Impact of Sequential Design on the Cost of Adiabatic Quantum-Flux Parametron Circuits

Siang-Yun Lee, Christopher Lawrence Ayala, *Senior Member, IEEE*, and Giovanni De Micheli, *Life Fellow, IEEE*

***Abstract*—Adiabatic quantum-flux-parametron (AQFP) logic is a superconductor logic family whose energy efficiency approaches theoretical limits. Because AQFP logic gates depend on a polyphase excitation current to perform their computation, gate fanins must arrive at the appropriate excitation phase. Such a technology constraint has conventionally been treated by inserting buffers to balance shorter paths. However, path-balancing buffers account for a large portion of the circuit area, limiting the scalability of AQFP circuits. In this article, we examine the necessity of AQFP design constraints and propose a more relaxed set of constraints, which still guarantees the correct operation of AQFP sequential circuits. In particular, we propose to consider phase alignment instead of path balancing. Experimental results show that adopting the relaxed constraints reduces 73% of buffers on average, and up to 90% in some particularly-imbalanced benchmarks.**

***Index Terms*—Adiabatic quantum-flux-parametron (AQFP), buffer insertion, sequential logic circuit, superconductor electronics.**

I. Introduction

ADIABATIC *quantum-flux-parametron* (AQFP) logic is an emerging superconductor digital logic technology receiving increased interest. By operating in the adiabatic mode, its energy efficiency is reaching theoretical limits [1]. Being a promising and attractive alternative to CMOS-based digital families for high-performance computing, design automation algorithms specialized for AQFP circuits are also being rapidly developed in recent years. As switching energy dissipation in AQFP is related to the number of *Josephson junctions* (JJs), reducing the JJ count of AQFP circuits has been the primary optimization goal along with reducing circuit latency. This, in turn, also helps to reduce the overall circuit area as AQFP primitives have a large footprint due to their output transformer. Surprisingly, research has found that a large portion of JJs in AQFP benchmark circuits is dedicated to buffering cells to fulfill technology constraints.

Manuscript received 19 May 2023; revised 12 August 2023; accepted 19 August 2023. Date of publication 24 August 2023; date of current version 7 September 2023. This work was supported by the SNF grant "Supercool: Design methods and tools for superconducting electronics" under Grant 200021_1920981. *(Corresponding author: Siang-Yun Lee.)*

Siang-Yun Lee and Giovanni De Micheli are with the Integrated Systems Laboratory, Swiss Federal Institute of Technology Lausanne, 1015 Lausanne, Switzerland (e-mail: siang-yun.lee@epfl.ch).

Christopher Lawrence Ayala is with the Institute of Advanced Sciences, Yokohama National University, Yokohama 240-8501, Japan.

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TASC.2023.3308408.

Digital Object Identifier 10.1109/TASC.2023.3308408

The AQFP technology imposes some special constraints uncommon to classical CMOS technologies. First, because every gate in an AQFP circuit is clocked, all input signals for any given logic gate have to arrive before the clock. To ensure this, shorter data paths need to be delayed with clocked buffers. Moreover, the output signal of AQFP logic gates cannot be directly branched to feed into multiple fanouts. Instead, splitters are placed at the output of multifanout gates to amplify the output current, which are also clocked. The AQFP splitter cell is based on the buffer cell and has the same cost of 2 JJs [2]. Thus, buffer insertion and splitter insertion are often considered together as an optimization problem [3], [4], [5], [6]. In previous works, the buffer and splitter insertion problem has been formulated as follows: All paths should be balanced to the same length (*path balancing*), and all gates, including primary inputs (PIs), with multiple fanouts must be branched (*fanout branching*).

While the path-balancing and fanout-branching constraints are absolutely required for the correct operation of an AQFP combinational[1] circuit without memory devices, in the context of a sequential computing model where combinational inputs and outputs are connected to registers, these constraints may be too conservative. According to the architectural clocking scheme currently used in AQFP sequential circuits, registers generally hold their values throughout the architectural clock cycle and their outputs can be taken by the next-stage combinational circuit multiple times. In other words, the same computation is repeated in waves in an AQFP combinational circuit. With a careful analysis, we argue that it is not always necessary to balance all paths to equal length. Instead, aligning the gate-level clock phases is enough.

In this work, we provide an overview of AQFP sequential circuit design and discuss how architectural clocking and register design affect the technology constraints. We argue that the commonly adopted constraint formulation is sometimes too conservative and propose relaxations to the constraints. Consequently, we also investigate how the relaxation of constraints affect the number of buffers needed, and discuss possible tradeoffs when the constraints are relaxed. Our following contributions are three-fold.

1) We re-examine the formulation of AQFP technology constraints and propose possible relaxations on these constraints: phase alignment instead of path balancing, and

[1]Although AQFP gates are clocked, we use the terms *combinational* and *sequential* here in a similar sense as in CMOS digital circuits, considering the (architectural) clock connected to registers.

the flexibilities on combinational inputs' splitting capacity and phases. We also discuss a potential issue with clock skew and the tradeoff of adopting relaxed constraints.

2) We implement and open source the first buffer-insertion framework, which considers detailed and realistic constraints and possible relaxations. The framework is parameterized for easy customization of constraint specification.
3) We investigate the influence of technology constraints on JJ count. Using the relaxed constraints, an average of 73% of buffers can be saved. This observation can help scale up AQFP circuits, which were bottle-necked by too many buffers before.

II. Adiabatic Quantum-Flux-Parametron

In AQFP digital circuits, logic "0" and "1" are represented by different current directions of the same magnitude, instead of low and high voltages as in CMOS. The basic circuit components in AQFP logic are the buffer cell and the branch cell. A majority-3 (MAJ3) logic gate can be constructed by combining three buffer cells with a reverted branch cell (i.e., a 3-to-1 merger). Other preliminary logic gates, such as the AND2 and OR2 gates, can be built from the MAJ3 gate with a constant input (constant 0 for AND2 and constant 1 for OR2) made of an asymmetric buffer cell. Input negation of logic gates is realized using a negative mutual inductance and is of no extra cost [7]. The commonly used cost metric for AQFP circuits is the JJ count. A buffer costs 2 JJs, a branch cell is of zero JJ-cost, and a logic gate based on MAJ3 costs 6 JJs [7].

A. Gate-Level Clocking Schemes

Logic gates, buffers, and splitters in AQFP are periodically activated and reset by alternating excitation current [1]. A gate takes its inputs, computes, and provides its output with the presence of the excitation current. In the absence of the excitation current, an AQFP gate produces no output current (i.e., neither logic "0" nor logic "1"). Thus, two cascaded gates must be fed with consecutive clocking phases, where the capturing gate is activated later than, but overlapping with, the activation of the launching gate, such that the information can be propagated along the circuit. We call the capturing gate a *fanout* of the launching gate, and the launching gate a *fanin* of the capturing gate.

Various clocking schemes have been proposed. *3-phase clocking* was used in earlier works [1], [7], [8], where three excitation currents with a phase shift of 120° to each other are fed into different levels of gates. A few years later, *4-phase clocking* was proposed [9] and has remained the most commonly used clocking scheme until now. In 4-phase clocking, the phase shift decreases to 90°, the number of alternating current sources decreases to 2, and the number of clocking phases in each clock cycle increases to 4, allowing for slightly lower latencies by enabling a logical depth of 4 gates instead of 3 per cycle. In both 3- and 4-phase clocking, logic gates in each level are assigned to one of the three or four phases and *phase synchronization* must be ensured: Any fanin of a gate g must be at the previous phase of g.

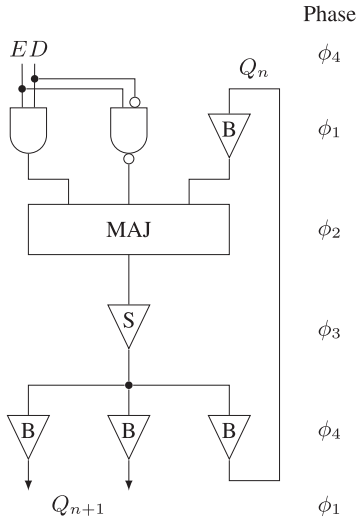

Fig. 1. Circuit schematic of an AQFP D-latch.

TABLE I
Truth Table of D-Latch and NDRO

E	D	Q_{n+1}	Action
0	0	Q_n	Hold
0	1	Q_n	Hold
1	0	0	Write 0
1	1	1	Write 1

Another clocking scheme is *delay-line clocking* [10], where a single alternating excitation current is used and transmission lines are inserted between levels to delay the clock. Delay-line clocking not only allows for even lower latency but also enables the *phase-skipping operation* [11], [12], reducing the number of path-balancing buffers.

In this article, we use p_{clk} to denote the number of phases in a (gate-level) clock cycle. Typically, $p_{\text{clk}} = 3$ or 4.

B. Memory Devices and Architectural Clocking

To implement sequential circuits using a similar finite-state-machine model as CMOS digital systems, AQFP memory devices are needed. At least two possible designs have been proposed in the literature: 1) D-latch and 2) QFPL-based NDRO.

A simplified AQFP *feedback delay latch* (D-latch) is depicted in Fig. 1, where the 4-phase clocking scheme is used. A D-latch takes an *Enable* signal E and a *Data* signal D as inputs. Its operation is illustrated by the truth table shown in Table I. When $E = 0$, the majority gate has input values $(0, 1, Q_n)$, thus keeping the same internal state $Q_{n+1} = Q_n$; when $E = 1$, the majority gate has input values (D, D, Q_n); thus, the internal state is overwritten by the new data D [13].

A *quantum-flux-parametron latch* (QFPL) is a special AQFP gate that can hold its state when the excitation current is low. The internal state of a QFPL is updated only when its two inputs A and B present the same value; otherwise, it keeps the previous state. Combining a QFPL and some logic gates, a *nondestructive-read-out* (NDRO) can be made, as shown in Fig. 2. An NDRO also takes an *Enable* signal E and a *Data*

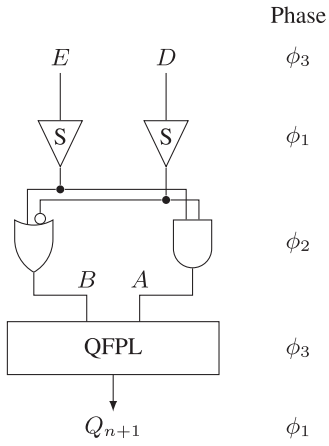

Fig. 2. Circuit schematic of a QFPL-based NDRO.

signal D as inputs and has the same truth table as in Table I. When $E = 0$, we have $A = 0$ and $B = 1$; thus, the QFPL holds its previous state; when $E = 1$, then $A = B = D$ and the new data is written into the QFPL [14].

For a D-latch, an update to the state, caused by a new value at the input D enabled with $E = 1$, is propagated through the circuit and changes the output Q_{n+1} 4 phases later. In contrast, an update to the state of an NDRO is available at the output 3 phases later.

In a classical sequential circuit model, the data D inputs of registers come from the outputs of the previous-stage combinational circuit, the outputs Q of registers are connected to the inputs of the next-stage combinational circuit, and the enable E input of registers comes from an architectural clock (in contrast to the gate-level clock discussed in Section II-A). In the CMOS paradigm, the enabling signal of registers is the rising edge or falling edge of a periodic clock signal. In contrast, in AQFP, the enabling signal E is kept at 0 most of the time and becomes 1 once every k gate-level clock cycle, where the value k depends on the length of the critical combinational path. In this article, we denote the number of phases in an architectural clock cycle by $p_{\text{arch}} = k \cdot p_{\text{clk}}$.

III. AQFP Design Constraints

A. Terminology

This work mainly focuses on the network-level abstraction of an AQFP circuit. A *logic network* is a directed acyclic graph (DAG) defined by a pair (V, E) of a set V of nodes and a set E of directed edges. The node set $V = I \cup O \cup G$ is disjointly composed of a set I of PIs, a set O of *primary outputs* (POs), and a set G of *(logic) gates* chosen from an AQFP technology library (e.g., composed of AND2, OR2, and MAJ3 with optional input negation). Each PI has in-degree 0 and unbounded out-degree, whereas each PO has in-degree 1 and out-degree 0. The out-degree of each gate is unbounded and the in-degree is a fixed number depending on the type of the gate. This abstraction models the combinational part of digital circuits. In practice, PIs of a logic network are often provided by the register outputs of the previous sequential stage and POs are connected to the register inputs of the next stage.

After synthesizing and optimizing a logic network using some logic synthesis algorithms, buffers and splitters need to be inserted to fulfill technology constraints, producing a *buffered network*. A buffered network $N' = (V', E')$ is a DAG extended from a logic network $N = (V, E)$. In particular, the node set $V' = V \cup B$ is supplemented with a set B of *buffers*. A buffer node models an AQFP buffer cell (when having out-degree 1) or an AQFP splitter cell (when having out-degree larger than 1) and always has in-degree 1.

When not specified, a *network* may be a logic network or a buffered network. In addition, a *schedule* can be defined for a network. A schedule of a network $N = (V, E)$ is a function $l : V \to \mathbb{Z}_{\geq 0}$ that assigns a nonnegative integer $l(n)$ to each node $n \in V$, called the *level* of n. The depth of a network N with a PO set O and associated with a schedule l is defined as $d(N) = \max_{o \in O} l(o)$.

We define the following properties for a buffered network $N' = (V' = I \cup O \cup G \cup B, E')$ and its associated schedule l related to the AQFP technology constraints, subject to some parameters: The *splitting capacities* $s_i = 1, s_g = 1$, and $s_b \geq 1$ of PIs, gates, and buffers, respectively, are the maximum number of fanouts each type of nodes may have. The clocking scheme p_{clk} is the number of phases in a gate-level clock cycle. Φ_{ro} is the set of phase differences a register may produce its output relative to its input phase.

1) Path balancing: N' is *path-balanced* if

$$\forall n_i, n_o \in V' : (n_i, n_o) \in E' \Rightarrow l(n_i) = l(n_o) - 1 \quad (1)$$

$$\forall i \in I : l(i) = 0, \text{ and} \quad (2)$$

$$\forall o \in O : l(o) = d(N'). \quad (3)$$

2) Phase alignment: N' is *phase-aligned* if

$$\forall n_i, n_o \in V' : (n_i, n_o) \in E' \Rightarrow l(n_i) \bmod p_{\text{clk}} = (l(n_o) - 1) \bmod p_{\text{clk}} \wedge l(n_o) > l(n_i) \quad (4)$$

$$\forall i \in I : \exists \phi_i \in \Phi_{\text{ro}} \; l(i) \bmod p_{\text{clk}} = \phi_i \bmod p_{\text{clk}} \wedge l(i) \geq \phi_i, \text{ and} \quad (5)$$

$$\forall o \in O : l(o) \bmod p_{\text{clk}} = 0. \quad (6)$$

3) Fanout branching: N' is *properly-branched* if every PI has an out-degree no larger than s_i, every gate has an out-degree no larger than s_g, and every buffer has an out-degree no larger than s_b.

In the following, we discuss which subset of these properties shall be required as AQFP technology constraints and the values of the parameters involved.

B. Phase Alignment Instead of Path Balancing

Existing works on AQFP sequential architectural design [14], [15], logic synthesis [2], [16], [17], [18], [19], [20], and buffer insertion-optimization [3], [4], [5], [6] conventionally adopt a more conservative set of constraints: path balancing and fanout

branching. Notice that fulfilling path balancing, with an additional constraint that $d(N') \bmod p_{\text{clk}} = 0$, implies fulfilling phase alignment with $\Phi_{\text{ro}} = \{0\}$. While this ensures correct and robust operation of the AQFP circuit even with fast clock frequencies, enforcing these constraints often leads to bulky circuits with more than half—sometimes up to 90%—area taken by buffers. In this section, we argue that in the context of synthesizing combinational logic between register stages, assuring phase alignment, instead of the stronger path-balancing constraint, is enough.

In [14], Saito et al. proposed that when registers in several sequential stages share the same enable signal, which arrives once per p_{arch} phases matching the depth of the deepest stage, shallower stages do not need to be balanced to the same length as the deepest stage. The main reason is that memory devices output their value every p_{clk} phase and do not change their internal state for the entire architectural clock cycle until the enable signal arrives. Thus, although shallower stages finish their computation earlier than when the registers are enabled to take the next values again, the same computation is repeated every (gate-level) clock cycle, and the same computational results are produced repeatedly until the registers are enabled again to accept them.

With a similar reasoning, we extend the argument further and propose that the path-balancing constraint can be relaxed to phase alignment, formally stated as follows.

Proposition 1: In an AQFP sequential circuit, let d be the longest path length between any two register stages, ϕ_{ro} be the phase difference between the register output Q_{n+1} and inputs D, E. Suppose that the register enables signal E is 1 for one phase in every $p_{\text{arch}} = k \cdot p_{\text{clk}}$ phase, where $p_{\text{arch}} \geq \phi_{\text{ro}} + d$, then fulfilling the phase-alignment constraint (4)–(6), in addition to fanout branching, is enough to ensure correct sequential operation of the circuit.

Proof: Without loss of generality, consider the computation propagated from one register stage I, through a combinational circuit N, to the next register stage O, in one architectural clock cycle. Suppose that $E = 1$ at time $t = 0$ and at time $t = p_{\text{arch}}$ (the unit of time is the number of phases) and that $E = 0$ all the other time. Let the (multi-input, multioutput) Boolean function computed by N be f_N and let the values presented at the outputs of registers I at time $t = \phi_{\text{ro}}$ be $\vec{x}$, we will prove that the values presented at the inputs of registers O at time $t = p_{\text{arch}}$ are exactly $f_N(\vec{x})$.

First, observe that the same $\vec{x}$ is produced at I every p_{clk} phase until (excluding) $t = p_{\text{arch}} + \phi_{\text{ro}}$, i.e., at

$$t = \phi_{\text{ro}}, \phi_{\text{ro}} + p_{\text{clk}}, \phi_{\text{ro}} + 2 \cdot p_{\text{clk}}, \ldots, \phi_{\text{ro}} + (k-1) \cdot p_{\text{clk}}. \quad (7)$$

Comparing Section III-B against (5), we conclude that for all combinational inputs i, its value is ready at time $t = l(i)$ corresponding to its assigned level, as well as every p_{clk} phase afterward.

Next, consider a gate n with two fanins[2] n_{i1} and n_{i2} and suppose that the values of n_{i1} and n_{i2} are ready at times corresponding to their assigned level, as well as every p_{clk} phase after these times, i.e., $t = l(n_{i1}) + j \cdot p_{\text{clk}}$ and $t = l(n_{i2}) + j \cdot p_{\text{clk}}$, respectively, where $j \in \mathbb{Z}_{\geq 0}$. By (4), we know that at time $t = l(n) - 1$, both fanins of n provide their correct values; thus, n computes its correct value at time $t = l(n)$. Moreover, as n_{i1} and n_{i2} produce the same values every p_{clk} phase, the same correct computation also repeats every p_{clk} phase since $t = l(n)$. Notice that this argument does not require $l(n_{i1})$ and $l(n_{i2}$ to be equal. By induction, we conclude that all gates compute and produce the correct value since time corresponding to their assigned levels and every p_{clk} phase afterward.

Finally, by definition of $d(N)$, we know that all combinational outputs o are ready since time $t = l(o) \leq d(N) \leq \phi_{\text{ro}} + d$; thus, at time $t = p_{\text{arch}} \geq \phi_{\text{ro}} + d$, correct values $f_N(\vec{x})$ are presented at the inputs of registers O. Equation (6) ensures that register inputs are placed at the correct phase. ■

Notice that in this analysis, the requirements for the architectural clock period $p_{\text{arch}} = k \cdot p_{\text{clk}}$ and $p_{\text{arch}} \geq \Phi_{\text{ro}} + d$ must hold regardless of adopting path-balancing or phase-alignment constraints. In other words, the proposed relaxation does not affect architectural clock frequency or latency.

[2]We consider two fanins in the analysis for convenience, but the argument can be extended to any number of fanins.

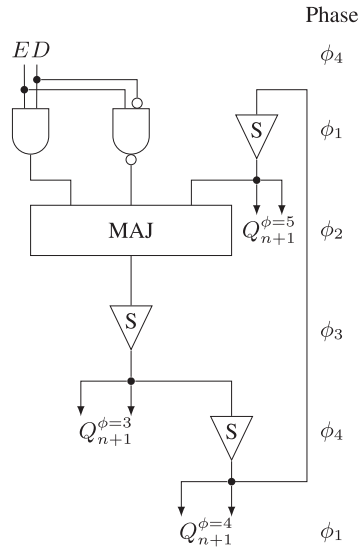

Fig. 3. Circuit schematic of an improved D-latch design.

C. PI Capacity and Phases

Based on the conventional D-latch as shown in Fig. 1, which adopts the 4-phase clocking scheme, we modify the design in Fig. 3 to show the possibility for memory devices to have an output capacity larger than 1 and to have their output signal available at multiple phases. In Fig. 3, buffers are replaced by splitters to drive up to $s_b - 1$ fanouts at various phases, not only phase 4. Adopting such D-latches as registers in a sequential circuit, PIs of the combinational network now have a splitting capacity $s_i = s_b - 1$ (where s_b is usually 3 or 4) instead of 1.

With the modified D-latch design in Fig. 3, instead of $\Phi_{\text{ro}} = \{4\}$ when adopting D-latches in Fig. 1, we may use $\Phi_{\text{ro}} = \{3, 4, 5\}$ for a more relaxed phase-alignment requirement because register outputs can be provided at various phases in the feedback loop in D-latch.

D. Consideration of Clock Skews

The analysis above assumes an ideal clock with zero clock skew. However, in real circuits, clock skews may arise when the clock signal travels along many logic levels. In other words, the activated time of a gate receiving a phase-1 clock closer to the clock source may be earlier than another gate receiving also a phase-1 clock but farther away from the clock source. The difference in the clock timing is called clock skew. One typical superconductor electronics process used to manufacture AQFP circuits is the National Institute of Advanced Industrial Science and Technology (AIST) 10 kAcm^{-2} Nb four-layer high-speed standard process (HSTP). In this process, microstriplines with a ground layer are used to deliver the ac power-clock signals to the AQFPs. A first-order approximation of the transport delay of a 5-μm-long microstripline in this process is approximately 6.20 psmm^{-1} [2]. This results in a nonzero clock skew that accumulates along the meandering power-clock network of the AQFPs [15]. With the existence of a nonzero clock skew, there is an upper limit on how many phases can be skipped without any buffer in between, in addition to the phase alignment constraint.

For large AQFP circuit designs such as a microprocessor, a meandering power-clock network may span across an entire chip, which is typically in the range of 5 × 5 mm to 10 × 10 mm in present-day superconductor fabrication processes. The accumulated skew at this scale is significant enough to produce timing errors at gigahertz range operating frequencies. In this case, it is important to physically constrain the clock skew by using microwave power dividers [15] or microwave H-tree networks [21] to reduce the physical size of the local meandering microstripline power-clock networks and, thus, reduce the accumulated clock skew. Timing characterization of AQFP cells indicates that for 5-GHz sinusoidal clocks, data can still be successfully captured with a clock skew of up to 30 ps between the launching and capturing AQFP [2], [22], [23]. This provides a nominal baseline target for how the power-clock network should be designed, and it also provides an upper limit on how much phase-skipping can be tolerated.

IV. Buffer/Splitter Insertion and Optimization

Research on the problem of buffer and splitter insertion and optimization for AQFP circuits has gained more interest in recent years. While it is possible to consider buffer and splitter cost early in logic synthesis, together with logic restructuring [18], [19], to simplify and focus on the problem and to provide fairer comparisons, many works consider only the problem of inserting the least buffers and splitters into a logic network, without logic restructuring, to fulfill the technology constraints (the *buffer insertion* problem). Starting as a postprocessing step with relatively lightweight optimization [2], [16], the buffer insertion problem has been identified as a scheduling problem in [4]. As finding the optimal schedule in terms of buffer count is likely an NP-hard problem due to the interplay between buffers and splitters, various heuristics have been proposed and improvements were made rapidly [3], [4], [5], [6]. All of these works assume the more conservative constraints, i.e., path balancing and fanout branching.

In this section, we first briefly introduce the buffer insertion algorithms this work is based on and, then, present an adapted framework considering the relaxed constraints, i.e., phase alignment and fanout branching, as discussed in Section III.

A. Related Works

1) Irredundant Local Insertion: In [4], a local insertion algorithm is proposed, which constructs the minimal buffer tree at the fanout of a gate g, given the level assignment of its fanout gates $\mathrm{FO}(g)$. The algorithm runs in linear time and is optimal subject to a given schedule.

2) Depth-Optimal Scheduling: [5] proposes to leverage the local insertion algorithm to determine the latest possible level for the root gate g, and to run this algorithm in an *as-late-as-possible* (ALAP) fashion, i.e., for each gate in a reversed topological order from POs to PIs. It is formally proved that this yields a depth-optimal scheduling. Intuitively, buffered networks with smaller depths likely also have fewer buffers.

3) Buffer Optimization by Chunked Movement: The optimality guarantee given by the local insertion algorithm is subject to a given schedule. In other words, it is possible to optimize a schedule for a smaller buffer count. Thus, given an initial schedule, the chunked movement algorithm proposed in [4] finds chunks of gates and buffers, which are tightly connected and try to move them altogether to reduce the number of buffers.

B. Adapted Framework Considering Relaxed Constraints

We integrated the three algorithms mentioned in Section IV-A in a buffer insertion framework. The implementation is available in the open-source logic synthesis library *mockturtle*[3] [24].

The overall buffer insertion flow combines the state-of-the-art algorithms. First, an initial schedule is obtained using the depth-optimal scheduling, and the minimum number of buffers needed for this schedule is counted using the irredundant local insertion algorithm. Then, the schedule is optimized for a smaller buffer count using the chunked movement algorithm, until no more improvement can be made.

To experiment on different formulations of the technology constraints, we adapted the algorithms to support customizable parameters involved in the constraints. These parameters include the following.

1) *Buffer's splitting capacity s_b:* The maximum out-degree of buffers. This is the same as in previous works.
2) *PI's splitting capacity s_i:* The maximum out-degree of PIs. s_i was fixed to 1 in previous works. However, as discussed in Section III-C, it is possible to have $s_i = s_b - 1$. Thus, we make this an integer parameter to be specified by the user.
3) A flag to switch between path balancing (1)–(3) and phase alignment (4)–(6): If phase alignment is adopted, modifications in the algorithms are made. First, levels of PIs and POs are not fixed anymore in both scheduling and chunked movement. Also, the chunked movement algorithm is modified to include PIs and POs in chunks

[3] Available: https://github.com/lsils/mockturtle

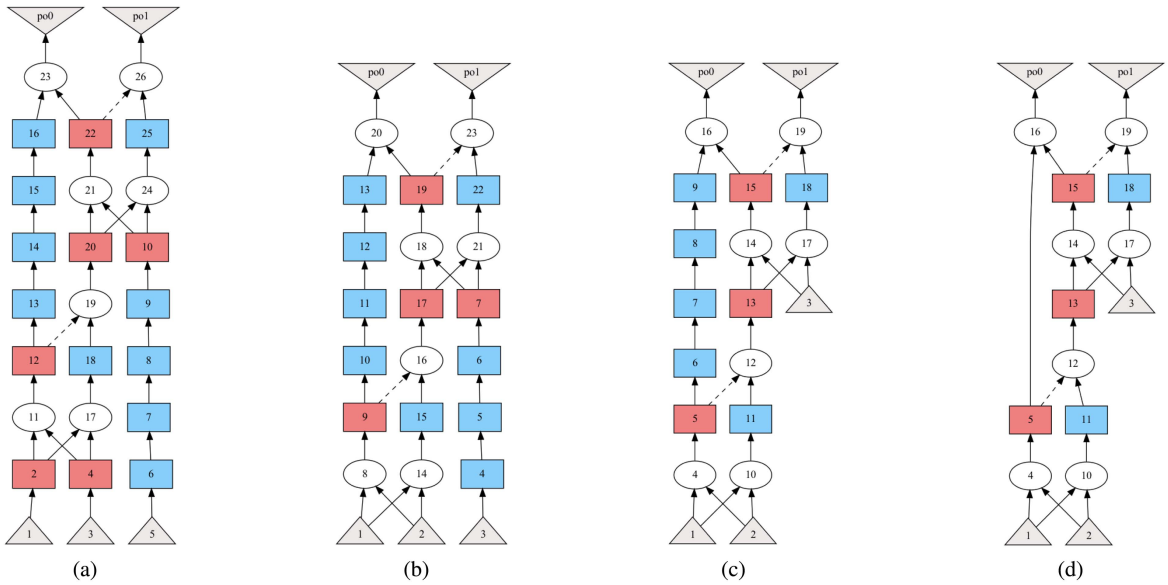

Fig. 4. Running example of how technology constraints affect the number of buffers in a small circuit. All subfigures show the optimal insertion subject to the specified constraints. (a) Path-balanced, $s_i = 1$ (16 buffers). (b) Path-balanced, $s_i = 2$ (13 buffers). (c) Imbalanced PIs and POs (9 buffers). (d) Remove buffer chains (5 buffers).

and try to move them. Special care is given to ensure that PIs and POs are always assigned to a legal phase with respect to p_{clk} and Φ_{ro}. Finally, chains of single-fanout buffers of a length being a multiple of p_{clk} are removed in a postprocessing step.

4) *Number of phases in a gate-level clock cycle p_{clk}:* When adopting path balancing, as in previous works, this parameter is not relevant. However, when relaxing path balancing to phase alignment, p_{clk} is involved in the constraints.
5) *Possible phase differences between register input and output Φ_{ro}:* Set of phases PIs are allowed to be scheduled (5). In previous works, PIs can only be scheduled at level 0 (2).
6) If clock skew is of concern, as discussed in Section III-D, then in any unbalanced path, a user-specified maximum phase-skip is ensured.

By default, our framework considers phase-alignment constraints and uses parameters $s_b = 3, s_i = 2, p_{\text{clk}} = 4, \Phi_{\text{ro}} = \{3, 4, 5\}$. This setting is expected to be the most realistic and result in the smallest size of buffered networks.

V. Impact of Technology Constraints on JJ Count

In this section, we demonstrate the impact of the proposed relaxation on technology constraints on the number of buffers and, thus, on the JJ count of an AQFP circuit. First, a small example circuit is presented, for which the optimum can be easily derived. Then, experimental results comparing different constraint formulations are listed.

A. Motivational Example

We use a 1-b full adder circuit as an example. In Fig. 4, PIs are at the bottom and POs on top; ellipse nodes are MAJ gates whose constant inputs are neglected for simplicity (i.e., AND gates or OR gates) and negated fanins are dashed; and square blue and red nodes are buffers and splitters, respectively.

The buffered network when adopting conventional constraints (path balancing and fanout branching, $s_i = 1$) is shown in Fig. 4(a), which is the optimal insertion with 16 buffers already shown in state-of-the-art works [4]. If s_i is increased to 2 as discussed in Section III-C, splitters at the first level are no longer needed, decreasing the network depth by 1 and reducing the number of buffers to 13, as shown in Fig. 4(b).

Moreover, as discussed in Section III-B, when enforcing the phase alignment constraint instead of path balancing, the number of buffers further reduces to 5, which is less than a third of the initial buffered network. This adjustment is done in two steps as described in Section IV-B. First, relaxing the constraints on PIs and POs (5), (6) instead of (2), (3) results in Fig. 4(c) with 9 buffers. Then, removing buffer chains [(4) instead of (1)] saves four more buffers.

B. Experimental Results on Constraint Relaxation

Table II shows the experimental results on the commonly used benchmark suite consisting of ISCAS benchmarks and some arithmetic circuits.[4] Five sets of constraints are presented and compared. To have a fair comparison, all of them use $s_b = 3$ and $p_{\text{clk}} = 4$ and the buffered networks are obtained using our adapted buffer insertion framework described in Section IV-B. Columns "#Bufs." list the number of buffers in the (optimized) buffered networks, columns "#JJs" list the JJ count of the buffered networks (obtained by $\#\text{JJs} = 6 \cdot \#\text{Gates} + 2 \cdot \#\text{Buf.}$), columns "$\Delta$B" and "$\Delta$JJ" list the reduction on buffer count and JJ count, respectively, and column "MPS" list the maximum phase skip.

[4]Available: https://github.com/lsils/SCE-benchmarks

TABLE II
EXPERIMENTAL RESULTS COMPARING DIFFERENT CONSTRAINTS

		Baseline		A		B		A+B		Best				
Register design		$s_i = 1$, $\Phi_{ro} = \{4\}$		$s_i = 2$, $\Phi_{ro} = \{3,4,5\}$		$s_i = 1$, $\Phi_{ro} = \{4\}$		$s_i = 2$, $\Phi_{ro} = \{3,4,5\}$		$s_i = 2$, $\Phi_{ro} = \{3,4,5\}$				
Balance PIs and POs		Yes		Yes		No		No		No				
Remove buffer chains		No		No		No		No		Yes				
Bench.	#Gates	#Buf.	#JJs	#Buf.	ΔB	#Buf.	ΔB	#Buf.	ΔB	#Buf.	ΔB	#JJs	ΔJJ	MPS
adder1	7	16	74	13	(19%)	12	(25%)	9	(44%)	5	(69%)	52	(30%)	4
adder8	77	400	1262	341	(15%)	172	(57%)	115	(71%)	87	(78%)	636	(50%)	24
mult8	439	1740	6114	1721	(1%)	1297	(25%)	1305	(25%)	681	(61%)	3996	(35%)	60
counter16	29	80	334	64	(20%)	56	(30%)	52	(35%)	52	(35%)	278	(17%)	20
counter32	82	170	832	158	(7%)	142	(16%)	139	(18%)	131	(23%)	754	(9%)	28
counter64	195	379	1928	360	(5%)	319	(16%)	317	(16%)	309	(18%)	1788	(7%)	36
counter128	428	801	4170	776	(3%)	685	(14%)	680	(15%)	656	(18%)	3880	(7%)	44
c17	6	18	72	5	(72%)	14	(22%)	5	(72%)	5	(72%)	46	(36%)	0
c432	121	904	2534	805	(11%)	582	(36%)	487	(46%)	147	(84%)	1020	(60%)	28
c499	387	1328	4978	1306	(2%)	1299	(2%)	1235	(7%)	407	(69%)	3136	(37%)	24
c880	306	1786	5408	1623	(9%)	982	(45%)	888	(50%)	516	(71%)	2868	(47%)	40
c1355	389	1330	4994	1321	(1%)	1302	(2%)	1242	(7%)	398	(70%)	3130	(37%)	24
c1908	289	1325	4384	1305	(2%)	1181	(11%)	1132	(15%)	364	(73%)	2462	(44%)	28
c2670	368	2036	6280	1812	(11%)	712	(65%)	459	(77%)	351	(83%)	2910	(54%)	32
c3540	794	2339	9442	2226	(5%)	1722	(26%)	1564	(33%)	1060	(55%)	6884	(27%)	44
c5315	1302	6013	19 838	5791	(4%)	2743	(54%)	2417	(60%)	1337	(78%)	10 486	(47%)	40
c6288	1870	9040	29 300	9008	(0%)	5924	(34%)	5886	(35%)	3206	(65%)	17 632	(40%)	168
c7552	1394	10 243	28 850	9521	(7%)	4373	(57%)	4108	(60%)	1860	(82%)	12 084	(58%)	56
sorter32	480	544	3968	448	(18%)	544	(0%)	448	(18%)	448	(18%)	3776	(5%)	0
sorter48	880	1008	7296	960	(5%)	1008	(0%)	960	(5%)	960	(5%)	7200	(1%)	0
alu32	1513	14 212	37 502	13 889	(2%)	7976	(44%)	7797	(45%)	1969	(86%)	13 016	(65%)	156
Total		55 712	179 560	53 453	(4%)	33 045	(41%)	31 245	(44%)	14 949	(73%)	98 034	(45%)	

Column "Baseline" is the most conservative constraint used in related works [2], [3], [4], [5], [6], [16], i.e., path balancing and fanout branching, plus an additional but realistic constraint that the network depth must be a multiple of $p_{clk} = 4$.[5] Column "A" uses the improved D-latch design discussed in Section III-C but still adopts path balancing. In contrast, column "B" still uses the classical register design, but does not balance PIs and POs. Column "A+B" combines both improvements. Finally, column "Best" further removes buffer chains in "A+B," shifting from path balancing to phase alignment and achieving the best constraint relaxation proposed in this article.

We observe from this experiment that considering phase alignment instead of path balancing reduces about 70% of buffers in AQFP circuits, among which about 40% are balancing PIs and POs, and the other 30% are chains of buffers within the network.

C. Experimental Results Using Larger Benchmarks

Table III shows the results of a similar experiment on the EPFL benchmark suite [25], which consists of up to 100× larger benchmarks than in the previous section. For the sake of simplicity, only the settings corresponding to columns "Baseline" and "Best" in Table II are shown. The number of buffers ("#Buf.") and the buffer-to-gate ratio ("$R_{B\text{-}G}$," the number of buffers divided by the number of gates) are listed for the two settings, as well as the reduction percentage of buffer count after relaxation ("ΔB").

It can be observed that many benchmarks have a high buffer-to-gate ratio when adopting the conventional conservative constraints, especially the arithmetic circuits (upper half). This is likely due to the imbalanced nature of these circuits. By relaxing the path-balancing constraint to phase alignment, a large portion of path-balancing buffers are eliminated, drastically reducing the number of buffers and making the buffer-to-gate ratio more reasonable. Take the *adder* benchmark as an example, with merely 384 gates in the original network, state-of-the-art buffer insertion algorithms adopting conservative constraints need to insert around 50k buffers to balance every path, 130× of the number of gates. Most JJs in the circuit and energy dissipation are wasted on these buffers. The resulting bulky buffered

[5]Many related works do not impose this constraint, although it is necessary. Enforcing this constraint adds about 1.7% buffers on this benchmark suite.

TABLE III
EXPERIMENTAL RESULTS ON EPFL BENCHMARKS

		Baseline		Best		
Register design		$s_i = 1$, $\Phi_{ro} = \{4\}$		$s_i = 2$, $\Phi_{ro} = \{3,4,5\}$		
Constraint		Path balancing		Phase alignment		
Bench.	#Gates	#Buf.	$R_{B\text{-}G}$	#Buf.	$R_{B\text{-}G}$	ΔB
adder	384	50 046	130.3	1 904	5.0	(96%)
bar	3 016	3 125	1.0	2 310	0.8	(26%)
div	57 300	1 883 971	32.9	148 268	2.6	(92%)
hyp	136 108	9 065 938	66.6	386 735	2.8	(96%)
log2	24 457	129 363	5.3	50 013	2.0	(61%)
max	2 413	71 841	29.8	3 341	1.4	(95%)
multiplier	19 716	103 153	5.2	40 263	2.0	(61%)
sin	4 307	19 261	4.5	8 450	2.0	(56%)
sqrt	23 238	1 796 085	77.3	50 299	2.2	(97%)
square	12 179	90 857	7.5	29 195	2.4	(68%)
arbiter	7 000	28 134	4.0	14 962	2.1	(47%)
cavlc	667	762	1.1	705	1.1	(7%)
ctrl	118	163	1.4	133	1.1	(18%)
dec	304	376	1.2	352	1.2	(6%)
i2c	1 246	2 921	2.3	1 549	1.2	(47%)
int2float	237	321	1.4	260	1.1	(19%)
mem_ctrl	42 714	224 766	5.3	61 114	1.4	(73%)
priority	988	17 546	17.8	1 466	1.5	(92%)
router	267	1 606	6.0	401	1.5	(75%)
voter	7 860	19 619	2.5	15 944	2.0	(19%)
Total/Average		13.5M	20.2	0.8M	1.9	(94%)

network also makes the following physical design and fabrication steps difficult. However, simply by relaxing the constraints to phase alignment, only about 1.9k buffers are actually needed, reducing the buffer count by 96%.

VI. Discussions

A. Tradeoff Between Throughput and Maximum Phase Skip

A disadvantage of replacing path balancing with phase alignment is that the possibility of wave-pipelining is disabled. *Wave-pipelining*, or multithreaded gate-level pipelining, is a technique to increase throughput by propagating more than one computation in one (architectural) clock cycle, which has been researched for classical CMOS-based digital systems [26] as well as emerging technologies [27], [28]. One important requirement for a wave-pipelined system is path balancing, thus making AQFP circuits a natural candidate to adopt this technique, although related research has not been proposed yet.

If an AQFP circuit is fully path-balanced, up to $k = p_{\text{arch}}/p_{\text{clk}}$ waves may be propagated between two register stages at the same time, increasing its throughput by $k\times$. When phase alignment is adopted instead to reduce JJ count, a tradeoff between throughput and buffer count (thus energy and area) arises. In such case, the number of waves allowed is bounded by the maximum phase skip, or inversely, given a desired throughput, the maximum allowed phase skip must be ensured, which can be achieved with our framework. Related work for the single-flux quantum (SFQ) technology family has been proposed [28], which uses ILP for scheduling and buffer insertion under similar constraints. However, for AQFP, because splitters are also clocked, this formulation cannot guarantee optimality and is also less scalable than our approach. Future AQFP circuit designers may choose path-balanced, wave-pipelined circuits for smaller components requiring higher throughput, and phase-aligned, nonpipelined circuits for larger parts consuming more energy.

B. N-Phase Clocking

Another buffer reduction method leveraging an n-phase clocking scheme has recently been proposed [11]. The basic idea is to multiply the number of phases in one (gate-level) clock cycle by an integer r, such that any chain of r buffers can be reduced to 1. The n-phase clocking technique is also very effective in reducing the number of buffers in AQFP circuits but it does not diminish the value of this work. Instead of comparing against n-phase clocking, we argue that these are two independent techniques that may work in collaboration to achieve the best results. Using our constraint formulation terminologies, n-phase clocking can be seen as using fractions instead of integers as the range of the schedule, i.e., a gate may be assigned to levels $1/r, 2/r, \ldots$, etc. n-phase clocking relaxes the path-balancing constraint by changing the clocking scheme, whereas we develop our argument from analysis of the sequential circuit model. Thus, these two relaxations affect the constraints independently and future work remains to formally consider them together. Also, as both techniques have their own drawbacks, engineers may choose between the two depending on the application requirements.

C. Physical Design and Postphysical-Design Legalization

In this article, we propose to relax path-balancing constraints to phase alignment, which will have an impact on physical design because current tools generally expect a path-balanced netlist as their input. Although adapting a physical design tool accordingly to generate realistic layouts is beyond the scope of this article, Fig. 4 serves as a good visualization of how a real layout would appear. Moreover, to truly exploit the possible area reduction due to the lower buffer count, the placement algorithm needs to be adapted to allow circuit folding. That is, instead of placing logic gates scheduled at the same level in the same physical row and having as many rows as logic levels, some gates could be placed in different rows with empty slots because of phase skipping. However, this would affect wire lengths and clock synthesis, with additional physical and timing constraints to be carefully considered.

The real clock skew between two gates in an AQFP circuit does not only depend on the number of phases in between but also on the microstripline length of the power-clock network between them [15]. Moreover, interconnect delay of data signals and longer wire lengths must also be considered to ensure the correct operation of an AQFP circuit. If the physical distance between the launching and capturing gates is too long (>0.7 mm for buffer-to-buffer connections), we may need to insert repeaters or use current boosters. However, these values are only available after physical design and are hard to predict during the buffer insertion stage. Thus, an estimation must be used in buffer insertion. More careful analysis and legalization, which may result in extra buffers being inserted, have to be done during or after physical design. Such overhead may occur in any AQFP synthesis flow regardless of adopting the proposals of this work or not but having a higher phase skip may cause the circuit being more prone to these issues, especially when operating in high frequency.

Assuming a layout realized similar to Fig. 4(d), we expect the power-clock margins to remain unchanged. However, we expect timing margins to reduce because larger phase skipping will likely incur more skew beyond the ideal timing of the capturing clocking phase. Thus, timing-aware placement [29] is important to make sure the circuit still meets sufficient timing margins.

D. Limitations and Future Directions

In this work, we experiment how assumptions on technology constraints impact the AQFP circuit cost using a postlogic-synthesis buffer insertion framework. Technology-aware logic synthesis is not considered because these algorithms need to be adapted to consider the relaxed constraints. Also, the buffer insertion and optimization algorithms in our framework do not guarantee optimum solutions because the AQFP buffer insertion problem is likely NP-hard due to the interplay between buffers and clocked splitters, and a scalable and globally optimal algorithm does not exist yet [4]. Nevertheless, this research is dedicated to explore different possibilities in formulating the technology constraints and to demonstrate their impact. In fact, it divides future research on the AQFP buffer insertion problem into two independent directions: On the one hand, considering path balancing makes the problem computationally

easier and maintains the possibility of wave-pipelining. Thus, existing algorithms are still valuable and are worth further improving, and wave-pipelining can be explored. On the other hand, considering phase alignment largely reduces JJ count, as shown in Section V-B, but its optimization problem becomes harder because of the increased flexibility. Thus, a second line of research is opened to better solve this newly-defined computational problem.

REFERENCES

[1] N. Takeuchi, D. Ozawa, Y. Yamanashi, and N. Yoshikawa, "An adiabatic quantum flux parametron as an ultra-low-power logic device," *Superconductor Sci. Technol.*, vol. 26, no. 3, 2013, Art. no. 035010.

[2] C. L. Ayala et al., "A semi-custom design methodology and environment for implementing superconductor adiabatic quantum-flux-parametron microprocessors," *Superconductor Sci. Technol.*, vol. 33, no. 5, 2020, Art. no. 054006.

[3] C.-Y. Huang, Y.-C. Chang, M.-J. Tsai, and T.-Y. Ho, "An optimal algorithm for splitter and buffer insertion in adiabatic quantum-flux-parametron circuits," in *Proc. IEEE/ACM Int. Conf. Comput. Aided Des.*, 2021, pp. 1–8.

[4] S.-Y. Lee, H. Riener, and G. De Micheli, "Beyond local optimality of buffer and splitter insertion for AQFP circuits," in *Proc. ACM/IEEE 59th Des. Automat. Conf.*, 2022, pp. 445–450.

[5] A. T. Calvino and G. De Micheli, "Depth-optimal buffer and splitter insertion and optimization in AQFP circuits," in *Proc. 28th Asia South Pacific Des. Automat. Conf.*, 2023, pp. 152–158.

[6] R. Fu, M. Wang, Y. Kan, N. Yoshikawa, T.-Y. Ho, and O. Chen, "A global optimization algorithm for buffer and splitter insertion in adiabatic quantum-flux-parametron circuits," in *Proc. 28th Asia South Pacific Des. Automat. Conf.*, 2023, pp. 769–774.

[7] N. Takeuchi, Y. Yamanashi, and N. Yoshikawa, "Adiabatic quantum-flux-parametron cell library adopting minimalist design," *J. Appl. Phys.*, vol. 117, no. 17, 2015, Art. no. 173912.

[8] C. L. Ayala, N. Takeuchi, Y. Yamanashi, T. Ortlepp, and N. Yoshikawa, "Majority-logic-Optimized parallel prefix carry look-ahead adder families using adiabatic quantum-flux-parametron logic," *IEEE Trans. Appl. Supercond.*, vol. 27, no. 4, Jun. 2017, Art. no. 1300407.

[9] N. Takeuchi et al., "Adiabatic quantum-flux-parametron cell library designed using a 10 kA cm^{-2} niobium fabrication process," *Superconductor Sci. Technol.*, vol. 30, no. 3, 2017, Art. no. 035002.

[10] N. Takeuchi, M. Nozoe, Y. He, and N. Yoshikawa, "Low-latency adiabatic superconductor logic using delay-line clocking," *Appl. Phys. Lett.*, vol. 115, no. 7, 2019, Art. no. 072601.

[11] R. Saito, C. L. Ayala, and N. Yoshikawa, "Buffer reduction via N-phase clocking in adiabatic quantum-flux-parametron benchmark circuits," *IEEE Trans. Appl. Supercond.*, vol. 31, no. 6, Sep. 2021, Art. no. 1302808.

[12] T. Yamae, N. Takeuchi, and N. Yoshikawa, "Adiabatic quantum-flux-parametron with delay-line clocking: Logic gate demonstration and phase skipping operation," *Superconductor Sci. Technol.*, vol. 34, no. 12, 2021, Art. no. 125002.

[13] N. Tsuji, C. L. Ayala, N. Takeuchi, T. Ortlepp, Y. Yamanashi, and N. Yoshikawa, "Design and implementation of a 16-word by 1-bit register file using adiabatic quantum flux parametron logic," *IEEE Trans. Appl. Supercond.*, vol. 27, no. 4, Jun. 2017, Art. no. 1300904.

[14] R. Saito, C. L. Ayala, O. Chen, T. Tanaka, T. Tamura, and N. Yoshikawa, "Logic synthesis of sequential logic circuits for adiabatic quantum-flux-parametron logic," *IEEE Trans. Appl. Supercond.*, vol. 31, no. 5, Aug. 2021, Art. no. 1301405.

[15] 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 J. Solid State Circuits*, vol. 56, no. 4, pp. 1152–1165, Apr. 2021.

[16] Q. Xu, C. L. Ayala, N. Takeuchi, Y. Murai, Y. Yamanashi, and N. Yoshikawa, "Synthesis flow for cell-based adiabatic quantum-flux-parametron structural circuit generation with HDL back-end verification," *IEEE Trans. Appl. Supercond.*, vol. 27, no. 4, Jun. 2017, Art. no. 1301905.

[17] R. Cai et al., "IDE development, logic synthesis and buffer/splitter insertion framework for adiabatic quantum-flux-parametron superconducting circuits," in *Proc. IEEE Comput. Soc. Annu. Symp. VLSI*, 2019, pp. 187–192.

[18] E. Testa, S.-Y. Lee, H. Riener, and G. De Micheli, "Algebraic and Boolean optimization methods for AQFP superconducting circuits," in *Proc. 26th Asia South Pacific Des. Automat. Conf.*, 2021. 779–785, doi: 10.1145/3394885.3431606.

[19] D. S. Marakkalage, H. Riener, and G. De Micheli, "Optimizing adiabatic quantum-flux-parametron (AQFP) circuits using an exact database," in *Proc. IEEE/ACM Int. Symp. Nanoscale Architectures*, 2021, pp. 1–6.

[20] G. Meuli et al., "Majority-based design flow for AQFP superconducting family," in *Proc. Des., Automat., Test Europe Conf. Exhib.*, 2022, pp. 34–39.

[21] Y. He et al., "Low clock skew superconductor adiabatic quantum-flux-parametron logic circuits based on grid-distributed blocks," *Superconductor Sci. Technol.*, vol. 36, no. 1, Dec. 2022, Art. no. 015006.

[22] C. L. Ayala et al., "Timing extraction for logic simulation of VLSI adiabatic quantum-flux-parametron circuits," IEICE Tech. Rep., vol. 115, no. 242, pp. 7–12, Oct. 2015.

[23] C. L. Ayala, O. Chen, and N. Yoshikawa, "AQFPTX: Adiabatic quantum-flux-parametron timing eXtraction tool," in *Proc. IEEE Int. Superconductive Electron. Conf.*, 2019, pp. 1–3.

[24] M. Soeken et al., "The EPFL logic synthesis libraries," 2022, *arXiv:1805.05121*. [Online]. Available: https://arxiv.org/abs/1805.05121

[25] L. Amarú, P.-E. Gaillardon, and G. De Micheli, "The EPFL combinational benchmark suite," in *Proc. Int. Workshop Log. Synth.*, 2015.

[26] W. P. Burleson, M. J. Ciesielski, F. Klass, and W. Liu, "Wave-pipelining: A tutorial and research survey," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 6, no. 3, pp. 464–474, Sep. 1998.

[27] O. Zografos et al., "Wave pipelining for majority-based beyond-CMOS technologies," in *Proc. Des., Automat., Test Europe Conf. Exhib.*, 2017, pp. 1306–1311.

[28] X. Li, M. Pan, T. Liu, and P. A. Beerel, "Multi-phase clocking for multi-threaded gate-level-pipelined superconductive logic," in *Proc. IEEE Comput. Soc. Annu. Symp. VLSI*, 2022, pp. 62–67.

[29] P. Dong et al., "TAAS: A timing-aware analytical strategy for AQFP-capable placement automation," in *Proc. ACM/IEEE 59th Des. Automat. Conf.*, 2022, pp. 1321–1326.

Siang-Yun Lee received the B.Sc. degree from the Department of Electrical Engineering, National Taiwan University (NTU), Taipei, Taiwan, in 2019. She is currently working toward the Ph.D. degree in logic synthesis and design automation for emerging technologies with Integrated Systems Laboratory, EPFL, Lausanne, Switzerland, led by Prof. G. De Micheli.

In NTU, she worked with Prof. J.-H. R. Jiang on threshold logic synthesis. She is currently a maintainer of the EPFL logic synthesis library mockturtle. Her research interests include logic synthesis and design automation for emerging technologies.

Christopher Lawrence Ayala (Senior Member, IEEE) received the combined B.Eng./M.Sc. and Ph.D. degrees in electrical and computer engineering from Stony Brook University, New York, NY, USA, in 2009 and 2012, respectively.

From 2013 to 2015, he was a Postdoctoral Fellow with IBM Research—Zurich, Switzerland. Since 2015, he has been with the Institute of Advanced Sciences, Yokohama National University, Yokohama, Japan, where he is currently an Associate Professor. His research interests include emerging circuit technologies, superconductor logic, NEMS-MEMS, transformative computer architectures, and electronic design automation.

Dr. Ayala is a member of The Japan Society of Applied Physics, Institute of Electronics, Information and Communication Engineers of Japan, Eta Kappa Nu Electrical and Computer Engineering Honor Society, and Tau Beta Pi Engineering Honor Society.

Giovanni De Micheli (Life Fellow, IEEE) received the Nuclear Engineer degree from Politecnico di Milano, Italy, in 1979, the M.S. and Ph.D. degrees in electrical engineering and computer science from University of California at Berkeley, Berkeley, CA, in 1980 and 1983. He is currently a Professor and the Director of the Integrated Systems Laboratory, EPFL Lausanne, Switzerland. Previously, he was a Professor of Electrical Engineering with Stanford University. His current research interests include several aspects of design technologies for integrated circuits and systems, such as synthesis for emerging technologies.

Prof. De Micheli is the recipient of the 2022 ESDA-IEEE/CEDA Phil Kaufman Award, the 2019 ACM/SIGDA Pioneering Achievement Award, and several other awards. He is a member of the Scientific Advisory Board of IMEC (Leuven, B) and STMicroelectronics. He is a Fellow of ACM and AAAS, a member of the Academia Europaea, and an International Honorary member of the American Academy of Arts and Sciences.