DLPlace: A Delay-Line Clocking-based Placement Framework for AQFP Circuits

Rongliang Fu[†], Olivia Chen[‡], Bei Yu[†], Nobuyuki Yoshikawa[‡], Tsung-Yi Ho[†]
[†]*The Chinese University of Hong Kong*
[‡]*Tokyo City University*
[‡]*Yokohama National University*
rlfu@cse.cuhk.edu.hk

Abstract*—Addressing the pressing need for energy-efficient computing technologies, innovations such as Josephson junctions-based superconducting logic circuits, particularly the Adiabatic Quantum-Flux-Parametron (AQFP) logic, have sparked increased research interest. AQFP logic, boasting superior energy efficiency, faces unique design challenges. The current 4-phase clocking scheme results in considerable circuit latency, a problem further amplified with larger logic depth in the circuit. A novel delay-line clocking scheme proposes increasing the number of clock phases, which could significantly improve circuit latency but also risks more severe timing violations. To address this issue, this paper proposes DLPlace, the first placement framework tailored for the delay-line clocking scheme, aiming to boost the performance of AQFP circuits. DLPlace formulates timing-aware global placement as a Lagrangian problem, targeting minimizing the circuit latency, to determine the positions of all gates and the delays of delay lines by the subgradient method. A timing-aware detailed placement approach is then proposed, where DLPlace introduces a row-wise gate order rearrangement method to reduce wirelength and timing violations in AQFP circuits. Furthermore, a dynamic programming approach is employed to achieve wirelength and timing legalization, thereby addressing the unique requirements of AQFP logic. The effectiveness of DLPlace is validated through AQFP benchmark experiments, demonstrating a significant reduction in both hardware footprint and circuit latency compared to the baselines. This new framework paves the way for the further optimization of AQFP circuit performance, offering a promising solution to the physical design challenges in superconductive electronics-based computing.

***Index Terms*—superconductor logic, AQFP, delay-line clocking, placement**

I. Introduction

Massive energy consumption has become an urgent concern in modern society, resulting in the imperative requirement for energy-efficient computing technologies. The power wall problem encountered by CMOS-based computing systems has accentuated the demand for energy-efficient computing systems, which in turn, has spurred the development of emerging technologies capable of providing comparable or superior performance to CMOS technology, but with less energy consumption. Among these technologies, Josephson junctions (JJs)-based superconducting logic circuits have garnered growing research attention. This includes the Rapid Single-Flux-Quantum (RSFQ) logic [1] and its more energy-efficient variants such as ERSFQ [2], Reciprocal Quantum Logic (RQL) [3] and Adiabatic Quantum-Flux-Parametron (AQFP) [4]. These emergent logic families are designed to specifically address the inherent static power problem present in RSFQ logic, with the primary goal of enhancing scalability.

Derived from Quantum-Flux-Parametron logic proposed in 1985 [5], AQFP logic significantly reduces energy dissipation. By re-parameterizing the device for adiabatic operation, AQFP achieves energy dissipation five or six orders of magnitude lower than its CMOS counterpart and boasts the highest energy efficiency among all superconducting logic families.

Despite its extraordinary energy efficiency, AQFP necessitates a unique design methodology, given its distinct data-propagation mechanism. Unlike CMOS, all AQFP gates operate in full synchronization with an AC power source that serves as both a power supply and a clock reference. Therefore, to facilitate data propagation in AQFP circuits, the utilization of multiple clock sources exhibiting phase differences is a prerequisite. This allows data to be conveyed across various logic stages during a substantial overlap of adjacent clock phases. On the other hand, logic bits in AQFP are represented by the superconductive current amplitude, which is delivered between logic gates by superconductive inductors, accounting for inevitable current attenuation. This requires a precise interconnect design to maintain an adequate bit-error rate, translating into different wire-length constraints for microstrip lines between various logic gates in the physical layout design. Consequently, these two distinctive requirements pose design difficulties, especially given the lack of support from automated design tools in the physical design domain.

To bridge the gap between logic design and physical design, methodologies for AQFP placement have been proposed in the past six years [6]–[9]. To resolve wire-length violations, the common solution involves inserting buffer rows to halve the original interconnect wire length. However, numerous buffers inserted result in more power consumption and significant circuit latency. Consequently, these aforementioned methods prioritize reducing violations while optimizing the total number of inserted buffer rows. Murai *et al.* [6] first employ a genetic algorithm (GA) to reposition gates within each logic row, aiming to reduce wire-length violations and subsequently decrease the number of inserted buffers, but suffers from unaffordable runtime, particularly for larger circuits, due to the computationally intensive nature of GA. ASAP [7] uses a method incorporating the analytical global placement and a row-wise detailed placement to significantly improve the

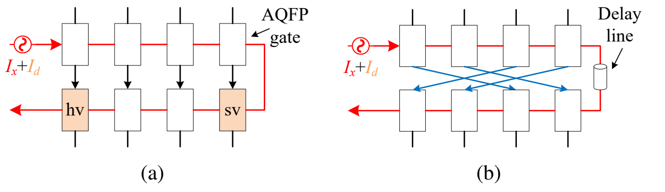

Fig. 1. Placement schematics of an AQFP circuit using the delay-line clocking scheme, where I_x is an AC clock signal, and I_d is a DC input for applying an offset flux to each gate. (a) adopts the 4-phase clocking-based placement method. For the bottom-right gate, its clock signal may arrive before its data signal due to the high-speed signal propagation, causing the setup time violation (sv); For the bottom-left gate, its next data signal may arrive before its previous clock signal due to the long clock wire, causing the hold time violation (hv). (b) shows a feasible method meeting its timing constraints by controlling the positions of gates and the delays of delay lines.

runtime. Li *et al.* [8] later introduce a GORDIAN algorithm-based placement framework tailored for a specially designed AQFP cell library, allowing more flexible routing spaces. As the first attempt to enable high-speed AQFP layout design, Dong *et al.* [9] propose TAAS, a placement framework further considering the timing violations introduced by the meandering clocking structure in the physical design of AQFP circuits.

In order to ensure adiabatic switching, AQFP gates are specifically designed to function at a maximum clock frequency of 5 GHz. However, the actual speed of data propagation between different logic stages hinges on the number of clock phases within a single clock cycle of 200 ps. For example, considering a 4-phase clocking operating at 5 GHz, data will traverse from the delivering stage to the receiving stage within 50 ps. Therefore, increasing the number of clock phases within a single cycle can significantly reduce the circuit latency. This enhancement not only accelerates data propagation but also leads to a substantial improvement in the circuit's overall performance. Presently, AQFP circuits utilize a 4-phase clocking method [10], favored for its simplicity and testing advantages. However, a drawback is its substantial circuit latency, which intensifies with increasing logic depth.

A recent delay-line-based approach [11] suggests enhancing the number of clock phases by up to ten times using a single AC source, a DC offset, and on-chip microstrip delay lines. This innovative method not only substantially improves circuit latency but also simplifies physical testing. In fact, a study [12] demonstrates that this delay-line clocking scheme significantly curtails the latency of an 8-bit ripple-carry adder, registering a decrease of about 60% compared to the 4-phase clocking scheme. However, the physical design of this clocking scheme calls for meticulous attention. This concern arises from the potential tenfold increase in effective data propagation speed between logic stages, amounting to 5 ps. Such a speed enhancement could lead to more severe timing violations between the arrival of the clock and data compared to the 4-phase clocking scheme, thereby making the delay-line clocking scheme unsuitable for existing placement frameworks. Fig. 1 illustrates a timing violation case and its feasible reposition approach in the delay-line clocking scheme.

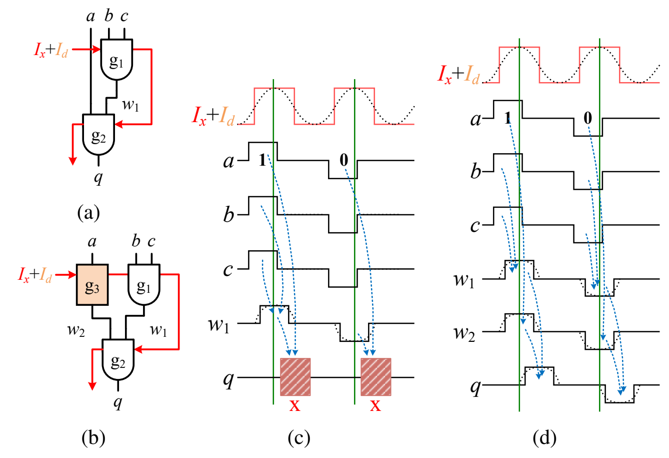

Fig. 2. A case shows the necessity of buffer insertion for correct operation in an AQFP circuit with the function $q = abc$ using the delay-line clocking scheme. (a) and (b) are gate-level schematics before and after buffer insertion, respectively, where a, b, and c are three data inputs, and q is a data output. (c) and (d) are timing schematics of (a) and (b), respectively.

To address the timing violation issue and support this innovative clocking scheme, this paper proposes DLPlace, the first placement framework specifically tailored for the delay-line clocking scheme. This framework aims to further elevate the performance of AQFP circuits to the next level.

This paper contributes significantly in the following ways:

- This paper proposes DLPlace, the first timing-aware placement framework for AQFP circuits using the delay-line clocking scheme. The framework aims to minimize circuit latency while ensuring adherence to wirelength and timing constraints.
- DLPlace formulates the global placement problem as the Lagrangian problem through Lagrangian relaxation concerning the maximum wirelength constraint and solves the Lagrangian subproblem by the subgradient method.
- To further enhance wirelength and timing optimization, DLPlace proposes a row-wise gate order rearrangement method to reduce wirelength and timing violations, along with a dynamic programming method to achieve wirelength and timing legalization.
- Experimental results on AQFP benchmarks illustrate DLPlace's effectiveness for the delay-line clocking-based placement of AQFP circuits in eliminating wirelength and timing violations and minimizing the circuit latency.

II. Preliminaries

A. AQFP Logic

AQFP circuits are realized by adiabatic superconductors and can operate with energy dissipation close to the thermodynamic and quantum limits [13], rendering it a promising candidate for building extremely energy-efficient computing

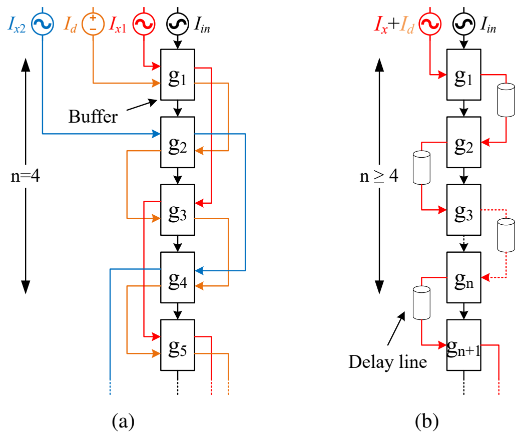

Fig. 3. Schematics of AQFP buffer chains adopting (a) 4-phase clocking scheme and (b) delay-line clocking scheme, where I_{x1}, I_{x2}, and I_x are AC clock signals, I_d is a DC input, I_{in} is the data input, and n indicates the number of clock stages of the signal propagation in one clock cycle.

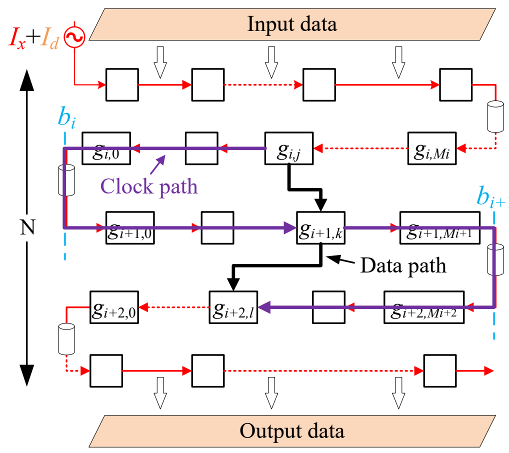

Fig. 4. Delay-line clocking-based placement schematic of an AQFP circuit, where b_i is gate row r_i's horizontal boundary.

systems. Since AQFP logic gates are driven by AC power, the propagation of a signal in AQFP logic is performed using multi-phase excitation clocks. To ensure the correct operation of AQFP circuits, buffers must be inserted so that all inputs to each logic gate have the same clock stages (named logic level), i.e., path balancing [14], [15]. As illustrated in Fig. 2, two inputs of gate $g2$ in Fig. 2(a) have different logic levels. When the clock signal arrives at gate $g2$, it has a large delay relative to input a. So, the output of $g2$ in Fig. 2(c) may be undetermined [16]. After buffer g_3 is inserted in Fig. 2(b), gate $g2$ can output '1' correctly in Fig. 2(d). Four-phase clock scheme and delay-line clock scheme are widely used in AQFP circuit design. Take an example of the AQFP buffer chains in Fig. 3. The 4-phase clocking scheme requires two AC clock signals, allowing for signal propagation through four clock stages ($n = 4$) in one clock cycle. In contrast, the delay-line clocking scheme only requires one AC clock signal, enabling signal propagation through four or more clock stages ($n \geq 4$) in one clock cycle by the delay configuration of delay lines. Hence, in cases where a data signal is fed into two buffer chains with an equal number of buffers, the chain using the delay-line clocking scheme can output it faster than that using the 4-phase clocking scheme. In other words, the use of the delay-line clocking scheme results in smaller circuit latency.

B. Placement Constraints of AQFP Logic

1) Timing Constraints: Unlike the 4-phase clocking scheme, the delay-line clocking scheme has strict timing constraints and usually needs to adjust the arrival time of the clock signal to each gate by delay lines. As shown in Fig. 4, the circuit is divided into N gate rows according to the logic level, where the i^{th} gate row r_i has M_i gates, $i \in [1, N]$. The clock routing of the delay-line clocking scheme follows a zigzag pattern, with even rows moving from right to left and odd rows moving from left to right. To ensure the correct operation of the circuit, the gate $g_{i+1,k}$ with setup time $st_{i+1,k}$ and hold time $ht_{i+1,k}$ must the following timing constraints:

- Setup time constraint

$$ct_{i,j;i+1,k} - dt_{i,j;i+1,k} \geq st_{i+1,k} + qt_{i,j} - lt_i, \quad (1)$$

- Hold time constraint

$$ct_{i,j;i+1,k} - dt_{i,j;i+1,k} \leq T - ht_{i+1,k} + qt_{i,j} - lt_i, \quad (2)$$

where $ct_{i,j;i+1,k}$ is the delay of the clock path between $g_{i,j}$ and $g_{i+1,k}$, $dt_{i,j;i+1,k}$ is the delay of the data path between $g_{i,j}$ and $g_{i+1,k}$, $qt_{i,j}$ is the delay from the receipt of the clock signal by gate $g_{i,j}$ to the release of its output signal, and lt_i is the delay of the delay line d_i between the i^{th} gate row r_i and the $(i+1)^{\text{th}}$ gate row r_{i+1}. Besides, T is the clock cycle of the circuit, usually set to $200\,\text{ps}$ in AQFP logic.

2) Wirelength Constraints: Each data wire in AQFP circuits consists of vertically and horizontally straight segments, corner segments, and intersection segments. The wirelength, defined as the total length of these segments, is a crucial consideration that is explored in this paper. Since each net in AQFP circuits has only two terminals, its wirelength is calculated as the half-perimeter wire length (HPWL). Different AQFP gates have varying output current magnitudes that determine the maximum length of their output wires. Due to signal current attenuation, it is crucial to restrain the length of each wire from surpassing its maximum limit. As shown in Fig. 5, when a wirelength violation occurs (orange wire), a buffer d insertion is required to split the wire. Considering the path balancing requirement, a buffer row r_{i+1} needs to insert between gate rows r_i and r_{i+1} (which becomes r_{i+2}) shown in Fig. 5(b).

C. Problem Formulation

This paper focuses on the placement of AQFP circuits using the delay-line clocking scheme. The process involves determining the delay of the delay line between gate rows and then establishing the position of each gate. The ultimate goal is to meet wirelength and timing constraints while minimizing

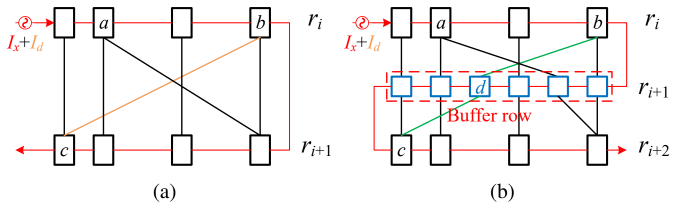

Fig. 5. Buffer row insertion for wirelength violations. (a)The length of the orange wire exceeds its maximum length. (b) A buffer row is inserted to eliminate this wirelength violation.

circuit latency. Hence, the delay-line clocking-based placement of AQFP circuits can be formulated as follows:

- Input:
  1) A given path-balanced AQFP netlist $G(V, E)$, where V is the gate set, and E is the set of data nets.
  2) A cell library, where the unit wirelength is l_μ, the signal propagation speed is v_w.
- Output:
  1) A legalized top-left position (x_v, y_v) of each gate v, $v \in V \cup B$, where B is the set of buffers inserted for wirelength violations.
  2) The delay lt_i of each delay line $d_i, i \in [1, N)$.
- Constraints:
  1) $\forall u, v \in r_i, y_u = y_v$, ensuring that gates in the same gate row have the same vertical position.
  2) Timing constraints shown in Equations (1)–(2).
  3) Wirelength constraints of data wires:

$$dl_{i,j;i+1,k} \le l_{max_{i,j}}, i \in [1, N), j \in [1, M_i], k \in [1, M_{i+1}], \quad (3)$$

  where $dl_{i,j;i+1,k}$ is the wirelength of the data path between gate $g_{i,j}$ and gate $g_{i+1,k}$, and $l_{max_{i,j}}$ is the maximum fan-out wirelength of gate $g_{i,j}$.
  4) Overlap constraints within the gate row:

$$x_{i,j} + w_{i,j} \le x_{i,j+1}, i \in [1, N], j \in [1, M_i), \quad (4)$$

  where $w_{i,j}$ denotes the width of gate $g_{i,j}$.
- Goal:
  Minimize the circuit latency of the generated delay-line clocking-based placement while meeting wirelength and timing constraints, formulated as follows:

$$\min \sum_{i\in[1,N)} lt_i + \sum_{i\in[1,N]} w_i/v_w, \quad (5)$$

  where w_i denotes the width of gate row r_i, calculated as

$$w_i = \begin{cases} b_i - b_{i-1}, & \text{if } i \bmod 2 == 1, \\ b_{i-1} - b_i, & \text{otherwise,} \end{cases} \quad (6)$$

  and

$$b_i = \begin{cases} x_{i,M_i} + w_{i,M_i} + l_\mu, & \text{if } i \bmod 2 == 1, \\ x_{i,1} - l_\mu, & \text{otherwise.} \end{cases} \quad (7)$$

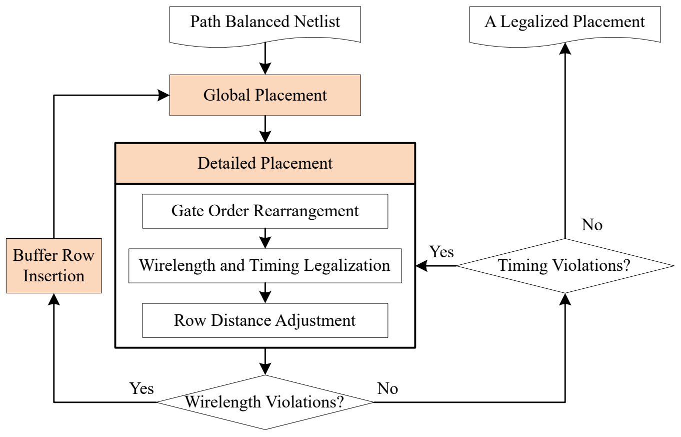

Fig. 6. The placement flow of DLPlace framework.

III. DLPlace

This section describes the implementation details of the proposed placement framework, whose overall flow is shown in Fig. 6, composed of three parts: global placement, detailed placement, and buffer row insertion.

A. Global Placement

For the placement of AQFP circuits using the delay-line clocking scheme, all attentions focus on wirelength violations, timing violations, and circuit latency. In the global placement phase, there are two main tasks: i) A refinement of gate positions with a global perspective and ii) a refinement of the delays of delay lines with a global perspective, which can reduce wirelength and timing violations while minimizing the circuit latency. To simplify the problem, DLPlace assumes that the gate order of each gate row and the distance between each two adjacent gate rows are fixed in the global placement phase, which will be updated in the detailed placement phase. Besides, since the locations of all pins in each gate are fixed, they are ignored in all equations to simplify the expressions. So, the wirelength $cl_{i,j;i+1,k}$ of the clock path and the wirelength $dl_{i,j;i+1,k}$ of the data path between gate $g_{i,j}$ and gate $g_{i+1,k}$ can be respectively calculated as

$$cl_{i,j;i+1,k} = \begin{cases} 2 * b_i - x_{i,j} - (x_{i+1,k} + w_{i+1,k}) + y_{i+1} - y_i, & \text{if } i \bmod 2 == 1, \\ (x_{i,j} + w_{i,j}) + x_{i+1,k} + y_{i+1} - y_i - 2 * b_i, & \text{otherwise,} \end{cases} \quad (8)$$

and

$$dl_{i,j;i+1,k} = |x_{i,j} - x_{i+1,k}| + y_{i+1} - y_i - h_{i,j}, \quad (9)$$

where $h_{i,j}$ is the height of gate $g_{i,j}$, and their delays can further be calculated as

$$ct_{i,j;i+1,k} = cl_{i,j;i+1,k}/v_w + lt_i, \quad (10)$$

$$dt_{i,j;i+1,k} = dl_{i,j;i+1,k}/v_w. \quad (11)$$

Hence, the circuit latency for the global placement can be formulated as follows:

$$\min \sum_{i\in[1,N)} lt_i + \sum_{i\in[1,N]} w_i/v_w, \quad (12)$$

$$\text{s.t.}\quad \text{Equations (1)} - \text{(2)}: \text{Timing constraints}, \tag{13}$$
$$\text{Equations (3)}: \text{Wirelength constraints}, \tag{14}$$
$$\text{Equation (4)}: \text{Overlap constraints}. \tag{15}$$

However, considering both wirelength constraints and timing constraints, this optimization problem is very difficult to solve. Equation (3) is relaxed into the objective function with Lagrangian multiplier λ_e, and the global placement can be formulated as a Lagrangian problem $\mathcal{L}(x, lt, \lambda)$ as follows

$$\min \sum_{i\in[1,N)} lt_i + \sum_{i\in[1,N]} w_i/v_w + \sum_{e\in E} \lambda_e(l_e - l_{max_e}), \tag{16}$$
$$\text{s.t.}\quad \text{Equations (1), (2), (4)}, \tag{17}$$
$$\lambda_e \geq 0, \tag{18}$$

where Lagrangian multiplier λ_e is updated by the subgradient method at the k^{th} iteration:

$$\lambda_e^{k+1} = \max\left\{0, \lambda_e^k + s_k(l_e - l_{max_e})\right\}, e \in E, \tag{19}$$

where s_k determines the step size of updating, which is used to control the solution quality and convergence speed. The convergence condition described in [17] suggests $s_k = \frac{1}{k^\alpha}$ with a constant value of $\alpha < 1$ to effectively achieve this goal. Moreover, to further optimize the solution speed, the introduction of extra variables $x^+_{i,j,k} \geq 0, x^-_{i,j,k} \geq 0$ can allow for the linearization of the absolute value operations required in the calculation of data wirelength by the following transformation:

$$x_{i,j} - x_{i+1,k} = x^+_{i,j,k} - x^-_{i,j,k}, \tag{20}$$
$$|x_{i,j} - x_{i+1,k}| = x^+_{i,j,k} + x^-_{i,j,k}. \tag{21}$$

It is worth noting that this transformation is equivalent because the objective is to minimize and $\lambda_e \geq 0$ [18]. Ultimately, solving this model can yield the initial positions of all gates and the delays of all delay lines.

B. Detailed Placement

Following the global placement phase, wirelength and timing violations may arise, predominantly due to the suboptimal ordering of gates within each gate row. To address these issues, the detailed placement phase aims to re-order gates according to the wirelength and timing constraints while proposing a dynamic programming method to rectify violations. Besides, the distance between adjacent gate rows is updated via channel routing, thereby adjusting the vertical position of each gate.

1) Gate Order Rearrangement: This paper proposes a heuristic algorithm that aims to rearrange the gate order in each gate row as a means to address violations in timing and wirelength constraints. As detailed in Algorithm 1 with the time complexity $O(|E|)$, the algorithm first calculates the weighted horizontal positions of each gate in a given row according to its wirelength and timing constraints (lines 2-32), and then re-orders all gates in gate row r_i according to their weighted horizontal positions (line 33). The weight applied to each position calculation is a critical design consideration, as it must account for both timing and wirelength constraints.

Algorithm 1: Gate order rearrangement.

```
Input: $N$ gate rows.
Output: Re-ordered gate rows.
1  for $i = 1 \to N,\ j = 1 \to M_i$ do
2      $update$ = True, $tw = 0, sx = 0$.
3      for $e$ in data edges of gate $g_{i,j}$ do
4          $l = dl_e$, $vio$ = True.
5          $t_{rs} = qt_{e_s} + dt_e + st_{e_t}$.
6          $t_{rh} = T + qt_{e_s} + dt_e - ht_{e_t}$.
7          $x = e_s == g_{i,j}\ ?\ x_{e_t} : x_{e_t}$.
8          if $e$ has setup time violation then
9              $\Delta = |ct_{e_t} - t_{rs}|$.
10             if $e_s == g_{i,j}$ then
11                 $x = x_{e_s} + (i \bmod 2 == 1\ ?\ -\Delta : \Delta)$.
12                 $l$ += $-|x_{e_s} - x_{e_t}| + |x - x_{e_t}|$.
13             else
14                 $x = x_{e_t} + (i \bmod 2 == 1\ ?\ \Delta : -\Delta)$.
15                 $l$ += $-|x_{e_s} - x_{e_t}| + |x - x_{e_s}|$.
16         else if $e$ has hold time violation then
17             $\Delta = |ct_{e_t} - t_{hs}|$.
18             if $e_s == g_{i,j}$ then
19                 $x = x_{e_s} + (i \bmod 2 == 1\ ?\ \Delta : -\Delta)$.
20                 $l$ += $-|x_{e_s} - x_{e_t}| + |x - x_{e_t}|$.
21             else
22                 $x = x_{e_t} + (i \bmod 2 == 1\ ?\ -\Delta : \Delta)$.
23                 $l$ += $-|x_{e_s} - x_{e_t}| + |x - x_{e_s}|$.
24         else if $e$ has no violation then
25             $vio$ = False.
26         if $vio$ then
27             $update$ = True.
28         $weight = \max(1, l/l_{max_e})$.
29         $tw$ += $weight$.
30         $sx$ += $x * weight$.
31     if $update$ then
32         $x_{i,j} = sx/tw$.
33 re-order the gates in each gate row by their abscissa.
```

Take an input edge e of gate $g_{i,j}$ as an example, where e_s and e_t are the source and target of edge e. Firstly, some metrics are calculated (lines 4-7), including the current wirelength l of edge e, the earliest clock arrival time t_{rs} required to satisfy the setup time constraint, and its latest clock arrival time t_{rh} required to satisfy the setup time constraint. If edge e cause the setup violation, the new horizontal position x of gate $g_{i,j}$ is estimated by the difference between the actual clock arrival time ct_{e_t} and t_{rs}, and wirelength l is also adjusted accordingly (lines 8-15). If edge e cause the hold time violation, the operation is similar (lines 16-23). Finally, the weight $weight$ assigned to new position x is set to the ratio of current wirelength l and its maximum wirelength l_{max_e} of edge e (line 28). The final horizontal position of gate $g_{i,j}$ is determined by

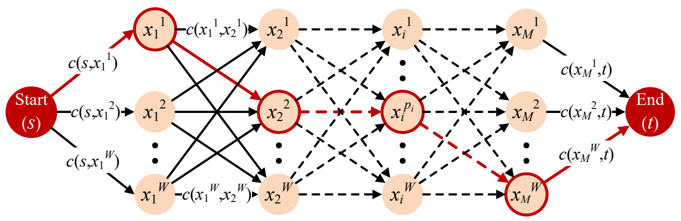

Fig. 7. Trellis for an in-row gate placement process, where the red line is the Viterbi path, i.e. the optimal placement solution.

the weighted average of these new positions (lines 29-32).

2) Wirelength and Timing Legalization: After the gate order rearrangement phase, wirelength and timing legalization become necessary to account for the changes in the horizontal positions of gates. So, this paper proposes a row-wise legalization algorithm based on the Viterbi algorithm [19], which is a dynamic programming algorithm used to find the most likely sequence (called the Viterbi path) of hidden states in a hidden Markov model. As shown in Fig. 7, each gate has W candidate positions in the gate row with M gates. Based on the position of the $(i-1)^{\text{th}}$ gate, the position of the i^{th} gate can be determined using the Viterbi algorithm. Firstly, a triplet $\{wv, tv, wl\}$ is defined as the cost of placing the i^{th} gate, where wv is the number of wirelength violations of the i^{th} gate, tv is the number of timing violations of the i^{th} gate, and wl is the total wirelength connected to the i^{th} gate. The comparison between two triplets is made by the lexicographic order, and the addition of two triplets is defined as the addition of corresponding elements separately. The cost $c(x_{i-1}^{p_{i-1}}, x_i^{p_i})$ of each edge is defined as the cost sum of the first $(i-1)$ gates, where the $(i-1)^{\text{th}}$ gate is placed at p_{i-1} and the i^{th} gate is placed at p_i. So, the minimum cost $c(x_i^{p_i})$ of placing i^{th} gate at p_i can be formulated as

$$c(x_i^{p_i}) = \min_{p_{i-1}=1}^{p_i - \lceil w_{i-1}/l_\mu \rceil} c(x_{i-1}^{p_{i-1}}, x_i^{p_i}) + cost(x_i^{p_i}), \tag{22}$$

where w_{i-1} is the width of the $(i-1)^{\text{th}}$ gate.

Algorithm 2 describes the complete process of the wirelength and timing legalization for a given gate row, whose time complexity of this algorithm is $O(M*W)$. The algorithm first calculates the total gate width W_c of all gates and the available width W for the in-row placement (lines 1-2), and then defines variables dp and $prev$ to record the cost and source of each edge in Fig. 7. Next, the algorithm places the i^{th} gate according to the placement of the $(i-1)^{\text{th}}$ gate to minimize the cost (lines 4-20). Finally, the algorithm updates the positions of all gates according to $prev$ and returns the minimum cost for the placement of the given gate row (lines 21-28).

3) Row Distance Adjustment: In consideration of the impact of the distance between adjacent gate rows on circuit latency, the vertical position of each gate is applied to the calculation of the wirelength and timing to strictly meet wirelength and timing constraints in this paper. All gates in the same gate row are assumed to share the same vertical position, and the distance between adjacent gate rows can be estimated by the number of tracks required by the channel routing.

Algorithm 2: Wirelength and timing legalization.

```
Input: A gate row with $M$ gates, left boundary $b_l$, and
       right boundary $b_r$.
Output: The total cost.
1  calculate the total gate width $W_c$ of all gates.
2  $W = \lfloor (b_r - b_l)/l_\mu \rfloor$, $cw = 0, nx = b_l$.
3  $dp[M][W] = \{\infty, \infty, \infty\}$, $prev[M][W] = -1$.
4  for $i = 1 \to M$ do
5      $pc = \{\infty, \infty, \infty\}, pi = -1$.
6      for $j = \lceil cw/l_\mu \rceil \to W - \lceil (W_c - cw)/l_\mu \rceil$ do
7          $x_i = b_l + j * l_\mu$.
8          $nx = x_i + w_i$.
9          if $nx > b_r$ then
10             break
11         calculate the cost $c$ of current gate placed at $j$.
12         if $i > 1$ then
13             $li = \lfloor (x_i - w_{i-1} - b_l)/l_\mu \rfloor$.
14             if $li \neq prev[i-1][li]$ then
15                 $li = \lfloor (x_i - l_\mu - w_{i-1} - b_l)/l_\mu \rfloor$
16             $c$ += $dp[i-1][li]$.
17         if $(i == 1 \wedge j == 1) \vee c < pc$ then
18             $pc = c$, $pi = j$.
19         $dp[i][j] = pc$, $prev[i][j] = pi$.
20     $cw$ += $w_i$.
21 $res = \{\infty, \infty, \infty\}$.
22 for $i = M \to 1$ do
23     $j = \lfloor (nx - w_i - b_l)/l_\mu \rfloor$.
24     if $i == M$ then
25         $res = dp[i][j]$.
26     $nx = b_l + prev[i][j] * l_\mu$.
27     $x_i = nx$.
28 return $res$
```

However, given that the wire congestion between adjacent gate rows may be altered after carrying out the gate order rearrangement, it is necessary to update the distance between adjacent gate rows through channel routing to effectively reduce the circuit latency.

C. Buffer Row Insertion

Upon completion of the detailed placement phase, wirelength violations may still arise, primarily due to the insufficient fan-out driving force of AQFP gates, particularly AQFP splitters. Since AQFP buffers possess a significant fan-out driving force, few JJs, and a small area, they can be inserted to eliminate wirelength violations. According to the description of Section II-B2, buffer rows need to insert between adjacent gate rows in cases where wirelength violations exist, with the horizontal position of each buffer set as the average of horizontal positions of the source and target of its corresponding wire. After this buffer insertion phase, DLPlace proceeds with the next iteration and continues to perform the global placement.

Algorithm 3: DLPlace algorithm.

```
Input: A path-balanced netlist $G(V, E)$, $T_L$, $T_D$, $T_R$.
Output: A corresponding legal placement.
1  group all gates into $N$ gate rows by the logic level.
2  initialize the gate order of each gate row and the
     distance between adjacent gate rows.
3  $\lambda_e = 1, \forall e \in E$
4  while wirelength or timing violations exist do
5      $t_L = 0$
6      while not converge $\wedge\ t_L$ ++ $< T_L$ do
7          solve the Lagrangian subproblem $\mathcal{L}(x, lt, \lambda)$
8          $t_D = 0$
9          while violations exist $\vee\ t_D$ ++ $< T_D$ do
10             $t_R = 0$
11             while violations exist $\vee\ t_R$ ++ $< T_R$ do
12                 re-order the gate order of each gate row.
13                 adjust row spacing.
14             legalize wirelength and timing violations.
15         update $\lambda_e, \forall e \in E$
16     insert buffer rows.
```

This paper has presented an overview of all the main phases involved in DLPlace. Algorithm 3 shows the complete algorithmic procedure of the delay-line clocking-based placement, where T_L, T_D, and T_R are used to control the iteration number of the Lagrangian relaxation, the detailed placement, and the gate order rearrangement, respectively. Initially, an initial placement is generated (lines 1-2), with Lagrangian multipliers initialized to 1 (line 3). The algorithm then enters the iteration process (lines 4-16). Firstly, DLPlace determines initial positions for all gates and delays for all delay lines through the adoption of Lagrangian relaxation. The algorithm then proceeds to the next iteration process (lines 9-14), which mainly involves the adjustment of the gate order of each gate row and the distance between adjacent gate rows. Following this layer iteration, violations are further eliminated by the dynamic programming algorithm, and the Lagrangian multipliers are updated (line 15). Upon completing the iteration of solving the Lagrangian problem, buffer rows are inserted wherever wirelength violations exist (line 16). In this way, DLPlace completes the placement process of AQFP circuits using the delay-line clocking scheme.

IV. Experimental Results

This section presents the implementation and evaluation of the proposed delay-line clocking-based placement framework for AQFP circuits. The proposed framework was developed using C++, and its execution was carried out on a machine running CentOS Linux 7, equipped with an Intel(R) Core(TM) i9-9900X CPU @ 3.50GHz and 125 GB memory. To solve the global placement problem, Gurobi [20] is selected as the solver due to its superior performance, with version 10.0.0 being utilized. The benchmark circuits are obtained from [9] and include an 8-bit Kogge-Stone adder (adder8), decoder,

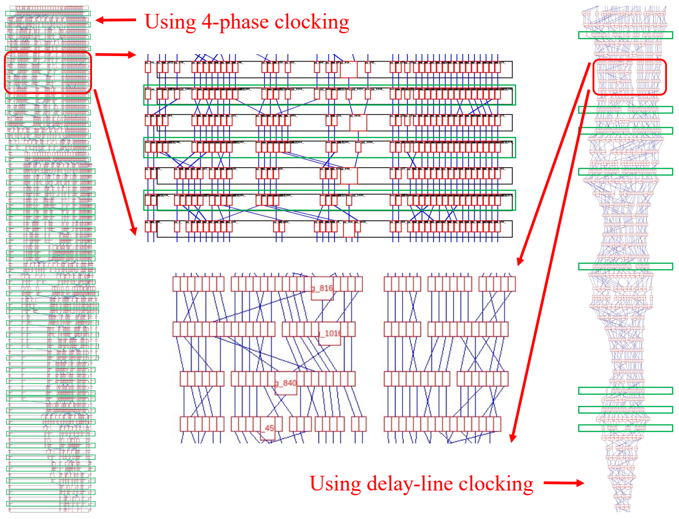

Fig. 8. The placement results of circuit c432, where small red rectangles are gates, each large green box marks a buffer row, and two zoom-in sub-figures show the placement results from the 5^{th} row to the 8^{th} row in the input circuit.

32-bit sorter (sorter32), 32-bit approximate parallel counter (apc32), 128-bit approximate parallel counter (apc128), as well as several ISCAS'85 benchmark circuits [21].

The utilization of the delay-line clocking scheme in physical design research on AQFP circuits has been limited due to two primary reasons: (i) the absence of prior research in this specific area and (ii) the challenges associated with the direct application of existing CMOS-based physical design frameworks on AQFP circuits utilizing the delay-line clocking scheme. As a result, the baselines in this paper use state-of-the-art methods for AQFP circuits using the 4-phase clocking scheme, including the GORDIAN-based AQFP placement method (GORDIAN-based) [8] and the timing-aware placement method (TAAS) [9]. As discussed in Section I, the GORDIAN-based method seeks to minimize the number of buffer rows inserted to address wirelength violations, while TAAS also further considers the timing constraints associated with the 4-phase clocking scheme.

Table I shows the detailed comparison results between the baselines and DLPlace, where the clock cycle of AQFP circuits was assumed as 200 ps. The parameters T_L, T_D, and T_R of DLPlace were set to 20, 2, and 10, respectively. The "Original" part displays the fundamental characteristics of input circuits, including the number of AQFP logic gates, denoted as "Gates", and the circuit depth, denoted as "Depth". On the other hand, the "GORDIAN-based" part shows the results of the 4-phase clocking-based method only targeting eliminating wirelength violations. The "BRI" column represents the number of buffer rows inserted to satisfy the wirelength constraint. The "Violations" column represents the total number of timing or wirelength violations under the clock cycle of 200 ps. The "Latency" column represents the circuit latency of the generated placement. Since the signal can propagate through four clock phases in one clock cycle in the 4-phase clocking scheme, their

TABLE I. The comparison between baselines and DLPlace with $T_L = 20$, $T_D = 2$ and $T_R = 10$ at a clock cycle of 200 ps.

Circuit	Original		GORDIAN-based [8]			TAAS [9]			DLPlace		
	Gates	Depth	BRI	Violations	Latency (ps)	BRI	Violations	Latency (ps)	BRI	Violations	Latency (ps)
adder8	446	25	24	0	2450	24	0	2450	0	0	594
apc32	379	27	26	0	2650	26	0	2650	0	0	625
apc128	1734	45	117	2157	8100	110	317	7750	24	0	2401
c432	1120	42	46	0	4400	45	0	4350	8	0	1310
c499	1908	31	62	1978	4650	62	234	4650	40	0	2926
c1355	1924	31	58	2011	4450	58	257	4450	32	0	2631
c1908	2212	39	67	1720	5300	66	213	5250	32	0	2681
decoder	764	20	34	360	2700	33	26	2650	13	0	1092
sorter32	960	30	29	274	2950	29	29	2950	27	0	2046

circuit latency can be estimated as $(\text{Depth} + \text{BRI})/4 * 200$. In addition, the "TAAS" part shows the results of the 4-phase clocking-based timing-aware method targeting satisfying both wirelength and timing constraints. The results of both methods are sourced from [9]. The "DLPlace" part shows the results of the proposed delay-line clocking-based method, where the latency is calculated as the total propagation delay of a clock signal from the first gate of the first row to the last gate of the last row. Compared to the baselines, there was no violation in all results of DLPlace for all testcases, while there were still a lot of violations in the results of the baselines. Meanwhile, DLPlace had a significant reduction in the number of buffer rows inserted and the circuit latency, specifically by 62.59% and 56.70% over "GORDIAN-based", and by 62.19% and 56.38% over "TAAS", respectively. This result shows that the DLPlace framework is effective for AQFP placement. Besides, Fig. 8 shows the placement result of circuit c432. For the baseline result [9] in the left part, a buffer row is inserted in almost every two adjacent gate rows, while only eight buffer rows are inserted in total in the DLPlace's result in the right part. This result further demonstrated DLPlace's significant advantage in the number of buffer rows inserted.

V. Conclusion

This paper introduces the placement problem of AQFP circuits using the delay-line clocking scheme, including wirelength and timing constraints. Meanwhile, this paper proposed DLPlace, the first delay-line clocking-based placement framework for AQFP circuits, which can eliminate wirelength and timing violations while minimizing the circuit latency. The experimental results on the AQFP logic benchmark circuits show the effectiveness of the proposed algorithm, making the generated placement satisfy the wirelength and timing constraints of the delay-line clocking scheme. Meanwhile, the number of buffer rows inserted and the circuit latency are significantly reduced compared to the baselines.

Acknowledgment

The research work described in this paper was conducted in the Intelligent Design Automation Lab funded by The Hong Kong Jockey Club Charities Trust.

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] D. E. Kirichenko, S. Sarwana, and A. F. Kirichenko, "Zero static power dissipation biasing of RSFQ circuits," *IEEE Transactions on Applied Superconductivity*, vol. 21, no. 3, pp. 776–779, 2011.

[3] Q. P. Herr, A. Y. Herr, O. T. Oberg, and A. G. Ioannidis, "Ultra-low-power superconductor logic," *Journal of applied physics*, vol. 109, no. 10, p. 103903, 2011.

[4] N. Takeuchi, D. Ozawa, Y. Yamanashi, and N. Yoshikawa, "An adiabatic quantum flux parametron as an ultra-low-power logic device," *Superconductor Science and Technology*, vol. 26, no. 3, p. 035010, 2013.

[5] K. Loe and E. Goto, "Analysis of flux input and output josephson pair device," *IEEE Trans. Magn.*, vol. 21, no. 2, pp. 884–887, 1985.

[6] Y. Murai, C. L. Ayala, N. Takeuchi, Y. Yamanashi, and N. Yoshikawa, "Development and demonstration of routing and placement EDA tools for large-scale adiabatic quantum-flux-parametron circuits," *IEEE Transactions on Applied Superconductivity*, vol. 27, no. 6, pp. 1–9, 2017.

[7] Y.-C. Chang, H. Li, O. Chen, Y. Wang, N. Yoshikawa, and T.-Y. Ho, "ASAP: An analytical strategy for AQFP placement," in *ICCAD*, pp. 1–7, 2020.

[8] H. Li, M. Sun, T. Zhang, O. Chen, N. Yoshikawa, B. Yu, Y. Wang, and Y. Lin, "Towards AQFP-capable physical design automation," in *DATE*, pp. 954–959, 2021.

[9] P. Dong, Y. Xie, H. Li, M. Sun, O. Chen, N. Yoshikawa, and Y. Wang, "TAAS: A timing-aware analytical strategy for AQFP-capable placement automation," in *DAC*, pp. 1321–1326, 2022.

[10] N. Takeuchi, S. Nagasawa, F. China, T. Ando, M. Hidaka, Y. Yamanashi, and N. Yoshikawa, "Adiabatic quantum-flux-parametron cell library designed using a 10 ka cm^2 niobium fabrication process," *Superconductor Science and Technology*, vol. 30, no. 3, p. 035002, 2017.

[11] N. Takeuchi, M. Nozoe, Y. He, and N. Yoshikawa, "Low-latency adiabatic superconductor logic using delay-line clocking," *Applied Physics Letters*, vol. 115, no. 7, p. 072601, 2019.

[12] T. Yamae, N. Takeuchi, and N. Yoshikawa, "An adiabatic quantum-flux-parametron 8-bit ripple carry adder using delay-line clocking," *IEEE Transactions on Applied Superconductivity*, vol. 33, no. 5, pp. 1–4, 2023.

[13] N. Takeuchi, Y. Yamanashi, and N. Yoshikawa, "Energy efficiency of adiabatic superconductor logic," *Superconductor Science and Technology*, vol. 28, no. 1, p. 015003, 2014.

[14] S.-Y. Lee, H. Riener, and G. De Micheli, "Beyond local optimality of buffer and splitter insertion for AQFP circuits," in *DAC*, pp. 445–450, 2022.

[15] R. Fu, M. Wang, Y. Kan, N. Yoshikawa, T.-Y. Ho, and O. Chen, "A global optimization algorithm for buffer and splitter insertion in adiabatic quantum-flux-parametron circuits," in *ASPDAC*, pp. 769–774, 2023.

[16] Q. Xu, C. L. Ayala, N. Takeuchi, Y. Yamanashi, and N. Yoshikawa, "Hdl-based modeling approach for digital simulation of adiabatic quantum flux parametron logic," *IEEE Transactions on Applied Superconductivity*, vol. 26, no. 8, pp. 1–5, 2016.

[17] M. Held, P. Wolfe, and H. P. Crowder, "Validation of subgradient optimization," *Math. Program.*, vol. 6, no. 1, pp. 62–88, 1974.

[18] D. Bertsimas and J. Tsitsiklis, *Introduction to Linear Optimization*. Athena Scientific, 1997.

[19] G. Forney, "The viterbi algorithm," *Proceedings of the IEEE*, vol. 61, no. 3, pp. 268–278, 1973.

[20] Gurobi Optimization, LLC, "Gurobi Optimizer Reference Manual." https://www.gurobi.com, 2023.

[21] F. Brglez and H. Fujiwara, "A neutral netlist of 10 combinational benchmark circuits and a target translator in Fortran," in *ISCAS*, pp. 677–692, 1985.