# Towards Multiphase Clocking in Single-Flux Quantum Systems

*Abstract*—**Rapid single-flux quantum (RSFQ) is one of the most advanced superconductive electronics technologies. SFQ systems achieve operate at tens of gigahertz with up to three orders of magnitude smaller power as compared to CMOS. In conventional SFQ systems, most gates require clock signal. Each gate should have the fanins with equal logic depth, necessitating insertion of path-balancing (PB) DFFs, incurring prohibitive area penalty.**

**Multiphase clocking is the effective method for reducing the path-balancing overhead at the cost of reduced throughput. However, existing tools are not directly applicable for technology mapping of multiphase systems. To overcome this limitation, in this work, we propose a technology mapping tool for multiphase systems. Our contribution is threefold. First, we formulate a phase assignment as a Constraint Programming with Satisfiability (CP-SAT) problem, to determine the phase of each element within the network. Second, we formulate the path balancing problem as a CP-SAT to optimize the number of DFFs within an asynchronous datapath. Finally, we integrate these methods into a technology mapping flow to convert a logic network into a multiphase SFQ circuit. In our case studies, by using seven phases, the size of the circuit (expressed as the number of Josephson junctions) is reduced, on average, by 59.94 % as compared to the dual (fast-slow) clocking method, while outperforming the state-of-the-art single-phase SFQ mapping tools.**

## I. INTRODUCTION

Superconductive electronics is one of the most promising beyond-CMOS technologies, with Rapid Single-Flux Quantum (RSFQ) [1] being one of the most prominent superconductive technologies. RSFQ systems consistently achieve operating frequencies on the order of tens of gigahertz [2], with particular structures accelerating to hundreds of gigahertz [3]. Furthermore, the operating power of the RSFQ systems is two to three orders of magnitude smaller than CMOS, even considering the refrigeration power [4].

Achieving the aforementioned advantages at scale however remains a challenging task. In CMOS, transistors encode information by voltage levels. In contrast, in RSFQ, Josephson Junctions (JJ) encode information using single flux quantum (SFQ) pulses [5]. The absence or presence of a SFQ pulse encode, respectively, a logical 0 or 1. The flow of pulses within a logic network requires synchronization. While a few structures, such as splitter and OR, can operate asynchronously, most logic gates in the original RSFQ cell library, such as AND, NOT and XOR, require clock signal [1]. This feature requires the data to be pipelined at the gate level.

Two features further complicate scaling the RSFQ systems. Due to the quantized nature of SFQ pulses, most RSFQ logic gates are limited to fanout of one. A special gate called splitter is necessary to duplicate a signal [5], [6], as illustrated in Fig. 1b. The need for splitters is particularly significant in clock distribution networks. Delivering a clock signal to $n$ clocked gates typically requires a binary tree of $(n-1)$ splitters [7]. In addition, path balancing is required to ensure the correct order of data propagation within the network, as indicated by four path balancing D-flip-flops (DFF) in Fig. 1b. The number of path balancing DFFs and splitters can be prohibitively large, degrading the area and yield of an integrated system. The complexity of the clock tree synthesis is also increased due to the large number of clocked elements. Despite the advances in technology mapping [8], [9], the overhead of path balancing and clock distribution remains large, motivating the investigation of alternative RSFQ architectures.

Alternative clocking methods have been explored in the literature alleviate the issue of path balancing. In dual clock methodology [10], a logic circuit is partitioned into separate clocking domains using the architectural nondestructive readout (NDRO) registers operating at low frequency. The clocked gates within a single domain are clocked at high frequency, while cross-domain communication is performed via the nondestructive readout (NDRO) registers operating at the low frequency. Using this approach, the number of path balancing
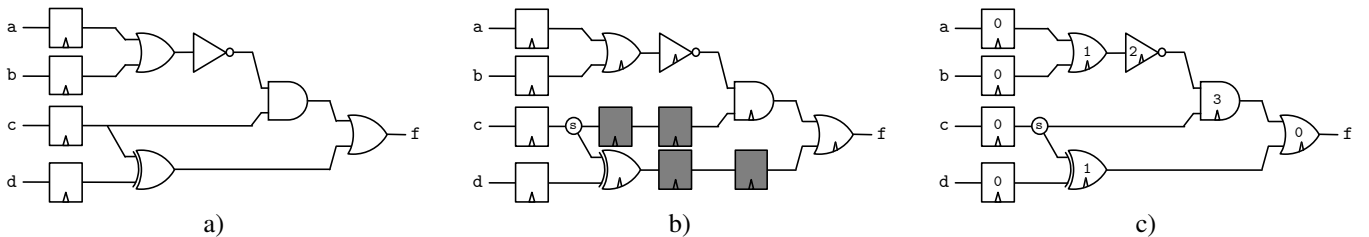


Fig. 1. a) An example of a CMOS circuit. b) Equivalent SFQ circuit with a splitter and two path balancing DFFs. c) A four-phase SFQ realization with no path balancing DFFs. The numbers indicate the phase corresponding to each clocked gate.

DFFs can be reduced by 35% by using the slow clock with the 1/5 frequency of the fast clock. The throughput of the system is however reduced by a factor of 5, establishing a tradeoff between the area and the throughput of the system. The primary limitation of the dual clock method is the use of the large registers to repeat the data signals and additional AND gates to suppress the spurious pulses within the system [11]. In addition, similar to full path balancing, distributing the high frequency clock signal makes the timing closure more challenging to achieve.

Multiphase clocking has been recently proposed for designing area-efficient SFQ systems [12]. The circuit operates using several phase-shifted clock signals allowing a pulse to propagate to the subsequent stages without a DFFs. For example, by using four-phase clocking, the path balancing DFFs can be completely removed, as shown in Fig. 1c. Using multiple phases was demonstrated effective in reducing the path balancing overhead in SFQ circuits. With only two phases, the number of DFFs is reduced by 55% [12], with further reduction brought by additional phases. Furthermore, in an $n$-phase system, the clock signals have $n$ times lower frequency, simplifying the clock distribution process. Similar to dual clock method, the area reduction is achieved by sacrificing the throughput. An $n$-phase system has $n$ times smaller throughput as compared to a single phase system.

Although the multiphase clocking has been shown to reduce the path balancing overhead, existing technology mapping tools offer limited support for the multiphase systems. For example, path balancing DFFs cannot be efficiently distributed within the complex datapaths utilizing asynchronous SFQ gates. To bridge this gap, in this paper, we propose a novel methodology to apply the multiphase clocking to SFQ circuits. First, we extend the DFF insertion methodology to assign the phase to each gate within the circuit, including unclocked elements. Next, we formulate PB DFF insertion as a Constraint Programming with SAT (CP-SAT) problem to satisfy the timing constraints with minimum number of PB DFFs. We improve the scalability by identifying independent paths allowing the asynchronous paths to be processed separately. Finally, we integrate phase assignment and DFF insertion into the technology mapping flow to realize an arbitrary logic network with multiple clock phases. With only two phases, our technology mapping flow outperforms the state-of-the-art single-phase methods. Compared with the dual clocking method, our technology mapping flow achieves up to 82% smaller size, while offering equal throughput.

The rest of the paper is organized as follows. The principles of SFQ technology and multiphase clocking are reviewed in Section II. The CP-SAT-based method for phase assignment is described in Section III PB DFF insertion are described in Section IV. Experimental results and comparison with the prior works are provided in Section V, followed by conclusions in Section VI.

## II. BACKGROUND

A logic network $N = (V, E)$ is a directed acyclic graph (DAG) with $V$ denoting a set of nodes and $E \subseteq V \times V$ is a set of edges. The set of nodes $V = I \cup O \cup G$ consists of three disjoint subsets representing, respectively, the primary inputs (PI), primary outputs (PO) and gates within the network. A set of fanins $FI(g)$ (fanouts $FO(g)$) of gate $g \in V$ denotes the nodes connected to $g$ via and incoming (outgoing) edge.

Technology mapping refers to the process of transforming an arbitrary logic network $N$ into a technology-specific representation $N'$ using a particular set of primitives. The technology mapping aims to optimize the target characteristics of the mapped network $N'$, such as area, delay, or yield The nature of a technology greatly influences the technology mapping process. In the next subsection, the properties of the SFQ technology are reviewed.

### A. 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. The information is transferred between the superconductive loops in form of SFQ pulses with the area of $\Phi_0 = \hbar/2e \approx 2.07$ mV·ps [6]. Based on synchronization mechanisms, the SFQ logic gates can be divided into three major categories [13].

The **Asynchronous input, Asynchronous output (AA)** components process the input information immediately upon arrival. The most common gates in this category are splitter and merger. A *splitter* cell produces two pulses for each incoming pulse. A *merger* cell, often referred to as *confluence buffer* (CB), directs signals from multiple (typically two) input branches into one output branch, effectively performing an OR function. The **Asynchronous input, Synchronous output (AS)** elements process the input information immediately upon arrival and release the output synchronously after the arrival of the clock signal. The components of this type include *D-flip-flop (DFF)*, *inverter* (NOT) and *exclusive-or* (XOR). The **Synchronous input, Asynchronous output (SA)** elements require the input signals to arrive simultaneously to operate correctly. The result of the computation is released immediately after processing. Assuming inputs arrive simultaneously, a CB can be tuned to produce at most a single output pulse, producing an OR element [14]. Furthermore, by adjusting the JJ size and bias current, the OR structure can be transformed into AND element, requiring both signals to arrive simultaneously to produce an output pulse. In conventional RSFQ [1], the OR and AND gates incorporate the DFFs at the inputs to ensure the simultaneous release of the data pulses. However, the simultaneity can be achieved by other AS gates, such as NOT and XOR [13], allowing richer functionality to be realized in fewer clock cycles.

To correctly consider the type of each element, we divide the set of logic gates into three disjoint subsets $G = G_{AA} \cup G_{AS} \cup G_{SA}$, where each subset represents the elements of the corresponding category. Note that only gates in the subset $G_{AS}$ require clock signal.

### B. Multiphase clocking

Multiphase clocking is a clocking technique based on several clock signals with equal clock period. The seminal work by Li *et al.* [12] demonstrated MPC as an effective remedy for path balancing in SFQ systems. A $n$-phase system utilizes $n$ periodic signals $\{t_0, \cdots, t_{n-1}\}$ operating at the same frequency. Each
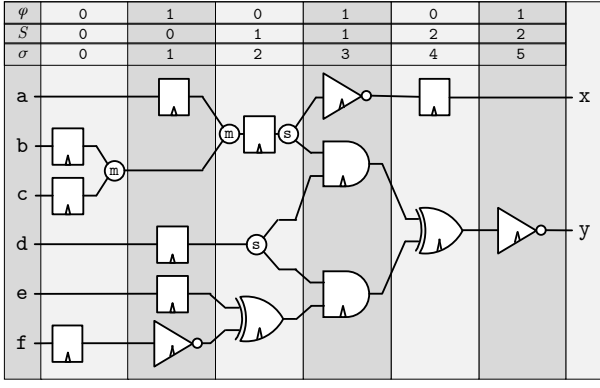
Fig. 2. An example two-phase network. Each PI is placed at stage 0 or 1, corresponding to epoch 0. The POs are placed at stages 4 and 5, corresponding to epoch 2.
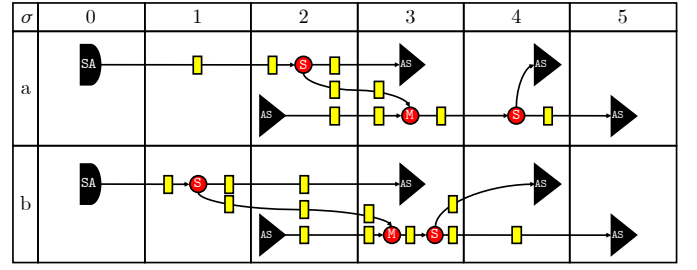


Fig. 3. The effect of phase assigned to splitters and mergers. Datapaths (a) and (b) execute the same function. The phases assigned to the splitters and mergers (denoted as s and m, respectively) are however different. The potential DFF locations (depicted as yellow rectangles) are therefore different in the two datapaths.

$$\left\lfloor \frac{\sigma(i)}{n} \right\rfloor = \left\lfloor \frac{\sigma(j)}{n} \right\rfloor \forall i, j \in O. \tag{5}$$

clocked element $g \in G_{AS}$ 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. 2. 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 consistency in I/O timing, the PIs are placed at the same epoch 0 and the POs are placed at the same epoch. For example, the POs are Fig. 2, the fanouts x and y are both placed at epoch 2. Observe that a path balancing DFF is inserted before x to equalize the epochs of both POs. For convenience, we define a *stage* $\sigma(g)$ of a gate $g$ as

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

Multiphase clocking relaxes the path balancing requirements of a clocked system. With $n$ phases, the stage difference $\Delta\sigma(i,j)$ between any pair of sequentially adjacent elements $(i,j)$ should be no greater than $n$. Compared with single phase systems where a $q$-stage difference is balanced by $q$ DFFs, the same difference within an $n$-phase system can be balanced with only $\left\lfloor \frac{q}{n} \right\rfloor$ DFFs, significantly reducing the path balancing overhead in imbalanced paths.

To realize the potential of multiphase clocking in SFQ systems, we propose the two-step DFF insertion methodology based on CP-SAT. During the phase assignment step, the SFQ gates are assigned a phase to reduce the number of DFFs within the system, as described in Section III. During the DFF insertion step, optimal DFF insertion is determined for each datapath, as described in Section IV.

### III. PHASE ASSIGNMENT

The phase assignment is the procedure of determining the stage (i.e., epoch and phase) of each gate within the network. An ILP-based methodology for phase assignment to reduce the number of path balancing DFFs is proposed in [12],

$$\min_{\sigma(g)\forall g \in G} \sum_{(i,j)\in E} \left\lfloor \frac{\sigma(j) - \sigma(i)}{n} \right\rfloor, \tag{2}$$

Subject to:

$$\sigma(i) < \sigma(j) \forall (i,j) \in E, \tag{3}$$

$$\left\lfloor \frac{\sigma(i)}{n} \right\rfloor = 0 \forall i \in I, \tag{4}$$

This formulation is highly effective when considering the systems consisting of only clocked gates, i.e. $G = G_{AS}$, $G_{AA} = G_{SA} = \varnothing$ This formulation however offers limited support for the unclocked gates, such as mergers and SA AND gate. Unclocked elements can greatly improve the quality of SFQ technology mapping by enriching the functionality available within a single clock cycle. In [13], for example, the area of the circuit was reduced by up to 53% by utilizing asynchronous elements. However, the unclocked gates impose timing constraints different from the clocked gates. To extend the support of this ILP model to unclocked gates, additional constraints are introduced in this section.

While the notions of phase, epoch, and stage are primarily relevant for the clocked gates, it is however important to also assign a stage to the AA and SA elements, since a stage of an unclocked element guides the DFF insertion. Consider two networks containing an asynchronous datapath illustrated in Fig. 3. Both networks realize the same function, however, the possible locations of path balancing DFFs are different. Assigning a stage to an unclocked gate is important to correctly place the path balancing DFFs.

#### A. SA element constraints

Recall that the inputs of the SA gates require simultaneous arrival of data pulses to the gate. To ensure the simultaneous arrival, the AS gate should be placed immediately before the SA gate, at the same clock stage, as shown in Fig. 4a. If the stage of AS gate is earlier than the SA gate, a path balancing DFF should be inserted. In addition, the SA gate cannot be placed at the same stage as the fanin, if the fanin is not an AS gate. For example, if the fanin of the SA gate has fanout greater than one, a splitter is inserted after the AS gate. Thus, the fanin type of the SA gate is a splitter (AA gate). The propagation delay incurred by the AA gates, such as splitters, may desynchronize the inputs arriving to the SA gate, producing a data hazard, as illustrated in Fig. 4b. To avoid this condition, the minimum stage of gate $g$ is increased by one, as shown in Fig. 4c.

#### B. Asynchronous paths

The sharing of DFFs by a complex unclocked path has been highlighted as early as 1991 in [15], where sharing of DFFs is maximized to minimize the circuit area. While estimating the minimum number of DFFs in special cases, such as splitter tree, is relatively simple, more complex asynchronous paths,
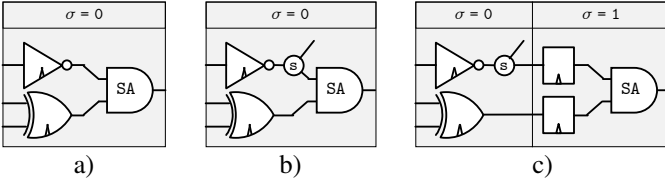
Fig. 4. Timing constraints of an SA gate. a) The gate should be preceded by an AS gate at the same stage. b) Example of an invalid assignment. The AS gate is preceded by an AA gate at the same stage. c) The issue is resolved by shifting the SA gate to the next stage and inserting two path balancing DFFs.

however, require more sophisticated processing and, hence, prohibitive runtime. During the phase assignment stage we therefore approximate the number of DFFs as

$$C\left(\sigma\left(g\right)\right) = \left\lfloor \frac{\max\limits_{a \in FO(g)} \left(\sigma\left(a\right)\right) - \sigma\left(g\right)}{n} \right\rfloor. \qquad (6)$$

The exact number of DFFs is determined during the DFF insertion stage described in Section IV. The combined optimization problem is formulated as

$$\min_{\sigma(g)\forall g \in G} \sum_{(i,j) \in E} \left\lfloor \frac{\sigma\left(j\right) - \sigma\left(i\right) + \left(j \in G_{SA}\right)}{n} \right\rfloor, \qquad (7)$$

Subject to:

$$\sigma\left(i\right) < \sigma\left(j\right) \forall \left(i,j\right) \in E, j \in G_{AS}, \qquad (8)$$

$$\sigma\left(i\right) \leq \sigma\left(j\right) \forall \left(i,j\right) \in E, j \notin G_{AS}, \qquad (9)$$

$$\sigma\left(g\right) \geq \sigma\left(a\right) + \left(a \notin G_{AS}\right) \forall g \in G_{SA}, a \in FI\left(g\right), \qquad (10)$$

$$\left\lfloor \frac{\sigma\left(i\right)}{n} \right\rfloor = 0 \forall i \in I, \qquad (11)$$

$$\left\lfloor \frac{\sigma\left(i\right)}{n} \right\rfloor = \left\lfloor \frac{\sigma\left(j\right)}{n} \right\rfloor \forall i,j \in O. \qquad (12)$$

Each gate $g$ with $\#FO(g) > 1$ requires a splitter tree to copy the signal to each of the fanouts. After completing the phase assignment, a splitter tree that maximizes sharing of path balancing DFFs can be produced. First, the fanouts $FO(g)$ of a gate $g$ are sorted by phase in the ascending order, producing a sequence $A = [a_0, a_1, \cdots, a_k]$, $\sigma(a_i) \leq \sigma(a_j) \iff i < j$. Next, the splitters $[s_0, ..., s_{k-1}]$ are inserted at phases $a_0, \cdots, a_{k-1}$. Each splitter $s_i$ has a splitter $s_{i-1}$ as a fanin, except $s_0$ whose fanin is the gate $g$. Each splitter $s_i$ has the $a_i$ and $s_{i+1}$ as a fanout, except $s_{k-1}$ whose fanouts are $a_{k-1}$ and $a_k$. Using this method, the splitters are placed as late as possible within the network, maximizing sharing of the datapath.

## IV. DFF PLACEMENT

The complex asynchronous datapaths greatly complicate the DFF insertion process. Consider for example the datapath shown in Fig. 3. The optimal placement of the DFFs cannot be easily inferred from the datapath topology. In this section, we propose a CP-SAT based placement methodology for determining the optimal location of the DFFs. Our methodology consists of two stages. First, we identify *independent datapaths* within an unbalanced logic network, as described in Subsection IV-A. Next, for each independent datapaths, we formulate a CP-SAT

problem where the timing constraints are satisfied using the minimum number of DFFs, as described in Subsection IV-B.

### A. Independent datapaths

Recall that only a single clocked gate can be placed at the same stage along a datapath. The AS and SA elements therefore prevent a DFF from being placed at the same stage. Furthermore, any timing constraint imposed by a DFF placed before the AS/SA element, will be dominated by the timing constraint imposed by the AS/SA element itself at any location after the AS/SA element. This feature allows representing a network-wide DFF insertion problem as multiple smaller problems. This partitioning capability brings two major advantages to the DFF insertion process. First, while the network-wide DFF insertion problem can be solved in $O(e^N)$ time, the complexity of a partitioned problem is $O(e^k N_{part})$, where $N$ is the network size, $N_{part} \propto N$ is the number of independent datapaths and $k$ is the size of the largest partition (typically, $k \ll N$). Second, partitioning facilitates multithreaded processing, allowing faster processing on a multicore system.

The proposed DFF insertion method is applied to a single independent datapath constrained by the synchronous gates at the input and the output, as illustrated in Fig. 5a. The datapath $P$ can be described as a portion of the logic network $P = (I, A, O)$, where set $A \subseteq G_{AA}$ contains the AA gates within the datapath and sets $I, O \subseteq G_{AS} \cup G_{SA}$ describe, respectively, the AS and SA gates at the input and output of the datapath. In Fig. 5a, the three independent paths are

$$P_1 = (I = \{\mathtt{A}, \mathtt{B}, \mathtt{C}\}, A = \{1, 2, 3, 4, 5\}, O = \{\mathtt{W}, \mathtt{X}, \mathtt{Y}, \mathtt{Z}\}),$$

$$P_2 = (I = \{\mathtt{E}\}, A = \varnothing, O = \{\mathtt{W}\}),$$

$$P_3 = (I = \{\mathtt{C}, \mathtt{F}\}, A = \{6\}, O = \{\mathtt{Z}\}).$$

### B. DFF insertion

After identifying each independent path within the network, we determine the potential DFF sites for subsequent DFF insertion. To uniquely identify each potential DFF site, we define $d = (g_d^i, g_d^o, \sigma(d))$, where $g_d^i \in I \cup A$ and $g_d^o \in A \cup O$ are the fanin and fanout elements of the DFFs and $\sigma(d)$ is the stage of the DFF site. We define a chain $Q = (d_{min}, \cdots, d_{max})$ as a sequence of sequentially adjacent DFF locations situated between $d_{min}$ and $d_{max}$. The examples of DFF sites and a chain are shown in Fig 5b. For convenience, we define the length of chain $\Delta\sigma(Q)$ as the stage difference between $d_{min}$ and $d_{max}$.

For each DFF site, 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

$$\sum_{d \in P} \delta(d). \qquad (13)$$

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. In Fig. 5b, for example, the stage $\sigma = 5$ has five potential DFF locations, $a, b, c, d,$ and $e$. The DFFs here should however satisfy the constraint

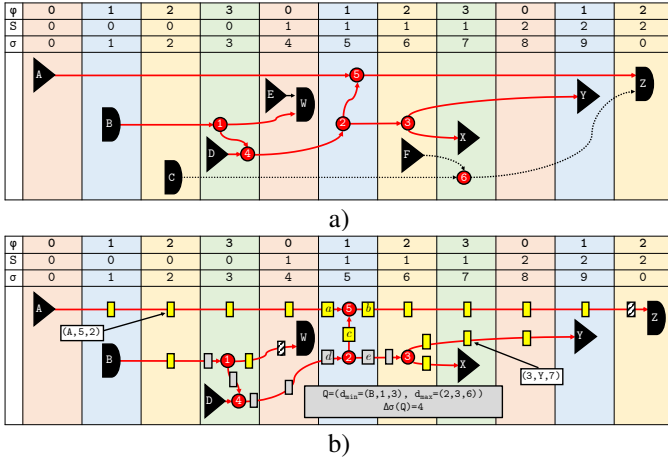$$\sum_{d \in Q, \sigma_d = i} \delta(d) \leq 1. \qquad (14)$$

a)



b)

Fig. 5. Example of an asynchronous 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 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.

The combinations of DFF sites satisfying this constraint are $\{a, c, e\}$, $\{a, d\}$, $\{b, e\}$, since only these sets do not place multiple DFFs along any chain. Other conflicting DFF sites are highlighted in blue in Fig. 5b.

Second, the distance between any pair of clocked elements along a chain should not exceed $n$ in an $n$-phase system. This requirement can be formulated as a condition

$$\bigvee_{d \in Q, \Delta\sigma(Q)=n} \delta(d) = 1 \qquad (15)$$

stating that at least one DFF should be placed along any chain of length $n$.

Finally, 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,

$$\delta(d) = 1 \forall g_d^o \in G_{SA}, \sigma(g_d) = \sigma_d. \qquad (16)$$

The DFF sites before the SA gates are shaded with diagonal hatch pattern in Fig. 5b.

Equations (13)-(16) constitute a CP-SAT problem where the conditions (14)-(16) are satisfied with the minimum number of DFFs (13).

## V. 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 a database of precomputed compound gate structures [13]. The circuits are next decomposed into the functional primitives while removing all DFFs within the circuit. We next apply out CP-SAT-based phase assignment and DFF insertion procedures using Google OR-Tools [16]. Each CP-SAT problem run has been limited by 60 seconds to ensure no excessive runtime. In all cases, a feasible phase assignment has been found within the allotted time. The solution to the DFF insertion specifies the location and phase of each DFF inserted into the network.
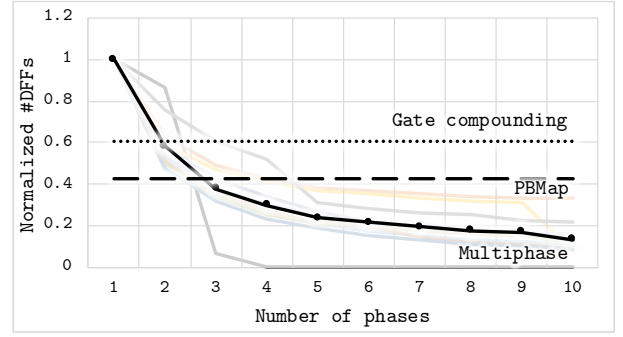


Fig. 6. Normalized average number of DFFs with different number of phases. On average, three phases are sufficient to outperform both gate compounding and PBMap. The lines in the background represent normalized number of DFFs for individual benchmarks.

We apply our mapping flow to synthesize a subset of EPFL [17] and ISCAS [18] benchmark circuits. We first compare our results with dual clocking method [11]. Two experiments are reported where the frequency of the slow clock is 7 and 12 times smaller than the frequency of the fast clock. For fair comparison, we use 7- and 12-phase, to equalize the throughput. The results are shown in Table I. Note that unlike [11], the DFFs within the AND and OR gate are explicitly counted in our work due to the use of the unclocked (SA) AND and OR elements. Furthermore, additional elements, such as pulse repeaters and NDRO flip flops are needed in DCM. We therefore focus on the number of JJs, commonly used in SFQ technology as an area metric. The number of JJs is reduced, on average, by 59.94% when using 7-phase clocking. The largest reduction is observed in the priority benchmark circuit containing many imbalanced paths that require long chains of path balancing DFFs to be inserted. With multiphase clocking, these chains can be balanced by approximately 7 times fewer DFFs, contributing to the area savings.

For 12-phase clocking, the area is reduced by 46.5%. Observe that the smallest reduction in area (and the increase in DFF count) are observed in circuits with relatively small depth. These circuits therefore have fewer imbalanced paths that could benefit from multiphase clocking. Despite relatively large number of DFFs, we still manage to reduce the total JJ count due to the use of unclocked elements (AA and SA) supported by our tool.

The Fig. 6 describes the relationship between the number of DFFs and the number of phases. The numbers of DFFs are normalized with respect to a single phase system mapped with our flow. Consistent with [12] the number of DFFs reduces with multiple phases. The area savings due to an additional phase gradually diminishes as more phases are added. We compare our results against PBMap [8] and gate compounding [13], the state-of-the-art single-phase SFQ mapping algorithms. With our mapping flow, two phases are sufficient, on average, to outperform gate compounding. With three phases our flow yields fewer DFFs than PBMap.

In RSFQ, the circuit area is commonly expressed as the number of JJs within the network. The Fig. 7 describes the relationship between the number of JJs and the number of phases. In terms of area, our flow outperforms PBMap with a single phase, primarily due to the use of asynchronous gates

| | 1/7 throughput | | | | | | 1/12 throughput | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | DCM | | Multiphase | | Change | | DCM | | Multiphase | | Change | |
| | #DFF | #JJ | #DFF | #JJ | #DFF | #JJ | #DFF | #JJ | #DFF | #JJ | #DFF | #JJ |
| int2float | 117 | 7'770 | 217 | 5'136 | +85.47% | −33.90% | 39 | 5'140 | 46 | 3'939 | +17.95% | −23.37% |
| priority | 8'562 | 257'252 | 3'285 | 45'094 | −61.63% | −82.47% | 4'225 | 158'568 | 1'775 | 34'524 | −57.99% | −78.23% |
| voter | 7'204 | 447'044 | 2'180 | 162'804 | −69.74% | −63.58% | 3'732 | 355'144 | 1'568 | 158'520 | −57.98% | −55.36% |
| c432 | 224 | 10'734 | 342 | 5'116 | +52.68% | −52.34% | 118 | 7'124 | 240 | 4'402 | +103.39% | −38.21% |
| c880 | 362 | 14'658 | 254 | 6'190 | −29.83% | −57.77% | 187 | 9'483 | 119 | 5'245 | −36.36% | −44.69% |
| c1908 | 282 | 13'169 | 125 | 3'529 | −55.67% | −73.20% | 144 | 8'739 | 69 | 3'137 | −52.08% | −64.10% |
| c3540 | 776 | 43'437 | 589 | 17'016 | −24.10% | −60.83% | 282 | 26'897 | 440 | 15'973 | +56.03% | −40.61% |
| c1355 | 193 | 8'739 | 46 | 4'515 | −76.17% | −48.34% | 119 | 6'149 | 44 | 4'501 | −63.03% | −26.80% |
| s13207 | 1'795 | 106'346 | 1'837 | 42'382 | +2.34% | −60.15% | 571 | 60'766 | 1'082 | 37'097 | +89.49% | −38.95% |
| s5378 | 645 | 50'766 | 808 | 21'761 | +25.27% | −57.13% | 255 | 34'053 | 368 | 18'681 | +44.31% | −45.14% |
| s382 | 56 | 4'448 | 89 | 2'411 | +58.93% | −45.80% | 9 | 2'750 | 48 | 2'124 | +433.33% | −22.76% |
| Geomean | 557.04 | 29'868.20 | 413.48 | 11'965.54 | −25.77% | −59.94% | 227.85 | 19'565.38 | 229.57 | 10'467.10 | +0.76% | −46.50% |
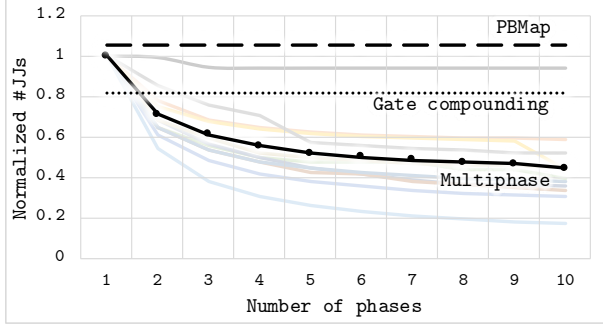


Fig. 7. Normalized average number of JJs (area metric) with different number of phases. On average, two phases are sufficient to outperform both gate compounding and PBMap. The lines in the background represent normalized number of JJs for individual benchmarks.

(e.g., confluence buffer instead of an OR gate). A two-phase network synthesized with our flow requires fewer JJs than a single-phase compound-gate network.

## VI. 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. Multiphase clocking can substantially reduce the path balancing overhead in SFQ systems. Existing path balancing tools however offer limited support of multiphase clocking in systems with asynchronous SFQ gates. In this work, we presented a technology mapping and path balancing methodology for multiphase SFQ systems based on constraint programming with satisfiability (CP-SAT). An SFQ logic network is initially synthesized with mockturtle logic synthesis library. Next, the circuit is decomposed into primitives and the DFFs are removed from the network. Each primitive gate is assigned a phase using the CP-SAT while satisfying the special timing constraints imposed by the unclocked SFQ elements. Finally, using the novel CP-SAT formulation, the minimum number of path balancing DFFs satisfying the timing constraints is determined. In the experimental results, we showed an average of 59% reduction in the number of JJs when compared to dual clocking method. Furthermore, with only two phases, our methodology yields smaller networks than the state-of-the-art single-phase SFQ mapping techniques.

## REFERENCES

[1] K. Likharev, O. Mukhanov, and V. Semenov, "Resistive Single Flux Quantum Logic for the Josephson-Junction Digital Technology," *Proc. International Conference on Superconducting Quantum Devices*, Vol. 85, pp. 1103–1108, June 1985.

[2] T. Kawaguchi, M. Tanaka, K. Takagi, and N. Takagi, "Demonstration of an 8-Bit SFQ Carry Look-Ahead Adder Using Clockless Logic Cells," *International Superconductive Electronics Conference*, July 2015.

[3] W. Chen, A. Rylyakov, V. Patel, J. Lukens, and K. Likharev, "Rapid Single Flux Quantum T-Flip Flop Operating up to 770 GHz," *IEEE Trans. Appl. Supercond.*, Vol. 9, No. 2, pp. 3212–3215, 1999.

[4] D. S. Holmes, A. L. Ripple, and M. A. Manheimer, "Energy-Efficient Superconducting computing—Power Budgets and Requirements," *IEEE Trans. Appl. Supercond.*, Vol. 23, No. 3, pp. 1701610–1701610, 2013.

[5] P. Bunyk, K. Likharev, and D. Zinoviev, "RSFQ Technology: Physics and Devices," *International Journal of High Speed Electronics and Systems*, Vol. 11, No. 01, pp. 257–305, 2001.

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

[7] R. Bairamkulov, T. Jabbari, and E. G. Friedman, "QuCTS — Single-Flux Quantum Clock Tree Synthesis," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, Vol. 41, No. 10, pp. 3346–3358, October 2022.

[8] G. Pasandi and M. Pedram, "PBMap: A Path Balancing Technology Mapping Algorithm for Single Flux Quantum Logic Circuits," *IEEE Trans. Appl. Supercond.*, Vol. 29, No. 4, June 2019.

[9] N. Kito, K. Takagi, and N. Takagi, "Logic-Depth-Aware Technology Mapping Method for RSFQ Logic Circuits With Special RSFQ Gates," *IEEE Trans. Appl. Supercond.*, Vol. 32, No. 4, 2021.

[10] G. Pasandi and M. Pedram, "An Efficient Pipelined Architecture for Superconducting Single Flux Quantum Logic Circuits Utilizing Dual Clocks," *IEEE Trans. Appl. Supercond.*, Vol. 30, No. 2, pp. 1–12, 2020.

[11] G. Pasandi and M. Pedram, "Depth-Bounded Graph Partitioning Algorithm and Dual Clocking Method for Realization of Superconducting SFQ Circuits," *ACM JETCAS*, Vol. 17, No. 1, October 2020.

[12] X. Li, M. Pan, T. Liu, and P. A. Beerel, "Multi-Phase Clocking for Multi-Threaded Gate-Level-Pipelined Superconductive Logic," *Proceedings of the IEEE Computer Society Symposium on VLSI*, pp. 62–67, 2022.

[13] R. Bairamkulov and G. De Micheli, "Compound Logic Gates for Pipeline Depth Minimization in Single Flux Quantum Integrated Systems," *Proceedings of the ACM Great Lakes Symposium on VLSI*, 2023.

[14] O. Mukhanov, V. Semenov, and K. Likharev, "Ultimate Performance of the RSFQ Logic Circuits," *IEEE Trans. Magn.*, Vol. 23, No. 2, pp. 759–762, 1987.

[15] C. E. Leiserson and J. B. Saxe, "Retiming Synchronous Circuitry," *Algorithmica*, Vol. 6, No. 1–6, pp. 5–35, June 1991.

[16] L. Perron, "Operations Research and Constraint Programming at Google," *Proceedings of the International Conference on Principles and Practice of Constraint Programming*, 2011.

[17] M. Soeken, H. Riener, W. Haaswijk, E. Testa, B. Schmitt, G. Meuli, F. Mozafari, S.-Y. Lee, A. Tempia Calvino, D. S. Marakkalage, and G. De Micheli, "The EPFL Logic Synthesis Libraries," *arXiv Preprint arXiv:1805.05121v3*, 2018.

[18] M. C. Hansen, H. Yalcin, and J. P. Hayes, "Unveiling the ISCAS-85 Benchmarks: A Case Study in Reverse Engineering," *IEEE Des. Test. Comput.*, Vol. 16, No. 3, pp. 72–80, 1999.