Unleashing the Power of T1-cells in SFQ Arithmetic Circuits

Rassul Bairamkulov*
rassul.bairamkulov@epfl.ch
Integrated Systems Laboratory, EPFL
Lausanne, VD, Switzerland

Mingfei Yu
mingfei.yu@epfl.ch
Integrated Systems Laboratory, EPFL
Lausanne, VD, Switzerland

Giovanni De Micheli
giovanni.demicheli@epfl.ch
Integrated Systems Laboratory, EPFL
Lausanne, VD, Switzerland

Abstract

Rapid single-flux quantum (RSFQ) is one of the most advanced cryogenic superconductive electronics technologies. With orders of magnitude smaller power dissipation, RSFQ is an attractive technology for cloud computing, aerospace electronics, and high-speed interfacing with quantum computing systems. Technological challenges however greatly complicate the realization of VLSI-complexity RSFQ systems. For example, gate-level pipelining in SFQ systems incurs a significant area overhead due to the need for path balancing. This issue is particularly detrimental to SFQ systems due to the limited layout density of RSFQ systems.

Multiple advanced SFQ logic cells exist that can be efficiently realized using SFQ technology. For example, a T1-cell can realize the full adder function with only half the area required by the conventional realization. This cell however imposes complex constraints on input signal timing, complicating its use.

Multiphase clocking, recently proposed to reduce the path balancing overhead, is an effective tool for controlling the timing of the signals within a network. By utilizing multiphase clocking, the timing of the input signals of the T1-cells can be efficiently satisfied, enabling its use within the SFQ networks. In this paper, we propose a two-stage SFQ technology mapping methodology supporting the T1-cells. During the logic synthesis stage, specific parts of the SFQ network are replaced by efficient T1-cells. During the retiming stage, phases are assigned to each logic gate within the network and DFFs are inserted to satisfy the timing constraints. Using our method, the area of the SFQ networks is reduced, on average, by 6% with up to 25% reduction in optimizing the 128-bit adder.

Keywords

Superconductive Electronics, Multiphase Clocking, Logic Synthesis

1 Introduction

Superconducting electronics are commonly regarded as a highly promising family of beyond-CMOS technologies. Rapid Single-Flux Quantum (RSFQ) [20] is a prominent representative within the realm of superconductive technologies. RSFQ systems consistently achieve operational frequencies in the tens of gigahertz range[8, 17], and specific structures are demonstrated to reach frequencies in the hundreds of gigahertz [5, 12, 14]. Furthermore, the power requirements for RSFQ systems are orders of magnitude lower as compared to CMOS, even accounting for refrigeration [13]. These advantages position RSFQ as a compelling candidate for large-scale stationary computing, such as data centers [13], and energy-efficient computing in space [18]. In addition, due to the minuscule heat generation and cryogenic operation, RSFQ can be used for interface circuitry for quantum computing systems [15]. All these applications require efficient arithmetic circuits.

In RSFQ systems, Josephson Junctions (JJ) and superconductive storage loops communicate using single flux quantum (SFQ) pulses [4]. The presence or absence of a flux quantum within a loop determines its logical state. Typically, a readout signal, such as a clock is required to release an SFQ pulse from the loop. Therefore, most SFQ gates, such as `NOT` and `XOR` require a clock signal, necessitating gate-level pipelining. This feature allows SFQ circuits to operate at tens to hundreds of gigahertz and achieve very large throughputs. Gate-level pipelining, however, requires fanins to each gate to have equal logic depth, an issue commonly known as *path balancing*. In addition, any datapath with a fanout greater than one requires an active splitter, consuming additional area.

To illustrate the issue of DFF and splitter insertion, consider a logic circuit in Fig. 1 [*left*]. Translating this circuit to SFQ requires two path-balancing DFFs and five splitters to ensure the correct order of data propagation, as shown in Fig. 1 [*center*]. In large circuits, the area consumed by path-balancing DFFs can occupy a significant portion of the layout. Considering the relatively small density of SFQ fabrication (~100 times smaller than CMOS) [27], area minimization becomes crucial for designing practical SFQ systems.

A variety of methods for area minimization have been proposed. In PBMap [22], the cost of path balancing is estimated during the optimization process. Several works extend the SFQ standard cell library with more complex cells, to realize the target functionality with fewer gates. In [16], for example, complex cells with up to five

*Corresponding author.
This research was supported by the SNF grant "Supercool: Design methods and tools for superconducting electronics", 200021 1920981.

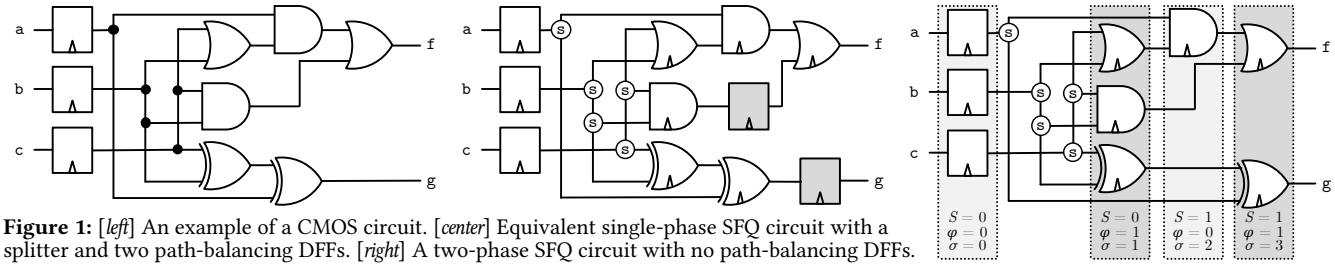

Figure 1: [*left*] An example of a CMOS circuit. [*center*] Equivalent single-phase SFQ circuit with a splitter and two path-balancing DFFs. [*right*] A two-phase SFQ circuit with no path-balancing DFFs.

inputs/outputs are proposed. Gate compounding technique [2] generalizes the design of complex logic cells by classifying SFQ primitives based on data synchronization. With compound logic cells, such as NIMPLY, the area of the networks is reduced, on average, by a factor of two [3]. Several works propose the use of unclocked gates to minimize the path balancing overhead [17, 26, 30].

The prior approaches to SFQ logic optimization are mostly inspired by the traditional logic gates. The SFQ technology, however, supports highly sophisticated elements capable of complex operations with minimal resources [10]. The most prominent example of such an element is a T1-cell [25], efficiently implementing a full adder with more than 50% fewer JJs, as compared to its conventional implementation. The T1-cell however requires input signals to be temporally separated i.e., the input pulses must never overlap. Ensuring this condition requires additional resources, such as delay elements or layout modifications. Therefore, the T1-cell is used in select regular layouts, such as counter [21] and bit-serial multiplier [24], where the correct order of signals can be maintained with relatively few resources. To this date, however, no generalized methodology for supporting the T1-cell exists in the literature.

Multiphase clocking has been recently proposed for designing area-efficient SFQ systems [19]. Using several phase-shifted clock signals allows data to propagate with fewer DFFs. For example, by using two-phase clocking, the path-balancing DFFs can be completely removed, as shown in Fig. 1 [*right*]. In addition to the area efficiency, using multiple phases enables precise control of the signal timing. This feature can be exploited to control the input signals in T1-cells, thus supporting the use of an efficient full adder in arbitrary logic networks.

In this paper, we propose a novel technology mapping methodology to synthesize multiphase SFQ networks utilizing T1-cells. Using *cut enumeration* and *Boolean matching*, we replace those parts of an SFQ circuit implementing functions compatible with a T1-cell. Next, we formulate an integer linear programming problem to assign phases to each gate within the network. Finally, the DFFs are inserted to satisfy the timing constraints of each gate within the network. In our case studies, the area of multiphase circuits with T1-cells can be reduced by up to 25%. The largest reduction is observed in such circuits as adder and voter with frequently appearing exclusive-or and majority operations.

The rest of the paper is organized as follows. The background on SFQ technology, multiphase clocking, and T1-cell is provided in Section 2. The technology mapping methodology utilizing the T1-cells is described in Section 3. Experimental results are presented in Section 4, followed by the conclusions in Section 5.

2 Background

A multi-output Boolean function $f : \mathbb{B}^k \mapsto \mathbb{B}^m$ maps k input signals to m output signals. A *Boolean function* [1] f can be represented by a *Boolean network* [2] $\mathcal{N} = (\mathcal{V} = \mathcal{I} \cup \mathcal{O} \cup \mathcal{G}, \mathcal{E})$ — a directed acyclic graph (DAG) representing the sequence of the Boolean operations applied to realize f. Set $\mathcal{G}$ is a set of gates, where each node $u \in \mathcal{G}$ applies a function f_u to its *fanins* $\mathtt{FI}(u)$ and passes the result to *fanouts* $\mathtt{FO}(u)$. Set $\mathcal{I}$ denotes the set of *primary inputs* (PI), i.e., nodes without fanins. Set $\mathcal{O}$ denotes the set of *primary outputs* (PO), i.e., nodes without fanouts.

A *cut* $C = (u, L)$ of a node u is defined as a set of nodes L such that any path from the PIs to u traverses a node $v \in L$. The cut is called *k-cut* if $|L| = k$.. The process of finding all k-cuts within the network is called *k-cut enumeration*, or simply *cut enumeration* [6]. Each cut implements a single-output Boolean function $f : \mathbb{B}^k \mapsto \mathbb{B}$ and can be represented as a truth table with 2^k rows. A truth table can be conveniently encoded as a 2^k-bit string $\mathtt{Y} = \overline{\mathtt{y}_{2^k-1} \cdots \mathtt{y}_0}$ where bit $\mathtt{y}_i$ denotes the output at the i^{th} row in the truth table. For example, $f_1(\mathtt{x}_1, \mathtt{x}_0) = \mathtt{x}_1 \vee \mathtt{x}_0$ is encoded as $\mathtt{Y}_1 = 1110_2$, since $f_1(1,1) = 1$, $f_1(1,0) = 1$, $f_1(0,1) = 1$, and $f_1(0,0) = 0$. For brevity, Y can be represented as a hexadecimal number, e.g., $\mathtt{Y}_1 = \mathtt{E}_{16}$.

A *maximum fanout free cone* $\mathtt{MFFC}(u)$ of a node u is a set of nodes such that deletion of the node u disconnects $\mathtt{MFFC}(u)$ from the fanouts. In this case, the nodes in $\mathtt{MFFC}(u)$ become *dangling* and have no effect on any PO.

2.1 Single-Flux Quantum technology

Rapid Single-flux Quantum (RSFQ) is a cryogenic superconductive computing logic family based on Josephson junctions (JJ). RSFQ gates consist of superconductive loops storing quantized magnetic flux. These superconductive loops exchange the data in the form of SFQ pulses with the area of $\Phi_0 = \hbar/2e \approx 2.07$ mV·ps [18].

According to [2], conventional SFQ logic gates can be divided into three major categories based on synchronization mechanisms;

- **Asynchronous input, Asynchronous output (AA)** components process the input information immediately upon arrival (e.g., splitter and confluence buffer (CB), a.k.a. merger).
- **Asynchronous input, Synchronous output (AS)** elements process the input information immediately upon arrival and release the output only after the arrival of the clock signal (e.g., DFF, NOT and XOR).
- **Synchronous input, Asynchronous output (SA)** elements require the input pulses to arrive simultaneously to operate correctly (e.g., AND and OR). The result of the computation is released immediately after processing.

Unlike the three types of logic gates described in the previous section, the timing requirements governing the T1-cell are more complex, not covered by the above categories.

2.2 T1-cell

The topology of the T1-cell is illustrated in Fig. 2a. The circuit has two inputs, T (*toggle*) and R (*reset*); and three outputs, S (*sum*), C (*carry*) and Q. Initially, the bias current I_0 flows along the blue dotted arrow towards junction $J_\mathtt{Q}$, corresponding to the storage of logical 0. A pulse arriving at input T switches $J_\mathtt{Q}$, producing the pulse at output $\mathtt{Q}^*$ (see Fig. 2b). The bias current is redirected along the red solid arrows, towards junctions $J_\mathtt{C}$ and $J_\mathtt{S}$, corresponding to the storage of logical 1. A second pulse at input T switches $J_\mathtt{C}$, producing the pulse at output $\mathtt{C}^*$, resetting the bias current towards $J_\mathtt{Q}$, i.e., logical 0. If the loop state is 1, an SFQ pulse at the input R switches $J_\mathtt{S}$, producing the pulse at output S, while resetting the loop state to 0. If the loop state is 0, an SFQ pulse at the input R is rejected by $J_\mathtt{R}$.

[1] For brevity, we use the term *function* to represent a *Boolean function*

[2] We use the terms *network* and *circuit* interchangeably to represent a *Boolean network*

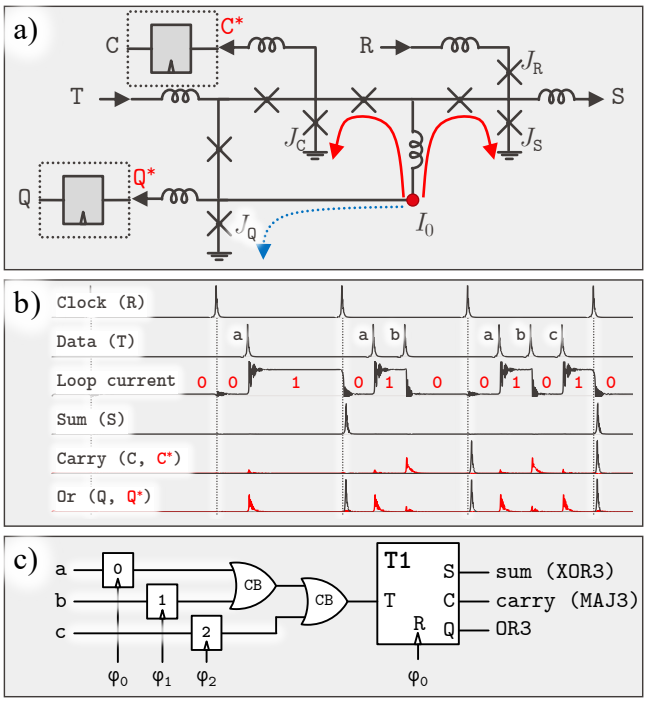

Figure 2: T1-cell. a) Schematic, b) simulation, c) full adder.

The T1-cell can be used to efficiently realize a full adder, as described in Fig. 2c. The three operands are connected to input T using two mergers. The clock signal is connected to the input R. Outputs R and C* execute, respectively, XOR3 and majority-3 (MAJ3) functions. The output C* produces the pulses asynchronously, making the output timing ambiguous. To alleviate this issue, the output C* is connected to the DFF, holding the pulse until the arrival of the clock signal. Similarly, by connecting a DFF to output Q*, a synchronous OR3 function can be realized (see shaded DFFs in Fig. 2a). In addition, the C* and Q* can be also connected to inverters to produce inverted XOR3 and MAJ3. Therefore, the extended T1-cell can efficiently produce up to five synchronous outputs. The variety of functions supported by the T1-cell can be further extended by allowing complemented inputs. With three inputs, a total of 2^3 combinations of three-input, five-output functions can be supported, as listed in Table 1. Note that, typically, using only two outputs is sufficient to significantly reduce the circuit area.

The major advantage of the T1-cell is its compactness, as compared to the regular SFQ cells. Compare the T1-cell-based full adder with the regular implementation shown in Fig. 1 [*center*]. The total number of JJs required for this circuit is 70 JJs according to CONNECT cell library [31]. In contrast, the T1-based full adder is 60% more compact, requiring only 29 JJs. However, the primary drawback of the T1-cell is its reliance on the order of input arrival. If any two input pulses arrive simultaneously to the merger element, only a single pulse propagates forward. Ensuring sufficient temporal separation of the input pulses requires precise delay tuning, potentially negating the benefits of using the T1-cell. For example, the multiplier unit proposed in [9] requires adjustment of the data and clock interconnects to control the order of input arrival.

Table 1: Truth tables (in hexadecimal format) realizable with T1-cell

Input negation			Sum	Carry	Or	Carry Inv.	Or Inv.
a	b	c	0x96	0xE8	0xFE	0x17	0x01
a	b	$\neg c$	0x69	0xD4	0xFD	0x2B	0x02
a	$\neg b$	c	0x69	0xB2	0xFB	0x4D	0x04
a	$\neg b$	$\neg c$	0x96	0x71	0xF7	0x8E	0x08
$\neg a$	b	c	0x69	0x8E	0xEF	0x71	0x10
$\neg a$	b	$\neg c$	0x96	0x4D	0xDF	0xB2	0x20
$\neg a$	$\neg b$	c	0x96	0x2B	0xBF	0xD4	0x40
$\neg a$	$\neg b$	$\neg c$	0x69	0x17	0x7F	0xE8	0x80

2.3 Multiphase clocking

Recently, multiphase clocking in SFQ has been suggested in [19] to tackle the issue of path balancing. A n-phase system utilizes n periodic signals $\{t_0, \cdots, t_{n-1}\}$ operating at the same frequency. Each clocked element g within the network is synchronized by only one clock signal at phase $\varphi(g)$. The epoch $S(g)$ of a gate g is defined as the number of clock cycles separating the gate g from the PIs, as illustrated in Fig. 1 [*right*]. The clock signals are ordered by phase $\varphi \in \{0, \cdots, n-1\}$, i.e., during any epoch, the clock signal t_i arrives before clock signal t_j if $i < j$. For convenience, we define a *stage* $\sigma(g)$ of a gate g as

$$\sigma(g) = nS(g) + \varphi(g). \tag{1}$$

We observe that, in addition to area efficiency, the multiphase clocking technique allows the time of input arrival to be precisely controlled. This capability can be utilized to correctly time the inputs to the T1 gate, as illustrated by phases φ_0, φ_1, and φ_2 in Fig. 2c. The inputs to the T1-cell are connected to the DFFs, with each DFF assigned a different phase Thus, after the T1-cell is reset, the input a is released to the T1-cell at phase 0, input b is next released at phase 1, and, finally, the input c is released at phase 2; i.e., assigning three different phases to the inputs of a T1-cell is sufficient to ensure no temporal overlap. During phase assignment and DFF insertion, described in Sections 3.2-3.3, this condition is encoded as a set of constraints to correctly assign phases and insert DFFs into a multiphase SFQ circuit with T1-cells.

3 T1-cell-aware technology mapping

Fig. 3 provides a flow diagram of the proposed methodology. To utilize a T1-cell within a circuit, we initially identify the gates that can be implemented using T1-cells. We replace these gates with more area-efficient T1-cells. Next, we ensure correct functioning of the resulting SFQ circuit with the T1-cells, by satisfying the timing requirements of each gate. The latter task can be further divided into the two subtasks. First, the each gate is assigned a phase based on our integer-linear program (ILP) that minimizes the expected number of DFFs. Second, the DFFs are inserted within the network to satisfy the timing constraints of each gate. Our formulation based on constraint programming with satisfiability (CP-SAT) determines the optimal number of DFFs for a given phase assignment.

3.1 T1-cell detection

Since a T1-cell realizes 3-input functions, to determine which parts of a given network can be substituted by T1-cells, we first perform cut enumeration. The 3-cuts are computed for all the nodes in the network in a topological order.

After enumeration, we identify those cuts that (1) have the same leaves and (2) implement compatible functions. Specifically, if a set

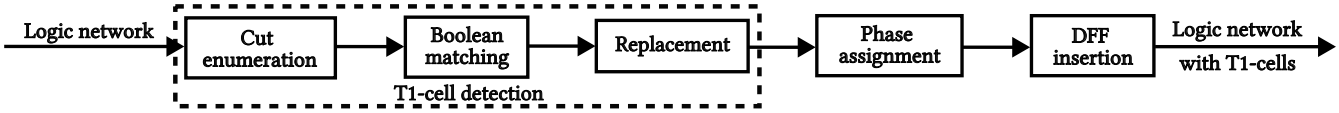

Figure 3: Proposed T1-cell insertion flow. The flow starts with the 3-cut enumeration. Those cuts compatible with the T1-cell are identified by Boolean matching. The compatible cuts are replaced with the T1 cells if the replacement yields smaller area. The correct timing is ensured by phase assignment and DFF insertion.

of cuts $C = \{C(u_1), \ldots, C(u_n)\}$, $2 \leq n \leq 5$ sharing the same leaves $\{a, b, c\}$ executes the functions realizable with the T1-cell, i.e. a subset of functions listed in Table 1, the cuts $\{C(u_1), \ldots, C(u_n)\}$ are considered for being replaced by a T1-cell. This procedure of matching a gate/cell and a cut by comparing their Boolean functions is commonly known as Boolean matching [7]. To ensure the substitution is beneficial, the area reduction ΔA due to replacement is calculated as

$$\Delta A = \sum_{i=1}^{n} A(\texttt{MFFC}(u_i)) - A_{\mathsf{T1}}(C), \tag{2}$$

where $A(\texttt{MFFC}(u_i))$ is the total area of the nodes within the MFFC of node u_i, and $A_{\mathsf{T1}}(C)$ is the area of the T1-cell implementing the functions realized by cuts in C considering possible input and output negations. Positive ΔA indicates that the area can be reduced by the substitution. The MFFCs of the nodes $u_1, \ldots, u_n$ are therefore replaced by the T1-cell.

3.2 Phase assignment

After the replacement process, the stage of each gate in the network is determined by the ILP-based *phase assignment* [1]. The goal of the ILP is minimization of the number of path-balancing DFFs while satisfying the timing requirements of each gate. Each gate type, including T1-cell, imposes different timing requirements. Here, we describe how these requirements inform the phase assignment process. For brevity, we divide the set of logic gates into three disjoint subsets $\mathcal{G} = G_{\mathsf{AA}} \cup G_{\mathsf{AS}} \cup G_{\mathsf{SA}} \cup G_{\mathsf{T1}}$, where each subset represents the elements of the corresponding category.

3.2.1 Constraints. To facilitate interfacing with the external circuitry, the epochs of the PIs and POs should be equalized, producing the following constraints

$$\sigma(g) = 0, g \in \mathcal{I}. \tag{3}$$

$$\sigma(g_1) = \sigma(g_2), g_1, g_2 \in \mathcal{O}. \tag{4}$$

For each pair $(i, j) \in \mathcal{E}$ of sequentially adjacent gates, the fanout j cannot have a clock stage earlier than i. However, specific gate combinations impose more stringent constraints on phase assignment as described in [1]. The constraints imposed by the regular AA, AS, and SA gates can be described as

$$\sigma(j) - \sigma(i) \geq \Delta\sigma_{\min}(i, j), (i, j) \in \mathcal{E} \tag{5}$$

where $\Delta\sigma_{\min}(i, j) \in \{0, 1\}$ is minimum stage difference depending on the types of i and j as summarized below.

i \ j	**AA**	**AS**	**SA**
AA	0	0	1
AS/T1	0	1	0
SA	0	1	1

The T1-cell requires the phases of the fanins to be unequal. Suppose the T1-cell with fanins $\{i_1, i_2, i_3\}$, $\sigma(i_1) \leq \sigma(i_2) \leq \sigma(i_3)$, is placed at stage $\sigma(j)$. Therefore,

- $\sigma(i_3) \leq \sigma(j) - 1$, since the T1-cell is clocked;
- $\sigma(i_2) \leq \sigma(j) - 2$, since $\sigma(j) - 1$ is occupied by i_3;
- $\sigma(i_1) \leq \sigma(j) - 3$, since $\sigma(j) - 1$ is occupied by i_3 and $\sigma(j) - 2$ is occupied by i_2.

Therefore, the phase of the T1-cell is constrained as

$$\begin{aligned} \sigma(j) \geq \max(\sigma(i_1) + 3, \sigma(i_2) + 2, \sigma(i_3) + 1), \\ (i, j) \in \mathcal{E}, j \in G_{\mathsf{T1}}, \sigma(i_1) \leq \sigma(i_2) \leq \sigma(i_3) \end{aligned} \tag{6}$$

3.2.2 DFF count. The goal of phase assignment is to find those stages of the gates minimizing the number of path-balancing DFFs inserted into the network. Finding the precise number of DFFs however requires excessively complex models, potentially degrading the runtime of the phase assignment. Therefore, more efficient models are adopted, while the precise placement of DFFs is determined during DFF insertion. For any pair of adjacent gates $(i, j) \in \mathcal{E}$, where j is not a T1-cell, the number of DFFs required for path balancing is estimated as

$$c_1(j) = \left\lfloor \frac{\sigma(j) - \sigma(i) + (j \in G_{\mathsf{SA}})}{n} \right\rfloor, j \in \mathcal{G}, i \in \texttt{FI}(j) \tag{7}$$

where an additional DFF is required if the fanout is an SA gate.

The complex constraints imposed by the T1-cell, require more sophisticated modeling. Since the T1-cell requires the inputs to arrive separately, additional DFFs are required if the phases of the inputs are equal,

$$\begin{aligned} c_2(j) = (\phi(i_1) = \phi(i_2)) \wedge (\sigma(j) - \sigma(i_1) \leq n) + \\ + (\phi(i_2) = \phi(i_3)) \wedge (\sigma(j) - \sigma(i_2) \leq n), \\ j \in G_{\mathsf{T1}}, i_1, i_2, i_3 \in \texttt{FI}(j) \end{aligned} \tag{8}$$

where $\sigma(i_1) \leq \sigma(i_2) \leq \sigma(i_3)$.

The combined optimization problem is formulated as

$$\min_{\sigma(g) \forall g \in G} \sum_{g \in \mathcal{G}} c_1(g) + \sum_{g \in G_{\mathsf{T1}}} c_2(g), \tag{9}$$

Subject to constraints (3)-(6).

3.3 DFF insertion

After assigning a stage to each gate within the circuit, the DFFs can be inserted to each datapath. We adopt a two-stage DFF insertion methodology based on CP-SAT. First, those portions of the networks bounded by the AS and SA gates are identified as *independent paths* [1]. The DFFs within distinct independent paths do not affect each other. Each independent path can be processed separately, greatly reducing the runtime. Next, for each independent path, we formulate the CP-SAT problem where the timing constraints for each gate are satisfied using the smallest number of DFFs.

The independent paths P is a portion of the logic network $P = (I, A, O)$, where set $A \subseteq G_{\mathtt{AA}}$ contains the internal unclocked AA gates and sets $I, O \subseteq G_{\mathtt{AS}} \cup G_{\mathtt{SA}} \cup G_{\mathtt{T1}}$ contain, respectively, the clocked gates at the input and output of the datapath. In Fig. 4a, the three independent paths are

$$P_1 = (I = \{\mathtt{A}, \mathtt{B}, \mathtt{C}\}, A = \{1, 2, 3, 4, 5\}, O = \{\mathtt{W}, \mathtt{X}, \mathtt{Y}, \mathtt{Z}\}),$$
$$P_2 = (I = \{\mathtt{E}\}, A = \varnothing, O = \{\mathtt{W}\}),$$
$$P_3 = (I = \{\mathtt{C}, \mathtt{F}\}, A = \{6\}, O = \{\mathtt{Z}\}).$$

For each independent path within the network, we identify the potential DFF sites for subsequent DFF insertion. To uniquely identify each potential DFF site, we define $d = (\mathtt{fi}(d), \mathtt{fo}(d), \sigma(d))$, where $\mathtt{fi}(d) \in I \cup A$ and $\mathtt{fo}(d) \in A \cup O$ are, respectively, the fanin and fanout elements of the DFF site, and $\sigma(d)$ is the stage of the DFF site. We define a chain $Q = (d_{min}, \cdots, d_{max})$ as a sequence of adjacent DFF locations situated between d_{min} and d_{max}. Examples of DFF sites and a chain are shown in Fig 4b. We define the length of chain $\Delta\sigma(Q)$ as the stage difference between d_{min} and d_{max}.

For each DFF site d, we introduce a binary variable $\delta(d)$ equal to 1 if the DFF is placed at d and 0 otherwise. The problem of minimizing the number of path-balancing DFFs can therefore be formulated as a CP-SAT problem minimizing $\sum_{d \in P} \delta(d)$.

Several constraints describe the valid placement of the DFFs within an independent path. First, for each stage, only a single DFF can be placed along a chain.

$$\sum_{d \in Q, \sigma_d = i} \delta(d) \leq 1. \tag{10}$$

For example, in Fig. 4b, the sites d and e cannot be simultaneously occupied. Second, the distance between adjacent clocked elements along a chain should not exceed n in an n-phase system. Therefore, at least one DFF should be placed along any chain of length n,

$$\bigvee_{d \in Q, \Delta\sigma(Q) = n} \delta(d) = 1. \tag{11}$$

In Fig. 4b, for example, at least one grey DFF site should be occupied, to avoid data hazard.

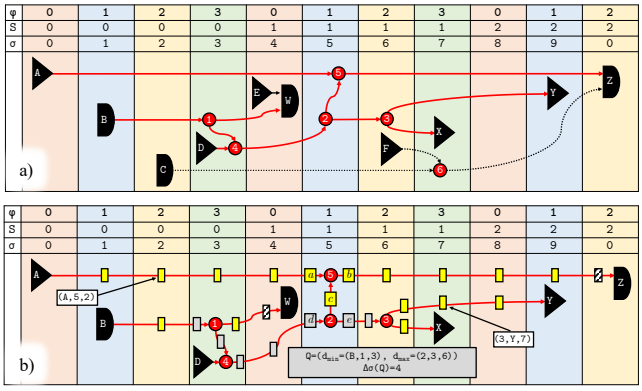

Figure 4: Example of a datapath within a four-phase network ($n = 4$). The black triangles and black curved shapes represent the AS and SA elements, respectively. Red circles denote the AA elements. a) Three independent paths are drawn with red solid, black solid, and black dotted arrows. b) DFF sites are shown as rectangles within the red independent path. The grey DFF sites represent the chain Q of length n. Diagonally shaded DFF sites precede the SA gates. DFFs should therefore be placed at these sites.

Third, recall that the SA gates should be preceded by an AS gate to function correctly. Thus, if an SA element is not directly preceded by an AS gate, a DFF should be placed before the SA element,

$$\delta(d) = 1 \quad \forall \mathtt{fo}(d) \in G_{\mathtt{SA}}, \sigma(\mathtt{fo(d)}) = \sigma(d). \tag{12}$$

Finally, the inputs to the T1-cell should never arrive simultaneously. Suppose the clocked elements (DFFs or gates) $d_1 \in Q_1$, $d_2 \in Q_2$, $d_3 \in Q_3$ precede the T1-cell. To ensure the inputs arrive at different stages, the stages of these DFFs should all be different,

$$\sigma(a) \neq \sigma(b) \Leftrightarrow a \neq b \quad \forall a, b \in \{d_1, d_2, d_3\} \tag{13}$$

An example of a correctly balanced datapath containing a T1-cell is shown in Fig. 5. Observe that the DFFs b and c are placed such that the phases of the clocked elements preceding the T1-cell (i.e., D, b, and c) are all different, ensuring no temporal overlap among the input pulses.

Eqs. (10)-(13) constitute a CP-SAT problem where the conditions (10)-(13) are satisfied with the minimum number of DFFs.

4 Experimental results

We integrate the DFF placement methodology into the technology mapping flow for SFQ compound gate circuits. The SFQ circuits are synthesized with `mockturtle` using the depth-oriented technology mapping applied to a database of pre-computed compound gate structures [3]. After inserting those T1-cells that provide improvement in area inserted into the network, we apply our CP-SAT-based phase assignment and DFF insertion procedures using Google OR-Tools [23]. Each CP-SAT problem run has been limited by 300 seconds to ensure no excessive runtime. In all cases, the phase assignment is satisfied within the allotted time. The DFF insertion determines the location and phase of each DFF within the network.

We apply our flow to synthesize a subset of EPFL [28] and ISCAS [11] benchmark circuits implementing arithmetic functions. We ran our experiments on a laptop with an Apple M1 10-core CPU with 64 GB of RAM. The number of path-balancing DFFs, circuit area (expressed in the number of JJs), and logic depth (in cycles) are shown in Table 2 (column T1). We compare our synthesis results against two baselines. The first baseline (1φ) is the single-phase circuits with full path balancing as described in [2, 3]. No T1-cells are used in the network, since the T1-cell requires at least three phases. Compared to the single-phase mapping, the area is reduced by 41% in our work, consistent with the observations in [1, 19]. The second baseline is the multiphase clocking without T1-cells [1] (4φ). Compared with the implementation without T1-cells our methodology achieves, on average a 6% better area and number of DFFs at the cost of 13% increase in the logic depth. The increase in depth can be explained by the additional stages necessary to accommodate the stringent timing constraints of the T1-cells. The benefits of adding the T1-cell vary among the circuits. The largest reduction is

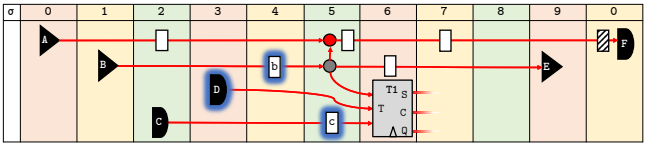

Figure 5: Example of a correctly balanced three-phase datapath containing a T1-cell. Gates B, C, and D are inputs to the T1-cell. DFFs b and c are placed such that $\varphi(\mathtt{D}) \neq \varphi(\mathtt{b})$, $\varphi(\mathtt{D}) \neq \varphi(\mathtt{c})$, and $\varphi(\mathtt{b}) \neq \varphi(\mathtt{c})$.

Table 2: Multiphase clocking with T1-cells applied to a subset of EPFL and ISCAS benchmark circuits. Our results (T1) are compared against the state-of-the-art single-phase mapping [3] (1φ) and four-phase mapping without T1 cells [1] (4φ).

	#T1-cells		#DFF			Ratio vs.		Area			Ratio vs.		Depth			Ratio vs.	
	found	used	1φ[3]	4φ[1]	T1	1φ[3]	4φ[1]	1φ[3]	4φ[1]	T1	1φ[3]	4φ[1]	1φ[3]	4φ[1]	T1	1φ[3]	4φ[1]
adder	127	127	32'768	7'963	5'958	0.18	0.75	238'419	64'784	48'844	0.20	0.75	128	32	33	0.26	1.03
c7552	17	9	2'489	713	765	0.31	1.07	32'038	19'606	19'907	0.62	1.02	16	4	5	0.31	1.25
c6288	142	142	2'625	1'431	1'349	0.51	0.94	47'198	38'840	35'386	0.75	0.91	29	8	10	0.34	1.25
sin	81	77	13'416	4'631	4'714	0.35	1.02	164'938	103'443	102'806	0.62	0.99	88	22	25	0.28	1.14
voter	252	252	10'651	5'779	5'584	0.52	0.97	222'101	187'997	182'972	0.82	0.97	38	10	11	0.29	1.10
square	861	806	44'675	16'645	14'304	0.32	0.86	525'311	329'101	301'287	0.57	0.92	126	32	32	0.25	1.00
multiplier	824	769	58'717	14'641	13'745	0.23	0.94	682'792	374'260	356'984	0.52	0.95	136	33	36	0.26	1.09
log2	644	593	86'985	33'790	33'946	0.39	1.00	978'178	605'813	598'292	0.61	0.99	160	40	47	0.29	1.18
Average						0.35	0.94				0.59	0.94				0.29	1.13

observed in adder where almost the entire circuit is replaced with the T1-cells, yielding a 25% smaller area. Significant reduction is also observed in voter, square and multiplier, while c7552 and sin yielded inferior area, likely due to the increase in the circuit depth, requiring additional path balancing DFFs.

By analyzing the final networks, we observed that none of the T1-cells utilize the output Q. All inserted T1-cells use only the S (*sum*) and C (*carry*) outputs. This trend can be attributed to a relatively low cost of the OR3 function – only two mergers and a single DFF. The area gained by replacing the OR3 function is therefore reduced. Therefore, according to Eq. (2), such T1-cell is less likely to be committed. This observation is consistent with [29]. The half and full adder were found to be the most frequently occurring multi-output cells, while other multi-output cells occurred relatively rarely.

5 Conclusions

RSFQ technology presents a remarkable opportunity to achieve unprecedented performance and energy efficiency of mainstream computing systems. Nonetheless, realizing its full potential necessitates the resolution of major technological issues, including path balancing. The T1-cells can substantially reduce the area of SFQ systems. This area reduction is particularly beneficial, since the current manufacturing technology severely limits the circuit size. Existing EDA tools however offer limited support of T1-cells in systems with asynchronous SFQ gates. In this work, we present a technology mapping for multiphase SFQ systems supporting T1-cells. Starting from an initial logic network, the T1-cells replace those parts of the network implementing the compatible functions yielding area improvements. Using the CP-SAT-based formulation, a multiphase path-balancing solution minimizing the path-balancing cost is determined. In the experimental results, we showed an average of 6% reduction in the number of JJs when compared to single-phase systems. Up to 25% reduction in area is achieved when using the four-phase clocking with T1-cells.

References

[1] R. Bairamkulov and G. De Micheli. 2024. Towards Multiphase Clocking in Single-Flux Quantum Systems. In *Proc. ASP-DAC*.

[2] R. Bairamkulov and G. De Micheli. 2023. Compound Logic Gates for Pipeline Depth Minimization in Single Flux Quantum Integrated Systems. In *Proc. GLSVLSI*.

[3] R. Bairamkulov, A. Tempia Calvino, and G. De Micheli. 2023. Synthesis of SFQ Circuits with Compound Gates. In *Proc. VLSI-SoC*.

[4] P. Bunyk, K. Likharev, and D. Zinoviev. 2001. RSFQ Technology: Physics and Devices. *IJHSES* 11, 01 (2001).

[5] W. Chen et al. 1999. Rapid Single Flux Quantum T-Flip Flop Operating up to 770 GHz. *IEEE TASC* 9, 2 (1999).

[6] J. Cong, C. Wu, and Y. Ding. 1999. Cut Ranking and Pruning: Enabling a General and Efficient FPGA Mapping Solution. In *Proc. FPGA*.

[7] G. De Micheli. 1994. *Synthesis and Optimization of Digital Circuits*. McGraw-Hill.

[8] Z. J. Deng et al. 1997. Data-Driven Self-Timed RSFQ High-Speed Test System. *IEEE TASC* 7, 4 (1997).

[9] M. Dorojevets et al. 2013. 20-GHz 8x8-Bit Parallel Carry-Save Pipelined RSFQ Multiplier. *IEEE TASC* 23, 3 (2013).

[10] K. Gaj, E. G. Friedman, and M. J. Feldman. 1997. Timing of Multi-Gigahertz Rapid Single Flux Quantum Digital Circuits. *J. VLSI Sig. Proc. Syst.* 16, 2 (1997).

[11] M. C. Hansen, H. Yalcin, and J. P. Hayes. 1999. Unveiling the ISCAS-85 Benchmarks: A Case Study in Reverse Engineering. *IEEE Des. Test. Comput.* 16, 3 (1999).

[12] Q. P. Herr, A. D. Smith, and M. S. Wire. 2002. High Speed Data Link between Digital Superconductor Chips. *Applied Physics Letters* 80, 17 (2002).

[13] D. S. Holmes, A. L. Ripple, and M. A. Manheimer. 2013. Energy-Efficient Superconducting computing—Power Budgets and Requirements. *IEEE TASC* 23, 3 (2013).

[14] T. Jabbari et al. 2020. Repeater Insertion in SFQ Interconnect. *IEEE TASC* 30, 8 (2020).

[15] M. R. Jokar et al. 2022. DigiQ: A Scalable Digital Controller for Quantum Computers Using SFQ Logic. In *IEEE HPCA*.

[16] N. K. Katam and M. Pedram. 2018. Logic Optimization, Complex Cell Design, and Retiming of Single Flux Quantum Circuits. *IEEE TASC* 28, 7 (2018).

[17] T. Kawaguchi et al. 2015. Demonstration of an 8-Bit SFQ Carry Look-Ahead Adder Using Clockless Logic Cells. In *Proc. ISEC*.

[18] G. Krylov and E. G. Friedman. 2022. *Single Flux Quantum Integrated Circuit Design*. Springer.

[19] X. Li, M. Pan, T. Liu, and P. A. Beerel. 2022. Multi-Phase Clocking for Multi-Threaded Gate-Level-Pipelined Superconductive Logic. In *Proc. ISVLSI*.

[20] K. Likharev, O. Mukhanov, and V. Semenov. 1985. Resistive Single Flux Quantum Logic for the Josephson-Junction Digital Technology. *Proc. SQUID* 85 (1985).

[21] T. Onomi, T. Kondo, and K. Nakajima. 2009. Implementation of High-Speed Single Flux-Quantum Up/Down Counter for the Neural Computation Using Stochastic Logic. *IEEE TASC* 19, 3 (2009).

[22] G. Pasandi and M. Pedram. 2019. PBMap: A Path Balancing Technology Mapping Algorithm for Single Flux Quantum Logic Circuits. *IEEE TASC* 29, 4 (2019).

[23] L. Perron. 2011. Operations Research and Constraint Programming at Google. In *Proc. CP*.

[24] S. Polonsky, J. C. Lin, and A. Rylyakov. 1995. RSFQ Arithmetic Blocks for DSP Applications. *IEEE TASC* 5, 2 (1995).

[25] S. Polonsky, V. Semenov, and A. Kirichenko. 1994. Single Flux, Quantum B Flip-Flop and Its Possible Applications. *IEEE TASC* 4, 1 (1994).

[26] S. V. Rylov. 2019. Clockless Dynamic SFQ and Gate with High Input Skew Tolerance. *IEEE TASC* 29, 5 (2019).

[27] L. Schindler, J. A. Delport, and C. J. Fourie. 2021. The ColdFlux RSFQ Cell Library for MIT-LL SFQ5ee Fabrication Process. *IEEE TASC* 32, 2 (2021).

[28] M. Soeken et al. 2018. The EPFL Logic Synthesis Libraries. *arXiv Preprint arXiv:1805.05121v3* (2018).

[29] A. Tempia Calvino and G. De Micheli. 2023. Technology Mapping Using Multi-output Library Cells. In *Proc. ICCAD*.

[30] G. Tzimpragos et al. 2020. A Computational Temporal Logic for Superconducting Accelerators. In *Proc.ASPLOS*.

[31] S. Yorozu, Y. Kameda, H. Terai, A. Fujimaki, T. Yamada, and S. Tahara. 2002. A Single Flux Quantum Standard Logic Cell Library. *Physica C: Superconductivity* 378-381 (2002).