek, E. Samadiani, C. Severn, G. Sizikov, M. Snelham, J. Souter, D. Steinberg, A. Swing, M. Tan, G. Thorson, B. Tian, H. Toma, E. Tuttle, V. Vasudevan, R. Walter, W. Wang, E. Wilcox, and D. H. Yoon, "In-datacenter performance analysis of a tensor processing unit," in *Proceedings of the 44th Annual International Symposium on Computer Architecture*, ser. ISCA '17. New York, NY, USA: Association for Computing Machinery, 2017, p. 1–12. [Online]. Available: https://doi.org/10.1145/3079856.3080246

[21] D. E. Kirichenko, S. Sarwana, and A. F. Kirichenko, "Zero static power dissipation biasing of rsfq circuits," *IEEE Transactions on Applied Superconductivity*, vol. 21, no. 3, pp. 776–779, 2011.

[22] S. R. Kunkel and J. E. Smith, "Optimal pipelining in supercomputers," *ACM SIGARCH Computer Architecture News*, vol. 14, no. 2, pp. 404–411, 1986.

[23] I. Kurosawa, H. Nakagawa, M. Aoyagi, M. Maezawa, Y. Kameda, and T. Nanya, "A basic circuit for asynchronous superconductive logic using rsfq gates," *Superconductor Science and Technology*, vol. 9, 1996.

[24] C. E. Leiserson and J. B. Saxe, "Retiming synchronous circuitry," *Algorithmica*, vol. 6, no. 1–6, p. 5–35, Jun. 1991. [Online]. Available: https://doi.org/10.1007/BF01759032

[25] K. Likharev and V. Semenov, "Rsfq logic/memory family: a new josephson-junction technology for sub-terahertz-clock-frequency digital systems," *IEEE Transactions on Applied Superconductivity*, vol. 1, no. 1, p. 3–28, Mar 1991. [Online]. Available: https://ieeexplore.ieee.org/document/80745

[26] G. M. Lilly, "Device for and method of one-way cryptographic hashing," Dec. 7 2004, uS Patent 6,829,355.

[27] A. Limaye and T. Adegbija, "A workload characterization of the spec cpu2017 benchmark suite," in *2018 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, 2018, pp. 149–158.

[28] M. Maezawa, I. Kurosawa, M. Aoyagi, H. Nakagawa, Y. Kameda, and T. Nanya, "Rapid single-flux-quantum dual-rail logic for asynchronous circuits," *IEEE Transactions on Applied Superconductivity*, vol. 7, no. 2, pp. 2705–2708, 1997.

[29] I. Nagaoka, M. Tanaka, K. Sano, T. Yamashita, A. Fujimaki, and K. Inoue, "Demonstration of an energy-efficient, gate-level-pipelined 100 tops/w arithmetic logic unit based on low-voltage rapid single-flux-quantum logic," in *2019 IEEE International Superconductive Electronics Conference (ISEC)*, 2019, pp. 1–3.

[30] G. Pasandi, A. Shafaei, and M. Pedram, "Sfqmap: A technology mapping tool for single flux quantum logic circuits," in *2018 IEEE International Symposium on Circuits and Systems (ISCAS)*, 2018, pp. 1–5.

[31] D. A. Reynolds and G. Metze, "Fault detection capabilities of alternating logic," *IEEE Transactions on Computers*, no. 12, pp. 1093–1098, 1978.

[32] S. V. Rylov, "Clockless dynamic sfq and gate with high input skew tolerance," *IEEE Transactions on Applied Superconductivity*, vol. 29, no. 5, pp. 1–5, 2019.

[33] V. K. Semenov, Y. A. Polyakov, and S. K. Tolpygo, "Very large scale integration of josephson-junction-based superconductor random access memories," *IEEE Transactions on Applied Superconductivity*, vol. 29, no. 5, pp. 1–9, 2019.

[34] V. Semenov, E. Golden, and S. K. Tolpygo, "Sfq bias for sfq digital circuits," *IEEE Transactions on Applied Superconductivity*, p. 1–1, 2021. [Online]. Available: https://ieeexplore.ieee.org/document/9381656

[35] V. K. Semenov, Y. A. Polyakov, and S. K. Tolpygo, "Ac-biased shift registers as fabrication process benchmark circuits and flux trapping diagnostic tool," *IEEE Transactions on Applied Superconductivity*, vol. 27, no. 4, p. 1–9, Jun 2017. [Online]. Available: https://arxiv.org/abs/1701.03837

[36] J. E. Smith, "On separable unordered codes," *IEEE transactions on computers*, vol. 100, no. 8, pp. 741–743, 1984.

[37] I. I. Soloviev, N. V. Klenov, S. V. Bakurskiy, M. Y. Kupriyanov, A. L. Gudkov, and A. S. Sidorenko, "Beyond moore's technologies: operation principles of a superconductor alternative," *Beilstein Journal of Nanotechnology*, vol. 8, p. 2689–2710, Dec 2017. [Online]. Available: http://dx.doi.org/10.3762/bjnano.8.269

[38] V. Srinivasan, D. Brooks, M. Gschwind, P. Bose, V. Zyuban, P. N. Strenski, and P. G. Emma, "Optimizing pipelines for power and performance," in *35th Annual IEEE/ACM International Symposium on Microarchitecture, 2002. (MICRO-35). Proceedings.*, 2002, pp. 333–344.

[39] G. Tang, P. Qu, X. Ye, and D. Fan, "Logic design of a 16-bit bit-slice arithmetic logic unit for 32-/64-bit rsfq microprocessors," *IEEE Transactions on Applied Superconductivity*, vol. 28, no. 4, pp. 1–5, June 2018.

[40] G. Tang, K. Takata, M. Tanaka, A. Fujimaki, K. Takagi, and N. Takagi, "4-bit bit-slice arithmetic logic unit for 32-bit rsfq microprocessors," *IEEE Transactions on Applied Superconductivity*, vol. 26, no. 1, pp. 1–6, Jan 2016.

[41] S. S. Tannu, P. Das, M. L. Lewis, R. Krick, D. M. Carmean, and M. K. Qureshi, "A case for superconducting accelerators," in *Proceedings of the 16th ACM International Conference on Computing Frontiers*, ser. CF '19. New York, NY, USA: Association for Computing Machinery, 2019, p. 67–75. [Online]. Available: https://doi.org/10.1145/3310273.3321561

[42] S. Tolpygo, V. Bolkhovsky, T. Weir, A. Wynn, D. Oates, L. Johnson, and M. Gouker, "Advanced fabrication processes for superconducting very large scale integrated circuits," *IEEE Transactions on Applied Superconductivity*, p. 1–1, 2016. [Online]. Available: https://ieeexplore.ieee.org/document/7386652

[43] S. K. Tolpygo, V. Bolkhovsky, R. Rastogi, S. Zarr, A. L. Day, T. J. Weir, A. Wynn, and L. M. Johnson, "Developments toward a 250-nm, fully planarized fabrication process with ten superconducting layers and self-shunted josephson junctions," *2017 16th International Superconductive Electronics Conference (ISEC)*, Jun 2017. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/8314189

[44] G. R. Trimble, "The ibm 650 magnetic drum calculator," *IEEE Annals of the History of Computing*, vol. 8, no. 01, pp. 20–29, jan 1986.

[45] G. Tzimpragos, J. E. Volk, D. Vasudevan, N. Tsiskaridze, G. Michelogiannakis, A. Madhavan, J. Shalf, and T. Sherwood, "Temporal computing with superconductors," *IEEE Micro*, pp. 1–1, 2021.

[46] G. Tzimpragos, D. Vasudevan, N. Tsiskaridze, G. Michelogiannakis, A. Madhavan, J. Volk, J. Shalf, and T. Sherwood, "A computational temporal logic for superconducting accelerators," in *Proceedings of the Twenty-Fifth International Conference on Architectural Support for Programming Languages and Operating Systems*, ser. ASPLOS '20. New York, NY, USA: Association for Computing Machinery, 2020, p. 435–448. [Online]. Available: https://doi.org/10.1145/3373376.3378517

[47] F. Ware, L. Gopalakrishnan, E. Linstadt, S. A. McKee, T. Vogelsang, K. L. Wright, C. Hampel, and G. Bronner, "Do superconducting processors really need cryogenic memories? the case for cold dram," in *Proceedings of the International Symposium on Memory Systems*, ser. MEMSYS '17. New York, NY, USA: Association for Computing Machinery, 2017, p. 183–188. [Online]. Available: https://doi.org/10.1145/3132402.3132424

[48] Y. Yamanashi, M. Tanaka, A. Akimoto, H. Park, Y. Kamiya, N. Irie, N. Yoshikawa, A. Fujimaki, H. Terai, and Y. Hashimoto, "Design and implementation of a pipelined bit-serial sfq microprocessor, core1β," *IEEE Transactions on Applied Superconductivity*, vol. 17, no. 2, pp. 474–477, June 2007.