J2Place: A Multiphase Clocking-Oriented Length-Matching Placement for Rapid Single-Flux-Quantum Circuits

Rongliang Fu[1†], Minglei Zhou[2,3†], Huilong Jiang[2], Junying Huang[2*], Xiaochun Ye[2], Tsung-Yi Ho[1]
[1]*Department of Computer Science and Engineering, The Chinese University of Hong Kong*, Hong Kong, China
[2]*SKLP, Institute of Computing Technology, CAS*, Beijing, China
[3]*University of Chinese Academy of Sciences*, Beijing, China

***Abstract*—Superconducting Rapid Single-Flux-Quantum (RSFQ) logic, characterized by low power consumption and high-frequency operation, has broad application prospects and holds substantial potential for future computing technologies. However, ensuring the correct operation of RSFQ circuits requires inserting numerous D flip-flops (DFFs), which substantially increase circuit area and energy dissipation. Recent studies have demonstrated that the multiphase clocking scheme can effectively reduce the number of required DFFs. Despite these advantages, existing placement tools do not support multiphase clocking RSFQ circuits. To address this limitation, this paper introduces J2Place, a novel multiphase clocking-oriented length-matching placement framework for RSFQ circuits. Our approach introduces two new RSFQ cells, TFFDO and TFFDE, to simplify the clock network in two-phase clocking designs. We propose a maximum flow-based method to generate the clock distribution column by column and utilize dynamic programming to minimize the total vertical wirelength while maintaining fixed placement orders. Additionally, to expand the solution space, we propose a length-aware reordering method to reduce the wirelength further. Experimental results on ISCAS85 and EPFL benchmarks demonstrate the effectiveness and efficiency of J2Place compared with state-of-the-art methods.**

***Index Terms*—RSFQ, placement, multiphase clocking**

I. Introduction

Rapid Single-Flux-Quantum (RSFQ) technology [1], based on superconducting Josephson junctions (JJs), has emerged as a transformative approach for future high-performance computing systems. This technology utilizes quantum coherence to achieve outstanding performance metrics. By encoding information in discrete magnetic flux quanta, RSFQ circuits achieve ultra-fast switching speeds on the order of picoseconds, which is significantly faster than conventional CMOS technology, while consuming ultra-low switching energy on the order of 1×10^{-19} J/bit [2], [3]. This combination of ultra-fast speed and ultra-low power dissipation positions RSFQ as a promising candidate for applications demanding extreme computational efficiency, such as high-performance microprocessors [4], quantum computer controllers [5], and power-efficient neural network accelerators [6].

Despite these significant performance advantages of RSFQ circuits, their unique physical characteristics hinder the direct application of conventional electronic design automation (EDA) tools specifically designed for CMOS technology to their physical design process. Most RSFQ logic gates require synchronization with a single clock to achieve clock-driven gate-level pipelining, necessitating the insertion of numerous D flip-flops (DFFs) to meet the path balancing constraint. This increases circuit power and incurs a prohibitive area penalty. For an 8-bit integer divider, the inserted DFFs are $4.5\times$ the original logic gate count [7]. Recent studies [8], [9] indicate that multiphase clocking can effectively reduce the number of inserted DFFs and improve the area efficiency of RSFQ circuits. With only two-phase clocking, the number of inserted DFFs and area can be reduced by 55% and 41% [9], respectively. However, n-phase clocking requires generating and distributing n separate clock signals with different phases, which increases the routing overhead and imposes more strict timing constraints. These pose challenges to the physical design of multiphase clocking RSFQ circuits.

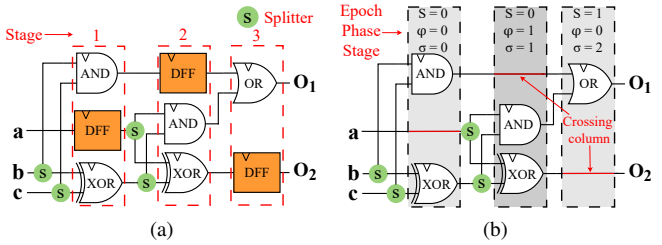


Fig. 1. Implementations of a 1-bit RSFQ full adder using (a) single-phase clocking, which requires 3 path-balancing DFFs, and (b) two-phase clocking, which eliminates the need for path-balancing DFFs.

Prevailing RSFQ placement methods predominantly focus on single-phase clocking, providing limited support for multiphase clocking. Kito *et al.* [10] proposed a single-phase clocking RSFQ placement algorithm based on the Simulated Annealing (SA) algorithm. Nevertheless, this SA-based methodology neglects the crucial timing constraints of RSFQ circuits and requires a significant runtime, particularly when dealing with large-scale circuits. Wang *et al.* [11] proposed a novel row-based clock-tree-aware placement methodology for RSFQ circuits to minimize the maximum path length. Chen *et al.* [12] proposed a clock-aware length-matching placement method for single-phase RSFQ circuits. This method simultaneously addresses the length-matching issues of both data and clock signals, effectively meeting the timing constraints of RSFQ circuits and significantly reducing the complexity of timing alignment during the routing phase. Unfortunately, these single-phase clocking-oriented RSFQ placement methods cannot be directly applied to RSFQ placement with multiphase clocking. Under a multiphase clocking scheme, clock signals with different phases increase the difficulty of timing alignment. Therefore, new RSFQ placement methods must be designed for multiphase clocking.

This paper proposes a multiphase clocking-oriented length-matching RSFQ placement algorithm to address the clock distribution and timing alignment issues in multiphase clocking RSFQ circuits. Overall, our contributions are as follows:

- We propose J2Place, a multiphase clocking-oriented length-matching placement algorithm that considers the timing constraints of multiphase clocking RSFQ circuits.
- We devise two novel RSFQ cells, TFFDO and TFFDE, to provide clock signals of different phases for the logic gates in the two-phase clocking RSFQ circuits.
- We propose a maximum flow-based clock distribution method

† Equal contribution. * Corresponding author: huangjunying@ict.ac.cn.

to minimize the vertical wirelength of the clock network column by column.
- We propose a dynamic programming (DP)-based column-wise placement method and a length-aware reordering for further wirelength reduction.
- On the ISCAS85 and EPFL benchmarks, J2Place reduces vertical wirelength by 50.09%, 57.12%, and 56.24%, respectively, compared to state-of-the-art methods.

II. Preliminary

A. RSFQ Logic

RSFQ logic [1] is a cryogenic superconducting computing technology based on JJs. In RSFQ circuits, information is transferred as SFQ pulses between superconductive loops storing quantized magnetic flux. In contrast to CMOS logic gates, RSFQ logic gates are typically clocked, generally limited to two inputs, and achieve multiple fanouts through unclocked splitter trees. These logic gates process input pulses by modifying internal current loops' states and generating output pulses synchronized with an input clock pulse, enabling natural gate-level pipelining.

In gate-level-pipelined RSFQ circuits, a single-phase clocking scheme is commonly employed, where the same clock synchronizes all gates. To ensure proper pulse alignment, all paths from a primary input (PI) to a primary output (PO) must traverse an identical number of clocked gates, a requirement known as path balancing. Achieving path balancing requires the insertion of numerous DFFs, which introduces significant area and power overhead.

The single-phase clocking RSFQ circuits usually adopt a stage-driven pipelined layout. Logic gates are arranged in columns according to their stages, with gates of the same stages placed in the same column (Fig. 1(a)). However, unlike single-phase clocking RSFQ circuits, where all fanin gates of a logic gate must be at the same stage, logic gates in multiphase clocking RSFQ circuits may have fanin gates with different stages (Fig. 1(b)), which pose challenges for the placement of multiphase clocking RSFQ circuits.

B. Multiphase Clocking

Multiphase clocking is a recently proposed clocking technique based on multiple clock signals with the same clock period but different phases [9]. An n-phase clocking RSFQ circuit employs a set of periodic signals $\{t_0, \ldots, t_{n-1}\}$ with identical frequencies but uniformly spaced phase offsets. Each logic gate g is driven by a clock signal of a specific phase denoted as $\varphi(g)$. The epoch $S(g)$ of a logic gate g is defined as the number of clock cycles required for signal propagation from the PI to the gate. Clock signals are ordered in ascending phase sequence $\varphi \in \{0, \ldots, n-1\}$, ensuring that for any two phases $i < j$, the clock signal t_i arrives before t_j. Following [8], we define the stage $\sigma(g)$ of logic gate g in the multiphase clocking RSFQ circuit as

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

Multiphase clocking significantly alleviates the path-balancing requirement in RSFQ circuits. Unlike single-phase clocking, which requires all fanin gates of a logic gate to be at the same stage, n-phase clocking allows a maximum stage difference of n between any two fanin gates (i, j), *i.e.*, $\Delta\sigma(i, j) \leq n$. This relaxation significantly reduces the number of DFFs needed to satisfy the path-balancing constraint. For example, two-phase clocking can completely eliminate the path balancing DFFs in the 1-bit RSFQ full adder design, as demonstrated in Fig. 1. However, this area reduction comes at the cost of throughput: an n-phase circuit operates at a throughput n times lower than that of a single-phase circuit.

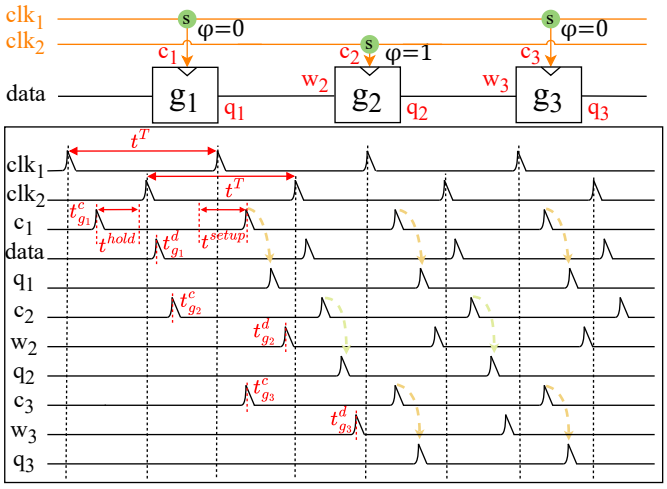


Fig. 2. A two-phase RSFQ sub-circuit and the timing of its gates, g_1, g_2, and g_3. $t^c_{g_i}$ and $t^d_{g_i}$ represents the arrival times of g_i's clock and data signals, respectively.

In the multiphase clocking pipelined RSFQ placement, we first calculate the stage $\sigma(g)$ of each logic gate g according to Equation (1). Then, the logic gates are placed in columns based on their stages. Logic gates with the same stage are placed in the same column, as shown in Fig. 1(b). These columns are arranged from left to right in ascending order of the stages, thus forming a multi-stage pipelined placement. However, since multiphase clocking relaxes the path balancing constraint, data pulses can propagate across columns to logic gates, as shown in Fig. 1(b). This necessitates space allocation for cross-column connections during placement.

C. Timing Constraints

In the physical design of RSFQ circuits, timing design [13] is crucial for achieving high performance. This paper utilizes the concurrent-flow clocking scheme for multiphase RSFQ circuits, where the clock and data signals flow in the same direction. To ensure the correct operation of the circuit, the arrival times of the clock and data signals of RSFQ logic gates must satisfy:

$$t^c + t^{hold} < t^{data} < t^c + t^T - t^{setup}, \tag{2}$$

where t^c and t^{data} refer to the arrival times of the clock and data signals, respectively. The timing parameters t^{setup} and t^{hold} are a gate's setup and hold times, which are determined by the specific characteristics of the logic gate. t^T represents the cycle of the clock signal.

Fig. 2 shows the timing of an RSFQ sub-circuit composed of three DFFs, g_1, g_2, and g_3, operating under two-phase clocking. The clock signals clk_1 and clk_2 share the same clock cycle but differ in phase: clk_1 has a phase of $\varphi = 0$ and drives g_1 and g_3, while clk_2 has a phase of $\varphi = 1$ and drives g_2. For each gate, the clock arrival time $t^c_{g_i}$ precedes its data arrival time $t^d_{g_i}$, ensuring proper timing. For example, for g_1, the clock (c_1) arrival time is $t^c_{g_1}$, and the data ($data$) arrival time is $t^d_{g_1}$. The output pulse (q_1) is generated after the next pulse of c_1.

In RSFQ circuits, the clock pulse must be delivered to almost all the logic gates, which imposes substantial demands on routing resources and leads to considerable power consumption. To address

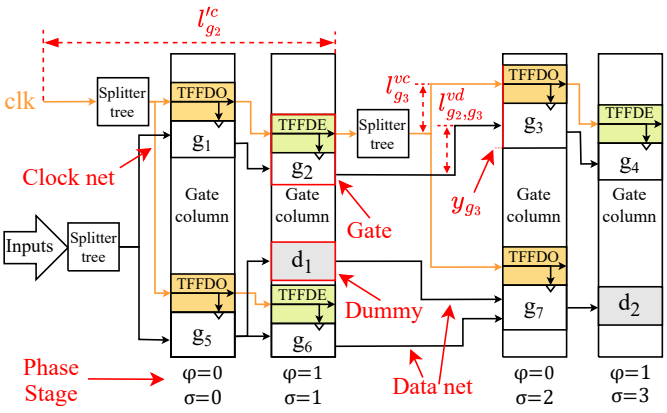


Fig. 3. Two-phase clocking-based placement schematic of an RSFQ circuit, where d_i represents the PTL crossing the column.

this issue, prior research has introduced tree-based clock distribution methods for single-phase RSFQ circuits, wherein clock pulses are propagated across columns via splitters, thus reducing the wirelength of the clock network [10], [14]. However, in n-phase clocking RSFQ circuits, n separate clock signals are required. Applying the tree-based scheme directly would necessitate n independent clock networks, greatly increasing clock distribution complexity and overhead. To overcome this limitation, we introduce two novel RSFQ cells, TFFDO and TFFDE, to replace splitters within gate columns for clock propagation. These cells are capable of generating clock pulses with different phases, thereby eliminating the need for multiple clock distribution networks. This approach simplifies the clock distribution in multiphase RSFQ systems, reduces routing complexity, and enhances scalability. Further implementation details are provided in Section III-B.

III. New RSFQ Cells and Problem Formulation

A. Terminology

A multiphase clocking RSFQ circuit can be represented by a directed graph $N(V, E)$. Considering the cross-column connections during the placement process, we introduce dummies representing the passive transmission lines (PTLs) crossing gate columns (Fig. 3), with dummy heights equal to the actual PTL size. After inserting dummies, the node set $V = G \cup D \cup I \cup O$, where G and D are the logic gate and dummy sets, respectively. I and O denote the PIs and POs, respectively. D^c and D^d denote the set of dummies for clock and data propagation, respectively. Nodes with a stage of i are included in the set $V_i = \{v|\sigma(v) = i, v \in G \cup D\}$. The height of a node v is denoted as h_v. The position of node v within a column, denoted as y_v, indicates the location of the lower boundary of the node. $\mathrm{FI}(v)$ denotes the set of nodes that serve as data inputs for node v, and $\mathrm{FO}(v)$ denotes the set of nodes that serve as data outputs for node v.

B. TFFDO and TFFDE Cell Design

As discussed in Section II-C, each logic gate g within a two-phase clocking RSFQ circuit is synchronized by the clock signal at a specific phase $\varphi(g)$. Unlike single-phase clocked RSFQ circuits, two-phase clocking requires two clock signals with distinct phases, significantly increasing the complexity of clock distribution. To address this issue, we designed two novel RSFQ cells, TFFDO and TFFDE, capable of generating clock pulses at phases 0 and 1, respectively.

As shown in Fig. 3, the clock pulse from the PI first flows into a splitter tree to generate clock pulses for the gates within the first column. For the remaining columns, the clock pulse propagates through the preceding column via TFFDO/E to reach the next column. Specifically, logic gates operating at phase 0 are paired with TFFDO cells to receive odd-numbered clock pulses, while gates at phase 1 are paired with TFFDE cells to receive even-numbered clock pulses. This approach enables two-phase clock distribution from a single clock signal, significantly simplifying the clock network.

Fig. 4 shows the designs of TFFDO and TFFDE. As shown in Fig. 4(a) and Fig. 4(b), each cell includes an input A, a toggle output $Q0$ (responding to the odd- and even-numbered pulses in TFFDO and TFFDE, respectively), and a direct output $Q1$ synchronized with the input. Both designs comprise 10 JJs, with a state storage loop formed by J7-LS-J8, where LS is a large inductor used to store one single magnetic flux quantum. We utilized the open-source TimEx tool [15], [16] to extract the corresponding Mealy state diagrams (Fig. 4(c) and Fig. 4(e)) and obtained the digital Verilog models with delay parameters. The analog design incorporating JJs was simulated using the JoSIM tool [17], [18], and the results are presented in Fig. 4(d) and Fig. 4(f). For TFFDO, during the transition from state '0' to '1', the inductor LS stores a single flux quantum, and both $Q0$ and $Q1$ generate outputs. Conversely, during the transition from state '1' to '0', LS releases the stored flux, and only $Q1$ generates an output, as shown in Fig. 4(c) and Fig. 4(d). The operation of TFFDE can be similarly analyzed concerning Fig. 4(e) and Fig. 4(f).

The physical layouts of TFFDO and TFFDE cells are shown in Fig. 5, which were designed using the open-source tools KLayout [19] and Spira [20], [21], along with cell library information provided by ColdFlux [22]. These designs were developed in strict compliance with the MIT-LL SFQ5ee process [23], [24], and design rule checks were performed to ensure fabrication compatibility. Each cell has two layout variants, occupying an area of 70 $\mu m \times 30$ μm. Fig. 5(a) and Fig. 5(b) depict two layout versions of TFFDO, where the $Q0$ port is positioned at the lower-right and lower-left corners, respectively, for routing convenience. Similarly, Fig. 5(c) and Fig. 5(d) show layout variants for TFFDE. Note that both layouts include PTL receivers and transmitters. In the SFQ5ee process, PTLs are typically implemented in either the M1 or M3 metal layers, sandwiched between two ground layers, while JJs are placed between the M5 and M6 layers.

C. Problem Formulation

The primary objective of RSFQ placement is to optimize the wirelength while meeting the timing constraints, thereby minimizing the overall layout area. In RSFQ circuits, extending the length of PTLs is a crucial approach to meeting the timing constraint. Since a PTL's delay is roughly proportional to its length, the need for timing adjustment can be translated into length-matching requirements in the placement process, *i.e.*, $l^c + l^{hold} < l^{data} < l^c + l^T - l^{setup}$. Given that connections between adjacent columns have identical horizontal length, connection length minimization can be confined to the vertical dimension, encompassing both the vertical minimum length and vertical matching length. The vertical minimum length refers to the vertical Manhattan distance between logic gates, while the vertical matching length represents the length of the detours to satisfy the timing constraint.

As shown in Fig. 3, the PTL length corresponding to the clock delay from g_2 to g_3 can be calculated as $l^{vc}_{g_3} + l^{mc}_{g_3} + l^{cell}$, where $l^{vc}_{g_3}$ and $l^{mc}_{g_3}$ are the vertical minimum length and vertical matching length of g_3's clock connection. l^{cell} represents the PTL length corresponding to the delay caused by splitters and TFFDO/E. Moreover, since

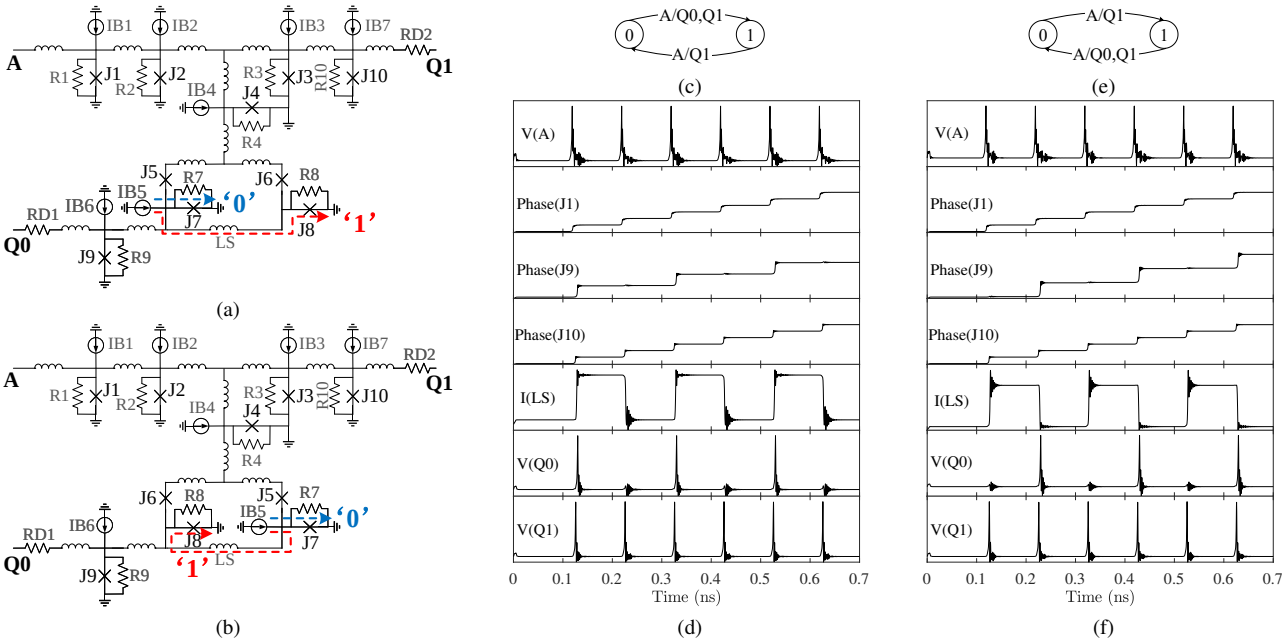


Fig. 4. (a) and (b) show the JJ-level schematics of TFFDO and TFFDE, respectively, while (c) and (e) show their corresponding state transform diagrams of TFFDO and TFFDE, and (d) and (f) show their corresponding timing schematics. Nominal parameters of the main components in (a) and (b) are as: $i_{c1} = 160\ \mu$A, $i_{c2} = 180\ \mu$A, $i_{c3} = 250\ \mu$A, $i_{c4} = 250\ \mu$A, $i_{c5} = 300\ \mu$A (TFFDO)/$260\ \mu$A (TFFDE), $i_{c6} = 180\ \mu$A, $i_{c7} = 250\ \mu$A, $i_{c8} = 230\ \mu$A, $i_{c9} = 250\ \mu$A, $i_{c10} = 250\ \mu$A, $R1 = 7.8\ \Omega$, $R2 = 7.2\ \Omega$, $R3 = 5.8\ \Omega$, $R4 = 5.8\ \Omega$, $R7 = 5.8\ \Omega$, $R8 = 6.2\ \Omega$, $R9 = 5.8\ \Omega$, $R10 = 5.8\Omega$, $RD1 = 1.36\ \Omega$, $RD2 = 1.36\ \Omega$, $LS = 8.25$ pH, $IB1 = 140\ \mu$A, $IB2 = 160\ \mu$A, $IB3 = IB4 = IB5 = IB6 = IB7 = 175\ \mu$A.

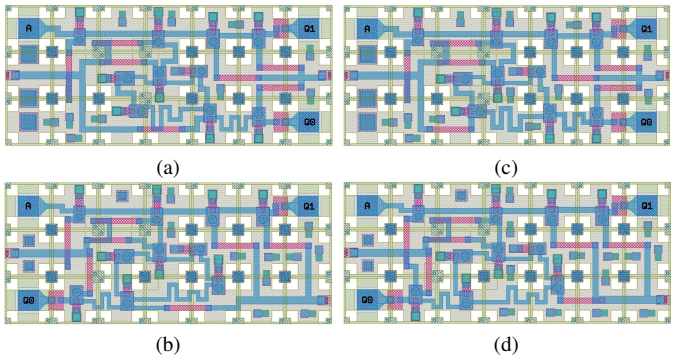


Fig. 5. The layouts of TFFDO/E cells, designed using open-source KLayout tool [19] in accordance with the SFQ5ee process [23] design rules, has dimensions of $70\,\mu\text{m} \times 30\,\mu\text{m}$. (a) and (b) show two layouts of TFFDO; (c) and (d) show two layouts of TFFDE.

the clock pulses propagate column by column from the PI, the delay of the clock pulse from the PI to g_2 must also be considered. The PTL length corresponding to the data delay can be analyzed similarly by analogy to the clock delay. Therefore, the timing constraint in Equation (2) can be further transformed into the length-matching constraint:

$$\begin{cases} l_v'^c + l_v^{vc} + l_v^{mc} + l^{cell} + l^{hold} < \min\limits_{u \in \mathrm{FI}(v)} \left(l_u'^d + l_{u,v}^{vd} + l_{u,v}^{md} + l^{cell} \right), \\ l_v'^c + l_v^{vc} + l_v^{mc} + l^{cell} + l^{T} - l^{setup} > \max\limits_{u \in \mathrm{FI}(v)} \left(l_u'^d + l_{u,v}^{vd} + l_{u,v}^{md} + l^{cell} \right), \end{cases} \tag{3}$$

where $l_v'^c$ denotes the PTL length corresponding to the clock pulse delay from the PI to the clock source of node v in the previous column, while $l_u'^d$ denotes the PTL length corresponding to the data pulse delay from the PI to node u. $l_{u,v}^{vd}$ and $l_{u,v}^{md}$ are the vertical minimum length and vertical matching length of the data connection between u and v.

Thus, the placement for multiphase clocking RSFQ circuits can be formulated as follows:

- **Input**:
  1) A path balanced multiphase clocking RSFQ circuit $N(V, E)$.
  2) An RSFQ cell library, including physical and timing parameters of logic gates, TFFDO, TFFDE, and PTLs.
- **Output**:
  1) A legal y_v for $\forall v \in V$.
  2) The multiphase clock distribution of the circuit.
- **Constraints**:
  1) Length-matching constraints defined by Equation (3).
  2) Overlap constraints: For two nodes u and v within the same column, $y_u + h_u \leq y_v$ if $y_u \leq y_v$.
  3) For every node $v \in V$, $y_v \geq 0$ and $y_v + h_v \leq H$, where H is the maximum height among all columns.
- **Goal**:
  Minimize the total vertical wirelength (*TVWL*) of the placement, which is the sum of the vertical minimum length and the vertical matching length, formulated as:

$$TVWL = \sum_{v \in V} \left[l_v^{vc} + l_v^{mc} + \sum_{u \in \mathrm{FI}(v)} \left(l_{u,v}^{vd} + l_{u,v}^{md} \right) \right]. \tag{4}$$

IV. J2Place

To solve the above RSFQ placement problem, we propose a multiphase clocking-oriented RSFQ placement framework consisting of four steps: 1) initial placement; 2) maximum flow-based clock distribution generation; 3) DP-based column-wise placement; and 4) length-aware reordering. During the initial placement phase, all gates are assigned to columns according to their stages, with dummies

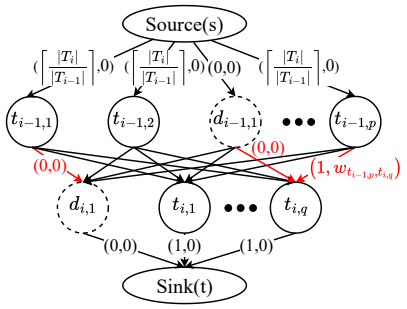


Fig. 6. Construction of a capacity-constrained flow graph for i-th column to generate clock distribution.

inserted as necessary. They are given random initial positions. The following subsections will detail the subsequent steps.

A. Maximum flow-based Clock Distribution Generation

In our placement, each logic gate g and the dummy d^c for clock propagation within a column are driven by clock pulses from the previous column. These clock pulses effectively serve as the clock sources for all cells in the current gate column, enabling a column-by-column construction of the clock distribution network. The location of the clock source not only determines the vertical wirelength of the clock connections for g and d^c, but also affects the vertical matching length required for the data connection of g to meet the length-matching constraint. Furthermore, to prevent routing congestion caused by large splitter trees, it is preferable to maintain a balanced fanout for each clock source. Therefore, we proposed a maximum-flow-based clock distribution method to minimize the wirelength and the number of splitters in the clock network.

The maximum flow-based method constructs a capacity-constrained flow graph for each column. Fig. 6 presents the graph of the i-th column, which contains nodes $\{s,t\} \cup V_{i-1} \cup V_i$. For nodes within the i-th column, the set of clocked nodes is denoted as $T_i = \{t_{i,1}, t_{i,2} \dots t_{i,q} \dots t_{i,|T_i|}\}$, while the set of dummies for data propagation is denoted as $D_i = \{d_{i,1} \dots d_{i,|D_i|}\}$. The capacity from the source node s to each node $t_{i-1,p} \in T_{i-1}$ is $\lceil |T_i|/|T_{i-1}| \rceil$ with a cost of 0, indicating the maximum fanout of the clock pulse sourced from $t_{i-1,p}$. For node $t_{i-1,p} \in T_{i-1}$, the capacity from $t_{i-1,p}$ to each node $t_{i,q} \in T_i$ is 1 and the cost $w(t_{i-1,p}, t_{i,q})$ represents the wirelength of the clock connection and the vertical matching length of the data connection of $t_{i,q}$ when its input clock pulse is sourced from $t_{i-1,p}$. If $t_{i-1,p}$ is a logic gate, the cost $w(t_{i-1,p}, t_{i,q}) = l^{vc}_{t_{i,q}} + l^{mc}_{t_{i,q}} + \sum_{u \in \mathrm{FI}(t_{i,q})} l^{md}_{u,t_{i,q}}$. If $t_{i-1,p}$ is a dummy for clock propagation, the cost $w(t_{i-1,p}, t_{i,q}) = l^{vc}_{t_{i,q}} + l^{mc}_{t_{i,q}}$. For each node $t_{i,q} \in T_i$, the capacity from $t_{i,q}$ to the sink t is 1 with a cost of 0, indicating that $t_{i,q}$ requires exactly one input clock pulse. Since dummies for data propagation cannot serve as clock sources and do not require input clock pulses, the capacity and cost of any edges flowing into or out of these dummies are zero. To achieve high efficiency, we adopt the successive shortest path method [25] to obtain the maximum flow with the minimum cost, which determines the optimal clock distribution for the logic gate within the i-th column. In this maximum flow, if the flow from the node $t_{i-1,p}$ to $t_{i,q}$ is greater than 0, it indicates that the input clock pulse of $t_{i,q}$ originates from $t_{i-1,p}$. Moreover, the flow volume of this maximum flow must be T_i, meaning that each node $t_{i,q} \in T_i$ has been assigned a clock source.

B. DP-based Column-wise Placement

After clock distribution generation, the objective is to determine the optimal position of each logic gate within the column to minimize the *TVWL*, while allocating space for the dummies representing the cross-column connections. In a two-phase clocking RSFQ circuit with C columns, there are no less than $\prod_i^C |V_i|!$ possible arrangements just by adjusting the order of the nodes within each column. Since clock pulses propagate column by column from the PI, the vertical wirelength of node v's clock and data connections depends solely on the positions of its clock source and data inputs in the previous column, independent of subsequent columns. This implies that we can decompose the global placement optimization problem into column-wise independent optimization problems, where the placement of each column depends only on the placement result of the previous column. During the column-wise placement, when placing the nodes within the i-th column, the positions of all the nodes in the previous column have been determined and remain fixed.

Considering that nodes in the i-th column have at least $|V_i|!$ possible arrangements, finding the optimal placement for this column remains a problem with high computational complexity, particularly for large-scale circuits. To address this issue, we introduce the fixed-order constraint and a predefined hyperparameter λ to restrict the possible positions of nodes. For nodes in the i-th column, they are sorted in ascending order based on y_v, denoted as $V_i' = \{v_1, v_2 \dots v_k \dots v_{|V_i|}\}$. When updating the positions of nodes within the i-th column, the relative order is fixed and the vertical position range for node v_k is set as $[y_{v_k} - \lambda, y_{v_k} + \lambda]$. Additionally, to satisfy the overlap constraint, the position of each node must leave sufficient space for other nodes in the same column within the range $[0, H]$. This constraint can be expressed as:

$$\forall v_k \in V_i', \sum_{j=1}^{k-1} h_{v_j} \le y_{v_k} \le H - \sum_{j=k}^{|V_i'|} h_{v_j}. \tag{5}$$

Thus, the candidate positions Y_k for v_k can be expressed as:

$$Y_k = \{y_{k,p} \mid y_{k,p} \in [y_{v_k} - \lambda, y_{v_k} + \lambda] \cap [\sum_{j=1}^{k-1} h_{v_j}, H - \sum_{j=k}^{|V_i'|} h_{v_j}]\}, \tag{6}$$

where $p = 1, 2, \dots, |Y_k|$. Assigning $y_{k,p}$ to node v_k ensures feasible placement space for other nodes in the same column while maintaining the fixed-order constraint. Additionally, according to Equation (4), the contribution of node v to *TVWL* can be expressed as follows:

$$l_v^{sum} = \begin{cases} l_v^{vc} + l_v^{mc} + \sum\limits_{u \in \mathrm{FI}(v)} \left(l_{u,v}^{vd} + l_{u,v}^{md} \right), & v \in G \\ l_v^{vc} + l_v^{mc}, & v \in D^c \\ \sum\limits_{u \in \mathrm{FI}(v)} \left(l_{u,v}^{vd} + l_{u,v}^{md} \right). & v \in D^d \end{cases} \tag{7}$$

The impact of v_k on *TVWL* depends solely on nodes $u \in \mathrm{FI}(v_k)$ and is independent of other nodes in the same column. This property allows us to reformulate the optimal placement problem for the i-th column as a shortest path problem. To address this problem effectively, we propose a DP-based algorithm to identify the path with the lowest cost in the graph, thereby minimizing *TVWL*.

The DP-based algorithm constructs a weighted directed graph for each column. As shown in Fig. 7, the graph of column i contains vertices $\{s,t\} \cap Y_1 \cap \dots \cap Y_{|V_i'|}$. The source vertex s connects to all vertices in Y_1, while all vertices in $Y_{|V_i'|}$ connect to the sink vertex t. The weight of the edges connected to t is zero, *i.e.*, $w(t) = 0$.

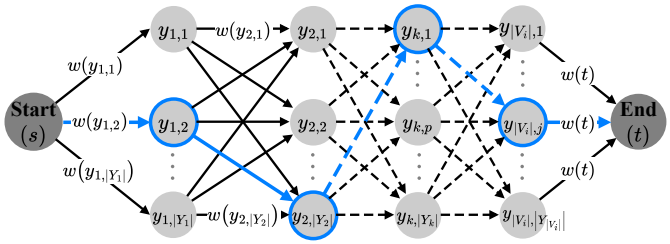


Fig. 7. Trellis for DP-based placement of the i-th column, where the blue line highlights the optimal placement solution.

For a vertex $y_{k,p} \in Y_k$, it connects to a vertex $y_{k+1,q} \in Y_{k+1}$ if and only if $y_{k,p} + h_{v_k} \leq y_{k+1,q}$, ensuring no overlap between nodes. Each edge connected to $y_{k,p}$ is assigned a weight $w(y_{k,p})$, which represents the cost of placing v_k at $y_{k,p}$. For columns consisting of gates and dummies, the edge weight is defined as $w(y_{k,p}) = l^{\text{sum}}_{v_k}(y_{k,p})$, where $l^{\text{sum}}_{v_k}(y_{k,p})$ represents the total wirelength of node v_k to *TVWL* when placed at $y_{k,p}$. For the column of PIs, the weight is calculated as $w(y_{k,p}) = \sum_{u \in \text{FO}(v_k)} l^{vd}_{v_k,u}$, which is the sum of the vertical minimum lengths of its outputs. The path with the minimum sum of edge weights from s to t determines the optimal placement positions for each node $v_k \in V'_i$. To find this path, we propose the following state transition model of our DP-based algorithm:

$$dp[k][y_{k,p}] = \begin{cases} \min\limits_{y < y_{k,p} - h_{v_{k-1}}} \left(dp[k-1][y] + w(y_{k,p}), dp[k][y_{k,p}]\right), & k > 1 \\ w(y_{k,p}), & k = 1 \end{cases}, \quad (8)$$

where the array dp stores the partial weight sums. For $k = 1$, $dp[k][y_{k,p}]$ is initialized as $w(y_{k,p})$. For $k > 1$, $dp[k][y_{k,p}]$ stores the minimum weight subsum for placing nodes v_1 to v_k with v_k at $y_{k,p}$. The minimum value in $dp[|V'_i|]$ records the path with the minimum sum of edge weights. The vertices along this path correspond to the optimal positions of each node. By backtracking the dp array, the algorithm determines the optimal placement for the i-th column.

C. Length-aware Reordering

We propose a length-aware reordering algorithm to overcome the limitations imposed by fixed node ordering in DP-based column-wise placement by dynamically adjusting the order of nodes within each column to minimize the *TVWL*. The main idea is to place each node v closer to its data input nodes $u \in \text{FI}(v)$ that have longer data connection wirelengths, thereby reducing the dominant component of *TVWL* associated with data connections after clock regeneration. As shown in Fig. 8, by placing g_3 closer to g_2 with the longest wirelength of the data connection, we can obtain a placement result with reduced *TVWL*.

This adjustment is performed by weighting the influence of each input node u on the position y_v of node v according to the sum of the data connection wirelength, as described in Equation (7). Considering the gate-level pipelining, we use the PTL length corresponding to the data delay from the PI through u to v to represent the weight of node u. So, the updated position y_v for v can be calculated as follows:

$$y_v = \begin{cases} \dfrac{\sum_{u \in \text{FO}(v)} l^{vd}_{v,u} \cdot y_u}{\sum_{u \in \text{FO}(v)} l^{vd}_{v,u}}, & v \in I \\ \dfrac{\sum_{u \in \text{FI}(v)} (l'^{d}_{u} + l^{vd}_{u,v}) \cdot y_u}{\sum_{u \in \text{FI}(v)} (l'^{d}_{u} + l^{vd}_{u,v})}, & v \in G \end{cases}. \quad (9)$$

For node $v \in I$, the position update is based on its output connections instead. Since dummies have a single input, their positions remain unchanged. By iteratively updating node positions and deriving new node orders for each column, the algorithm effectively expands the solution space beyond fixed ordering, leading to a significant reduction in *TVWL* and improved placement quality.

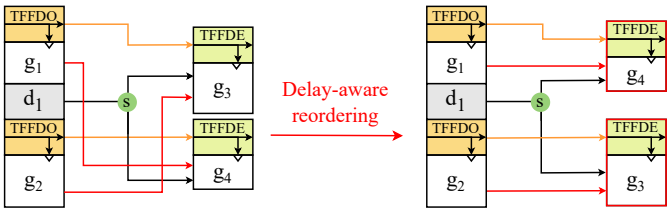


Fig. 8. Length-aware reordering for the column-wise placement. After reordering, g_3 and g_4 are positioned closer to g_2 and g_1, respectively, which obtain a placement with reduced *TVWL*.

Algorithm 1: J2Place algorithm.

Input: A path-balanced netlist $N(V, E)$, iteration I.
Output: A corresponding legal placement.

1 $C \leftarrow$ assign gates to C gate columns based on their stages.
2 Insert dummies to replace the PTLs crossing the gate columns.
3 Randomly initialize the positions of nodes within each column.
4 **for** $i \in [1, I]$ **do**
5 Place the primary inputs.
6 **for** *gate column* $c \in C$ **do**
7 Generate the clock distribution for column c.
8 Perform DP-based placement for column c.
9 Reorder the gates and dummies within each column.

D. The Flow of J2Place and Time Complexity Analysis

Algorithm 1 outlines the complete procedure of the multiphase clocking-oriented length-matching placement for RSFQ circuits, where I denotes the total number of iterations. Initially, gates are assigned to columns according to their stages (line 1). Dummies are then inserted to replace PTLs crossing the gate columns (line 2), followed by random initialization of all node positions (line 3). The algorithm proceeds with an iterative refinement process (lines 4-9). In each iteration, PIs are placed first (line 5). Next, the algorithm performs a column-wise placement update (lines 6-8), which involves two key steps: computing the optimal clock distribution for each column using a maximum flow algorithm (line 7), and minimizing the *TVWL* through DP-based placement that considers data and clock connections (line 8). After all columns are processed, nodes within each column are reordered to further optimize the placement (line 9). By iteratively applying these steps, J2Place completes multiphase clocking RSFQ placement column by column.

The analysis of the time complexity of our placement algorithm is as follows. During the initial placement, assigning a random position to each node requires $O(|V|)$ time. Determining the optimal clock distribution for each column by the maximum flow-based algorithm takes $O(|V|^2)$ time. During the DP-based column-wise placement phase, according to Equation (8), the time complexity of DP for each column of nodes is $O(|V| \cdot \lambda^2)$. Subsequently, the length-aware reordering updates the positions of nodes based on their inputs. Since all nodes have at most three inputs, the time complexity of this step is linear. Additionally, sorting after updating the node positions takes $O(|V| \log |V|)$ time. Overall, the time complexity of one iteration of our placement algorithm is $O(|V|^2 + \lambda^2|V| + |V| \log |V|)$. Since λ is a small constant (set to 50 in our experiment), the overall

TABLE I. DFF insertion of multiphase and single-phase clocking.

Circuit	Gates	Stages	DFF	
			Single-phase	Two-phase
c432	259	27	756	344
c499	195	12	409	179
c880	345	28	1031	470
c1355	205	12	430	190
c1908	208	22	941	381
c3540	1063	33	1216	507
c5315	1411	29	4092	1880
c6288	1789	76	2976	1304
c7552	1355	48	6216	2920
adder16	96	33	754	360
int2float	248	16	246	103
sin	5269	169	11247	5149
priority	885	215	16893	8319
multiplier	21550	257	55988	26909
max	4193	207	79740	39563
Ave.ratio	/	/	2.21	1

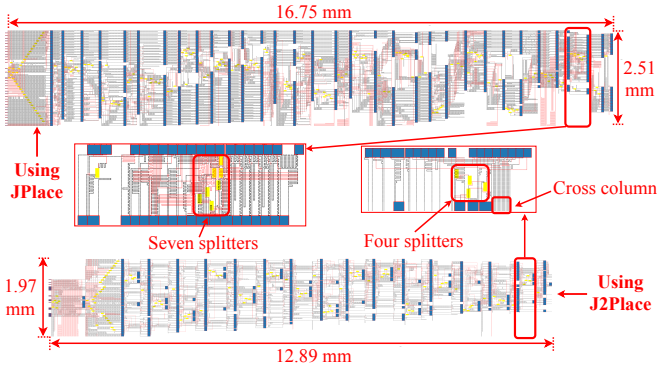


Fig. 9. Routed layout of the adder16 circuit. The layout generated by JPlace has an area of $16.75\,\mathrm{mm} \times 2.51\,\mathrm{mm}$ ($42.04\,\mathrm{mm}^2$), while our placement achieves a layout of $12.89\,\mathrm{mm} \times 1.97\,\mathrm{mm}$ ($25.39\,\mathrm{mm}^2$).

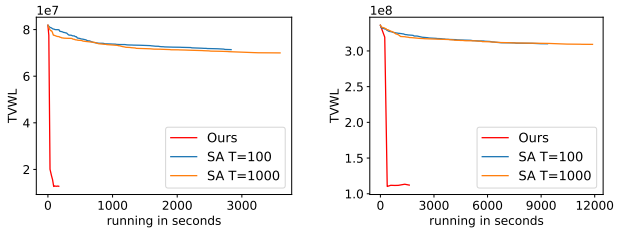
time complexity can be simplified to $O(|V|^2)$. Considering that the algorithm runs for I iterations (set to 100 in our experiment), the total time complexity of our algorithm is $O(I \cdot |V|^2)$.

V. Experimental Results

J2Place was implemented in Python and evaluated on an Intel(R) Xeon(R) Gold 5218R CPU with 128 GB of memory. The evaluation utilized combinational benchmark circuits from ISCAS85 [26] and EPFL [27]. Gate-level netlists were synthesized using the ABC tool [28] with the "resyn2" and "map" scripts, followed by path balancing through the DFF insertion according to the method in [8], [29]. Gate sizes and timing parameters are obtained from the RSFQ cell library [22]. Besides, all gates in the circuit are composite gates, with a uniform layout width of $70\,\mu\mathrm{m}$ after composition.

TABLE I summarizes the DFF insertion results under single-phase and two-phase clocking schemes. The "Gates" column represents the number of logic gates excluding DFFs, while "Stages" denotes the number of circuit stages. The "Single-phase" and "Two-phase" columns display the number of inserted DFFs for each clocking scheme. Results show that two-phase clocking can effectively reduce the number of inserted DFFs, with an average reduction of 54.65%.

We first compared our method with JPlace [12], the state-of-the-art single-phase RSFQ placement algorithm. Our method and JPlace were executed using parameters $I = 100$ and $\lambda = 50$, with automatic termination after five consecutive iterations without improvement. TABLE II presents the placement quality and runtime for our method



(a) sin (b) multiplier

Fig. 10. *TVWL* changes over time for (a) sin and (b) multiplier circuits, comparing J2Place with the SA [10] at two initial temperatures.

and JPlace, where "H" represents layout height. Compared with JPlace, our placement algorithm achieves average reductions of 50.09% in *TVWL* and 19.47% in layout height. Fig. 9 illustrates the routed layout of the adder16 circuit placed by JPlace and J2Place, with routing performed by JRouter [30]. The placement generated by our algorithm results in an area of $25.39\,\mathrm{mm}^2$, compared to $42.04\,\mathrm{mm}^2$ using JPlace, demonstrating a significant area reduction.

To further demonstrate the effectiveness of our placement algorithm, we compared it with an improved SA-based method [10] for two-phase clocking. In this SA-based approach for two-phase clocking, gates within the same column were paired, and multiple pairs were adjusted simultaneously during each placement perturbation to enhance efficiency. The objective metric was set to minimize *TVWL*. The SA parameters were configured as follows: the number of iterations was set to 10, the cooling ratio was 0.95, and the termination criterion was set to a temperature threshold of 0.01. Two initial annealing temperatures were performed, $T = 100$ and $T = 1000$, respectively. Placement quality and runtime results are shown in TABLE II. Overall, our placement algorithm achieved average reductions of 57.12% and 56.24% in *TVWL*, alongside runtime improvements of 62.01% and 69.91%, respectively, compared to the SA-based method. Moreover, Fig. 10 shows the placement processes of our placement algorithm and the SA-based methods on the sin and multiplier circuits from the EPFL benchmark [27]. The results clearly demonstrate that, compared with the SA-based methods, J2Place converges more rapidly and consistently produces superior placement solutions with smaller *TVWL* than the SA-based approaches.

VI. Conclusion

This paper proposed a length-matching placement method for multiphase clocking RSFQ circuits, addressing the complex timing constraints inherent in such designs. For this purpose, we first introduced two novel RSFQ cells, TFFDO and TFFDE, to simplify the clock network in two-phase clocking designs. A maximum flow-based method was employed to effectively minimize the clock network wirelength. Next, a fine-grained dynamic programming method was adopted to minimize the vertical wirelength, complemented by a length-aware reordering technique for further wirelength optimization. Experimental results demonstrated that our method significantly improves wirelength, layout height, and runtime.

Acknowledgment

This work was conducted in the JC STEM Lab of Intelligent Design Automation funded by The Hong Kong Jockey Club Charities Trust and was supported in part by the Research Grants Council of Hong Kong SAR (Grant No. CUHK14207523); in part by the National Natural Science Foundation of China (Grant No. 62302477); in part by the State Key Lab of Processors, Institute of Computing

TABLE II. Comparison between our placement algorithm and the baselines.

Circuit	JPlace [12]			SA [10] for two-phase clocking						Ours		
				T=100			T=1000					
	H(μm)	Time(s)	TVWL	H(μm)	Time(s)	TVWL	H(μm)	Time(s)	TVWL	H(μm)	Time(s)	TVWL
c432	5030	6.156	436435	3590	22.215	309669	3590	27.036	307835	3590	3.062	203452
c499	4830	3.406	251154	3750	12.324	214541	3750	15.319	220881	3750	2.546	178757
c880	5910	8.688	601397	4980	37.195	571998	4980	45.483	557227	4980	3.828	228088
c1355	5200	3.219	272560	4000	13.562	250152	4000	16.331	235648	4000	1.609	172428
c1908	5800	6.891	423619	4270	26.749	417401	4270	33.095	400304	4270	2.046	224394
c3540	20480	22.828	2147667	17670	165.684	3877517	17670	205.06	3812148	17670	15.734	1082770
c5315	29540	144.531	5843549	25560	537.546	12452730	25560	665.788	12000114	25560	56.843	2876765
c6288	6510	16.812	4404936	5250	120.946	2481759	5250	151.875	2406573	5250	23.203	1472803
c7552	23080	89	4677287	17450	531.173	11014361	17450	660.422	11017244	17450	117.234	2263693
adder16	2510	0.656	185928	1970	14.962	198571	1970	18.588	198770	1970	1.14	71262
int2float	5580	3.844	201678	4160	10.9813	166755	4160	13.68	154636	4160	0.5	76393
sin	47790	520.593	25240196	40900	2833.752	71398678	40900	3591.3514	69990118	40900	2788.906	20457036
priority	9390	280.921	11050073	7830	740.181	10489735	7830	918.335	10387312	7830	67.469	3304702
multiplier	47950	503.468	521097358	35330	9358.752	310057704	35330	11877.98	309206553	35330	1853.312	123515601
max	51870	655.187	77790331	51590	17332.36	346390287	51590	22083.509	347976657	51590	52834.453	65660287
Ave. ratio	1.25	1.96	2.26	1.00	8.37	2.83	1.00	10.36	2.78	1.00	1.00	1.00

Technology, CAS (Grant No. CLQ202402); and in part by the Shandong Province Natural Science Foundation, China (Grant No. ZR2024MF073).

REFERENCES

[1] K. Likharev and V. Semenov, "RSFQ logic/memory family: a new josephson-junction technology for sub-terahertz-clock-frequency digital systems," *IEEE Transactions on Applied Superconductivity*, vol. 1, no. 1, pp. 3–28, 1991.

[2] T. Van Duzer, C. Turner, D. McDonald, and A. F. Clark, "Principles of superconductive devices and circuits," *Physics Today*, vol. 35, no. 2, p. 80, 1982.

[3] D. S. Holmes, A. L. Ripple, and M. A. Manheimer, "Energy-efficient superconducting computing—power budgets and requirements," *IEEE Transactions on Applied Superconductivity*, vol. 23, no. 3, pp. 1 701 610–1 701 610, 2013.

[4] J. Choi, I. Byun, J. Hong, D. Min, J. Kim, J. Cho, H. Jeong, M. Tanaka, K. Inoue, and J. Kim, "SuperCore: An ultra-fast superconducting processor for cryogenic applications," in *IEEE/ACM International Symposium on Microarchitecture (MICRO)*, 2024, pp. 1532–1547.

[5] M. R. Jokar, R. Rines, G. Pasandi, H. Cong, A. Holmes, Y. Shi, M. Pedram, and F. T. Chong, "Digiq: A scalable digital controller for quantum computers using SFQ logic," in *2022 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*. IEEE, 2022, pp. 400–414.

[6] K. Ishida, I. Byun, I. Nagaoka, K. Fukumitsu, M. Tanaka, S. Kawakami, T. Tanimoto, T. Ono, J. Kim, and K. Inoue, "SuperNPU: An extremely fast neural processing unit using superconducting logic devices," in *IEEE/ACM International Symposium on Microarchitecture (MICRO)*, 2020, pp. 58–72.

[7] G. Pasandi and M. Pedram, "An efficient pipelined architecture for superconducting single flux quantum logic circuits utilizing dual clocks," *IEEE Transactions on Applied Superconductivity*, vol. 30, no. 2, pp. 1–12, 2019.

[8] R. Bairamkulov, M. Yu, and G. De Micheli, "Unleashing the power of T1-cells in SFQ arithmetic circuits," in *ACM/IEEE Design Automation Conference (DAC)*, 2024.

[9] X. Li, M. Pan, T. Liu, and P. A. Beerel, "Multi-phase clocking for multi-threaded gate-level-pipelined superconductive logic," in *IEEE Annual Symposium on VLSI (ISVLSI)*, 2022, pp. 62–67.

[10] N. Kito, K. Takagi, and N. Takagi, "A fast wire-routing method and an automatic layout tool for RSFQ digital circuits considering wire-length matching," *IEEE Transactions on Applied Superconductivity*, vol. 28, no. 4, pp. 1–5, 2018.

[11] C.-C. Wang and W.-K. Mak, "A novel clock tree aware placement methodology for single flux quantum logic circuits with maximum path length consideration," *IEEE Transactions on Applied Superconductivity*, vol. 32, no. 5, pp. 1–12, 2022.

[12] S. Chen, R. Fu, J. Huang, Z. Zhang, X. Ye, T.-Y. Ho, and D. Fan, "JPlace: A clock-aware length-matching placement for rapid single-flux-quantum circuits," in *IEEE/ACM Proceedings Design, Automation and Test in Eurpoe (DATE)*, 2024, pp. 1–6.

[13] K. Gaj, E. G. Friedman, and M. J. Feldman, "Timing of multi-gigahertz rapid single flux quantum digital circuits," *Journal of VLSI signal processing systems for signal, image and video technology*, vol. 16, no. 2, pp. 247–276, 1997.

[14] J.-T. Yan, "Tree-based clock distribution of multiple-stage pipelined architecture in rapid single-flux-quantum circuits," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, vol. 41, no. 4, pp. 1090–1102, 2022.

[15] C. J. Fourie, "Extraction of DC-biased SFQ circuit verilog models," *IEEE Transactions on Applied Superconductivity*, vol. 28, no. 6, pp. 1–11, 2018.

[16] sunmagnetics and CoenradFourie, "Tzzzzzzzz/TimEx," https://github.com/Tzzzzzzzz/TimEx, accessed: April 01, 2025.

[17] "JoSIM Documentation," https://joeydelp.github.io/JoSIM/#building-from-source, accessed: April 01, 2025.

[18] J. Delport, "JoeyDelp/JoSIM," https://github.com/JoeyDelp/JoSIM, accessed: April 01, 2025.

[19] "KLayout Layout Viewer And Editor," https://www.klayout.de/, accessed: April 01, 2025.

[20] "Welcome to the SPiRA Framework! — SPiRA Auron [Beta] documentation," https://spira.readthedocs.io/en/latest/, accessed: April 01, 2025.

[21] R. v. Staden, "rubenvanstaden/spira," https://github.com/rubenvanstaden/spira, accessed: April 01, 2025.

[22] L. Schindler and T. Hall, "RSFQ cell library," https://github.com/sunmagnetics/RSFQlib, version: 3.0, Release date: 21 March 2023.

[23] S. K. Tolpygo, V. Bolkhovsky, T. J. Weir, A. Wynn, D. E. Oates, L. M. Johnson, and M. A. Gouker, "Advanced fabrication processes for superconducting very large-scale integrated circuits," *IEEE Transactions on Applied Superconductivity*, vol. 26, no. 3, pp. 1–10, 2016.

[24] L. Schindler, P. le Roux, and C. J. Fourie, "Impedance matching of passive transmission line receivers to improve reflections between rsfq logic cells," *IEEE Transactions on Applied Superconductivity*, vol. 30, no. 2, pp. 1–7, 2020.

[25] B. R. Treat and J. R. Mote, "On successive shortest path algorithms for network flow models," Ph.D. dissertation, The University of Texas at Austin, 1993, aAI9323571.

[26] M. C. Hansen, H. Yalcin, and J. P. Hayes, "Unveiling the ISCAS-85 benchmarks: A case study in reverse engineering," *IEEE Design & Test*, vol. 16, no. 3, pp. 72–80, 1999.

[27] L. Amarù, P.-E. Gaillardon, and G. De Micheli, "The EPFL combinational benchmark suite," in *IEEE/ACM International Workshop on Logic Synthesis*, 2015.

[28] R. Brayton and A. Mishchenko, "ABC: An academic industrial-strength verification tool," in *International Conference on Computer-Aided Verification (CAV)*. Springer-Verlag, 2010, pp. 24–40.

[29] M. Zhou, R. Fu, R. Zhang, X. Ye, T.-Y. Ho, and J. Huang, "An optimal DFF-oriented technology legalization algorithm for rapid single-flux-quantum circuits," in *ACM Great Lakes Symposium on VLSI (GLSVLSI)*, 2025.

[30] X. Chen, R. Fu, J. Huang, H. Cao, Z. Zhang, X. Ye, T.-Y. Ho, and D. Fan, "JRouter: A multi-terminal hierarchical length-matching router under planar manhattan routing model for RSFQ circuits," in *ACM Great Lakes Symposium on VLSI (GLSVLSI)*, 2023, pp. 515–520.