Technology-Aware Logic Synthesis for Superconducting Electronics

Rassul Bairamkulov, Siang-Yun Lee, Alessandro Tempia Calvino,
Dewmini Sudara Marakkalage, Mingfei Yu, and Giovanni De Micheli
Integrated Systems Laboratory, EPFL
Lausanne, Switzerland
giovanni.demicheli@epfl.ch

***Abstract*—Superconducting electronics provide us with cryogenic digital circuits that can rival established technologies in performance and energy consumption. Today, the lack of tools for the design of large-scale integrated superconducting circuits is a major obstacle to their deployment. Few research institutions and companies have contributed to making such tools available. This review focuses on methods, algorithms, and open-source design tools for logic synthesis of superconducting circuits in two major families: single-flux quantum (SFQ) circuits and adiabatic quantum flux parametron (AQFP).**

I. Introduction

Superconducting electronics (SCE) offer effective solutions to the challenges faced by modern CMOS systems, particularly stagnating clock frequencies and prohibitive power density. SCE systems can achieve 100× lower operating power and 10-100× higher clock frequencies than CMOS. Furthermore, SCE systems operate at cryogenic temperatures with millivolt-level signals, producing minimal noise. Whereas this paper deals with superconductive electronic digital circuits only, superconducting sensors and communication primitives can also be realized. These advantages have led to SCE applications in areas such as high-resolution sensors for medical and scientific measurements, fast signal processing for wireless communications, and interfacing with superconductive qubits for quantum computing.

Despite these successful applications, the scope of SCE applications remains narrow as compared to its potential. Most circuits have been designed with abundant human intervention. To reap the benefits of the technology by scaling circuits up in complexity, new *electronic design automation* (EDA) tools are required, as conventional EDA tools for CMOS are not well-suited for SCE due to fundamental differences between the two technologies.

II. Background

Modern SCE technology is based on two major superconductive effects, namely, *Josephson effect* and *magnetic flux quantization*. The fundamental building block of any SCE system is a *Josephson junction (JJ)*, consisting of superconductors separated by a barrier. A current I propagates through the barrier without any voltage drop (Josephson effect), if its magnitude is below the critical current $\texttt{I}_\texttt{c}$, determined by the JJ dimensions and fabrication technology. Increasing the current above $\texttt{I}_\texttt{c}$ disrupts the superconductive mode, producing voltage across the JJ. An example is given in Fig. 1. The voltage pulse with an area of $\Phi_0 = \frac{h}{2e} \approx 2.07\text{mV}\times\text{ps}$ is produced, commonly referred to as a *Single-Flux Quantum (SFQ)* pulse. These pulses encode information in *Rapid Single-Flux Quantum* (RSFQ) technology, described in Section II-A. Magnetic flux Φ through a superconducting loop only exists as an integer multiple of Φ_0. The current within such a loop adjusts to satisfy this condition. This effect is exploited to amplify input current in *Adiabatic Quantum-Flux Parametron* (AQFP) technology described in Section II-B.

A. SFQ

The *Rapid Single-Flux Quantum* (RSFQ) technology was developed in the late 1980s by K.Likharev and Semenov [1]. Information is carried by voltage pulses: a single pulse encodes logic `1` and the absence of a pulse encodes logic `0`. The fundamental structure of RSFQ systems is a storage loop, illustrated in Fig. 1b. The external bias current flows through junction J_0, bringing J_0 close to the critical current. This state of the loop encodes logic `0`. An SFQ pulse at the terminal `D` increases the current beyond the critical value. J_0 becomes resistive, redirects the bias current towards J_1 and becomes superconductive again. The magnetic flux through the loop becomes Φ_0, encoding logic `1`.

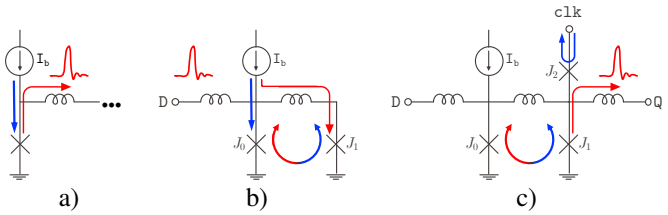


Fig. 1: Basic SCE structures. a) Josephson junction (JJ). If the bias current $\texttt{I}_\texttt{b}$ through the junction is small, no voltage drop across the junction occurs. Increasing $\texttt{I}_\texttt{b}$ over critical value $\texttt{I}_\texttt{c}$ causes JJ to *switch*, producing an SFQ pulse opposing the bias current. b) Storage loop. Initially, small bias current flows through J_0, denoting the logical `0`. An SFQ pulse at input `D` switches J_0 and redirects the bias current towards J_1, storing logical `1`. c) The D flip-flop operates similarly to the storage loop. An incoming `clk` pulse will switch J_2 if the state is `0`. If the state is `1`, J_1 switches and produces an SFQ pulse at `Q`.

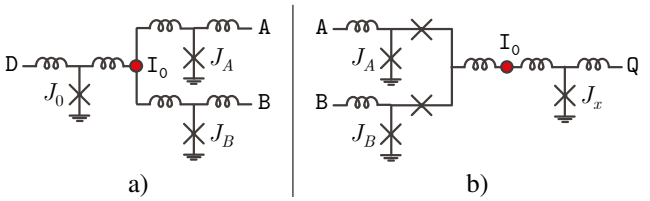


Fig. 2: a) Splitter gate duplicating an SFQ pulse into two branches. b) Merger gate (confluence buffer) directing SFQ pulses towards a common output branch. I_0 denotes the bias current injection.

To determine the state of the loop, an SFQ pulse is applied at the input `clk`, increasing the current through J_1 and J_2, as shown in Fig. 1c. If the loop state is `0`, J_1 is not biased and junction J_2 reaches its critical current first, producing no effect on the storage loop. If the loop state is `1`, J_1 switches, redirecting the bias current towards J_0. An SFQ pulse produced by J_1 travels towards the output `Q`. Due to this operating principle, this structure is called a D-flip-flop. Note that it is equivalent to a clocked buffer.

Due to the quantized nature of an SFQ pulse, producing multiple fanout requires a *splitter*, converting a single input pulse into two identical output pulses, as shown in Fig. 2a. By reversing the splitter, a *merger* structure is produced, directing the input pulses to a single output, as shown in Fig. 2b. By arranging the loops, splitters, and mergers, a variety of gates can be produced, including `NOT`, `XOR`, `AND`, and `OR` [2]. Most of these gates require clock signal to produce logic output and thus necessitate gate-level pipelining [3]. The gate-level pipelining however significantly complicates the logic synthesis, requiring the fanins of each gate to have equal logic depth. This issue, called *path balancing*, requires dummy DFFs to be inserted into the circuit. The number of DFFs and spli tters can be very large, occupying a significant portion of an SCE circuit layout. For example, in Figs. 3a-3b, a splitter and four DFFs are required to transform the CMOS circuit to its SFQ equivalent. A major task of EDA tools is to insert a minimal number of elements as described in the sequel. Note that in RSFQ the splitters are not clocked, and so balancing and splitter insertion can be performed independently. A few modifications of SFQ technology are in use today; particularly ERFSQ [4] and eSFQ [5]. These technologies differ from RSFQ [1] in their biasing structure. Therefore their logic design principles are essentially the same and are described in Section III.

B. AQFP

Adiabatic quantum-flux parametron (AQFP) is a superconducting electronics technology developed by N. Yoshikawa and coworkers at Yokohama National University [6], where the fundamental cell is a buffer and consists of two coupled loops, as illustrated in Fig. 4. Such loops are magnetically coupled to a clock line that serves as an adiabatic power supply to the buffer, as well as to the output stage. The buffer input is a current pulse and its direction corresponds to either a logic `1` or a logic `0`. Similarly the output is a current pulse. Information is stored in either loop, and transitions happen as clock transitions enable the circuit. By operating in the adiabatic mode, AQFP circuits achieve very small dynamic power consumption [7].

A majority-3 (`MAJ3`) logic gate can be constructed by combining three buffer cells with a 3-to-1 branch cell, from which other logic gates, such as the `AND` gate and the `OR` gate, can be built with constant cells (biased buffer cells). Input negation of logic gates is realized using a negative mutual inductance and is of no extra cost [8]. The commonly-used cost metric of AQFP circuits is the JJ count. A buffer costs 2 JJs, a branch cell incurs no JJ cost, and a logic gate based on majority-3 costs 6 JJs [8].

As in RSFQ, every gate in an AQFP circuit is clocked, and all input signals have to arrive in the same clock cycle. Thus the circuit needs to be balanced. Moreover, the output signal of AQFP logic gates cannot directly feed multiple fanouts. Instead, splitters are placed at the output of multi-fanout gates to amplify the output current. Unlike RSFQ, splitters in AQFP circuits are clocked and so the balancing and splitter insertion problems are intertwined, as shown in Fig 3c. Techniques for AQFP logic synthesis are described in Section IV.

III. Design of SFQ circuits

Conventional academic and commercial design flows offer little support for SCE design, as models, operations, cell layout and interconnect issues are very different from CMOS. In this section, we review some of the recent progress.

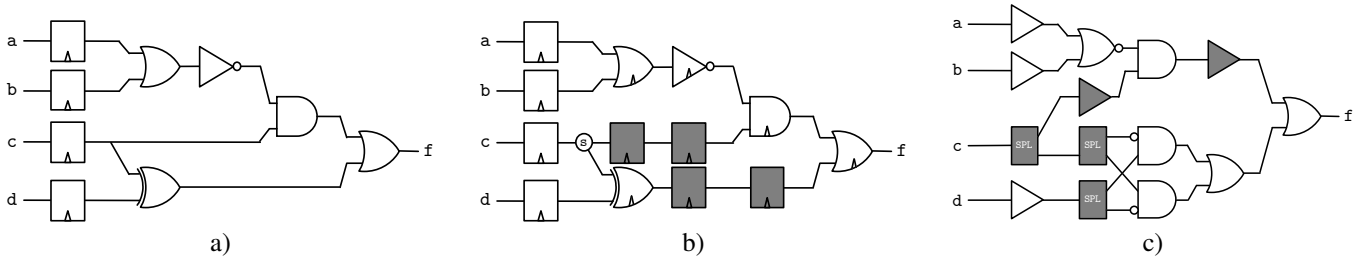


Fig. 3: a) An example of a CMOS circuit. b) Equivalent RSFQ circuit with a splitter and four path balancing DFFs. c) Equivalent AQFP circuit with three splitters and two buffers. The `XOR` gate is not available in AQFP technology and is realized with two `AND` gates. Note that all AQFP gates are clocked but the clock input is typically omitted in the literature.

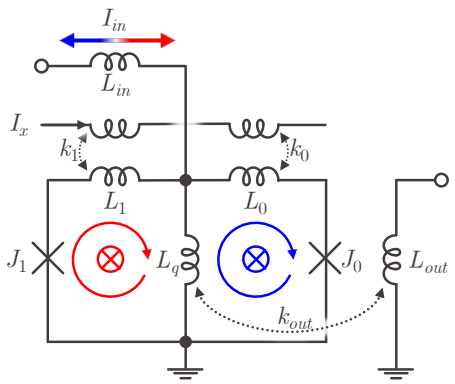


Fig. 4: AQFP buffer. The AC current I_x energizes the loop J_1-L_1-L_0-J_0. The input current I_{in} induces a flux quantum to be stored in the left loop (logic 1) or right loop (logic 0). A current is produced through L_q with the same direction as I_{in} and is inductively coupled to the output.

A. Modeling, physical design and simulation

Due to the peculiar information encoding, SCE interconnects operate differently from the CMOS interconnects. For example, a simple wire can only transfer an SFQ signal over a short distance before attenuation renders the pulse undetectable [9]. Placement and routing (P&R) tools however require interconnects of arbitrary length [10]. Therefore, early superconductive cell libraries, such as CONNECT [11] do not support automated P&R. The advent of *passive transmission lines* (PTL) [12] enabled reliable large-distance transfer of SFQ signals. To ease the design process, modern SCE cell libraries include PTL drivers and receivers in the standard cell [9], [13], [14]. In addition, standard cell dimensions are adjusted to support grid-based routing [15], [16]. These innovations enabled the development of SCE-specific routing tools, including qPlace [17] and JRouter [18], as well as the tools for clock distribution network design [19], [20].

The core circuit simulation tool for most conventional electronic systems, SPICE, offers limited support for JJs [21]. Based on the *resistively and capacitively shunted junction* (RCSJ) model [22], [23], a number of specialized circuit simulators have been developed, including JSIM [24], PSCAN [25], [26], WRspice [21], and JoSIM [27]. PSCAN and WRspice also support more accurate (but more computationally expensive) *tunneling junction model* (TJM) [28], [29] offering superior accuracy, particularly in modern fabrication technologies with high critical current density [30].

B. Logic synthesis for SFQ

Logic synthesis optimizes a gate-level description based on graphs properties and cost functions that correlate with the quality of the final circuit. On the one hand, a synthesis that is *technology-aware* has several advantages since it closely reflects the underlying technology, better correlating with quality improvements. On the other hand, synthesis should work on simple data structures composed of a few primitives to easily navigate the graph, extract properties, and apply Boolean rules.

Traditional technology-independent logic synthesis for CMOS systems is mostly based on the *and-inverter graph* (AIG), due to its simplicity, versatility, and correlation with the cost at transistor level [31]. Unlike CMOS, however, most 2-input logic cells in SFQ have similar implementation costs. In particular, `XOR` cells demonstrate a similar level of efficiency as `AND` cells. Based on this observation, a technology-independent synthesis flow for SFQ on *xor-and graphs* (XAGs) was proposed in [32] to better correlate to the technology. XAGs are more compact than AIGs since they contain one additional primitive, which is implemented using three 2-input `AND` gates in AIGs. Consequently, circuits represented by XAGs tend to be smaller and shallower. Moreover, XAGs offer more opportunities to restructure logic through additional rewriting rules and Boolean methods.

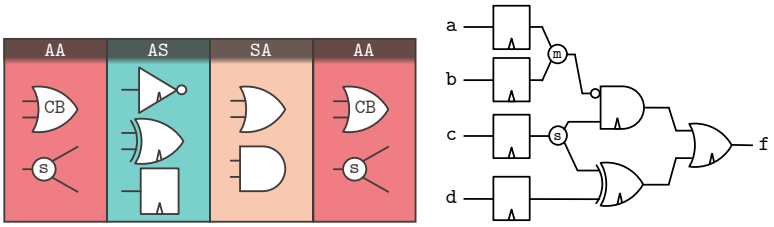
The performance of SFQ circuits highly depends on the number of logic levels, since each gate is clocked. Moreover, delay optimization helps to minimize the number of necessary balancing DFFs. Intuitively, longer critical paths require more DFF cells due to longer paths to balance. Consequently, logic synthesis for SFQ needs to implement a logic restructuring flow that primarily focuses on delay-oriented optimization. Additionally, logic sharing needs careful evaluation since it may produce high fanouts, which translate into high splitting costs.

Multiple synthesis algorithms and a flow based on the XAG representation of logic have been proposed in [32]. The flow interleaves specialized delay-oriented optimization methods with area-oriented ones. To describe the initial logic as an XAG, a versatile graph mapping algorithm [33] is employed. This method maps an initial graph representation into an arbitrary one using pre-computed structures, while optimizing for area and delay. It has been shown that this algorithm leads quickly to a good-quality XAG without involving many sophisticated optimization steps. Then, delay-oriented optimization is applied that combines three strategies:

- *Algebraic rewriting* applies the associativity, distributivity, and commutativity axioms of Boolean algebra over the primitives `AND2` and `XOR2` to minimize the depth
- *ESOP balancing* collapses sections of the circuit into optimized *exclusive-sum-of-products* (ESOPs) and decomposes them into XAGs using Huffman decomposition while minimizing the depth
- *XAG remapping* rewrites the entire network to minimize the depth using the versatile mapper [33].

Finally, we apply area recovery based on XAG-based rewriting and resubstitution.

The methods presented in this section lead to a solid logic synthesis framework capable of leveraging the logic primitives of SFQ circuits and addressing technology-specific SFQ quality metrics early on in the EDA flow.



a) b)

Fig. 5: a) Generic compound gate structure. b) Compound-gate realization of a network in Fig. 3 with no path balancing DFFs.

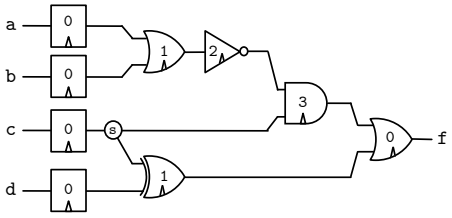


Fig. 6: Four-phase realization of network in Fig. 3 with no path balancing DFFs. The numbers indicate the phase of each clocked gate.

C. Technology mapping, path balancing and splitter insertion

After technology-independent optimization, technology mapping describes the optimized network in terms of the connection of cells from an SFQ cell library. This process involves 3 steps: mapping to cells, path balancing, and splitter insertion.

One of the earliest works in this area is PBMap [34], which roughly estimates the cost of path balancing during mapping. The state-of-the-art method [32] adopts a direct mapping strategy that starts from XAGs as the subject graph. To achieve reduced latency and area, the mapper is configured to prioritize performance. To further enhance the quality of the mapping and minimize delays, *supergates* [35], i.e., pre-computed connection of cells, or *compound gates* [36] have been used in literature [3], [32], [34].

Afterwards, SFQ circuits are path-balanced by inserting DFFs. In [3], [32], DFFs are inserted by placing them ASAP, i.e., ensuring that the arrival times at each cell's input are synchronized (balancing constraint), while maintaining the same delay (ASAP balancing policy). To optimize the area, DFFs among nodes connected to the same input node at the same clock level are shared. After initial insertion, post-optimization of balancing DFFs has been proposed in [37] using minimum-area retiming [38], leading to a minimal number of balancing DFFs. Finally, any instance of multiple fanout is replaced with a balanced tree of splitters to address the fanout constraint.

After technology mapping, technology-dependent logic synthesis algorithms can further optimize the logic while accounting for balancing and splitting costs. For instance, in [39] the authors extended classical synthesis algorithms to consider balancing costs, thus applying logic transformations that reduce the number of balancing DFFs. Additionally, a large part of the methods in this section are adopted in the industrial flow for RSFQ circuits proposed in [40].

D. Gate compounding

The design of larger gates, as compositions of smaller ones, is a way to counter the inherent limitations of path balancing and achieve lower latency. Whereas most SFQ logic cells are synchronous, some others operate without synchronization mechanisms. Examples are the splitter and the merger cells. The recent introduction of the gate compounding technique [3], [36], leverages synchronization mechanisms of SFQ cells to realize more complex functionalities within fewer clock cycles. The primary innovation of this method is the classification of gates into three types – *AA*, *AS*, and *SA* – depending on whether inputs and outputs are synchronized. *Asynchronous-asynchronous* (AA) gates, such as the merger and splitter, handle data immediately upon arrival and release the outputs shortly afterwards. *Asynchronous-synchronous* (AS) gates, such as the DFF, `NOT`, and `XOR2`, process inputs immediately but release the computation after the clock signal. *Synchronous-asynchronous* (SA) gates, such as asynchronous `AND2` and `OR2` cells based on the merger, require inputs to arrive simultaneously for immediate processing and output.

Gate compounding combines these three gate types to produce complex functionalities while minimizing the number of synchronization mechanisms required, with benefits in performance and sometimes also in area. The compound gate is composed of four stages of logic as shown in Figure 5. According to this structure, an AS gate synchronizes the inputs arriving at the SA gate, as required by the latter. Traditional SFQ systems ensure this requirement by adding input DFFs to SA cells. In gate compounding, AS cells replace DFFs, expanding the range of logic functions possible in a single clock cycle. This enables the creation of complex gates such as `XNOR2` and `NIMPLY2`. Applications of compound gates have been shown to produce up to 53% faster circuits compared to the conventional SFQ ones [3].

E. Multiphase clocking

Multi-phase clocking (MPC) is a synchronization technique that uses multiple clock signals with an identical clock period. This approach was first effectively demonstrated for path balancing in SFQ systems by Li *et al.* [41]. In an n-phase system, periodic signals t_i, with $i \in [0, n-1]$, operate at the same frequency but with different phases, typically uniformly spaced as $\frac{2\pi i}{n}$. Each clocked element g is assigned to a phase $\varphi(g)$. Utilizing this strategy, a new logic synthesis flow supporting multi-phase clocking was introduced in [42]. This framework extends MPC to compound gates and supports retiming and DFF insertion. This novel approach complements the state-of-the-art SFQ logic synthesis methods achieving superior area.

A major advantage provided by multi-phase clocking is the finer control of the signal arrival time. This capability enables the use of highly-expressive SFQ gates that rely on the precise order of input arrival. For instance, a full-adder, frequently occurring in practical logic networks [43], can be

realized using the SFQ T1-flip-flop, requiring 60% smaller area compared to its regular realization. The T1-flip-flop, however, requires the inputs to never arrive simultaneously, greatly complicating its use. This condition can be efficiently satisfied using multi-phase clocking, as demonstrated in [44], where the area of arithmetic circuits is reduced by up to 25%.

IV. Synthesis of AQFP circuits

AQFP design requires both path balancing and splitter insertion. As mentioned before, AQFP splitters are clocked, making the two design problems entangled. Moreover, the basic computing element in AQFP is the majority gate, and complementation is achieved with zero cost. This makes the *majority-inverter graph* (MIG) [45] a perfect logic representation for AQFP synthesis.

A. MAJ-based logic synthesis

The MIG is a homogeneous logic network where each node represents a `MAJ3` gate and optional inverters can be placed on edges. MIGs have been used for the optimization of both CMOS-based and emerging technologies, either for their advantage in depth optimization or because the underlying technology is majority-based, like AQFP. An MIG can be translated directly from an AIG by replacing each `AND2` gate with a `MAJ3` gate with a constant-0 input. Alternatively, in [33], a versatile graph mapping algorithm is proposed, which is capable of mapping from and to any network types, including MIGs, while optimizing for depth and/or size in the process and leveraging Boolean don't cares [46].

Similar to XAG discussed in the previous section, many logic optimization algorithms tailored for MIGs have been proposed. *Algebraic rewriting* applies special Boolean algebraic rules to reduce MIG depth [45]. MIG-based *resubstitution* resynthesizes a small part of the network using majority gates to reduce MIG size [47]. *Boolean rewriting* [46] replaces small sections of the network with size-optimum implementations. In [48], these algorithms are first combined to form a `MAJ`-based logic synthesis flow for AQFP synthesis. By using MIGs, there is no need for an extra step of technology mapping for AQFP circuits. Instead, only buffer and splitter insertion is needed before physical design. Furthermore, in [49], optimal databases consisting of `MAJ3`, `MAJ5`, buffers, and splitters are used to optimize AQFP circuits considering the technology constraints at the same time.

B. Fanout-bounded synthesis

Given a logic network, as well as fanout bounds and area costs of different gate/buffer types, the goal of *Fanout Bounded Synthesis* (FBS) is to synthesize the network through duplications of gates and insertion of buffers such that all fanout constraints are met. Considering AQFP splitters as buffers with bounded fanout capacity, in [50], the buffer and splitter (B/S) insertion problem is considered as an FBS problem in the unit delay model with the additional requirement of path-balancing.

For a given target delay, considering 1) the number of duplicates of each gate in each level, and 2) the number of buffers associated with each gate at each level as integer variables, an *integer linear program* (ILP) formulation is presented in [50] to obtain the optimum area solutions. The results obtained from the exact method show that it is possible to simultaneously achieve better delays and area by allowing duplication of gates. Additionally, [50] also introduced a scalable heuristic FBS algorithm for AQFP synthesis using a top-down approach, where gate and buffer counts of each level are decided, starting from the primary output side and continuing towards primary inputs.

C. Buffer and splitter insertion

In [51], it is shown that the AQFP B/S insertion problem is a scheduling problem. A *schedule* is a level assignment to each node in the network. With the proposal of an irredundant B/S insertion algorithm that is size-optimal subject to a given schedule, the B/S insertion problem is equivalent to finding a legal schedule and optimizing the schedule. In [52], a depth-optimal, linear-time scheduling algorithm is proposed. Using a depth-optimal legal schedule as the starting point, two orthogonal heuristic optimization algorithms can be applied to further optimize the B/S count: *chunked movement* tries to move groups of tightly connected nodes together [51], and *retiming* [52] leverages an existing register retiming algorithm to reduce the number of buffers. These algorithms form an AQFP legalization and optimization flow, consisting of first obtaining a depth-optimal schedule and then iteratively optimizing it.

D. Sequential designs and constraint relaxation

In [53], an overview of AQFP sequential circuit design is presented, and the authors discuss how architectural clocking and register design affect the technology constraints. The commonly-adopted constraint formulation is sometimes too conservative and relaxations to the constraints are proposed. To be more specific, the *path-balancing* constraint, which requires all inputs n_i to a gate n to be placed at exactly one level below ($l(n_i) = l(n) - 1$), can be relaxed to the *phase alignment* constraint, which only asks for a similar relation for the clock phases assigned to each gate. In other words, the modulus of p_{clk}, the number of phases per (gate-level) clock cycle decided by the clocking scheme (usually 3 or 4), is taken on both sides of the constraint equation ($l(n_i) \bmod p_{\text{clk}} = (l(n)-1) \bmod p_{\text{clk}}$). This means long chains of buffers can be removed. Experiments show that adopting the relaxed constraints reduces 73% of buffers on average, and up to 90% in some particularly imbalanced benchmarks. Trade-offs are possible when the constraints are relaxed.

V. Conclusions

This paper has described SCE technologies for digital design, with a major emphasis on RSFQ and AQFP, as well as highlighting some recent methods for logic synthesis of these circuits in a brief and comparative way. The methods described are fully detailed in the literature. In particular, the software packages produced by the Authors are available at

[54] as open source, and detailed results are presented in [3], [32], [33], [36], [42]–[53], [55], [56] and not reported here for lack of space. Overall, these tools do not yet provide a comprehensive flow for digital SCE design, but show already how far can EDA support the solution of key point problems for SCE design. We consider this area as quite timely and challenging and as an enabler for the realization of fast data processing systems.

VI. Acknowledgments

This work was supported in part by the SNF grant "Supercool: Design methods and tools for superconducting electronics", 200021_1920981 and by Synopsys Inc.

References

[1] K. K. Likharev and V. K. Semenov, "RSFQ Logic/Memory Family: A New Josephson-Junction Technology for Sub-Terahertz-Clock-Frequency Digital Systems," *IEEE TASC*, 1991.
[2] P. Bunyk, K. Likharev, and D. Zinoviev, "RSFQ Technology: Physics and Devices," *Int. J. High Speed Electron. Syst.*, 2001.
[3] R. Bairamkulov, A. Tempia Calvino, and G. De Micheli, "Synthesis of SFQ Circuits with Compound Gates," *Proc. VLSI-SoC*, 2023.
[4] D. Kirichenko, A. Kirichenko, and S. Sarwana, "No Static Power Dissipation Biasing of RSFQ Circuits," *IEEE TASC*, 2010.
[5] O. A. Mukhanov, "Energy-Efficient Single Flux Quantum Technology," *IEEE TASC*, 2011.
[6] N. Yoshikawa, D. Ozawa, and Y. Yamanashi, "Ultra-Low-Power Superconducting Logic Devices Using Adiabatic Quantum Flux Parametron," *Proc. SSDM*, 2011.
[7] N. Takeuchi *et al.*, "An Adiabatic Quantum Flux Parametron as an Ultra-Low-Power Logic Device," *Supercond. Sci. Technol.*, 2013.
[8] N. Takeuchi, Y. Yamanashi, and N. Yoshikawa, "Adiabatic Quantum-Flux-Parametron Cell Library Adopting Minimalist Design," *J. Appl. Phys.*, 2015.
[9] Y. Kameda, S. Yorozu, and Y. Hashimoto, "A New Design Methodology for Single-Flux-Quantum (SFQ) Logic Circuits Using Passive-Transmission-Line (PTL) Wiring," *IEEE TASC*, 2007.
[10] C. J. Fourie *et al.*, "Design and Characterization of Track Routing Architecture for RSFQ and AQFP Circuits in a Multilayer Process," *IEEE TASC*, 2020.
[11] S. Yorozu *et al.*, "A Single Flux Quantum Standard Logic Cell Library," *Physica C Supercond.*, 2002.
[12] T. Jabbari *et al.*, "Interconnect Routing for Large-Scale RSFQ Circuits," *IEEE TASC*, 2019.
[13] S. Rylov *et al.*, "Superconducting VLSI Logic Cell Library Using DC-Powered Clockless Dynamic SFQ Gates and ASIC-Style Layout Template," *IEEE TASC*, 2023.
[14] H. Akaike *et al.*, "Design of Single Flux Quantum Cells for a 10-Nb-Layer Process," *Physica C Supercond.*, 2009.
[15] S. Nath *et al.*, "An Automated Place and Route Methodology for Asynchronous SFQ Circuit Design," *Proc. ISEC*, 2019.
[16] S. K. Tolpygo *et al.*, "Properties of Unshunted and Resistively Shunted Nb/AlOx-Al/Nb Josephson Junctions with Critical Current Densities from 0.1 to 1 mA/μm2," *IEEE TASC*, 2017.
[17] M. Pedram, "Superconductive Single Flux Quantum Logic Devices and Circuits: Status, Challenges, and Opportunities," *Proc. IEDM*, 2020.
[18] X. Chen *et al.*, "JRouter: A Multi-Terminal Hierarchical Length-Matching Router under Planar Manhattan Routing Model for RSFQ Circuits," *Proc. GLSVLSI*, 2023.
[19] R. Bairamkulov, T. Jabbari, and E. G. Friedman, "QuCTS — Single-Flux Quantum Clock Tree Synthesis," *IEEE TCAD*, 2022.
[20] C.-C. Wang and W.-K. Mak, "A Novel Clock Tree Aware Placement Methodology for Single Flux Quantum (SFQ) Logic Circuits," *Proc. ICCAD*, 2021.
[21] S Whiteley, "WRspice Circuit Simulator," 2017.
[22] W. C. Stewart, "Current-Voltage Characteristics of Josephson Junctions," *Appl. Phys. Lett.*, 1968.
[23] D. E. McCumber, "Effect of AC Impedance on DC Voltage-Current Characteristics of Superconductor Weak-Link Junctions," *J. Appl. Phys.*, 1968.
[24] E. S. Fang and T. Van Duzer, "A Josephson Integrated Circuit Simulator (JSIM) for Superconductive Electronics Application," *Proc. ISEC*, 1989.
[25] S. Polonsky, V. Semenov, and P. Shevchenko, "PSCAN: Personal Superconductor Circuit Analyser," *Supercond. Sci. Technol.*, 1991.
[26] S. Polonsky *et al.*, "PSCAN'96: New Software for Simulation and Optimization of Complex RSFQ Circuits," *IEEE TASC*, 1997.
[27] J. A. Delport *et al.*, "JoSIM—Superconductor SPICE Simulator," *IEEE TASC*, 2019.
[28] A. Odintsov, V. Semenov, and A. Zorin, "Specific Problems of Numerical Analysis of the Josephson Junction Circuits," *IEEE Trans. Magn.*, 1987.
[29] D. R. Gulevich, "MiTMoJCo: Microscopic Tunneling Model for Josephson Contacts," *Comput. Phys. Commun.*, 2020.
[30] S. Whiteley *et al.*, "Observations in Use of a Tunnel Junction Model in Simulations of Josephson Digital Circuits," *IEEE TASC*, 2022.
[31] A. Tempia Calvino *et al.*, "Improving Standard-Cell Design Flow Using Factored Form Optimization," *Proc. DAC*, 2023.
[32] A. Tempia Calvino and G. De Micheli, "Algebraic and Boolean Methods for SFQ Superconducting Circuits," *Proc. ASP-DAC*, 2024.
[33] A. Tempia Calvino *et al.*, "A Versatile Mapping Approach for Technology Mapping and Graph Optimization," *Proc. ASP-DAC*, 2022.
[34] G. Pasandi and M. Pedram, "PBMap: A Path Balancing Technology Mapping Algorithm for Single Flux Quantum Logic Circuits," *IEEE TASC*, 2019.
[35] S. Chatterjee *et al.*, "Reducing Structural Bias in Technology Mapping," *Proc. ICCAD*, 2005.
[36] R. Bairamkulov and G. De Micheli, "Compound Logic Gates for Pipeline Depth Minimization in Single Flux Quantum Integrated Systems," *Proc. GLSVLSI*, 2023.
[37] N. K. Katam and M. Pedram, "Logic Optimization, Complex Cell Design, and Retiming of Single Flux Quantum Circuits," *IEEE TASC*, 2018.
[38] C. E. Leiserson and J. B. Saxe, "Retiming Synchronous Circuitry," *Algorithmica*, 1991.
[39] G. Pasandi and M. Pedram, "Balanced Factorization and Rewriting Algorithms for Synthesizing Single Flux Quantum Logic Circuits," *Proc. GLSVLSI*, 2019.
[40] E. Mlinar *et al.*, "An RTL-to-GDSII Flow for Single Flux Quantum Circuits Based on an Industrial EDA Toolchain," *IEEE TASC*, 2023.
[41] X. Li *et al.*, "Multi-Phase Clocking for Multi-Threaded Gate-Level-Pipelined Superconductive Logic," *Proc. ISVLSI*, 2022.
[42] R. Bairamkulov and G. D. Micheli, "Towards Multiphase Clocking in Single-Flux Quantum Systems," *Proc. ASP-DAC*, 2024.
[43] A. Tempia Calvino and G. De Micheli, "Technology Mapping Using Multi-Output Library Cells," *Proc. ICCAD*, 2023.
[44] R. Bairamkulov, M. Yu, and G. De Micheli, "Efficient Full Adders in SFQ Arithmetic Circuits," *Proc. DATE*, 2024.
[45] L. G. Amarù, P.-E. Gaillardon, and G. De Micheli, "Majority-Inverter Graph: A New Paradigm for Logic Optimization," *IEEE TCAD*, 2016.
[46] A. Tempia Calvino and G. De Micheli, "Scalable Logic Rewriting Using Don't Cares," *Proc. DATE*, 2024.
[47] S.-Y. Lee and G. De Micheli, "Heuristic Logic Resynthesis Algorithms at the Core of Peephole Optimization," *IEEE TCAD*, 2023.
[48] E. Testa *et al.*, "Algebraic and Boolean Optimization Methods for AQFP Superconducting Circuits," *Proc. ASP-DAC*, 2021.
[49] D. S. Marakkalage, H. Riener, and G. De Micheli, "Optimizing Adiabatic Quantum-Flux-Parametron (AQFP) Circuits Using an Exact Database," *Proc. NANOARCH*, 2021.
[50] D. S. Marakkalage and G. De Micheli, "Fanout-Bounded Logic Synthesis for Emerging Technologies," *IEEE TCAD*, 2023.
[51] S.-Y. Lee, H. Riener, and G. De Micheli, "Beyond Local Optimality of Buffer and Splitter Insertion for AQFP Circuits," *Proc. DATE*, 2022.
[52] A. Tempia Calvino and G. De Micheli, "Depth-Optimal Buffer and Splitter Insertion and Optimization in AQFP Circuits," *Proc. ASP-DAC*, 2023.
[53] S.-Y. Lee, C. L. Ayala, and G. De Micheli, "Impact of Sequential Design on The Cost of Adiabatic Quantum-Flux Parametron Circuits," *IEEE TASC*, 2023.
[54] M. Soeken *et al.*, "The EPFL Logic Synthesis Libraries," *https://github.com/lsils/mockturtle arXiv:1805.05121v3*, 2018.
[55] G. De Micheli, "Logic Synthesis for Emerging Technologies," *Proc. ASICON*, 2023.
[56] G. Meuli *et al.*, "Majority-Based Design Flow for AQFP Superconducting Family," *Proc. DATE*, 2022.