An Optimal DFF-Oriented Technology Legalization Algorithm for Rapid Single-Flux-Quantum Circuits

Minglei Zhou
SKLP, Institute of Computing Technology, CAS
School of Computer Science and Technology, University of Chinese Academy of Sciences
Beijing, China
zhouminglei24s@ict.ac.cn

Rongliang Fu
Department of Computer Science and Engineering, The Chinese University of Hong Kong
Hong Kong, China
rlfu@cse.cuhk.edu.hk

Ran Zhang
SKLP, Institute of Computing Technology, CAS
School of Computer Science and Technology, University of Chinese Academy of Sciences
Beijing, China
zhangran23s@ict.ac.cn

Xiaochun Ye
SKLP, Institute of Computing Technology, CAS
Beijing, China
yexiaochun@ict.ac.cn

Tsung-Yi Ho
Department of Computer Science and Engineering, The Chinese University of Hong Kong
Hong Kong, China
tyho@cse.cuhk.edu.hk

Junying Huang*
SKLP, Institute of Computing Technology, CAS
Beijing, China
huangjunying@ict.ac.cn

Abstract

Superconducting rapid single-flux-quantum (RSFQ) logic has garnered considerable attention as a prospective technology for future computing systems, thanks to its superior high-speed and low-power characteristics. However, conventional semiconductor logic synthesis tools can not guarantee the functional correctness of the generated RSFQ circuits. RSFQ circuits require DFF and splitter insertion to satisfy the path balancing requirement and the fanout limitation, thereby legitimizing the circuit design. Furthermore, DFFs and splitters inserted in RSFQ circuits introduce delays, occupy area, and substantially increase energy dissipation. To address this problem, this paper proposes an optimal DFF-oriented technology legalization algorithm. First, an integer linear programming algorithm is proposed for the logic level assignment to minimize the number of inserted DFFs. Then a splitter tree is constructed for each net of the circuit to minimize the timing discrepancies among the various fanouts. The experimental results on ISCAS'85 and EPFL benchmarks demonstrate the effectiveness and efficiency of our proposed algorithm compared with the state-of-the-art, particularly with significant advantages on large circuits.

CCS Concepts

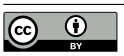
• **Hardware → Emerging technologies**.

Keywords

RSFQ, Superconducting logic circuits, DFF and splitter insertion, Integer linear programming

ACM Reference Format:
Minglei Zhou, Rongliang Fu, Ran Zhang, Xiaochun Ye, Tsung-Yi Ho, and Junying Huang. 2025. An Optimal DFF-Oriented Technology Legalization Algorithm for Rapid Single-Flux-Quantum Circuits . In *Great Lakes Symposium on VLSI 2025 (GLSVLSI '25), June 30-July 2, 2025, New Orleans, LA, USA.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3716368.3735163

*Corresponding author: huangjunying@ict.ac.cn



This work is licensed under a Creative Commons Attribution 4.0 International License.
GLSVLSI '25, June 30-July 2, 2025, New Orleans, LA, USA
© 2025 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-1496-2/2025/06
https://doi.org/10.1145/3716368.3735163

1 Introduction

Josephson junction (JJ) based superconducting rapid single-flux-quantum (RSFQ) technology [19] has become a popular emerging technology, potentially offering high-performance digital systems at low power. The Josephson junction exhibits the Josephson effect at approximately 4 K, demonstrating rapid switching speeds and low switching energy. Previous research has demonstrated the potential of RSFQ technology for high-performance operation (tens of GHz) in large-scale circuits, such as microprocessors [2, 17] and hardware accelerators for machine learning [8, 15, 16]. However, the pulse-based logic, different active and passive components, certain interconnect structures, and tens of gigahertz clock frequencies present unique challenges during the RSFQ circuit design process.

Among these challenges, two critical issues stand out as particularly demanding: the clock-synchronized data propagation requirement and the fanout limitation. First, nearly all RSFQ logic gates operate in synchronization with the clock to achieve clock-driven gate-level pipelining. To ensure the correct operation of RSFQ logic gates, all input signals to a gate must arrive within the same clock cycle. This necessitates the insertion of Delay flip-flops (DFFs) to equalize path lengths from primary inputs (PIs) to each gate. Second, unlike CMOS gates, most RSFQ logic gates have a limited fanout of one. Therefore, splitters are required to distribute signals to multiple destinations. Consequently, DFF and splitter (D/S) insertion is a critical step in RSFQ circuit design, and their legal implementation is essential to guarantee the correct functionality of the RSFQ circuit.

However, DFFs and splitters occupy a significant portion of RSFQ circuits, seriously affecting the area, delay, and energy consumption of RSFQ circuits. Although numerous studies [9, 10, 14] have addressed this issue in AQFP circuits, a different superconducting technology, these methods are not applicable to RSFQ circuits. This is because splitters in AQFP logic require clocking, whereas those in RSFQ logic do not, resulting in fundamentally different constraints. Additionally, Katam et al. [18], and Fu et al. [11] used ABC [4], an open-source logic synthesis and verification tool, to generate an intermediate circuit, followed by the insertion of DFFs and splitters using a retiming-like approach to satisfy path balancing and

fanout constraints. Ghasem et al. [20] proposed a path balancing technology mapping algorithm, PBMap, for RSFQ circuits, which can provide an optimal solution of DFF insertion for RSFQ circuits with a tree-like structure. However, this algorithm is not optimal for circuits with a general directed acyclic graph (DAG) structure. Rassul et al. [3] proposed a two-stage SFQ technology mapping method that supports the T1 cell and uses multi-phase clocking to meet the timing requirements for the input signals of the T1 cell. The increased complexity of clock distribution due to multi-phase clocking makes the physical design of RSFQ circuits more challenging. Furthermore, in the current RSFQ physical design, the splitter insertion is typically associated with the optimization during placement [6] and routing [7] stages. So, technology legalization for RSFQ circuits should focus on DFF insertion.

This paper focuses on minimizing the number of inserted DFFs while ensuring the correct functionality of RSFQ circuits from a global circuit optimization perspective. It provides a comprehensive analysis of the operational mechanisms of RSFQ logic and explains the reasons behind the insertion of DFFs and splitters. To address this problem, we propose an optimal DFF-oriented technology legalization algorithm. Specifically, we make the following contributions:

- We propose an integer linear programming-based algorithm to minimize the number of inserted DFFs by minimizing the maximum logic level gap between the source and sinks of each net.
- We construct a splitter tree for each net in the RSFQ circuit with the minimum splitters to minimize the timing discrepancies among the various fanouts.
- The proposed algorithm achieves an average reduction of 44.05% in the number of inserted DFFs and 38.41% in the number of JJs on the ISCAS'85[5] and EPFL[1] benchmarks, respectively, compared to GLSVLSI'20[11]. Furthermore, it reduces the number of inserted DFFs by 11.88% and the circuit depth by 19.01% on average compared to PBMap[20].

2 Background

2.1 Rapid Single-Flux-Quantum Circuits

RSFQ circuits employ JJ as their active device for high-speed digital signal processing. Information is stored in the form of magnetic flux quantum and transferred in the form of single-flux-quantum (SFQ) voltage pulses. The SFQ within the JJ serves as the fundamental information carrier, similar to the voltage level in traditional CMOS circuits. The logical states of '1' and '0' are represented by the presence or absence of an SFQ pulse, respectively.

In RSFQ logic, clock pulses are essential for transferring stored SFQs between adjacent gates. RSFQ gates are clocked and require the clock signal to trigger the transfer of their stored SFQ pulse to their output. This inherent sequential nature of RSFQ gates enables natural gate-level pipelining, leading to significant differences in clocking methodologies compared to CMOS [12]. To further clarify the operational principles and design constraints of RSFQ circuits, we will introduce a representative RSFQ logic gate, D flip-flop (DFF), and discuss two critical aspects of RSFQ circuit design: the fanout limitation and the path balancing requirement.

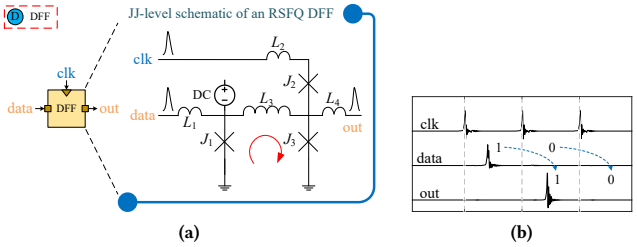


Figure 1. (a) and (b) are the JJ-level and timing schematic diagrams of the RSFQ DFF, respectively.

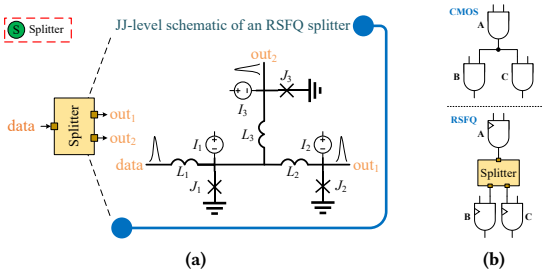


Figure 2. (a) The JJ-level schematic of SPL2. (b) An instance of splitter insertion for a 2-fanout net. In CMOS circuits, the output of A can be directly connected to B and C, but in RSFQ circuits, an SPL2 must be inserted to connect A to B and C.

2.1.1 RSFQ DFF. RSFQ DFF provides a simple way to demonstrate the operational mechanism of RSFQ logic. It is mainly designed to store SFQ pulses and consists of a superconducting ring, as shown in the JJ-level schematic in Fig. 1(a). The superconducting ring, labeled as J_1-L_3-J_3, serves as the storage device for SFQ pulses. It has two stable states: '1' and '0', representing the presence and absence of a magnetic flux quantum within the ring, respectively.

When an SFQ voltage pulse arrives at the input port (*data*), the current flowing through junction J_1 will approach its critical current. This causes J_1 to switch from the superconducting state to the voltage state, generating an SFQ voltage pulse. This pulse is then stored as an SFQ within the superconducting loop formed by J_1-L_3-J_3, setting the loop state to '1'. Subsequently, if a clock pulse arrives while the loop is in the '1' state, the junction J_3 switches, releasing the stored SFQ, and resetting the loop state to '0'. Conversely, if the clock pulse arrives while the loop is in the '0' state, the current flowing through J_3 remains below its critical current, and no voltage pulse is generated at the output port. Fig. 1(b) illustrates the timing diagram of DFF operating under concurrent-flow clocking, where the clock and data signals flow in the same direction and the clock signal precedes the data signal. As a result, when an input pulse ('1') arrives during a clock cycle, an output pulse is produced in the subsequent clock cycle.

2.1.2 Fanout limitation. Unlike conventional CMOS gates, RSFQ logic gates typically can drive only a single output due to their limited output driving capability. Therefore, for a multi-fanout net in RSFQ circuits, a splitter tree must be constructed to split the pulse from the net source to multiple sinks. This requires a dedicated JJ-based gate called the splitter, typically SPL2 for two

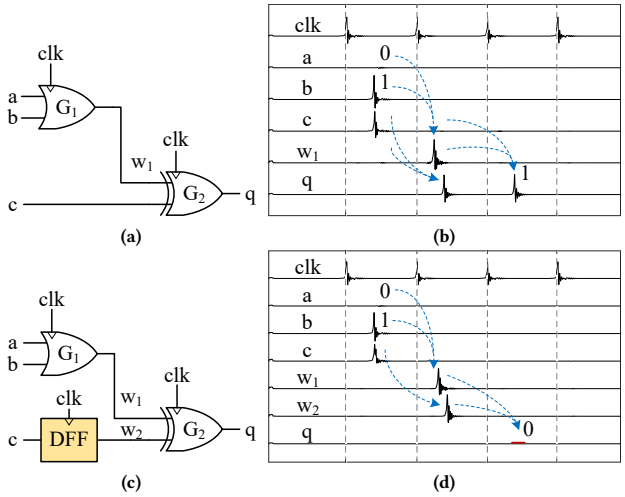


Figure 3. (a) An instance of path imbalance. (b) The path imbalance leads to a functional error where the output becomes 1 when a = 0, b = 1, and c = 1. (c) Introducing a DFF at the second input of the G_2 gate. (d) The correct waveform after introducing the path-balanced DFF.

and SPL3 for three fanouts. Fig. 2(a) depicts the simplified schematic of SPL2, featuring three JJs, i.e., J_1-J_3. At appropriate bias currents (I_1-I_3), when an SFQ pulse is generated at the input port (*data*), it propagates simultaneously to both output ports (out_1 and out_2). As illustrated in Fig. 2(b), CMOS circuits allow gate A to connect directly to gates B and C. Conversely, in a superconducting RSFQ circuit, an SPL2 is required at the output of gate A to drive the other two gates. In an RSFQ circuit, it is necessary to insert $N-1$ splitters for an N-fanout net if only SPL2 is used to construct the fanout tree. Therefore, it is crucial to insert splitters appropriately to minimize the gate count of RSFQ circuits, especially for circuits with many multi-fanout nets.

2.1.3 Path balancing. Almost all RSFQ logic gates require a clock pulse to transfer stored quantum and synchronize gates, allowing for the natural gate-level pipeline. Consequently, to ensure the correct logic operation of RSFQ logic gates, all fanin gates of a given RSFQ gate should have the same logic level, known as the path balancing constraint. The logic level of a gate is defined as the maximum number of clocked gates from any PI of the circuit to this gate. If discrepancies in logic levels exist among the fanins of a gate, it is necessary to insert DFFs at the outputs of the fanin gates with lower logic levels to achieve path balancing.

Fig. 3(a) illustrates an instance of path imbalance. In this example, the G_2 gate's two input gates (one of which is the PI) have logic levels of 1 and 0, thereby violating the path balance condition. This imbalance leads to a functional error where the output becomes 1 when $a = 0$, $b = 1$, and $c = 1$, as shown in Fig. 3(b). Due to the absence of a path-balanced DFF at input c, during the clock cycle when the G_2 gate performs an XOR operation, an erroneous output of 1 is generated. However, by introducing a DFF at the second input of the G_2 gate (depicted in Fig. 3(c)), the output changes to 0, effectively rectifying the circuit's functionality. The corresponding waveform is shown in Fig. 3(d).

3 Methodology

3.1 Terminology

An RSFQ circuit can be represented by a network, denoted as $N(V, E)$, where V is the set of nodes, and E is the set of nets. The node set V consists of three subsets: PI for PIs, PO for primary outputs (POs), and G for logic gates. Each edge in E corresponds to a net. For an edge $e \in E$, e_s is its net source, and e_t is the set of its net sinks. If edge e is a 2-pin net, then $|e_t| = 1$, otherwise $|e_t| > 1$. For a node $v \in V$, $L[v]$ denotes its logic level, while $\mathrm{FI}(v)$ and $\mathrm{FO}(v)$ represent the sets of its fanin nodes and fanout nodes, respectively. Particularly, for a node $i \in PI$, the set of its fanin nodes is empty. For a node $o \in PO$, the set of its fanout nodes is also empty. After D/S insertion, an extended network $N'(V', E')$ can be obtained, where $V' = V \cup D \cup S$, with D and S representing the sets of DFFs and splitters, respectively. In addition, as shown in Fig. 4(b), all nodes are organized into multiple columns according to their logic levels. The logic level for any node $v \in V'$ in the i^{th} column is defined as i. Specifically, this can be expressed as $i = L(v) = \max_{u \in \mathrm{FI}(v)} L(u) + 1$, where $v \in V'$. This paper assumes that all PIs arrive at the same clock cycle and that all POs are generated in the same clock cycle. Therefore, the logic level for each PI is set to 0, while the logic level for each PO is the maximum logic level $l_{\max}$.

3.2 Problem Formulation

In RSFQ circuits, the JJ complexity is usually a vital metric, representing the number of required JJs to design a logic block. However, to satisfy the path balancing requirement and fanout limitation, a large number of DFFs and splitters must be inserted in RSFQ circuits. Therefore, minimizing the total JJ counts caused by D/S insertion is equivalent to minimizing the total number of inserted DFFs and splitters. Our objective is to minimize the number of inserted DFFs while adhering to the design requirements of RSFQ circuits. Therefore, the D/S insertion problem for RSFQ circuit design can be formulated as follows:

- Input: A given circuit $N(V, E)$ and the RSFQ cell library.
- Output: An extended RSFQ circuit $N'(V', E')$ inserted DFFs and splitters.
- Constraints:
  (1) Fanout limation: $\forall i \in G$, the fanout of i is limited to one.
  (2) Path balance: $\forall v \in V', u \in \mathrm{FI}(v), L(v) = L(u) + 1$.
  (3) PI alignment: $\forall i \in PI$, $L(i) = 0$, which ensures that PIs arrive at the same clock cycle.
  (4) PO alignment: $\forall o \in PO$, $L(o) = \max_{v \in V'} L(v) + 1$, which ensures that POs are generated in the same clock cycle.

  Constraints (1) and (2) guarantee the circuit N' is legal after DFF and splitter insertion.
- Goal:

$$\min_{N'} |D|, \tag{1}$$

  which means minimizing the number of inserted DFFs in the circuit N' while guaranteeing the functional correctness of the original circuit N.

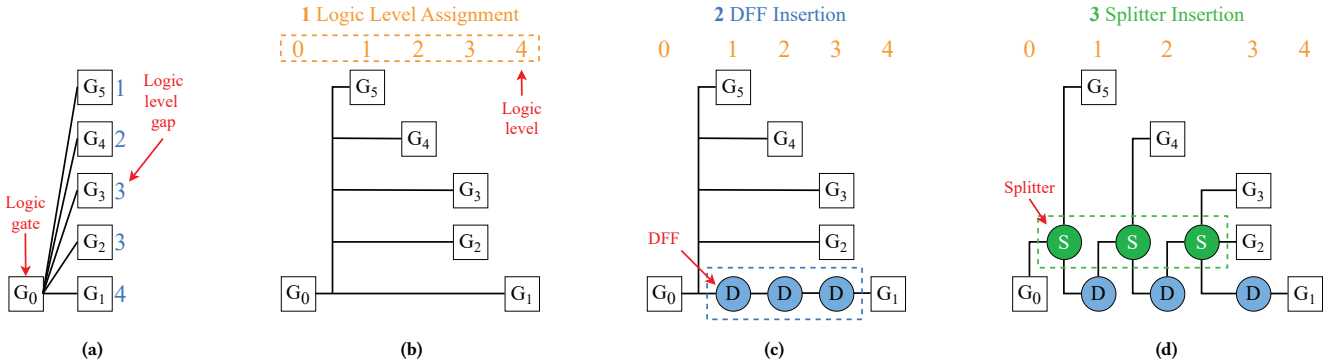


Figure 4. An example of the D/S insertion. (a) is the input sub-circuit, where each blue number is the minimum logic level gap of the corresponding connection. (b) gives a logic level assignment for all gates. (c) inserts DFFs for the connection with the largest logic level gap between the source and sinks. (d) inserts splitters for other connections.

3.3 ILP-based D/S Insertion Algorithm

Ensuring the correct functionality of RSFQ circuits relies on meeting two crucial constraints: path balancing and fanout limitation. In cases where these constraints are naturally violated, the circuit necessitates the insertion of additional DFFs and splitters. However, it is worth noting that the clock-driven feature of RSFQ DFFs introduces implications for both the circuit performance and the clock distribution complexity. Therefore, we propose an optimal DFF-oriented technology legalization algorithm. The D/S insertion process is divided into three phases. First, an integer linear programming model is proposed to minimize the maximum logic level gap between the source and sinks of each net. Next, based on the predefined logic level assignments, DFFs are inserted between source and sink nodes that exhibit the maximum logic level gap within each net to ensure path balancing. Finally, splitter trees are constructed to achieve multi-fanout for all nets. The primary objective of our algorithm is to minimize the number of inserted DFFs, which also facilitates circuit depth minimization. Circuit depth is defined as the PO's logic level minus 1. A lower circuit depth implies a shorter amount of time required to accomplish a task.

3.3.1 *DFF Insertion.*

Lemma 1. *Given a net e after assigning logic levels, the minimum number of DFFs that must be inserted to achieve path balancing is* $\max_{e_{t_i} \in e_t} \left(L[e_{t_i}] - L[e_s] - 1\right)$.

Proof. Assume that the number of inserted DFFs in net e is d_e, where $d_e < \max_{e_{t_i} \in e_t} \left(L[e_{t_i}] - L[e_s] - 1\right)$. So, there must exist a sink e_{t_k} such that $L[e_{t_k}] - L[e_s] - 1 > d_e$. However, to achieve path balancing, the number of inserted DFFs must be greater than or equal to the logic level gap between source e_s and sink e_{t_k}. This contradicts the given condition that $L[e_{t_k}] - L[e_s] - 1 > d_e$. Therefore, the minimum number of DFFs that must be inserted in net e to achieve path balancing is $\max_{e_{t_i} \in e_t} \left(L[e_{t_i}] - L[e_s] - 1\right)$. □

For each net after assigning logic levels, DFFs are inserted only between the source and the sink that possesses the largest logic level gap. This strategy aims to minimize the number of inserted DFFs for the given circuit(Lemma 1). Fig. 4 shows an example of inserting D/S using our algorithm. As shown in Fig. 4(a), the circuit features a net where the minimum logic level gap for G_5 is 1, while G_4, G_3, G_2, and G_1 have minimum logic level gaps of 2, 3, 3, and 4, respectively. Assuming that the logic level is assigned according to the minimum logic level gap of each gate, as shown in Fig. 4(b), the first step is to insert DFFs at the sink with the largest logic level gap. Specifically, this involves inserting DFFs on the connection $G_0 \rightarrow G_1$. As shown in Fig. 4(c), after inserting three DFFs, G_1 achieves path balancing. At this point, it is only necessary to insert splitters between the DFFs; no additional DFFs need to be inserted to satisfy the path balancing for all gates, as shown in Fig 4(d). Relevant details about splitter insertion will be elaborated in Section 3.3.2. So, the minimum number of inserted DFFs for a net e can be calculated by $\max_{e_{t_i} \in e_t} \left(L[e_{t_i}] - L[e_s] - 1\right)$.

Consequently, the quality of the D/S insertion depends on the logic level assignment. Nevertheless, how to determine an optimal logic level assignment is challenging due to the intricate interactions among nets. To address this problem, we first give the following mathematical formulation for the logic level assignment:

$$\min \quad \sum_{e \in E} \max_{e_{t_i} \in e_t} \left(L[e_{t_i}] - L[e_s] - 1\right), \tag{2}$$

$$s.t. \quad L[e_{t_i}] - L[e_s] - 1 \geq 0, \forall e \in E, \forall e_{t_i} \in e_t \tag{3}$$

$$L[pi] = 0, \forall pi \in PI \tag{4}$$

$$L[u] = L[v], \forall u, v \in PO, \tag{5}$$

where the objective function is to minimize the total number of inserted DFFs. Besides, Equation (3) guarantees that the logic level assigned to the source of each net must be strictly less than that of its corresponding sinks. Equations (4)-(5) ensure that all PIs have a logic level of 0 and that all POs share the same logic level.

Notably, the RSFQ splitter does not require clock driving, so it does not need to be considered when assigning logic levels. To simplify the min-max formulation in Equation (2), the variable d_e is introduced, which actually represents the number of inserted DFF required by net e and must satisfy the constraint in Equation (7). So, the aforementioned logic level assignment formulation can be

equivalently transformed into the following ILP problem.

$$\min \quad \sum_{e \in E} d_e \tag{6}$$

$$s.t. \quad L[e_{t_i}] - L[e_s] - 1 \leq d_e, \forall e \in E, \forall e_{t_i} \in e_t \tag{7}$$

$$Equations\ (3) - (5) \tag{8}$$

After solving this problem using the ILP-based solver, the logic level of each gate can be determined. Subsequently, DFFs are inserted between gates with the maximum logic level gap in each net, thereby minimizing the number of DFFs inserted.

3.3.2 Splitter Insertion. After assigning logic levels to each logic gate and inserting DFFs, the primary concern shifts to the insertion of splitters to satisfy the fanout limitation and path balancing requirement. For a 2-pin net, it is sufficient to insert DFFs without the need for any splitters. However, for an over-2-pin net, it is necessary to construct a splitter tree composed of splitters to achieve multi-fanout. As shown in Fig. 5, logic gates are grouped according to their logic levels. For each group, a splitter tree is constructed using RSFQ splitters, achieving multi-fanout for all gates.

Fig. 5(a) illustrates a single-branch splitter tree. With each additional fanout, the depth of the single-branch splitter tree increases by one. This splitter tree is relatively straightforward to implement during the placement and routing stages. However, due to the temporal delays introduced by SFQ pulses when passing through a splitter, there may be significant timing discrepancies between different fanouts of the same output. This situation is detrimental to meeting the clock-synchronized data propagation of RSFQ circuits. As shown in Fig. 5(a), all fanouts of I_5 are arranged in height and split sequentially through splitters. The red numbers above the logic gate G_1, G_2, G_3 and G_4 indicate the number of splitters that the output traverses when traveling from I_5 to each corresponding logic gate. To reach G_1 and G_2, the output from I_5 must pass through four splitters while reaching G_4 requires only two splitters. Fig. 5(b) illustrates an approximate complete binary splitter tree to achieve multi-fanout. This splitter tree can minimize the timing discrepancies among the various fanouts. As shown in Fig. 5(b), the number of splitters traversed by different fanouts differs by no more than one, thereby mitigating the timing discrepancies among the various fanouts. Therefore, this paper employs the construction of a complete binary tree to insert splitters, thereby minimizing the timing discrepancies among the various fanouts.

4 Experiment

4.1 Experiment Setup

The experiments are executed on Intel Core i7-13650HX 2.60 GHz CPU with 16 GB memory. The optimal DFF-oriented technology legalization algorithm is implemented by Python 3.8, and the Gurobi [13] is used as the ILP solver. The RSFQ cell library is from the open-source ColdFlux logic cell library for the MIT-LL SFQ process [21]. The JJ count of an SPL3 is evaluated as 1.5 times that of an SPL2 [21]. The benchmarks are from ISCAS85 [5], EPFL [1] and MCNC[22]. Gate-level netlists were synthesized using the ABC tool [4] with the `recadd3` and `map` scripts.

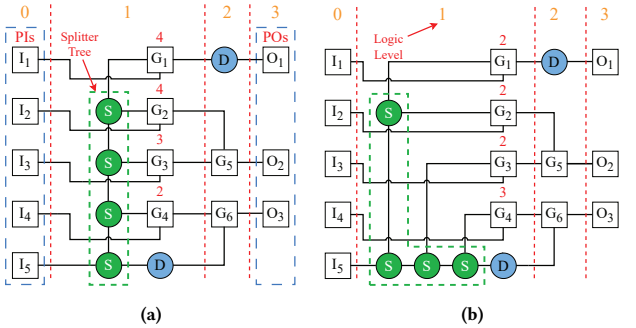


Figure 5. Two different types of splitter trees. (a) constructs a single-branch splitter tree, where each red number is the number of splitters that the output traverses when traveling from I_5 to each corresponding logic gate. (b) constructs an approximate complete binary splitter tree.

Table 1. Experimental results on the ISCAS'85 and EPFL benchmarks

Circuit	GLSVLSI'20[11]				Ours			
	DFF	SPL	JJ	Depth	DFF	SPL	JJ	Depth
c432	895	219	7136	21	569	250	4885	21
c499	360	166	3196	10	308	187	2853	10
c880	1164	230	9112	19	749	274	6251	19
c1355	514	162	4270	12	387	184	3403	12
c1908	1371	378	11211	19	790	434	7200	19
c3540	3066	1211	26543	29	1449	1324	15337	29
c5315	5272	1221	42063	21	2994	1378	26274	21
c6288	50199	6151	377974	68	7429	6957	79390	68
c7552	6243	1285	49034	24	3761	1524	31899	24
int2float	283	137	2588	10	128	158	1524	10
priority	2366	549	18611	30	1392	644	11888	30
adder	23125	816	164999	75	14812	934	106926	75
max	43429	3593	318800	59	29865	3907	224166	59
sin	57274	8813	438117	104	11952	9852	121902	104
Average	2.24	0.88	1.88	1	1	1	1	1

4.2 D/S Insertion Evaluation

We compare our proposed algorithm with the baseline algorithm from GLSVLSI'20 [11] and PBMap [20]. As the code and cell library of PBMap are not available, the data of PBMap in Table 2 is taken from [20]. Since the maximum fanout of the splitter in GLSVLSI'20 is 3 and in PBMap it is 2, we maintain consistency in the maximum fanout of splitters when comparing with the baselines.

Table 1 shows the experimental results of D/S insertion on the ISCAS'85 and EPFL benchmarks. "DFF" denotes the number of inserted DFFs; "SPL" denotes the total number of inserted splitters (SPL2 and SPL3); "JJ" denotes the JJ count of inserted gates; and "Depth" stands for the circuit depth after D/S insertion. The results indicate a significant advantage of the proposed algorithm in terms of the number of inserted DFFs, about a reduction of 44.05% on average. Although our proposed algorithm leads to more splitters inserted due to distinct strategies employed in the D/S insertion process, overall, it achieves an average reduction of 38.41% in the JJ count compared with GLSVLSI'20. Furthermore, our proposed

Table 2. Comparison between our algorithm and PBMap[20]

Circuit	PBMap[20]			Ours		
	DFF	Depth	Time(s)	DFF	Depth	Time(s)
c499	476	13	0.064	308	10	0.335
c880	774	22	0.16	749	19	0.386
c1908	696	20	0.14	790	19	0.418
c3540	1159	31	0.56	1449	29	1.603
c5315	2908	23	1.4	2994	21	1.855
c7552	2429	19	1.04	3761	24	2.218
decoder	8	4	0.012	8	4	0.353
9sym	143	14	0.05	129	12	0.376
int2float	270	16	0.082	128	10	0.299
i9	647	12	0.26	523	10	0.597
cavlc	522	17	0.19	347	12	0.721
priority	9064	124	41.9	1392	30	0.844
sin	13666	182	409.8	11952	104	16.364
Average	1.58	1.44		1	1	

algorithm demonstrates significant advantages over GLSVLSI'20 when applied to large circuits. For instance, in the case of the ISCAS'85 benchmark circuit `c7552`and the EPFL benchmark circuit `sin`, our algorithm achieves substantial improvements in the DFFs inserted and JJ count compared to the GLSVLSI'20, with average reductions of 82.17% and 75.59%, respectively.

Table 2 shows the comparison between PBMap and our proposed ILP-based algorithm. "Time(s)" denotes the runtime in seconds. Compared to PBMap, our proposed algorithm consistently achieves circuit designs with smaller depth, averaging a 19.01% decrease. Our proposed algorithm also reduces the number of inserted DFFs, achieving an average reduction of 11.88%. Additionally, when handling large circuits, our proposed algorithm exhibits a significant advantage in runtime compared to PBMap. For instance, for the large circuits `priority` and `sin` from the EPFL benchmark, our algorithm reduces the runtime by 97.99% and 96.01%, respectively.

5 Conclusion

Superconducting RSFQ logic is an up-and-coming solution in the post-Moore era. Still, current RSFQ-based EDA tools have some limitations due to the unique characteristics of RSFQ circuits. This paper proposed an optimal DFF-oriented technology legalization algorithm for RSFQ circuits, which satisfies the path balancing requirement and fanout limitation with the target of minimizing the number of inserted DFFs. First, an ILP-based algorithm was proposed to determine the optimal optimization logic level assignment. For each net after assigning logic levels, DFFs are inserted only between the source and the sink that possesses the largest logic level gap. Next, splitter trees were constructed for all nets to minimize the timing discrepancies among the various fanouts. Experimental results demonstrated that the proposed algorithm outperformed state-of-the-art D/S insertion algorithms on benchmark circuits in terms of the number of inserted DFFs and circuit depth, particularly on large circuits.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (Grant No.62302477).

References

[1] Luca Gaetano Amarú, Pierre-Emmanuel Gaillardon, and Giovanni De Micheli. 2015. The EPFL Combinational Benchmark Suite. In *Proc. IWLS*.

[2] Yuki Ando, Ryo Sato, Masamitsu Tanaka, Kazuyoshi Takagi, Naofumi Takagi, and Akira Fujimaki. 2016. Design and Demonstration of an 8-bit Bit-Serial RSFQ Microprocessor: CORE e4. *IEEE Transactions on Applied Superconductivity* 26, 5 (2016), 1–5. https://doi.org/10.1109/TASC.2016.2565609

[3] Rassul Bairamkulov, Mingfei Yu, and Giovanni De Micheli. 2024. Unleashing the Power of T1-cells in SFQ Arithmetic Circuits. In *Proc. DAC*. 6 pages. https://doi.org/10.1145/3649329.3658267

[4] Robert Brayton and Alan Mishchenko. 2010. ABC: An Academic Industrial-Strength Verification Tool. In *Proc. CAV*. Springer-Verlag, 24–40. https://doi.org/10.1007/978-3-642-14295-6_5

[5] Franc Brglez. 1985. A neutral netlist of 10 combinatorial benchmark circuits and a target translator in FORTRAN. In *Proc. ISCAS*. 663–698.

[6] Siyan Chen, Rongliang Fu, Junying Huang, Zhimin Zhang, Xiaochun Ye, Tsung-Yi Ho, and Dongrui Fan. 2024. JPlace: A Clock-Aware Length-Matching Placement for Rapid Single-Flux-Quantum Circuits. In *Proc. DATE*. 1–6. https://doi.org/10.23919/DATE58400.2024.10546887

[7] Xinda Chen, Rongliang Fu, Junying Huang, Huawei Cao, Zhimin Zhang, Xiaochun Ye, Tsung-Yi Ho, and Dongrui Fan. 2023. JRouter: A Multi-Terminal Hierarchical Length-Matching Router under Planar Manhattan Routing Model for RSFQ Circuits. In *Proc. GLSVLSI*. 515–520. https://doi.org/10.1145/3583781.3590267

[8] Rongliang Fu, Junying Huang, Haibin Wu, Xiaochun Ye, Dongrui Fan, and Tsung-Yi Ho. 2022. JBNN: A Hardware Design for Binarized Neural Networks Using Single-Flux-Quantum Circuits. *IEEE Trans. Comput.* 71, 12 (2022), 3203–3214. https://doi.org/10.1109/TC.2022.3215085

[9] Rongliang Fu, Mengmeng Wang, Yirong Kan, Olivia Chen, Nobuyuki Yoshikawa, Bei Yu, and Tsung-Yi Ho. 2024. Buffer and Splitter Insertion for Adiabatic Quantum-Flux-Parametron Circuits. *IEEE TCAD* (2024), 1–14. https://doi.org/10.1109/TCAD.2024.3461573

[10] Rongliang Fu, Mengmeng Wang, Yirong Kan, Nobuyuki Yoshikawa, Tsung-Yi Ho, and Olivia Chen. 2023. A Global Optimization Algorithm for Buffer and Splitter Insertion in Adiabatic Quantum-Flux-Parametron Circuits. In *Proc. ASPDAC*. 769–774. https://doi.org/10.1145/3566097.3567936

[11] Rongliang Fu, Zhi-Min Zhang, et al. 2020. Design Automation Methodology from RTL to Gate-Level Netlist and Schematic for RSFQ Logic Circuits. In *Proc. GLSVLSI*. 145–150. https://doi.org/10.1145/3386263.3406898

[12] Kris Gaj, Eby G Friedman, and Marc J Feldman. 1997. Timing of multi-gigahertz rapid single flux quantum digital circuits. *Journal of VLSI signal processing systems for signal, image and video technology* 16, 2 (1997), 247–276.

[13] Gurobi Optimization, LLC. 2022. Gurobi Optimizer Reference Manual. https://www.gurobi.com

[14] Chao-Yuan Huang, Yi-Chen Chang, et al. 2021. An Optimal Algorithm for Splitter and Buffer Insertion in Adiabatic Quantum-Flux-Parametron Circuits. In *Proc. ICCAD*. 1–8. https://doi.org/10.1109/ICCAD51958.2021.9643456

[15] Koki Ishida, Ilkwon Byun, Ikki Nagaoka, Kosuke Fukumitsu, Masamitsu Tanaka, Satoshi Kawakami, Teruo Tanimoto, Takatsugu Ono, Jangwoo Kim, and Koji Inoue. 2020. SuperNPU: An Extremely Fast Neural Processing Unit Using Superconducting Logic Devices. In *Proc. MICRO*. 58–72. https://doi.org/10.1109/MICRO50266.2020.00018

[16] Mustafa Altay Karamuftuoglu, Beyza Zeynep Ucpinar, Arash Fayyazi, Sasan Razmkhah, Mehdi Kamal, and Massoud Pedram. 2025. Scalable superconductor neuron with ternary synaptic connections for ultra-fast SNN hardware. *Superconductor Science and Technology* 38, 2 (2025), 025014. https://doi.org/10.1088/1361-6668/adaaa9

[17] Ryota Kashima, Ikki Nagaoka, Masamitsu Tanaka, Taro Yamashita, and Akira Fujimaki. 2021. 64-GHz Datapath Demonstration for Bit-Parallel SFQ Microprocessors Based on a Gate-Level-Pipeline Structure. *IEEE Transactions on Applied Superconductivity* 31, 5 (2021), 1–6. https://doi.org/10.1109/TASC.2021.3061353

[18] Naveen Katam, Alireza Shafaei, and Massoud Pedram. 2017. Design of Complex Rapid Single-Flux-Quantum Cells with Application to Logic Synthesis. In *International Superconductive Electronics Conference*. 1–3. https://doi.org/10.1109/ISEC.2017.8314236

[19] Konstantin K Likharev and Vasilii K Semenov. 1991. RSFQ logic/memory family: A new Josephson-junction technology for sub-terahertz-clock-frequency digital systems. *IEEE Transactions on Applied Superconductivity* 1, 1 (1991), 3–28.

[20] Ghasem Pasandi and Massoud Pedram. 2019. PBMap: A Path Balancing Technology Mapping Algorithm for Single Flux Quantum Logic Circuits. *IEEE Transactions on Applied Superconductivity* 29, 4 (2019), 1–14. https://doi.org/10.1109/TASC.2018.2880343

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

[22] S. Y. Yang. 1991. *Logic Synthesis and Optimization Benchmarks User Guide: Version 3.0.* Technical Report. MCNC Technical Report.