ELMBA: Escape from Local Minima in Buffer and Splitter Insertion for AQFP Circuits

Yanshuang Teng
SSSLab, China University of Petroleum-Beijing

Rongliang Fu
The Chinese University of Hong Kong

Xijie Wang
SSSLab, China University of Petroleum-Beijing

Dan Niu
Southeast University

Zhou Jin*
Zhejiang University
z.jin@zju.edu.cn

Abstract

Adiabatic Quantum-Flux-Parametron (AQFP) is a promising superconducting logic family that combines ultra-low power consumption with high-speed switching. However, the extensive insertion of buffers and splitters (B/S) to satisfy fan-out and synchronization constraints significantly increases circuit area and depth, becoming a key bottleneck in AQFP synthesis. Existing heuristic approaches are lightweight but prone to local optima, while global or exact methods achieve higher solution quality at the expense of runtime and scalability. In this work, we propose the first Large Neighborhood Search (LNS)-guided framework for AQFP B/S insertion, which combines multi-granularity group movement with a destruct-and-repair paradigm to systematically escape local minima while ensuring legality through constraint-aware repair. Extensive experiments on ISCAS'85 and EPFL benchmarks show that our framework achieves up to 14.0% fewer B/S insertions and 13.1% fewer junctions on large EPFL circuits, while yielding the lowest circuit depth among state-of-the-art methods. It also attains up to a 3.1× runtime speedup on large benchmarks, and on the ISCAS'85 suite reduces B/S and JJs by 13.0% and 8.1% respectively.

Keywords

Superconducting logic, AQFP circuits, Buffer and splitter insertion, Large neighborhood search

ACM Reference Format:
Yanshuang Teng, Rongliang Fu, Xijie Wang, Dan Niu, and Zhou Jin. 2026. ELMBA: Escape from Local Minima in Buffer and Splitter Insertion for AQFP Circuits. In *63rd ACM/IEEE Design Automation Conference (DAC '26), July 26–29, 2026, Long Beach, CA, USA.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3770743.3803938

1 Introduction

Superconducting logic based on Josephson junctions (JJs) offers high switching speed and low power consumption [19, 20, 22, 23, 26], making it a promising candidate beyond the limits of complementary metal-oxide-semiconductor (CMOS) circuits [12]. Information is encoded as voltage pulses propagating through lossless superconducting lines, enabling high-throughput and energy efficiency. Among various logic families, the Adiabatic Quantum-Flux Parametron (AQFP) stands out for its exceptional energy efficiency [25]. Evolved from the Quantum-Flux Parametron (QFP) [18], AQFP employs adiabatic switching and uses alternating current (AC) for both power delivery and clocking, thereby eliminating direct current (DC) bias and enabling gigahertz-level operation. This AC-powered clocking scheme imposes fundamental design constraints on AQFP circuits[24]. As illustrated in Fig. 1, the AC clock simultaneously powers logic gates and synchronizes their operation across discrete phases. Consequently, precise delay matching is required. The limited output drive strength of AQFP gates, which allows each gate to directly drive only one successor, mandates the insertion of dedicated splitters. AQFP design automation shows significant progress, with frameworks proposed for multiple stages of the design flow [4, 6, 8, 9, 21, 27]. Among these, optimizing splitter and buffer (B/S) insertion is crucial, as it determines the number of B/S for gate synchronization and affects overall circuit architecture, power, and delay.

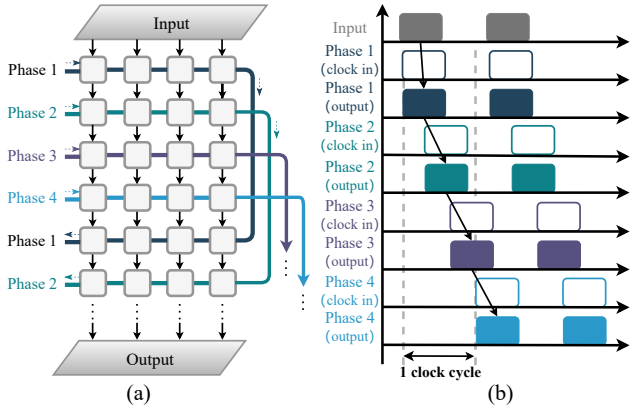


Figure 1: (a) 4-phase clocking scheme for AQFP circuits. (b) Data propagation between neighbouring clock phases.

Several approaches address the B/S optimization problem. Cai et al. [3][5] perform B/S insertion after logic synthesis and apply timing-driven heuristics. Lee et al. [15] propose a block-movement

*Corresponding author.

DAC '26, Long Beach, CA, USA

ACM ISBN 979-8-4007-2254-7/2026/07
https://doi.org/10.1145/3770743.3803938

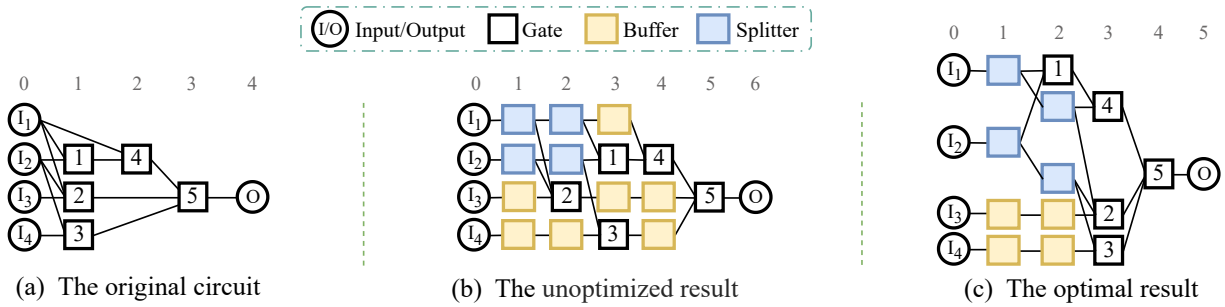


Figure 2: Example of B/S insertion with a splitter fan-out limit of 2. For the input network (a), gray numbers indicate gate levels; (b) results in greater depth and three extra buffers compared to (c).

heuristic for redundancy-free insertion. Huang et al. [13] formulate the problem using dynamic programming to minimize B/S under fixed delay constraints. Lee et al. [16] reformulate the task as a scheduling problem using satisfiability modulo theory (SMT) combined with block movement. More recently, Saito et al. [2] employ a Phase-Skipping clocking scheme to reduce B/S, while Fu et al. [11][10] decompose the B/S insertion process into two parts and propose an integer linear programming (ILP) model for logic level assignment and the optimal fan-out tree for each net to achieve minimization. However, existing approaches still face several significant challenges. Algorithms based on local search [5] and heuristics [16] are prone to becoming trapped in local optima and are unable to achieve a globally optimal or near-optimal B/S configuration. Furthermore, methods that rely on complex algorithms, such as ILP or dynamic programming [11] [10], are computationally inefficient and unscalable for the growing number of large-scale and complex circuits.

To address these limitations, we introduce the first Large Neighborhood Search (LNS)-guided framework for AQFP B/S insertion. The framework switches between multi-granularity group movements for local B/S reduction and structured destruct-and-repair perturbations that systematically overcome deep local minima. A constraint-aware repair step ensures legality, and the iterative process continues until convergence, enabling thorough exploration of the solution space. Fundamentally different from existing heuristics and ILP/DP-based methods, our LNS-guided framework combines global search capability with polynomial-time scalability, making it both effective in escaping local minima and efficient for large-scale AQFP circuits. As a result, it achieves superior B/S minimization, depth control, and runtime efficiency across diverse benchmarks. The contributions are highlighted as follows:

- To the best of our knowledge, this is the first LNS-guided framework for B/S insertion in AQFP circuits, overcoming the local-optima limitations of heuristics and the scalability bottlenecks of ILP/DP approaches.
- Our framework achieves polynomial-time complexity, ranging from $O(T \cdot |V| \log |V|)$ (typical case) to $O(I \cdot |V|^2)$ (worst case): $|V|$ is the total number of circuit nodes, I is the LNS iteration count.
- Extensive experiments on ISCAS'85 and EPFL benchmarks show that our framework achieves the fewest #B/S and the lowest depth among all state-of-the-art approaches. It reduces #B/S by up to 14.0% and #JJs by 13.1%, while simultaneously offering up to a 3.1× runtime speedup.

2 Preliminaries

2.1 Adiabatic Quantum-Flux Parametron

AQFP is an adiabatic superconducting logic family with energy dissipation near thermodynamic and quantum limits [20]. Its distinct physical traits impose strict design constraints, requiring fan-out branching and path balancing for circuit design. For fan-out branching, if a logic gate g drives multiple successors, its limited driving capability requires the insertion of clocked splitters to distribute the output signal. Each splitter has a maximum branching capacity s_p, defined as the maximum number of fan-outs it can directly support. For path balancing, all inputs of a gate g must arrive in the same clock cycle, one cycle before g executes, to ensure correct latching. Primary outputs (POs) are assumed to be aligned at the same depth, while primary inputs (PIs) are placed at depth 0. PIs obey the fan-out branching constraint.

As shown in Fig. 2, a naive level assignment can yield suboptimal insertion, with Fig. 2(b) inserting three more buffers than the optimized configuration in Fig. 2(c).

2.2 Terminology

A logical network is modeled as a directed acyclic graph (DAG) $N = (V, E)$, where the vertex set is partitioned into $V = I \cup O \cup G$, representing primary inputs (PIs), primary outputs (POs), and logic gates, respectively. Each PI has an in-degree of 0 and an unbounded out-degree, while each PO has an out-degree of 0. Each gate $g \in G$ has a fixed in-degree and unbounded out-degree. For any node $n \in I \cup G$, its fan-in $\mathrm{FI}(n)$ and fan-out $\mathrm{FO}(n)$ denote its immediate predecessors and successors in $G \cup O$, respectively.

A mapped network $N' = (V', E')$ extends N by inserting B/S, with $V' = V \cup B \cup S$. Both B/S are timing cells with unit delay ($\delta = 1$). Buffers compensate for delay along single paths, while splitters support multi-fanout distribution. In N', the effective $\mathrm{FI}(g)$ and $\mathrm{FO}(g)$ for any $g \in G$ are defined only over nodes in $G \cup O$, excluding intermediate B/S.

The fan-out tree $\mathrm{FOT}(n)$ of a node $n \in I \cup G$ is the set of B/S along the paths from n to its fan-out loads in N', forming the distribution

network required to satisfy both branching and synchronization constraints.

A schedule is a function $d : V \to \mathbb{Z}\geq 0$ that assigns each node $n \in V$ a non-negative integer $d(n)$, called its depth. The depth of a network N is defined as $d(N) = \max o \in Od(o)$. The relative depth between $n \in I \cup G$ and one of its fan-outs $n_o \in \mathrm{FO}(n)$ is defined as:

$$\mathrm{rd}(n, n_o) = d(n_o) - d(n) \tag{1}$$

To capture the feasible placement of each node, we compute both the earliest possible depth (ASAP schedule) and the latest possible depth (ALAP schedule). The ASAP schedule d_{ASAP} is obtained by forward propagation and represents the minimum cycle in which a node can be activated, while the ALAP schedule d_{ALAP} is obtained by backward propagation under the target network depth and represents the latest feasible cycle without violating timing. Together, these define a timing window for each node $v \in V$: $[d_{\mathrm{ASAP}}(v), d_{\mathrm{ALAP}}(v)]$ which bounds all valid assignments in subsequent optimization phases. d is called legal if it allows the construction of a mapped network $N' = (V', E')$ that is both path-balanced and properly branched, and all fan-outs are supported by valid B/S structures.

2.3 Problem Definition

To satisfy the delay and fan-out requirements, B/S must be inserted into the AQFP circuit. This paper focuses on the B/S insertion problem for AQFP gate-level circuits after logic synthesis, formulated as follows:

- Input:
  (1) A given initial network $N(V, E)$.
  (2) The maximum fan-out capacity of the splitter s_p.
- Output:
  A mapping network that satisfies AQFP constraints $N'(V', E')$ with an equal delay for all inputs to any gate and a single fan-out for all gate outputs.
- Constraints:
  (1) N' is path-balanced and properly-branched.
  (2) For all $g \in G$, the effective $\mathrm{FI}(g)$ and $\mathrm{FO}(g)$ in N' are identical to those in N, ignoring $B \cup S$.
- Goal:
  (1) Minimize the number of $|B| + |S|$.
  (2) Reduce runtime.

3 Methodology

To address the B/S insertion problem formulated in Section 2.3, we design a structured optimization framework that combines multi-granularity group movement with a LNS strategy. The overall procedure consists of three stages: first constructing a legal AQFP network through depth-optimal initial mapping, then applying optimization via group movement, and finally performing global optimization through LNS-based destruct-and-repair to expand the solution space beyond local adjustments. Combination multi-granularity group movement and destruct-and-repair distinguishes our approach from prior heuristic methods that only perform local rewiring, and from global optimization frameworks that incur high runtime cost, enabling both scalability and high-quality solutions.

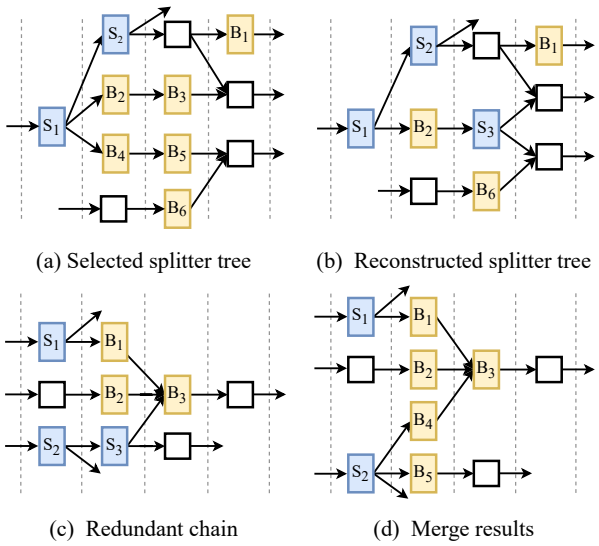


Figure 3: (a) shows how the redundant $B_2 - B_5$ chains in the network are merged and inserted into S_3 for signal distribution. (c) and (d) show the process of rebuilding the splitter tree, converting S_3 into B_4 and B_5.

3.1 Depth-Optimal Initial Mapping

To construct an initial AQFP network, we formulate B/S insertion as a scheduling problem and build upon the framework of [7], which assigns gate levels and inserts splitters under path-balancing constraints. To obtain a higher-quality initialization for subsequent optimization, We augment the basic mapping with two complementary enhancements: Predictive Fanout Scheduling and fan-out tree optimization with redundancy elimination, yielding a compact N'_{init} with reserved timing flexibility for subsequent optimization.

Predictive Fanout Scheduling. Beyond conventional scheduling, we assign each gate a depth within its feasible timing window $[d_{\mathrm{ASAP}}(g), d_{\mathrm{ALAP}}(g)]$ using a look-ahead cost model that anticipates downstream B/S demand. This timing-aware assignment lays the foundation for balanced fan-out tree construction and avoids excessive B/S insertion caused by poor depth choices. Specifically, for each candidate depth d, we estimate its future splitter overhead by evaluating the projected fanout distribution after potential downstream movements. The selected depth minimizes

$$\mathrm{cost}(g, d) = |\mathrm{FOT}(g)| + \gamma \cdot \widehat{b_{\mathrm{f}}}(g, d), \tag{2}$$

where $\widehat{b_{\mathrm{f}}}$ predicts buffer usage introduced by deeper fanout divergence. A fanout-weighted traversal,

$$\mathrm{priority}(v) = \beta_1 d_{\mathrm{ASAP}}(v) - \beta_2 |\mathrm{FO}(v)|, \tag{3}$$

ensures that high-fanout regions receive additional slack, improving later global optimization.

Fan-out Tree Optimization. Building on the depth assignment from Predictive Fanout Scheduling, we construct a depth-balanced splitter tree $\mathrm{FOT}(n)$ for each multi-fanout gate n under branching

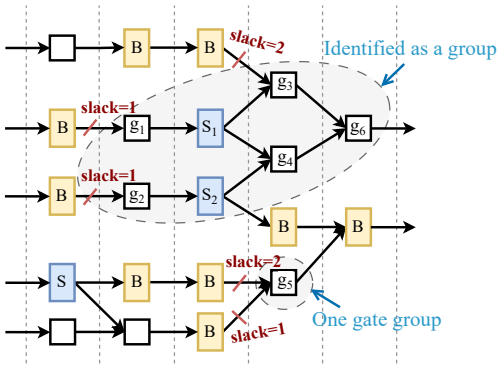


Figure 4: Example of two groups to be downward movement.

capacity s_p:

$$\mathrm{dp}(\mathrm{FOT}(n)) = \lceil \log_{s_p} f_n \rceil, \quad |\mathrm{FOT}(n)| = \left\lceil \frac{f_n - 1}{s_p - 1} \right\rceil. \tag{4}$$

This guarantees consistent branch depths and satisfies AQFP path-balancing requirements. To suppress unnecessary tree expansion, we apply a redundancy-aware compaction step that merges partially filled splitter chains whenever spare capacity exists, producing tighter structures without increasing depth (Fig. 3). After tree construction, structurally equivalent buffer chains feeding identical fanout sets are further collapsed through redundancy matching (e.g., the buffer chain in Fig. 3). The resulting N'_{init} satisfies AQFP depth and fanout constraints while preserving additional timing flexibility, providing a high-quality starting point for subsequent multi-granularity group movement and LNS reconstruction.

3.2 Multi-granularity Group Movement

To address residual local redundancy and limited exploration scope of gate-level adjustments in N'_{init}, while satisfying AQFP constraints, we adopt a multi-granularity adaptive movement strategy with phase-sync boundary calibration to reposition correlated gate clusters across timing levels, further reducing B/S.

A group $C = (M_C, \mathcal{I}_C, O_C)$ is formed by greedy expansion under a hybrid proximity metric:

$$\mathrm{prox}(g_i, g_j) = \frac{1}{1 + \mathrm{rd}(g_i, g_j)} + \lambda \, \mathrm{corr}(g_i, g_j), \tag{5}$$

which integrates structural distance ($\mathrm{rd}(g_i, g_j)$) and fanout-depth similarity, where $\mathrm{corr}(g_i, g_j)$ denotes the Pearson correlation of fanout sizes and depths to quantify structural similarity. This yields locally coherent timing regions at adaptive granularities (fanout chain, clustered logic, e.g.) instead of predefined chunks. At each iteration, we select the granularity maximizing the efficiency ratio $\Delta|B|/\Delta|M_C|$, enabling broad exploration tailored to circuit structure.

Group Formation. Given a group C, its legal movement range is determined by interface slack:

$$l^{\downarrow}_{\max} = \min_{(u,v)\in\mathcal{I}_C} \big(\mathrm{rd}(u,v) - \delta_{FO}(u)\big),$$
$$l^{\uparrow}_{\max} = \min_{(v,w)\in O_C} \big(\mathrm{rd}(v,w) - \delta_{FO}(v)\big), \tag{6}$$

Algorithm 1 LNS-based Destruct-and-Repair Process

Require: Current network N', splitting capacity s_p, buffer count B_{current}
Ensure: Updated network N'' after LNS repair
1: Compute scores for all $v \in V'$ by Eq. (8)
2: Select high-score subgraph $\mathcal{U} \subseteq N'$ with adaptive size $|\mathcal{U}|$
3: Estimate $\hat{B}(\mathcal{U})$ by Eq. (9)
4: **if** $\hat{B}(\mathcal{U}) > \alpha \cdot B_{\text{current}}$ **then**
5: **return** N' {Skip if unpromising}
6: **end if**
7: Remove all B/S in $\mathcal{U}$
8: Reschedule gates of $\mathcal{U}$ within feasible timing windows
9: Reinsert B/S under fanout limit s_p
10: Update only $\mathcal{U}$ and boundary incrementally
11: Reuse cached result if subgraph pattern repeats
12: **return** N'' if $B(N'') < B_{\text{current}}$, else N'

where $\delta_{FO}(u)$ equals 1 for single-fanout nodes and 2 otherwise. Each candidate displacement is evaluated by $\eta(C, l) = \Delta B(l)/\Delta D(l)$, which measures the buffer reduction per unit depth deviation, balancing two conflicting objectives rather than optimizing for one. Movements with $\eta(C, l) > \tau$ are applied, where τ adapts to global optimization progress. Fig. 4 shows a typical downward movement that removes redundant buffering while maintaining legal fanout balance.

Phase-Sync Boundary Calibration. A critical limitation of prior chunk movement is the lack of phase consistency across group boundaries, leading to clock-phase mismatches in AQFP circuits. After a group moves, we perform boundary-phase calibration to ensure clock-phase continuity at inter-group fanout interfaces. Neighboring groups with conflicting depth assignments are adjusted locally by ±1 level, preventing oscillation and allowing multiple groups to update concurrently. When multiple interacting groups share timing windows or overlapping fanout cones, a coordination graph is constructed and a lightweight solver jointly updates their depth adjustments by solving

$$\max_{\{\Delta d_i\}} \sum_i \Delta B_i \quad \text{s.t. } |\Delta d_i - \Delta d_j| \le 1, \ \forall (i,j) \in E, \tag{7}$$

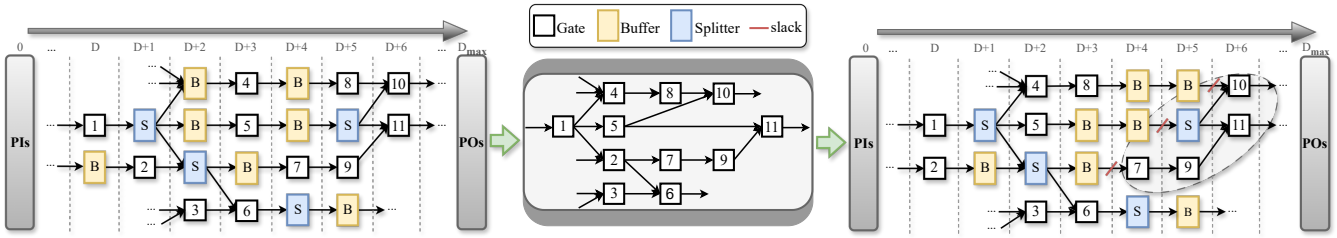
Although coordination improves stability, it may converge into restrictive equilibria where adjacent groups mutually constrain one another. This motivates our subsequent global refinement step using LNS to break stagnation and release new timing slack.

3.3 LNS-based Global Iterative Optimization

To overcome local optima from greedy insertion and limited group movement, we integrate a Large Neighborhood Search (LNS) framework into the iterative optimization loop. Unlike local gate-level adjustments, LNS periodically reconstructs selected dense subgraphs $\mathcal{U} \subset V'$ via a destruct-and-repair process—removing all B/S, rescheduling gates in feasible timing windows, and reinserting B/S under fanout constraints. This global refinement breaks rigid buffer chains and releases timing slack for subsequent group movements, forming a cooperative cycle: LNS expands design flexibility, and group movement exploits it for further reduction.

To ensure scalability, the LNS module integrates three lightweight strategies. First, it performs subgraph selection by scoring each node according to its structural importance:

$$\mathrm{score}(v) = w_1 |\mathrm{FO}(v)| + w_2 |\mathrm{FI}(v)|. \tag{8}$$



(a) Selected high fan-out subgraph (b) Subgraph after removing B/S (c) Reconstructed subgraph

Figure 5: Illustration of LNS repair process. (a) A high-fanout subgraph is first selected. (b) All buffers and splitters (B/S) in this subgraph are then removed while preserving gate connections. (c) Finally, the subgraph is reconstructed by re-scheduling gates within their timing windows and reinserting B/S to reduce depth and improve flexibility.

A subgraph $\mathcal{U}$ of adaptive size (3%–10% of $|V|$) is extracted from high-score regions to focus on buffer-redundant areas. Next, a fast estimation step approximates the buffer potential of the subgraph as

$$\hat{B}(\mathcal{U}) = \sum_{v \in \mathcal{U}} \left(\left\lceil \frac{f_v - 1}{s_p - 1} \right\rceil + \Delta_d(v) \right), \tag{9}$$

and LNS proceeds only if $\hat{B}(\mathcal{U}) \leq \alpha \cdot B_{\text{current}}$. Finally, an incremental evaluation updates only $\mathcal{U}$ and its boundary, avoiding global recomputation; cached results are reused for repeating substructures to boost efficiency.

The LNS procedure is summarized in Algorithm 1. It begins with subgraph selection. It evaluates $\hat{B}(\mathcal{U})$ and destructs the region when $\hat{B}(\mathcal{U}) \leq \alpha \cdot B_{\text{current}}$. It performs timing-aware rescheduling and reinserts B/S elements under s_p. Incremental updates and cache reuse maintain efficiency and reshape the timing landscape at a global level.

Regarding computational complexity, the LNS framework is dominated by three steps. The iteration count I (empirically 10–100 across our benchmarks and only weakly dependent on $|V|$) acts as a constant factor. Subgraph selection computes node scores in $O(|V|)$ and sorts them in $O(|V| \log |V|)$. Fast estimation, B/S removal, and localized rescheduling for $\mathcal{U}$ each run in $O(|\mathcal{U}|)$ or $O(|\mathcal{U}| \log |\mathcal{U}|)$. These steps remain lightweight because $\mathcal{U}$ typically occupies only 3–10% of $|V|$. Incremental updates of $\mathcal{U}$ and its boundary take $O(|\mathcal{U}|)$ and do not affect the dominant cost. Each LNS iteration therefore costs $O(|V| \log |V|)$. The overall complexity $O(I \cdot |V| \log |V|)$. In practice, the small subgraph size and incremental repair yield near-linear scalability, as confirmed by the sub-quadratic runtime trend on large circuits.

The complete optimization framework (Algorithm 2) alternates between multi-granularity group movement and LNS repair. Starting from the initial mapping N'_{init}, group movement first removes local redundancy, and when no further improvement is possible, the LNS phase resets constrained subregions to reopen the search space. The system then reapplies group movement to exploit the newly released slack, updating N'_{best} iteratively until convergence.

Algorithm 2 Overall Optimization Framework Process

Require: Network N, splitting capacity s_p
Ensure: Optimized network N'_{best} with B/S
$N' \leftarrow$ construct_legal_circuit(N, s_p)
$N'_{\text{best}} \leftarrow N'$, $B_{\text{best}} \leftarrow$ buffer_count(N')
while not converged **do**
  Apply multi-granularity group movement on N'
  if no further reduction **then**
    Perform LNS repair on selected subgraph $\mathcal{U}$ (Algorithm 1)
    Apply boundary alignment and incremental update
  end if
  if buffer_count$(N') < B_{\text{best}}$ **then**
    $N'_{\text{best}} \leftarrow N'$, $B_{\text{best}} \leftarrow$ buffer_count(N')
  else
    $N' \leftarrow N'_{\text{best}}$ {Revert to best}
  end if
end while
return N'_{best}

4 Experiments

4.1 Experimental Setup

We implement the optimization framework for B/S insertion of AQFP logic in C++-17 and carry out the experiments on a machine equipped with an AMD EPYC 7C13 64-core CPU @ 2.0 GHz and 128 GB of memory. Two categories of benchmark circuits are selected: (1) small-scale circuits from ISCAS'85 and the arithmetic benchmark [14], and (2) large-scale circuits from the EPFL benchmark [1]. All mapped circuits are verified to satisfy the AQFP constraints. We evaluate methods by the original circuit depth (D), the total number of inserted buffers and splitters (#B/S), the total number of Josephson junctions (#JJs), post-mapping circuit depth (D'), and runtime (T). The proposed framework is compared with four state-of-the-art (SOTA) methods [10, 13, 16, 17]. The adaptive subgraph size $|\mathcal{U}|$ varies between 3% and 10% of the total nodes to balance exploration and runtime.

4.2 Comparison with Prior Work

Tables 1 and 2 summarize the comparison with prior B/S insertion frameworks. Across all benchmarks, our method achieves the lowest #B/S, D', and #JJs while maintaining competitive runtime, confirming that the proposed initialization, group movement, and LNS reconstruction jointly improve mapping quality under AQFP constraints.

On ISCAS'85 and arithmetic circuits, #B/S is reduced by up to 13.0% and #JJs by 8.1% over ICCAD'21, and further by 9.9% and 6.0% over DAC'22. Large-fanout designs such as *c7552* and *c6288* benefit the most, reflecting the effectiveness of our fanout-aware compaction and slack-guided group movement. Runtime improves substantially compared with ICCAD'21 and remains comparable to

Table 1: Experimental results on the ISCAS'85 and simple arithmetic benchmarks.

Testcase	Original Circuit			ICCAD'21 [13]				DAC'22 [16]				TCAD'24 [17]				TCAD'25 [10]				Our method			
	Gates	D	MFO	#B/S	#JJs	D'	T(s)	#B/S	#JJs	D'	T(s)	#B/S	#JJs	D'	T(s)	#B/S	#JJs	D'	T(s)	#B/S	#JJs	D'	T(s)
adder1	7	4	2	16	74	8	0.15	16	74	8	0.01	16	74	8	0.01	16	74	8	0.02	**16**	**74**	8	0.01
adder8	77	17	3	404	1270	33	0.08	371	1204	33	0.01	371	1204	33	0.01	371	1204	33	0.03	**371**	**1204**	33	0.01
alu32	1513	100	128	15244	39566	171	17.39	14742	38562	173	0.48	13836	36750	169	2.74	13991	37060	169	2.16	**13631**	**36340**	169	0.50
multiplier8	439	35	9	1907	6448	70	0.21	1861	6356	71	0.04	1690	6014	70	0.18	1681	5996	70	0.16	**1656**	**5946**	70	0.10
counter16	29	9	4	99	372	17	0.05	65	304	17	0.01	65	304	17	0.01	65	304	17	0.02	**64**	**302**	17	0.01
counter32	82	13	4	201	894	23	0.04	155	802	23	0.01	155	802	23	0.01	155	802	23	0.03	**154**	**800**	23	0.01
counter64	195	17	4	412	1994	30	0.05	352	1874	30	0.02	347	1864	30	0.02	349	1868	30	0.04	**347**	**1864**	30	0.02
counter128	428	22	4	836	4240	38	0.10	760	4088	38	0.04	747	4062	38	0.07	751	4070	38	0.05	**747**	**4062**	38	0.04
c17	6	3	2	17	70	5	0.05	12	60	5	0.01	12	60	5	0.01	12	60	5	0.02	**12**	**60**	5	0.01
c432	121	26	10	889	2504	37	0.07	886	2498	39	0.01	839	2404	37	0.02	843	2412	37	0.04	**829**	**2394**	37	0.02
c499	387	18	8	1238	4798	29	0.11	1270	4862	31	0.03	1173	4668	29	0.09	1173	4668	29	0.06	**1173**	**4668**	29	0.05
c880	306	27	9	1702	5240	40	0.11	1665	5166	41	0.03	1511	4858	40	0.15	1536	4908	40	0.06	**1511**	**4858**	40	0.04
c1355	389	18	9	1240	4814	29	0.11	1271	4876	31	0.03	1184	4702	29	0.06	1186	4706	29	0.06	**1178**	**4690**	29	0.04
c1908	289	21	14	1405	4544	35	0.12	1344	4422	37	0.03	1234	4202	34	0.09	1246	4226	34	0.06	**1232**	**4198**	34	0.06
c2670	368	21	32	2014	6240	28	0.24	2092	6392	30	0.04	1912	6032	28	0.32	1876	5964	28	0.68	**1792**	**5796**	28	0.11
c3540	794	32	38	3419	11606	53	0.63	2281	9326	56	0.18	1943	8650	52	0.81	1964	8696	52	0.60	**1918**	**8604**	52	0.22
c5315	1302	26	41	6016	19922	40	1.08	5989	19790	42	0.46	5640	19092	40	2.06	**5531**	**18952**	40	0.64	5538	18938	40	0.49
c6288	1870	89	17	9476	30172	179	2.62	9016	29252	180	0.64	8647	28514	179	2.56	8832	28884	179	3.41	**8612**	**28444**	179	0.74
c7552	1394	33	170	8090	24560	58	15.81	8598	25560	66	0.43	7437	23238	56	4.20	6740	21860	58	1.93	**6602**	**21584**	56	0.55
sorter32	480	15	2	510	3900	30	0.12	480	3840	30	0.04	480	3840	30	0.06	480	3840	30	0.05	**480**	**3840**	30	0.05
sorter48	880	20	3	915	7110	35	0.17	880	7040	35	0.11	880	7040	35	0.20	880	7040	35	0.07	**880**	**7040**	35	0.15
Total	-	-	-	56050	180338	988	39.31	54106	176348	1016	**2.66**	50119	156154	982	13.68	49678	167594	984	10.19	**48743**	**165706**	982	3.23

Table 2: Experimental results on the larger-scale circuits from the EPFL benchmark.

Testcase	Original Circuit			DAC'22 [16]				TCAD'25 [10]				Our method			
	Gates	D	MFO	#B/S	#JJs	D'	T(s)	#B/S	#JJs	D'	T(s)	#B/S	#JJs	D'	T(s)
div	57247	4372	370	3117173	6577828	8703	8.97	2785942	5915486	8601	7.70	**2468226**	**5280054**	**8530**	**5.21**
hyp	214335	24801	255	13912571	29111152	41464	-	12642718	26571446	41292	-	**12072651**	**25431312**	**41083**	35.72
log2	32060	444	253	188374	569108	788	7.00	158122	508604	773	6.21	**148036**	**488432**	**771**	**4.17**
multiplier128	27062	274	145	134413	431198	531	5.98	121620	405612	526	4.74	**120182**	**402736**	526	**3.82**
sin	5416	225	84	29140	90776	360	2.63	**25999**	**84494**	352	3.32	26834	86164	352	**2.01**
sqrt	24618	5058	126	1453718	3055144	8288	**5.86**	1416653	2981014	8199	6.27	**1372642**	**2892992**	**8098**	6.02
square	18484	250	75	102638	316180	414	5.01	**97268**	**305440**	409	3.5	97514	305932	409	**2.31**
Total	-	-	-	18938027	40151386	60548	35.45	17248322	36772096	60152	31.74	**16306085**	**34887622**	**59769**	**23.54**

DAC'22. Relative to TCAD'24, our method reduces #B/S by 2.75% with a 4.25× faster runtime on ISCAS'85. Compared with TCAD'25, it provides further reductions in #B/S and D' while retaining a 3.1× speedup. These results reflect that compact initialization and LNS-based reconstruction remove more redundancy than TCAD'24 and avoid the scheduling overhead of TCAD'25, achieving both higher quality and lower runtime.

On large EPFL circuits, #B/S and #JJs drop by 14.0% and 13.1% over DAC'22, and by 5.5–5.1% over TCAD'25. Benchmarks such as *div*, *log2*, and *sqrt*, illustrating how multi-granularity grouping paired with LNS reconstruction effectively handles the deep, high-fanout, and highly regular structures common in EPFL designs. These results also reveal an algorithmic insight. Previous methods often become trapped in rigid splitter trees, whereas our framework periodically resets congested subgraphs through LNS and then re-exploits the freed slack via group movement. Runtime scales favorably as well, *div* accelerates from 8.97 s to 5.21 s, and total runtime decreases from 35.45 s to 23.54 s. This trend aligns with the near-linear practical complexity discussed earlier, confirming that destruct-and-repair remains lightweight even on the largest EPFL circuits.

Overall, these results demonstrate that the proposed framework achieves the best trade-off among #B/S, D', and runtime across all benchmarks. The favorable runtime trend is consistent with the near-linear practical complexity in Section 3, as localized movement and small LNS regions keep optimization scalable on large circuits.

5 Conclusion

This paper presented a novel framework for B/S insertion in AQFP circuits, based on LNS-guided subgraph optimization. By enabling non-local transformations through multi-granularity group movement and a destruct-and-repair paradigm, our approach effectively escapes deep local minima that limit prior methods. Experiments on ISCAS'85 and EPFL benchmarks demonstrate that our method consistently achieves the best results across all key metrics, including the lowest #B/S, #JJs, and D', while maintaining competitive runtime. Compared to local heuristics, our framework consistently yields superior solution quality. Compared to global optimization, it further offers substantial runtime gains. These results highlight the practicality of our approach for integration into modern AQFP synthesis flows.

References

[1] L. G. Amarù, P.-E. Gaillardon, and G. De Micheli. 2015. The EPFL combinational benchmark suite. *in International Workshop on Logic and Synthesis (IWLS)* (2015).

[2] R. S. Aviles and P. A. Beerel. 2024. A Joint Optimization of Buffer and Splitter Insertion for Phase-Skipping Adiabatic Quantum-Flux-Parametron Circuits. *in IEEE International Conference on Computer Design (ICCD)* (2024).

[3] C. L. Ayala, R. Saito, T. Tanaka, O. Chen, N. Takeuchi, Y. He, and N. Yoshikawa. 2020. A semi-custom design methodology and environment for implementing superconductor adiabatic quantum-flux-parametron microprocessors. *Superconductor Science and Technology* (2020).

[4] R. Cai, O. Chen, A. Ren, N. Liu, C. Ding, N. Yoshikawa, and Y. Wang. 2019. A Majority Logic Synthesis Framework for Adiabatic Quantum-Flux-Parametron Superconducting Circuits. *Great Lakes Symposium on VLSI (GLSVLSI)* (2019).

[5] R. Cai, O. Chen, A. Ren, N. Liu, N. Yoshikawa, and Y. Wang. 2019. A Buffer and Splitter Insertion Framework for Adiabatic Quantum-Flux-Parametron Superconducting Circuits. *in IEEE International Conference on Computer Design (ICCD)* (2019).

[6] R. Cai, X. Ma, O. Chen, A. Ren, N. Liu, N. Yoshikawa, and Y. Wang. 2019. IDE Development, Logic Synthesis and Buffer/Splitter Insertion Framework for Adiabatic Quantum-Flux-Parametron Superconducting Circuits. *IEEE Computer Society Annual Symposium on VLSI (ISVLSI)* (2019).

[7] Alessandro Tempia Calvino and Giovanni. De Micheli. 2023. Depth-optimal Buffer and Splitter Insertion and Optimization in AQFP Circuits. *in Asia and South Pacific Design Automation Conference (ASP-DAC)* (2023).

[8] Yi-Chen Chang, Hongjia Li, Olivia Chen, Yanzhi Wang, Nobuyuki Yoshikawa, and Tsung-Yi Ho. 2020. ASAP: An Analytical Strategy for AQFP Placement. *in IEEE/ACM International Conference On Computer Aided Design (ICCAD)* (2020).

[9] Rongliang Fu, Olivia Chen, Bei Yu, Nobuyuki Yoshikawa, and Tsung-Yi Ho. 2023. DLPlace: A Delay-Line Clocking-Based Placement Framework for AQFP Circuits. *in IEEE/ACM International Conference On Computer Aided Design (ICCAD)* (2023).

[10] R. Fu, M. Wang, Y. Kan, O. Chen, N. Yoshikawa, B. Yu, and T.-Y. Ho. 2025. Buffer and Splitter Insertion for Adiabatic Quantum-Flux-Parametron Circuits. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems publication information* (2025).

[11] R. Fu, M. Wang, Y. Kan, N. Yoshikawa, T.-Y. Ho, and O. Chen. 2023. A Global Optimization Algorithm for Buffer and Splitter Insertion in Adiabatic Quantum-Flux-Parametron Circuits. *in Asia and South Pacific Design Automation Conference (ASP-DAC)* (2023).

[12] D. S. Holmes, A. L. Ripple, and M. A. Manheimer. 2013. Energy-Efficient Superconducting Computing—Power Budgets and Requirements. *IEEE Transactions on Applied Superconductivity* (2013).

[13] C.-Y. Huang, Y.-C. Chang, M.-J. Tsai, and T.-Y. Ho. 2021. An Optimal Algorithm for Splitter and Buffer Insertion in Adiabatic Quantum-Flux-Parametron Circuits. *in IEEE/ACM International Conference On Computer Aided Design (ICCAD)* (2021).

[14] ISCAS85 2023. ISCAS 85 and simple arithmetic benchmarks. https://github.com/lsils/SCE-benchmarks/tree/main/ISCAS.

[15] S.-Y. Lee, H. Riener, and G. De Micheli. 2021. Irredundant Buffer and Splitter Insertion and Scheduling-Based Optimization for AQFP Circuits. *in International Workshop on Logic and Synthesis (IWLS)* (2021).

[16] S.-Y. Lee, H. Riener, and G. De Micheli. 2022. Beyond local optimality of buffer and splitter insertion for AQFP circuits. *in ACM/IEEE Design Automation Conference (DAC)* (2022).

[17] Siang-Yun Lee, Alessandro Tempia Calvino, Heinz Riener, and Giovanni De Micheli. 2024. Technology Legalization and Optimization for Adiabatic Quantum-Flux Parametron. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* (2024).

[18] K. Loe and E. Goto. 1985. Analysis of flux input and output Josephson pair device. *IEEE Transactions on Magnetics* (1985).

[19] Takashi Mukaiyama, Naoki Takeuchi, Yuki Yamanashi, and Nobuyuki Yoshikawa. 2013. Design and demonstration of an on-chip AC power source for adiabatic quantum-flux-parametron logic. *Superconductor Science Technology* (2013).

[20] Oleg A. Mukhanov. 2011. Energy-Efficient Single Flux Quantum Technology. *IEEE Transactions on Applied Superconductivity* (2011).

[21] G. Pasandi and M. Pedram. 2019. A Dynamic Programming-Based, Path Balancing Technology Mapping Algorithm Targeting Area Minimization. *in IEEE/ACM International Conference On Computer Aided Design (ICCAD)* (2019).

[22] R. Sato, Y. Hatanaka, Y. Ando, M. Tanaka, A. Fujimaki, K. Takagi, and N. Takagi. 2017. High-Speed Operation of Random-Access-Memory-Embedded Microprocessor With Minimal Instruction Set Architecture Based on Rapid Single-Flux-Quantum Logic. *IEEE Transactions on Applied Superconductivity* (2017).

[23] N. Takeuchi, K. Ehara, K. Inoue, Y. Yamanashi, and N. Yoshikawa. 2013. Margin and Energy Dissipation of Adiabatic Quantum-Flux-Parametron Logic at Finite Temperature. *IEEE Transactions on Applied Superconductivity* (2013).

[24] Naoki Takeuchi, Mai Nozoe, Yuxing He, and Nobuyuki Yoshikawa. 2019. Low-latency adiabatic superconductor logic using delay-line clocking. *Applied Physics Letters* (2019).

[25] N. Takeuchi, D. Ozawa, Y. Yamanashi, and N. Yoshikawa. 2013. An adiabatic quantum flux parametron as an ultra-low-power logic device. *Superconductor Science and Technology* (2013).

[26] N. Takeuchi, T. Yamae, C. Ayala, H. Suzuki, and N. Yoshikawa. 2019. An adiabatic superconductor 8-bit adder with $24k_BT$ energy dissipation per junction. *Applied Physics Letters* (2019).

[27] Eleonora Testa, Siang-Yun Lee, Heinz Riener, and Giovanni De Micheli. 2021. Algebraic and Boolean Optimization Methods for AQFP Superconducting Circuits. *in Asia and South Pacific Design Automation Conference (ASP-DAC)* (2021).