Temporal Computing With Superconductors

Georgios Tzimpragos and Jennifer Volk, *University of California, Santa Barbara, Santa Barbara, CA, 93106, USA*

Dilip Vasudevan, *Lawrence Berkeley National Laboratory, Berkeley, CA, 94720, USA*

Nestan Tsiskaridze, *Stanford University, Stanford, CA, 94305, USA*

George Michelogiannakis, *Lawrence Berkeley National Laboratory, Berkeley, CA, 94720, USA*

Advait Madhavan, *University of Maryland, College Park, MD, 20742, USA*

John Shalf, *Lawrence Berkeley National Laboratory, Berkeley, CA, 94720, USA*

Timothy Sherwood, *University of California, Santa Barbara, Santa Barbara, CA, 93106, USA*

Creating computing systems able to address our ever-increasing needs, especially as we reach the end of CMOS transistor scaling, will require truly novel methods of computing. However, the traditional logic abstractions and the digital design patterns we understand so well have coevolved with the hardware technology that has embodied them. As we look past CMOS, there is no reason to think that those same abstractions best serve to encapsulate the computational potential inherent to emerging devices. We posit that a new and radically more efficient foundation for computing lies at the intersection of superconductor electronics and delay-coded computation. Building on recent work in race logic, we show that superconducting circuits can naturally compute directly over temporal relationships between pulse arrivals; that the computational relationships between those pulse arrivals can be formalized through a functional extension to a temporal predicate logic used in the verification community; and that the resulting architectures can operate asynchronously and describe real and useful computations. We verify our hypothesis through a combination of detailed analog circuit models and layout designs, a formal analysis of our abstractions, and an evaluation of several superconducting temporal accelerators.

Superconductor electronics offer the potential to perform computations at tremendous speeds and energy savings—especially on large scales.[1] Unfortunately, a semantic gap exists between the level-driven logic, which CMOS designs accept as a foundation, and the pulse-driven logic that is naturally supported by the most compelling superconducting technologies. This gap creates a variety of challenges across the full hardware stack, from the circuits up through the CAD and architecture levels.

0272-1732 © 2021 IEEE
Digital Object Identifier 10.1109/MM.2021.3066377
Date of publication 17 March 2021; date of current version 25 May 2021.

A pulse, unlike a stable voltage level, will fire through a channel for only an instant. Arranging the network of superconducting components so that input pulses—driven by the transfer of magnetic flux quanta—always arrive simultaneously to logic gates to maintain the illusion of a strictly Boolean evaluation is a significant engineering hurdle and results in unavoidable overheads. If we instead think about these pulses as the representation of nonbinary data, the natural language for expressing computations over those data would be one that can efficiently describe the temporal relationships between these pulses. Here, we draw upon two distinct lines of research, both currently disconnected from superconducting.

First, recent work has shown that delay-based encoding has impressive expressive power and

practical utility in implementing important classes of accelerators.[2–4] The principles of the delay-coded logic described in that prior work apply directly to problems in superconducting. However, the fact that its primitive operations have been so far implemented only in CMOS under specific assumptions—e.g., edges are used to denote event occurrences—brings up questions about their implementability and efficiency in the much different superconducting technology.

Second, there is a long history of work in temporal logic. Temporal logic systems allow for the representation and reasoning of propositions qualified in terms of time. Although expressing and evaluating constraints on the order of events is particularly useful for formal verification purposes, the predicate nature of existing temporal logics makes them insufficient for desired computing tasks. We instead need a temporal logic with computational capabilities that takes events as inputs and creates new events as outputs based on the input relationships.

In this article, we address these issues and explore the potential of superconducting temporal computing by proposing a new computational temporal logic. This computational temporal logic subsumes classical temporal predicate logic and gives clear, precise, and useful semantics to delay-based computations. Moreover, we demonstrate how this approach enables a tradeoff between implementation complexity and delay; show how one can realize a functionally complete set of temporal operations in superconducting circuits; and create useful designs that effectively capture the potential of these circuits. Finally, we identify critical timing constraints and develop proof-of-concept accelerator designs that demonstrate the functional correctness and high performance of the proposed logic scheme, methodology, and architectures. Open-source implementations of our superconducting temporal primitives and accelerators can be found on github.

BACKGROUND

Computing With Superconductors

Pulse-based superconductor electronics are characterized by three basic features: 1) the absence of resistance in static circuits at superconducting temperatures, 2) the Josephson effect, which governs the fundamental switching element in superconductor circuits, the Josephson junction (JJ), and 3) the propagation of single flux quanta (SFQ), appearing as ps-duration, mV-amplitude pulses across switching JJs, instead of static-voltage levels.

Unlike semiconductor devices, the resistance of superconducting circuits is zero, and thus, their speed is not limited by RC time constants. Moreover, the propagation of digital data (in superconducting interconnects) is near the speed of light, lossless, and consumes no power. These speed and power benefits do not come without challenges, though. One of the most profound is the pulse-based nature of computation. Developing logic using only pulses presents intellectual and application challenges alike. Pulses cannot be sampled like voltage levels because they do not coincide with ps precision (because of this reason, we usually end up with synchronous superconducting gates and fine-grained clocking). Also, the relatively low integration density of JJs along with the need for Splitters to fan out can inflate the circuit size considerably. Finally, the lack of reliable and high-capacity superconducting memories imposes its own distinctive constraints.

Because of these unique characteristics (and limitations) of superconducting, innovation requires that research efforts focus on computing paradigms and architectures that depart from classic CMOS-inspired solutions. Examples of such alternative solutions are those that 1) use fewer JJs than transistors for the same information throughput, 2) allow for easier clocking, and 3) have lower memory requirements.[5] We claim that temporal computing satisfies all these "constraints" and paves a promising path forward.

Race Logic

Race logic is a prime example of computing in the time domain.[2,4] Under race logic, information is represented as a delay; and a set of basic temporal operations—FirstArrival (*FA*), LastArrival (*LA*), Delay (*D*), and Inhibit (I_S)—replaces the OR, AND, and NOT we know from Boolean logic.

Figure 1 illustrates the functionality of these temporal operations. As their names indicate, *FA* and *LA*

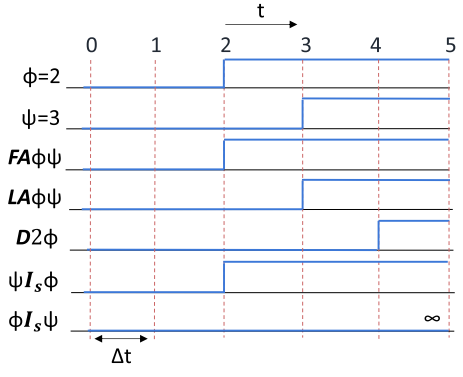

FIGURE 1. Basic operations of race logic are FirstArrival (*FA*), LastArrival (*LA*), Delay (*D*), and strict Inhibit (I_S). Δt corresponds to the delay associated with one time step.

"fire" as soon as the first and last high inputs arrive, respectively. Under the assumption that smaller delays in rise time encode smaller magnitudes, whereas larger magnitudes are encoded as longer delays, *FA* realizes MIN function and *LA* MAX function. The *D* operation delays an input event by *c* units of time; in the shown example, $c = 2$. Since the arrival time of the rising edge is what encodes information, delaying the $0 \rightarrow 1$ transition by a fixed amount of time is equivalent to CONSTANT − ADDITION. Finally, the l_S operation, inspired by the behavior of inhibitory postsynaptic potentials in the neurons of the neocortex,[3] works as a nonlinear filter that has two inputs: 1) an inhibiting signal and 2) a data signal. If the data signal arrives first, it is allowed to pass through the gate unchanged; otherwise, no event occurs on the output line, which denotes ∞. Prior to the next computation, race logic-driven circuitry must be reset.

Space–Time Algebra

Space–time algebra provides a mathematical underpinning for temporal processing.[3] Any function defined over the domain $S = \{0, N^+, \infty\}$, where N^+ is the set of positive natural numbers, that satisfies the properties of *invariance* and *causality* complies with this algebra. Interestingly, one can always make an arbitrary function satisfy these properties with the use of a reference signal *r*. Thus, any *n*-ary function defined over *S* is expressible in race logic with at most $n + 1$ variables.

FORMALIZATION

The transition from Boolean to race logic gives us the freedom to set up a clean infrastructure and avoid retrofitting superconductor electronics to existing abstractions and architectures. To establish a solid foundation and lay the groundwork for the development of systematic methods at all levels, we formally define the above-described temporal operations.

Computational Temporal Logic

Given the long history of temporal logics in the verification community, we use the well-established setting of linear temporal logic (LTL) as a starting point. The basic LTL operators are: ◇ *sometime in the future*, □ *always in the future*, ◯ *next time* (*tomorrow*), *U until*, and *R release*. Past LTL (PLTL) extends LTL with past-time operators, which are the temporal duals of LTLs future-time operators, and allows one to concisely express statements on the past time instances, such as: ◆ *sometime in the past*, ■ *always in the past* (*historically*), ● *previous time* (*yesterday*), *S since*, and *T trigger*.[6,7]

TABLE 1. Semantics of PLTL.

$\langle S,t\rangle$	$\models$	$\blacklozenge\phi$	iff	$\exists k{:}\ k \in [0,t].\ \langle S,k\rangle \models \phi$
$\langle S,t\rangle$	$\models$	$\blacksquare\phi$	iff	$\forall k{:}\ k \in [0,t].\ \langle S,k\rangle \models \phi$
$\langle S,t\rangle$	$\models$	$\bullet\,\phi$	iff	$t > 0$ and $\langle S,t-1\rangle \models \phi$
$\langle S,t\rangle$	$\models$	$\phi \mathbf{S} \psi$	iff	$\exists k{:}\ k \in [0,t].\ (\langle S,k\rangle \models \psi$ and $\forall j{:}\ j \in (k,t].\ \langle S,j\rangle \models \phi)$
$\langle S,t\rangle$	$\models$	$\phi \mathbf{T} \psi$	iff	$\forall j{:}\ j \in [0,t].\ (\langle S,j\rangle \models \psi$ or $\exists k{:}\ k \in (j,t].\ \langle S,k\rangle \models \phi)$

A definition of the semantics of PLTL is provided in Table 1. The notation $\langle S,t\rangle$ is used to signify a system *S* at time step *t*. We say that an *event* ϕ *occurs* at time step *t* in the system *S*, if ϕ holds at time step *t* in *S*, denoted by $\langle S,t\rangle \models \phi$.

These operators allow for the efficient representation and reasoning of propositions qualified in terms of time; e.g., an event in a system *S* has happened sometime in the past. However, they are incapable of encapsulating *when* a formula ϕ holds, which is essential for race logic. To address this issue, we introduce the *earliest-occurrence* function $\mathcal{E}_{\langle S,t\rangle}(\phi)$

$$\mathcal{E}_{\langle S,t\rangle}(\phi) = \begin{cases} t_{\min}, & \exists\ t_{\min}{:}\ t_{\min} \in [\![\phi]\!]_{\langle S,t\rangle}.\langle S,t_{\min}\rangle \models \phi \\ & \text{and } \forall j : j \in [0,t_{\min}).\langle S,j\rangle \not\models \phi \\ \infty, & \text{otherwise.} \end{cases}$$

This earliest occurrence function receives as input a formula ϕ and returns the *earliest* time step $t_{\min} \in [\![\phi]\!]_{\langle S,t\rangle}$, where $[\![\phi]\!]_{\langle S,t\rangle}$ is the scope of ϕ at time step *t* in the system *S* (the scope of ϕ denotes an interval of time steps ϕ operates on at time step *t*), such that $\langle S,t_{\min}\rangle \models \phi$. If ϕ does not hold at any time step within $[\![\phi]\!]_{\langle S,t\rangle}$, the earliest occurrence function returns ∞, which represents an unreachable time step.

Race Logic Semantics

The semantics of *FA*, *LA*, *D*, and l_S operations can be formally defined using existing PLTL operators; as shown in Table 2 ($\bullet^c$ denotes the application of ● operator *c* times). To extract the step at which these formulas evaluate to *True* for the first time in their scope, $\mathcal{E}_{\langle S,t\rangle}()$ can be used. For example, $\mathcal{E}_{\langle S,t\rangle}(FA\phi\psi)$ will return the first time step that either ϕ or ψ hold.

TABLE 2. PLTL-based semantics of race logic operations.

$\langle S,t\rangle$	$\models$	$\mathbf{FA}\phi\psi$	iff	$\langle S,t\rangle \models \blacklozenge\phi \vee \blacklozenge\psi$
$\langle S,t\rangle$	$\models$	$\mathbf{LA}\phi\psi$	iff	$\langle S,t\rangle \models \blacklozenge\phi \wedge \blacklozenge\psi$
$\langle S,t\rangle$	$\models$	$\mathbf{D}c\phi$	iff	$\langle S,t\rangle \models \bullet^c \blacklozenge\phi$
$\langle S,t\rangle$	$\models$	$\psi \mathbf{I}_S \phi$	iff	$\langle S,t\rangle \models \blacklozenge(\phi \wedge \blacksquare\neg\psi)$

Note that an important property of race logic is that at most one event is allowed to occur per "wire" across the entire computation. Bounding switching activity leads to simple and power-efficient hardware designs, and it may even allow for an overhead-free transition between temporal and binary domains.[4] In the case of superconducting, verifying that this property holds is crucial to the correct operation of the circuits presented in the section that follows. For example, if more than one pulse appears per line, the proposed I_S design will not satisfy its specification anymore. A formal definition of this property—represented here by $\mathcal{A}$—using PLTL operators follows:

$$\langle S, t \rangle \models \mathcal{A}\phi \text{ iff } \blacksquare(\phi \rightarrow \bullet\blacksquare\neg\phi).$$

SUPERCONDUCTING TEMPORAL ARCHITECTURES

Realizing Temporal Operations in Superconducting

The way in which events are encoded plays a critical role in selecting the hardware that most efficiently implements logic operators. For example, in CMOS, in which (under race logic) temporal events are represented by rising edges, *FA* and *LA* functions are realized with a single OR and AND gate, respectively.[2,4] An important property of edge-based event encoding is that it automatically keeps track of the input state at all times—a signal that has made a transition from a "low" to a "high" state will not make a transition back to low in the same computation. This feature breaks down, though, when dealing with pulses—as discussed in the "Background" section, superconducting technologies encode digital data as SFQ voltage pulses. Pulses naturally return back to their low state, preventing downstream nodes from implicitly knowing the state of its predecessors.

A potential solution to this problem is to embed the state into each gate. Interestingly, the majority of superconducting single flux quantum (SFQ) elementary cells have both logic and storage abilities;[8] thus, they can be thought of as simple state machines. To facilitate the mapping between our temporal operations and existing stateful SFQ elements, we draw *FA*, *LA*, *D*, and I_S as Mealy machines—as shown in Figure 2. Following this representation, we build *FA* with a Merge and a Destructive Read Out (DRO) cell, *LA* with a C-element cell, *D* with a sequence of Josephson Transmission Lines (JTL), and I_S with an Inverter cell. The length of the JTL chain depends on the selected Δt. In contrast to the other shown cells, JTLs do not hold state; thus, more than one SFQ pulses can propagate through a JTL chain

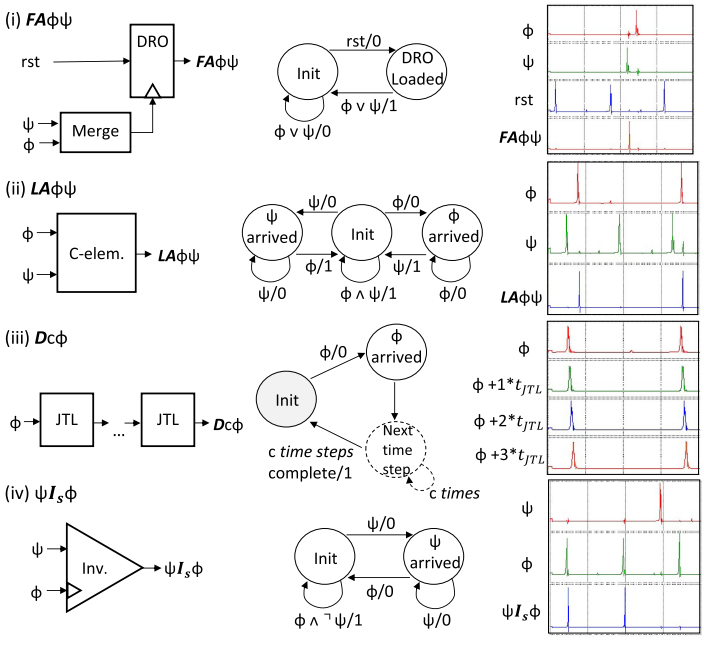

FIGURE 2. Block diagrams, Mealy machine representations, and WRSPICE simulations of race logic operations implemented in Rapid SFQ.

TABLE 3. Area and latency estimates.

Function	Area (#JJs)	Latency (ps)
FA	10	13
LA	6	8
D	2/JTL	5/JTL
I_s	8	11

concurrently. To highlight this difference, we color the "Init" node in the *Dc*ϕ Mealy machine gray.

Area and latency results for each of these operators are provided in Table 3. The shown estimates are based on our WRSPICE simulations using the MIT-LL SFQ5ee 10-kA/cm^2 process. Note that one of the goals of this article is to repurpose existing superconducting cells where possible, like those from Stony Brook.[8]

Self-Clocked Temporal SFQ Circuits

In contrast to more traditional superconducting approaches, where each gate is synchronous and fine-grained clocking is necessary, the proposed scheme relies on asynchronous operations. That is, even in the cases of *FA* and I_s, where synchronous cells are used, a data signal is routed to their "clock" port. However, the use of synchronous components may sometimes be beneficial. For example, although Coincidence—which gets satisfied if and only if both ϕ and ψ arrive within the same interval—can be built from the above-described operations,[3] a more efficient implementation is possible: all that is needed for its realization is a synchronous AND gate.

To avoid costly clock trees and the clock skew problems that come with them, we propose a data-driven self-timing (DDST) scheme—as shown in Figure 3. In a DDST system, timing information is carried by data. More specifically, the required clock signal is generated locally by a logical OR function between the data lines, which in the provided example

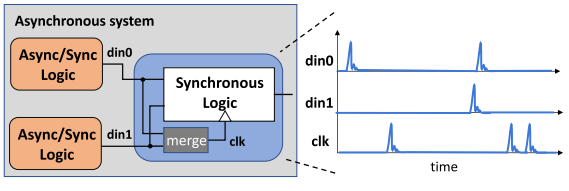

FIGURE 3. Proposed DDST scheme. The clock signal can be locally generated from input data at each gate. If no input pulse arrives, it is safe to assume the operator idle; thus no clock pulse is required.

is implemented by a Merger. Obviously, the shown DDST design is not suitable for Boolean circuits; e.g., a NOT gate must be clocked even in the absence of an incoming pulse. However, this is not the case for temporal systems, as temporal operators can be safely considered idle for the time steps that no input pulse arrives. To extend the "evaluation" window of a synchronous logic block (defined by the delay between data and clock pulse arrivals), JTLs can be added after the Merger cell.

EVALUATION

For the evaluation of the proposed logic scheme and methodology, we build, simulate, and measure the performance of various superconducting temporal accelerators. A detailed description of our race trees[4] implementation follows. Interested readers can find additional designs and information in the original paper[9] and our github repository: https://github.com/UCSBarchlab/Superconducting-Temporal-Logic.

Experimental Setup and Design Principles

We perform circuit simulations and analysis in both the open-source WRSPICE and Cadence Spectre platforms using MIT-Lincoln Lab's SFQ5ee 10-kA/cm^2 process node. The layout is completed in Cadence Virtuoso version 16.1. For gate isolation, delay balancing, and interconnects implementation, JTLs are used. For fan out, we use Splitters—denoted by *s* in the figure that follows. Finally, we define minimum Δt in a way that is similar to finding the cycle-time in wave-pipelined architectures.

Proof-of-Concept Design: Race Trees

In the case of race trees (ensembles of decision trees implemented in race logic), each tree node is considered an independent temporal threshold function and realized with a single Inhibit operator. Figure 4(ii) shows the block diagram of an SFQ race tree corresponding to the decision tree seen in Figure 4(i). For the construction of NOT gates, which are required for the implementation of the label decoder, Inhibits and an upper bound reference signal t_{ub} are used. This reference signal denotes the end of a specific time interval of interest (directly related to the inputs resolution in this case); thus, NOT will fire at $t = t_{ub}$ if and only if the gate has received no input spikes from time reference 0 until that moment.

Figure 4(iii) and (iv) provides simulation results and a layout diagram of this design. In the shown example, *x* and *y* are set to 4 and 1, and t_{ub} is equal to 5. As

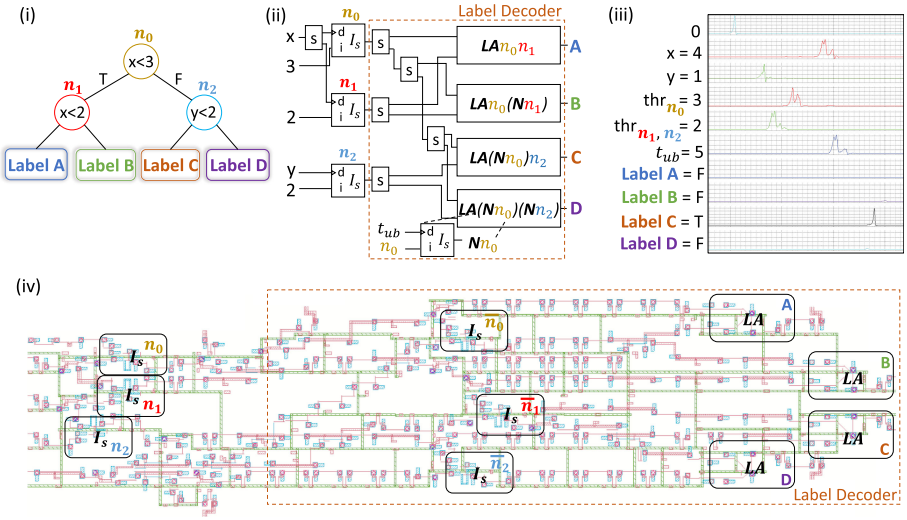

FIGURE 4. Panel (i): Decision tree with three nodes. Panel (ii): Block diagram of an SFQ race tree. Panel (iii): Simulation results for $x = 4$ and $y = 1$. Panel (iv): Layout diagram (unlabeled JJs are used for interconnection, splitting, routing, and testing purposes).

TABLE 4. Estimated latency results for hardwired race trees in both CMOS (f = 1 GHz)[4] and SFQ ($\Delta t = 25$ ps).

Depth	Input res.	CMOS Latency	SFQ Latency	Improvement
6	4 bits	17 ns	0.464 ns	37×
6	8 bits	257 ns	6.464 ns	40×
8	4 bits	17 ns	0.490 ns	35×
8	8 bits	257 ns	6.490 ns	40×

expected, the final outcome is Label *C*. The total latency, for $\Delta t = 25$ ps, is 150 ps and the design consists of 164 JJs; 72 JJs for logic elements, 24 JJs for Splitters, and 68 JJs for JTLs. The number of JJs in the layout is greater than 164 because of the additional cost of routing and the inclusion of testing circuitry. More results and a comparison with CMOS can be found in Table 4.

CONCLUSIONS AND FUTURE IMPACT

CMOS scaling has been a driving factor for computing technology for many decades and the transistors manufactured today are multiple orders of magnitude more efficient, compact, and less expensive than those built 30 years ago. However, as CMOS approaches its thermal limits, keeping this trend going becomes increasingly difficult.[10] Given this reality, post-Moore technologies are likely to play an important role in the future of high-performance computing. While there are certainly challenges remaining, the high-speed and ultralow energy operation of superconductor electronics makes them a promising candidate for this new era.

Continuous and extended effort, mostly at the device level, has already carried the superconducting field from the first fabricated JJ in 1970 through the development of RSFQ logic in 1985 to chips with densities on the order of several million JJs today. With the realization of self-shunted JJs in 2017,[11] chips with 10 M JJs are now in sight. The excitement around quantum computing further drives the demand for improvement in superconducting circuit fabrication. But these efforts to advance superconducting-based computation also highlight a fundamental mismatch between the computational abstractions provided by these pulse-based devices and the core "logic" we often assert prematurely as the basis of any computing system.

A change in the underlying technology can disrupt convention and necessitate a complete rethinking of computers from logic to circuits to architecture. While it remains to be seen if such a radical reconsideration (of computing) is necessary, it is clearly far more likely now than at any other time since the creation of the first integrated circuits. The approach taken in this article, which integrates ideas from languages, logic design, verification, circuits, and formal methods, and is informed by the physical phenomena underlying the operation of these novel devices, is a useful model

for exploring other approaches to computation as well.

The versatility of this method is a broader vision for post-Moore technological evaluation than is realized here. However, in line with this objective, we do specifically propose and evaluate the idea that a radically more efficient foundation for computing lies at the intersection of superconductor electronics and delay-coded computation. In particular, we claim that the natural language for expressing computations in superconducting is one that can precisely and efficiently describe the temporal relationships between SFQ pulses. To support this argument, we present a functional extension to LTL that provides the needed abstractions to formally capture the capabilities of computing in the time domain; we show how existing RSFQ elementary logic cells can be repurposed to realize the desired temporal operations; and we develop architectures that can safely leverage the extremely tight timing of superconducting circuits while avoiding the clocks and memories that shackle more incremental approaches.

THE CONNECTION THAT THIS ARTICLE ESTABLISHES BETWEEN PREDICATE TEMPORAL LOGICS AND COMPUTING PROVIDES A FRESH PERSPECTIVE ON THE DEVELOPMENT OF ABSTRACTIONS THAT CAN LEVERAGE THE UNIQUE PROPERTIES OF SUPERCONDUCTOR ELECTRONICS AND ARE SEMANTICALLY CLOSE TO THE THEORIES USED BY MODEL CHECKERS AND FORMAL ANALYSIS TOOLS.

Looking forward, we expect a growing interest in emerging technologies, less-traditional computing paradigms, and device-specific architectures. We see this work as an important step toward the transition from strictly Boolean circuits and the von Neumann computer model to "languages" and designs that better exploit the unique characteristics of post-Moore electronics. Moreover, the connection that this article establishes between predicate temporal logics and computing provides a fresh perspective on the development of abstractions that are semantically close to the theories used by model checkers and formal analysis tools.

ACKNOWLEDGMENTS

This work was supported by the National Science Foundation under Grant 1763699, Grant 1730309, Grant 1717779, and Grant 1563935. The research conducted at LBNL was supported by the Director, Office of Science of the U.S. Department of Energy under Contract DE-AC02-05CH11231. We would like to thank James E. Smith, David Donofrio, Dmitri Strukov, Alexander Wynn, Isaac Mackey, and the anonymous reviewers for their helpful comments.

REFERENCES

1. D. S. Holmes, A. M. Kadin, and M. W. Johnson, "Superconducting computing in large-scale hybrid systems," *Computer*, vol. 48, pp. 34–42, Dec. 2015, doi: 10.1109/MC.2015.375.
2. A. Madhavan, T. Sherwood, and D. Strukov, "Race logic: A hardware acceleration for dynamic programming algorithms," *SIGARCH Comput. Archit. News*, vol. 42, pp. 517–528, Jun. 2014, doi: 10.1145/2678373.2665747.
3. J. E. Smith, "Space-time algebra: A model for neocortical computation," in *Proc. 45th Annu. Int. Symp. Comput. Archit.*, 2018, Art. no. 289–300, doi: 10.1109/ISCA.2018.00033.
4. G. Tzimpragos, A. Madhavan, D. Vasudevan, D. Strukov, and T. Sherwood, "Boosted race trees for low energy classification," in *Proc. 24th Int. Conf. Archit. Support Program. Lang. Operating Syst.*, 2019, pp. 215–228, doi: 10.1145/3297858.3304036.
5. S. K. Tolpygo, "Superconductor digital electronics: Scalability and energy efficiency issues," *Low Temp. Phys.*, vol. 42, no. 5, pp. 361–379, 2016, doi: 10.1063/1.4948618.
6. M. Benedetti and A. Cimatti, "Bounded model checking for past LTL," in *Tools and Algorithms for the Construction and Analysis of Systems*, H. Garavel and J. Hatcliff, Eds. Berlin, Germany: Springer, 2003, pp. 18–33, doi: 10.1007/3-540-36577-X_3.
7. F. Laroussinie, N. Markey, and P. Schnoebelen, "Temporal logic with forgettable past," in *Proc. 17th IEEE Annu. Symp. Logic Comput. Sci.*, 2002, pp. 383–392, doi: 10.1109/LICS.2002.1029846.
8. K. K. Likharev and V. K. Semenov, "RSFQ logic/memory family: A new Josephson-junction technology for sub-terahertz-clock-frequency digital systems," *IEEE Trans. Appl. Supercond.*, vol. 1, no. 1, pp. 3–28, Mar. 1991, doi: 10.1109/77.80745.
9. G. Tzimpragos *et al.*, "A computational temporal logic for superconducting accelerators," in *Proc. 25th Int. Conf. Archit. Support Program. Lang. Operating Syst.*, 2020, pp. 435–448, doi: 10.1145/3373376.3378517.

10. V. V. Zhirnov, R. K. Cavin, J. A. Hutchby, and G. I. Bourianoff, "Limits to binary logic switch scaling—A Gedanken model," *Proc. IEEE*, vol. 91, no. 11, pp. 1934–1939, 2003, doi: 10.1109/JPROC.2003.818324.
11. S. K. Tolpygo *et al.*, "Developments toward a 250-nm, fully planarized fabrication process with ten superconducting layers and self-shunted Josephson junctions," *in Proc. 16th Int. Supercond. Electron. Conf.*, Jun. 2017, pp. 1–3, doi: 10.1109/ISEC.2017.8314189.

GEORGIOS TZIMPRAGOS is currently working toward a Ph.D. degree with the Department of Computer Science, University of California, Santa Barbara, Santa Barbara, CA, USA, and is a Research Affiliate with Lawrence Berkeley National Laboratory, Berkeley, CA, USA. His research interest includes computer architecture, currently focusing on emerging computing paradigms and technologies for energy-efficient processing—ranging from tiny embedded systems to exotic supercomputers. Tzimpragos received an M.S. degree in electrical and computer engineering from University of California, Davis, Davis, CA, USA. His alma mater is the National Technical University of Athens, Greece. He is the corresponding author of this article. Contact him at gtzimpragos@cs.ucsb.edu.

JENNIFER VOLK is currently working toward a Ph.D. degree with the Department of Electrical and Computer Engineering, University of California, Santa Barbara, Santa Barbara, CA, USA. Her research interests include the exploitation of circuit-based quirks for architectural advancements in novel technologies, as well as the exploration of tradeoffs between area, energy, robustness, and sustainability. Volk received a B.S. degree in electrical engineering from the University of California, Santa Cruz, Santa Cruz, CA, USA. She is a Member of IEEE and ACM. Contact her at jevolk@ucsb.edu.

DILIP VASUDEVAN is currently a Research Scientist with the Computer Architecture Group, Lawrence Berkeley National Laboratory, Berkeley, CA, USA. His research interests include post-Moore architectures, reconfigurable spintronic devices, superconductor electronics, and neuromorphic computing. Vasudevan received a B.E. degree in electronics and communications engineering from the University of Madras, Chennai, India, an M.S. degree in computer systems engineering from the University of Arkansas, Fayetteville, AR, USA, and a Ph.D. degree in informatics (computer engineering) from the University of Edinburgh, Edinburgh, U.K. He is a Professional Member of ACM. Contact him at dilipv@lbl.gov.

NESTAN TSISKARIDZE is currently a Research Engineer with the Department of Computer Science, Stanford University, Stanford, CA, USA. From 2011 to 2019, she was a Researcher with Princeton University, Princeton, NJ, USA, the University of Iowa, Iowa City, IA, USA, and the University of California, Santa Barbara, Santa Barbara, CA, USA. Her research work focuses on developing and adapting cutting-edge formal verification techniques to automate the design and analysis of hardware and software systems. Tsiskaridze received a Ph.D. degree in computer science from the University of Manchester, Manchester, U.K. Contact her at nestan@stanford.edu.

GEORGE MICHELOGIANNAKIS is currently a Research Scientist with Computer Architecture Group, Lawrence Berkeley National Laboratory, Berkeley, CA, USA. He has extensive experience in on- and off-chip networking and computer architecture. His current research work focuses on emerging technologies and 3-D integration in the post-Moore era, as well as optics and architecture for HPC and datacenter networks. Michelogiannakis received a Ph.D. degree from Stanford University, Stanford, CA, USA, in 2012. He is a Member of IEEE and ACM. Contact him at mihelog@lbl.gov.

ADVAIT MADHAVAN is currently a Faculty Specialist with the University of Maryland, College Park, MD, USA, affiliated with the National Institute of Standards and Technology. His research interests include novel encoding schemes, emerging technologies and architectures, and CMOS VLSI aimed toward building next generation computing systems. Madhavan received a Ph.D. degree from the University of California, Santa Barbara, Santa Barbara, CA, USA, in 2016. He is a Member of IEEE. Contact him at advait.madhavan@nist.gov.

JOHN SHALF is currently with the Department Head for Computer Science, Lawrence Berkeley National Laboratory, Berkeley, CA, USA, and was formerly the Deputy Director of Hardware Technology for the DOE Exascale Computing Project and the Leader of the Green Flash Project. He is a coauthor of more than 90 publications in the field of parallel computing software and HPC technology and corecipient of three best paper awards. Before joining Lawrence Berkeley National Laboratory in 2000, he was with the National Center for Supercomputing Applications, University of Illinois, and was a Visiting Scientist with the Max-Planck-Institute for Gravitational Physics/Albert Einstein Institute, Potsdam, Germany. Contact him at jshalf@lbl.gov.

TIMOTHY SHERWOOD is currently a Professor of Computer Science with the University of California, Santa Barbara, Santa Barbara, CA, USA, specializing in the development of processors that exploit novel technologies (e.g., superconductors and memristors), provable properties (e.g., security, privacy, and correctness), and hardware-accelerated algorithms (e.g., high-throughput scanning and new logic representations). In 2013, he cofounded Tortuga Logic to bring rich security analysis to hardware and embedded system design processes. Sherwood received a B.S. degree in computer science and engineering from the University of California, Davis, Davis, CA, USA, in 1998, and M.S. and Ph.D. degrees in computer science and engineering from the University of California San Diego, San Diego, CA, USA, in 2003. He is a recipient of the Northrop Grumman Teaching Excellence Award and the ACM SIGARCH Maurice Wilkes Award, an ACM Distinguished Scientist, and corecipient of 17 different "best paper" and "top pick" article awards. Contact him at sherwood@cs.ucsb.edu.