TAAS: A Timing-Aware Analytical Strategy for AQFP-Capable Placement Automation

Peiyan Dong*[1] Yanyue Xie*[1] Hongjia Li*[1] Mengshu Sun[1] Olivia Chen[2] Nobuyuki Yoshikawa[3] Yanzhi Wang[1,4]
[1]Northeastern University [2]Tokyo City University [3]Yokohama National University [4]CoCoPIE LLC
[1]{dong.pe, xie.yany, li.hongjia, sun.meng, yanz.wang}@notheastern.edu, [2]olivia.chen@ieee.org, [3]nyoshi@ynu.ac.jp

ABSTRACT

Adiabatic Quantum-Flux-Parametron (AQFP) is a superconducting logic with extremely high energy efficiency. AQFP circuits adopt the deep pipeline structure, where the four-phase AC-power serves as both the energy supply and the clock signal and transfers the data from one clock phase to the next. However, the deep pipeline structure causes the stage delay of the data propagation is comparable to the delay of the zigzag clocking, which triggers timing violations easily. In this paper, we propose a timing-aware analytical strategy for the AQFP placement, TAAS, that immensely reduces timing violations under specific spacing constraints and wirelength constraints of AQFP. TAAS includes two main characteristics: 1) a timing-aware objective function that incorporates a four-phase timing model for the analytical global placement. 2) a unique detailed placement including the timing-aware dynamic programming technique and the time-space cell regularization. To validate the effectiveness of TAAS, various representative circuits are adopted as benchmarks. As shown in the experimental results, our strategy can increase the maximum operating frequency by up to 30% ∼ 40% with a negligible wirelength increase -3.41%∼1%.

ACM Reference Format: Peiyan Dong, Yanyue Xie, Hongjia Li, Mengshu Sun, Olivia Chen, Nobuyuki Yoshikawa, Yanzhi Wang. 2022. TAAS: A Timing-Aware Analytical Strategy for AQFP-Capable Placement Automation. In *2022 59th ACM/IEEE Design Automation Conference (DAC), July 2022, San Francisco, CA, USA.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3489517.3530487

1 INTRODUCTION

AQFP is a superconducting logic with very low energy dissipation, which can play as a promising energy-efficient alternative to complementary metal-oxide-semiconductor (CMOS) circuits. AQFP logic delivers extraordinary high energy efficiency by adopting adiabatic switching [9], in which the potential energy profile evolves from a single well to a double well so that the logic state can change quasi-statically. AQFP circuits use the fabrication processes such as the AIST standard process 2 (STP2) and the MIT-LL SFQ process [11]. It can potentially achieve $10^4 - 10^5$ energy-efficiency gain compared with state-of-the-art CMOS and operate at a clock frequency of several GHz [3].

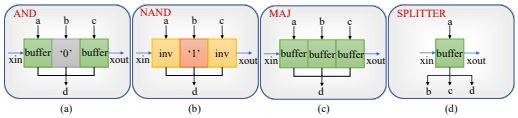

Figure 1: (a) The symbol of AND gate; (b) The symbol of NAND gate; (c) The symbol of MAJ gate; (d) The symbol of 1-to-3 splitter [3].

To deliver the promising energy efficiency, it is crucial to investigate the choice of logic gates, circuits, architecture and develop efficient design automation tools for AQFP circuits. However, AQFP differs from CMOS circuits in terms of active components (transistor in CMOS vs. Josephson Junction (JJ) in AQFP), passive components, logic gates (AQFP relies on the efficient realization of majority/minority gates), clocking scheme (four-phase clocking scheme in AQFP), and data propagation. Moreover, the timing requirements of AQFP differ from CMOS as well. In specific, multi-level gates are supposed to meet timing constraints as a group in CMOS circuits, i.e multi-level gates for pipelining, while each gate must satisfy the timing in AQFP circuits, i.e the gate-level pipelining. So the currently mature design automation tools for CMOS cannot be directly applied to the design of superconducting electronics.

Timing is also one of the most important concerns of AQFP. Since AQFP circuits employ the deep pipeline architecture (each logic gate is modulated by a clock signal), the clock signal runs through each of the AQFP logic gates in a zigzag manner, as shown in Figure 2. This deep pipeline architecture also leads to a short stage delay, which is comparable to the time for the clock signal to pass through each zigzag row. This reveals that the strong timing constraints in the physical design of the AQFP, and the positions of the cells and the wirelength of the cells in a row have a great impact on the timing closure. Besides, there are still various other constraints, including spacing constraints and max-wirelength constraints [8], which indicates that all the above constraints must be considered together during the placement process. Traditional placers [2, 8] that address wirelength constraints alone are no longer sufficient to close timing for AQFP. To address it, we introduce a timing-aware placement framework for AQFP circuits, which considers timing and cell positions in different phases during the physical design and generates placement results free of timing violations.

Our major contributions are summarized as follows:

- We design an analytical global placement method with a novel objective function which is carefully crafted with the four-phase timing model of AQFP. Furthermore, we solve the global placement by DREAMPlace [10] tools under our multi-k approximation strategy.
- We introduce a detailed placement method, which maintains the quality inherited from the global placement and refines the timing-wirelength trade-offs by the timing-aware dynamic programming and the time-space cell regularization.

* The first three authors contribute equally to this research.

DAC '22, July 10–14, 2022, San Francisco, CA, USA

ACM ISBN 978-1-4503-9142-9/22/07…$15.00
https://doi.org/10.1145/3489517.3530487

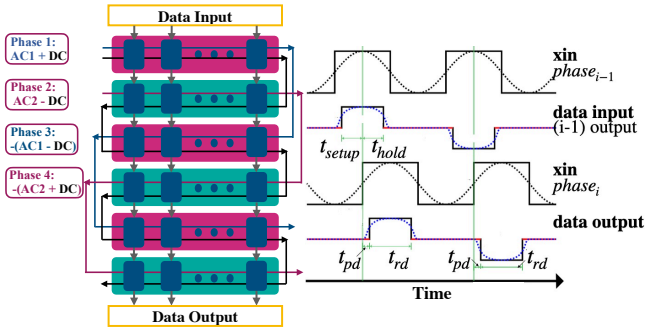

Figure 2: The four-phase clocking scheme and the data propagation for AQFP circuits.

- Experimental results show that compared with the existing AQFP placement works, TAAS improves the maximum operating frequency of the circuits by 19.2% on average at the expense of a wirelength increase of 3.5% on average. When it comes to the large-scale complex circuits, TAAS can increase the maximum operating frequency by 30%∼40% with a negligible increase (<1%) in wirelength.
- TAAS is the pioneering work to simultaneously optimize the timing performance, spacing constraints, and wirelength constraints of AQFP circuits in the physical design stage.

2 BACKGROUND AND MOTIVATION

2.1 AQFP Superconducting Logic

The standard cell library of AQFP includes basic logic gates such as BUFFER, INVERTER, AND, OR, MAJORITY, and SPLITTER. The most basic structure of AQFP circuits is the AQFP buffer, which consists of a double-Josephson-Junction SQUID [5]. The AQFP inverter and constant cell are designed based on the AQFP buffer [1]. The AQFP inverter is designed by negating the coupling coefficient of the output transformer in the AQFP buffer. And the AQFP constant gate is implemented based on the asymmetry of excitation flux inductances in the AQFP buffer. Please note that, unlike CMOS circuits which use wires to connect to the same port for fan-out directly, the AQFP circuits utilize splitters for fan-out (when the number of fan-outs is 2 or more). Figure 1 shows some examples of AQFP logic gates, such as AND, NAND, MAJORITY, and SPLITTER.

Different from the conventional CMOS technology, both combinational and sequential AQFP logic cells are driven by AC-power, which serves as not only the excitation current but also the synchronization mechanism, i.e., a clock signal to synchronize the outputs of all gates in the same clock phase. Therefore, data propagation in AQFP circuits requires overlapping clock signals from the neighboring phase. An example of a four-phase clocking scheme of AQFP circuits and corresponding data flow is shown in Figure 2. As shown in Figure 2, each AQFP logic gate is driven by an AC clock signal and assigned with one specific clock phase, which makes AQFP circuits "deep-pipelining" in nature. In this clocking scheme, all inputs for a logic gate should have the same delay (clock phases) from the primary inputs, i.e., the strict path balancing shall be enforced.

2.2 AQFP Placement and Timing Problem

After the logic synthesis and the buffer/splitter insertion for path balancing, the AQFP circuit is transformed to a netlist of AQFP logic cells. The next step is to place each AQFP cell in a specific position within its assigned row. To fit the AQFP clock architecture, each cell in the netlist is assigned with a specific, fixed clock phase, which corresponds to a specific row in Figure 2. The placement algorithm will not change the specific row index that a logic cell resides in, but will only optimize the relative, horizontal locations of logic cells within each row. After the fixed-order, row-wise placement algorithm, the routing algorithm will only need to connect cells from row i to row $i+1$. Therefore, we could perform routing separately for every two consecutive rows and greatly accelerate the routing process.

On the other hand, the AQFP placement algorithm needs to address various spacing constraints [8], including cell spacing and zigzag spacing, which lead to a large number of combinatorial constraints in the AQFP placement problem. For cell spacing, two adjacent cells in one row need to be either abutting or keeping a minimum spacing. For zigzag spacing, if vias are adopted to perform the wire direction shift in AQFP circuits, then these wire zigzags need to have at least certain pre-defined spacing (e.g. $10\mu m$ for MIT-LL process). Apart from these spacing constraints, AQFP circuits should also satisfy the max-wirelength constraint, which defines that a single connection is limited to WL_{max} (e.g., $1mm$ in the MIT-LL process). If a single connection exceeds the max-wirelength constraint, then it is required to add an entire row of buffers between the two rows where the violation appears.

Since the AQFP circuits operate at the clock frequency of several GHz, strict timing requirements must be met. AQFP circuits employ the deep pipeline architecture, where the clocking signal runs through each of the AQFP logic gates in a zigzag manner. This zigzag clocking imposes great challenges to AQFP placement algorithms as the cell positions in a fixed row have a great impact on the timing closure. Therefore, traditional placers [2, 8] that address wirelength alone may lead to placement results with severe timing violations. Specifically, placement results need to meet setup and hold timing requirements. Setup time is the minimum amount of time before the active edge of the clock that the data must be stable, while hold time is the minimum amount of time after the active edge of the clock during which data must be stable. For example, if two AQFP cells in consecutive rows (phase 2 and phase 3, respectively) are placed both on the left side, then it is possible that data from the previous row could not keep stable after the active edge of the clock in the following row, therefore resulting in hold time violation. The placer should become timing-aware to address the potential timing violations during the placement stage.

3 THE PROPOSED OVERALL FRAMEWORK

Figure 3 demonstrates the overall flow of our framework, which can be divided into three parts: global placement, detailed placement, and placement legalization. In this section, we elucidate the implementation details of these steps.

3.1 Global Placement

Three requirements should be satisfied in the global placement: (1) A refinement of cell locations from the perspective of the global performance; (2) Minimize the total wirelength under the specific spacing constraints and wirelength constraints of AQFP; (3) With the deep pipeline structure, timing violations should be eliminated to achieve the final high-frequency circuit. Accordingly, our timing-aware global placement introduces three parts: (1) Four-phase timing model, where through the static timing analysis process (STA) of

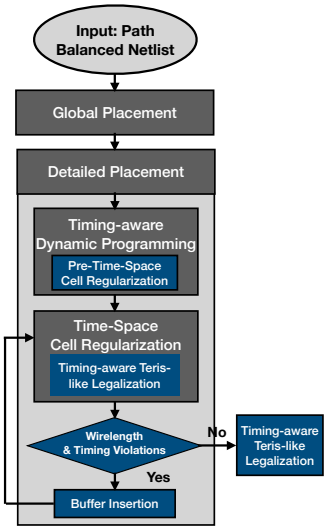

Figure 3: The overall workflow of the timing-aware analytical strategy: TAAS.

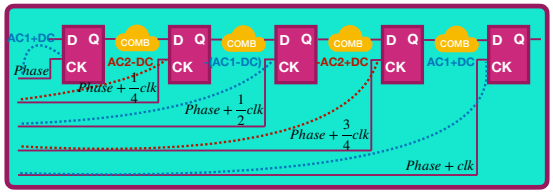

Figure 4: The STA equivalent circuit of AQFP with 4-phase clocking.

the equivalent circuit of AQFP as Figure 4, we obtain the timing constraints and the relative positions constraints between connected cells. (2) Timing-aware analytical function, where the objective function of the global placement in ASAP [2] is reconstructed with the timing penalty, and solved by DREAMPlace. (3) Multi-k approximation strategy, by which we search the optimal timing-wirelength trade-offs in a fine-grained wise. Please note that we fix the distance between adjacent phases (rows), while only adjusting the x-direction. And we also perform the placement process row by row from top to bottom, in which we place the current row with the previous ones fixed. Moreover, we only calculate one data path each time because the cell has the one-to-one connection pattern.

3.1.1 Four-phase Timing Model. Based on the deep pipeline structure, we transform the AQFP circuit into Figure 4 to demonstrate the STA. The timing model is derived as follows:

$$\begin{cases} t_{rd} = \frac{1}{2}clk - 2t_{pd}, \\ t_{pd} + t_0 + delay(x_p) \le \frac{1}{4}clk - t_{setup} + t_0 + delay(x_s), \\ t_{pd} + t_{rd} + t_0 + delay(x_p) \ge \frac{1}{4}clk + t_{hold} + t_0 + delay(x_s), \\ t_{pd} + t_0 + delay(x_p) \ge \frac{1}{4}clk - clk + t_{hold} + t_0 + delay(x_s). \end{cases} \tag{1}$$

where t_{rd} is the generated signal length in the next phase as the Figure 2; t_{pd} is the clock-to-q time of cell; clk is the clock cycle; t_0 is the time when one data coming; $delay(x_p)$ is the clock delay of the cell in the upper phase while $delay(x_s)$ for the lower one; t_{setup} is the setup time of cell, while t_{hold} is the hold time. The

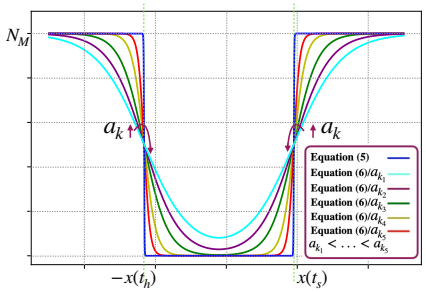

Figure 5: The comparison of the numerical distribution between the Equantion (5) and the Equantion (6) ($k = 5$ in experiments).

second equation in Equation (1) can be easily violated under a high operating frequency (small clk). Given the specific clock setting, we obtain the timing window:

$$\begin{cases} -t_h \le Timing\_Window \le t_s, \\ -t_h = \frac{1}{4}clk + t_{setup} + t_{pd}, \\ t_s = -\frac{1}{4}clk - t_{hold} + t_{pd} + t_{rd}. \end{cases} \tag{2}$$

where $TW(x_s, x_p) = delay(x_s) - delay(x_p)$; when $TW = -t_h$, $x = -x(t_h)$; when $TW = t_s$, $x = x(t_s)$. (TW stands for $Timing\_Window$.)

3.1.2 Timing-aware Analytical Function. ASAP integrates the maximum wirelength constraints of AQFP into the RePlace [4]:

$$F_e = e^{\frac{L_e - WL_{max}}{WL_{max}}} \tag{3}$$

where E is the set of nets that the cells are incident to and $e \in E$; L_e is the wirelength cost using the weighted average wirelength model [7] to approximate half-perimeter wirelength (HPWL) accurately; With WL_{max} being the maximum wirelength, the Equantion (3) will be punished exponentially when the wirelength over WL_{max}. However, this optimization goal focuses on the wirelength constraints of AQFP and the overlap rate of the cell, while ignoring the timing limitations of AQFP with the four-phase clocking.

More concretely, the four-phase clocking scheme and its one-to-one cell connection can trigger timing violations easily. Since under the high operating frequency, the delay of the zigzagging clock of each cell has the same order of magnitude as the stage delay so that it is challenging for a single cell with a narrow timing window to capture the correct data. Accordingly, we introduce the timing-aware analytical function and embed it into DREAMPlace, which maps the cell position (x_i, y_i) into the weights of the neural networks (NN) and performs NN training. The proposed timing-aware analytical function is as follows:

$$\min \sum_{e \in E} (F_e + \beta T(x)) + \lambda D(c) \tag{4}$$

where to reduce timing violations, we model the timing penalty, $T(x)$, as a separated adding item with a β correction coefficient:

$$T(x) = \begin{cases} 0 & \text{if } x \in [-x(t_h), x(t_s)] \\ N_M & \text{if } x \in (-\infty, -x(t_h)) \bigcup (-x(t_s), +\infty) \end{cases} \tag{5}$$

where $x(t_s)$ and $x(t_h)$ obtained in Equation (2) serve as the commensurate delay length; N_M can be set based on specific circuits.

3.1.3 Multi-k Approximation Strategy. Equation (5) is a non-smooth Heaviside Function and will generate invalid gradients in NN training so that we apply the multi-k logic function to progressively

approximate the Equation (5):

$$T(x) = N_M \left\{ \frac{1}{2} + \frac{1}{4} tanh \left\{ a_k [-x - l(t_h)] + \frac{1}{4} tanh \left\{ a_k [x - l(t_s)] \right\} \right\} \right\} \tag{6}$$

And Figure 5 presents the smooth effect of Equation (6) which is amicable to the backpropagation of NN training. We asymptotically approach the Equation (5) to reinforce the timing constraints in a fine-grained wise. In the initialization of each k-stage, we model the $\beta = \beta_k$ to keep the numerical dimension uniform between the timing effect and the wirelength effect:

$$\beta_k = \frac{\sum_{e \in E} |\partial F_e|}{\sum_{e \in E} |\partial T_x|} \tag{7}$$

3.2 Detailed Placement

In the detailed placement, we introduce three parts: (1) the timing-aware dynamic programming technique, which minimizes the timing cost and the total wirelength of AQFP circuits with the fixed-order cells under the spacing constraints and the wirelength constraints of AQFP. (2) the time-space regularization, which reorders cells to reduce unresolvable timing violations in the fixed-order dynamic programming. (3) the Tetris-like legalization incorporating timing constraints of AQFP, through which we finalize the placement to ensure circuits execute properly.

3.2.1 Timing-aware Dynamic Programming. We ameliorate the cell positions within each row. The algorithm is operated row by row with other rows fixed. The mathematical formulation for this workflow can be simplified as follows:

$$\min_{x_i} \quad \sum_{e \in E} L(e), \tag{8a}$$

$$\text{s.t.} \quad L(e) \le WL_{max}, \qquad \forall e \in E, \tag{8b}$$

$$x_{i-1} + w_{i-1} \ge x_i - BI_i, \qquad i = 2, \ldots, N, \tag{8c}$$

$$x_{i-1} + w_{i-1} \le x_i - p_i^{min} I_i, \qquad i = 2, \ldots, N, \tag{8d}$$

$$x_i \in \{x_i^1, x_i^2, \ldots, x_i^M\}, \qquad i = 1, 2, \ldots, N \tag{8e}$$

$$z_i \in \{0, 1\}, \qquad i = 1, 2, \ldots, N-1, \tag{8f}$$

where with I_i being a binary variable, B being a big positive constant, (8c) and (8d) guarantee two horizontal-neighboring cells can either be abutting or keeping a minimum spacing p_i^{min}. The two equations can be integrated into $x_{i-1} + w_{i-1} = x_i$ for $I_i = 0$. Equation (8b) prevents wirelength violations. Equation (8e) ensures the legal locations satisfying the horizontal spacing requirements.

This is a discrete optimization problem so that we simulate the detailed placement as the shortest path problem and apply the dynamic programming algorithm. We first exploit the Lagrangian multiplier method to obtain the Lagrangian subproblem $\$(x; \lambda)$:

$$\arg\min_{x_i} \sum_{e \in E} L_e + \sum_{e \in E} \lambda_e (L_e - L_{max}), \tag{9a}$$

$$\text{s.t.} \quad \lambda_e > 0, \tag{9b}$$

$$Equation\,(8c) \sim (8f)\,. \tag{9c}$$

In Figure 6, the blue circles indicate the candidate positions set of each cell. With the N being the number of cells in the current phase, the M denote the M candidate positions. Integrating the timing attribute and wirelength attribute of AQFP into the candidate

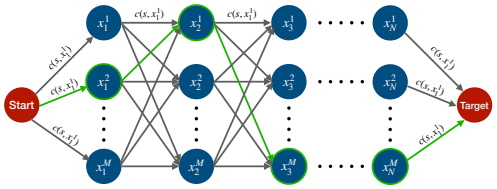

Figure 6: The shortest path model for Lagrangian subproblem $\$(x, \lambda)$. And the green line is the selected shortest path.

position, the cost of each edge is defined as follows:

$$c(x_{i-1}^u, x_i^v) = SP(x_{i-1}^u, x_i^v) + \sum_{e \in E} (1 + \lambda_e) WL_e \cdot TP(x_{i-1}^u),$$

$$SP(x_{i-1}^u, x_i^v) = \begin{cases} 0, & \text{if } x_{i-1}^u + w_{i-1} = x_i^v \text{ or} \\ & x_{i-1}^u + w_{i-1} \le x_i^v - p_i^{min}, \\ \infty, & \text{others,} \end{cases} \tag{10}$$

$$TP(x_{i-1}^u) = \begin{cases} 1, & \text{if } TW, \\ e^{-t_s - t_h + |x_{i-1}^u - t_s| + |x_{i-1}^u + t_h|}, & \text{others.} \end{cases}$$

where SP is the spacing cost keeping no overlap between adjacent cells; TP is the timing cost merging into the wirelength as a penalty item, and TW follows Equantion (2). Since timing violations can be improved or even eliminated by the subsequent time-space cell regularization and timing-violation-free legalization, we set TP as a soft constraint but in form of an exponential penalty.

We adopt the subgradient method to update the multiplier λ at the kth iteration and gradually solve the generated subproblem. According to [6], λ_e^{k+1} is updated as: $\max\left(0, \lambda_e^k + t_k\,(L_e - WL_{max})\right)$, where t_k controls the step size of updating.

Algorithm 1: Timing-aware Dynamic Programming

```
Given an ordered sequence of cells 1, 2, ..., N;
λ_e ← 1, ∀ e ∈ E, pre_regulized ← RegNum;
while not converged and not reach maximum iteration do
    Solve Lagrangian subproblem ℒ(x; λ) with the shortest path
      algorithm;
    Update λ according to Equation (11);
    if iteration ≥ pre_regulized then
        Preform Pre-Time-Space Cell Regularization;
    end
end
Update the positions of the cells in one row.
```

Pre-Time-Space Cell Regularization. To alleviate the conflict with the numerical solution space of the next stage, the time-space cell regularization, we insert the pre-time-space cell regularization in the last few iterations of the timing-aware dynamic programming. Algorithm 1 illustrates the whole procedure on one row.

3.2.2 Time-Space Cell Regularization. The fixed-order strategy compresses the search space of the dynamic programming. However, the dynamic order is necessary for timing optimization when AQFP circuits contain compact cells, such as sorter32. To fix remaining timing violations, we introduce the time-space cell regularization. Specifically, we pin the cell in the previous rows and adjust the position of the cell. As shown in Figure 7(a), when no timing violations, the penalty for the wire is 1 and the cell has no moving

Table 1: The comparison on the total HPWL and the timing performance between baselines and TAAS.

Circuits	GORDIAN-based[8]					TAAS					Impro. Ratio		
	HPWL	Buffers	Violations (5 GHz)	Frequency (GHz)	WNS (ps)	HPWL	Buffers	Violations (5 GHz)	Frequency (GHz)	WNS (ps)	WLI. Ratio	FRI. Ratio	WNS. Ratio
adder8	10948	24	#	6.1	#	12360	24	#	6.2	#	**12.90%**	**1.64%**	#
apc32	15915	26	#	5.8	#	15915	26	#	5.8	#	0	0	#
c432	51009	46	#	5.4	#	52208	45	#	5.5	#	**2.35%**	**1.85%**	#
decoder	141151	34	360	4.3	-8.8	156213	33	26	5.1	-1.4	**10.67%**	**18.60%**	**3.50%**
sorter32	168208	29	274	4.4	-6.9	180427	29	29	4.7	-3.3	**7.26%**	**6.82%**	**1.74%**
apc128	254068	117	2157	2.8	-40.7	245416	110	317	3.9	-10.1	**-3.41%**	**39.29%**	**12.72%**
c499	430659	62	1978	3.1	-29.9	431108	62	234	4.3	-8.9	**1.00%**	**38.71%**	**9.13%**
c1355	422556	58	2011	3.1	-31.4	426099	58	257	4.1	-9.1	**0.84%**	**32.26%**	**9.64%**
c1908	358271	67	1720	3.3	-25.5	361071	66	213	4.4	-6.9	**0.78%**	**33.34%**	**8.25%**

Note: *HPWL are the summation of all nets measured using μm; *Violations are test under 5GHz; *Frequency is the maximum operating frequency; *Frequency is the maximum operating frequency; *WLI Ratio is the increased ratio of the total wirelength; *FRI Ratio is the increased ratio of the maximum operating frequency; *WNS Ratio is the decrease ratio of the worst negative slack.

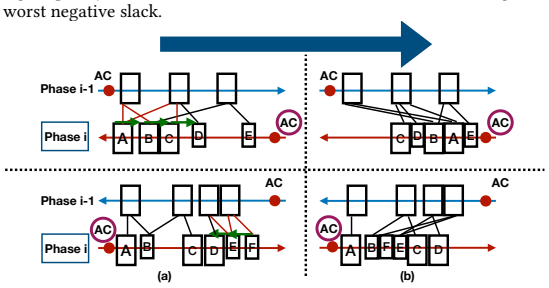

Figure 7: Two examples of the time-space cell regularization. The upper side shows $phase_{i-1}\%4 = 1$ and the lower side shows $phase_{i-1}\%4 = 3$. (a) Before the cell regularization, there are 4 timing violations (red lines) on the upper side and 3 on the lower side. (b) After the cell regularization, cells with timing violations will be driven close to the AC entrance.

tendency. If timing violations exist in a data path, the corresponding wire will be punished by $a(1 + t_{slack})$ or $a\frac{1}{1+t_{slack}}(phase_{i-1}\%4 = 1$ or $phase_{i-1}\%4 = 3)$, where t_{slack} denotes the value part beyond the timing window and a restricts the new wirelength $< WL_{max}$. The cell will be approaching the clock entrance in that phase, reducing the phase difference with the connected cells. Finally, we employ the Tetris-like legalization modified by the four-phase timing model (timing-aware legalization). Maintaining the legal timing, the legalization sorts the cells in the row by ascending x-coordinate and places them from left to right without overlap. After the time-space cell regularization, the position of the cell can be changed as Figure 7(b). The relevant details are shown in Algorithm 2.

3.2.3 Overall Detailed Placement. The overall detailed placement shown in Algorithm 3. Lines 1 ~ 3 performs the fixed-order timing-aware dynamic programming, and 4 ~ 8 for the time-space cell regularization to reorder the cells in each row. Then insert a row of buffers between the two rows with the violations and then conduct the time-space cell regularization as described in rows 9 ~ 16. The procedure terminates until there are no violations in the circuit. Finally, we finalize the placement with the timing-aware legalization.

4 IMPLEMENTATION AND RESULTS

We implement our framework in Python and the procedure runs on an AMD Ryzen Threadripper 2920X Processor with 12 high-performance cores and 24 parallel threads. We test our proposed

Algorithm 2: Time-Space Cell Regularization

```
1  Given a vector vec includes cells that needs to be relocated;
2  Let a be the penalty for the wire with timing violations, a ← a_max;
3  foreach cell_i ∈ vec do
4      flag = False;
5      while flag == False do
6          sum ← 0, weight ← 0, vkep ← cell_i.x;
7          foreach cell_j connects with cell_i do
8              if (t_slack ← TimingCheck(cell_j,cell_i)) ! = 0 then
9                  sum + = (a(1 + t_slack)cell_i.x) or
                       (a/(1+t_slack) cell_i.x);
10             else
11                 sum + = cell_i.x;
12             end
13             weight + = 1;
14         end
15         cell_i.x ← sum/weight;
16         if WirelengthCheck() == False then
17             a − = δ, cell_i.x ← vkep, flag = False;
18         else
19             flag = True;
20         end
21     end
22 end
23 Timing-aware Legalization().
```

placement framework using classic benchmark circuits for AQFP testing [3], including 32-bit approximate parallel counter (apc32), 8-bit Kogge-Stone adder (adder8), decoder, 27-channel interrupt controller (c432), 32-bit sorter (sorter32), 32-bit approximate parallel counter (apc128). We also validate it on the ISCAS'85 benchmark circuits to demonstrate the effectiveness of our proposed framework on large circuits. In Table 1, we compare our results with the GORDIAN-based AQFP placement method in [8].

4.1 Results of the Overall Workflow of TAAS

As shown in Table 1, the GORDAN-based framework can generate placement results with a shorter wirelength (5% shorter on average for small circuits, e.g. adder8, apc32, c432), which is not an obvious advantage on larger circuits compared to our method (2.4% shorter on average for large circuits, e.g. decoder, sorter32,

Algorithm 3: Overall Detailed Placement

```
foreach row ∈ Nestlist do
    Timing-aware Dynamic Programming (row);
end
while not converge do
    foreach row ∈ Nestlist do
        Time-Space Cell Regularization(row);
    end
end
while wire & timing violation exists do
    foreach row ∈ Nestlist do
        if wire & timing violation exists then
            Insert a Buffer row;
            Time-Space Cell Regularization(row);
        end
    end
end
RowDistanceAjustment();
Timing-aware Legalization().
```

apc128, c499, c1355, c1908). However, the GORDIAN-based method suffers from timing violations when the circuits target at a high clock frequency, since it does not analyze the timing constraints of AQFP circuits with the deep pipeline architecture. At 5 GHz, the generated placement results can work correctly for small circuits such as the adder8, while serious timing violations will occur for a slightly larger circuit such as the sorter32, not to mention the circuits with more complex functions (e.g. c499, c1355, c1908).

In contrast, TAAS improves the timing performance extremely while maintaining comparable advantages on the total wirelength. For small circuits, such as adder8, apc32, c432, TAAS can further increase the FRI of the circuit by 0% ∼ 12.9%. For large circuits with complex logic such as apc128, c499, c1355, c1908, TAAS can increase the FRI by 30%∼40% with a negligible -3.41% ∼ 1% increase of the WLI compared with the GORDIAN-based placement. In brief, TAAS improves the FRI of the circuits by 19.2% on average at the cost of the WLI increase of 3.5% on average, and lessens the WNS of the circuit by 5.0% on average. Note that our framework does not augment the number of buffers while optimizing the timing performance, and reaches a slight decrease in buffers, such as apc128.

4.2 Results of TAAS without Time-Space Cell Regularization

We implement TAAS without the cell regularization to show the effects of the timing-aware dynamic programming and the time-space cell regularization respectively as shown in Table 2. For small circuits (adder8, apc32, c432), TAAS without the cell regularization achieves the same effect as the complete TAAS workflow. It implies that the timing-aware dynamic programming is sufficient to solve the timing violations of small circuits. For larger circuits (sorter32, decoder), the FRI increase is a little limited (6.98% vs. 18.6%, 2.27 vs. 6.82%), even though WLI with a <5% increase. Moreover, if the complete workflow on the decoder, its frequency can be increased to 5.1 GHz (>5 GHz), which is higher than the target frequency as Table 1. It proves that the time-space cell regularization plays a pivotal role for circuits containing compact cells. For more complex circuits (apc128, c499, c1355, c1908), the WLI has only a slight increase of <1%, with the FRI being optimized by 19.4% on average. It shows that the timing-aware dynamic programming can effectively improve timing performance for circuits with relatively scattered cells after the placement.

Table 2: The comparison on the total HPWL and the maximum frequency between baselines and TAAS (w/o time-space cell regularization).

Circuits	GORDIAN-based		TAAS (w/o Cell Regularization)		Impro. Ratio	
	HPWL	Frequency (GHz)	HPWL	Frequency (GHz)	WLI. Ratio	FRI. Ratio
adder8	10948	6.1	12360	6.2	**12.90%**	**1.64%**
apc32	15915	5.8	15915	5.8	0	0
c432	51009	5.4	52208	5.5	**2.35%**	**1.85%**
decoder	141151	4.3	147841	4.6	**4.74%**	**6.98%**
sorter32	168208	4.4	172124	4.5	**2.33%**	**2.27%**
apc128	254068	2.8	255169	3.2	**0.43%**	**14.29%**
c499	430659	3.1	434214	3.9	**0.83%**	**25.81%**
c1355	422556	3.1	422919	3.7	**0.09%**	**19.35%**
c1908	358271	3.3	359142	3.9	**0.24%**	**18.18%**

5 CONCLUSION

In this work, we propose a timing-aware framework, TAAS, for the AQFP placement, which is the first to simultaneously optimize the timing performance and spacing constraints of AQFP in the physical design. Experiments show that TAAS can greatly improve timing performance with a negligible HPWL increase.

REFERENCES

[1] Ruizhe Cai, Olivia Chen, Ao Ren, Ning Liu, Caiwen Ding, Nobuyuki Yoshikawa, and Yanzhi Wang. 2019. A majority logic synthesis framework for adiabatic quantum-flux-parametron superconducting circuits. In *Proceedings of the 2019 on Great Lakes Symposium on VLSI*. 189–194.

[2] Yi-Chen Chang, Hongjia Li, Olivia Chen, Yanzhi Wang, Nobuyuki Yoshikawa, and Tsung-Yi Ho. 2020. ASAP: an analytical strategy for AQFP placement. In *2020 IEEE/ACM International Conference On Computer Aided Design (ICCAD)*. IEEE, 1–7.

[3] Olivia Chen, Ruizhe Cai, Yanzhi Wang, Fei Ke, Taiki Yamae, Ro Saito, Naoki Takeuchi, and Nobuyuki Yoshikawa. 2019. Adiabatic quantum-flux-parametron: Towards building extremely energy-efficient circuits and systems. *Scientific reports* 9, 1 (2019), 1–10.

[4] Chung-Kuan Cheng, Andrew B Kahng, Ilgweon Kang, and Lutong Wang. 2018. Replace: Advancing solution quality and routability validation in global placement. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 38, 9 (2018), 1717–1730.

[5] John Clarke and Alex I Braginski. 2006. *The SQUID handbook: Applications of SQUIDs and SQUID systems*. John Wiley & Sons.

[6] Michael Held, Philip Wolfe, and Harlan P Crowder. 1974. Validation of subgradient optimization. *Mathematical programming* 6, 1 (1974), 62–88.

[7] Meng-Kai Hsu, Yao-Wen Chang, and Valeriy Balabanov. 2011. TSV-aware analytical placement for 3D IC designs. In *Proceedings of the 48th Design Automation Conference*. 664–669.

[8] Hongjia Li, Mengshu Sun, Tianyun Zhang, Olivia Chen, Nobuyuki Yoshikawa, Bei Yu, Yanzhi Wang, and Yibo Lin. 2021. Towards AQFP-Capable Physical Design Automation. In *Design, Automation and Test in Europe Conference*.

[9] 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.

[10] Yibo Lin, Zixuan Jiang, Jiaqi Gu, Wuxi Li, Shounak Dhar, Haoxing Ren, Brucek Khailany, and David Z Pan. 2020. Dreamplace: Deep learning toolkit-enabled gpu acceleration for modern vlsi placement. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 40, 4 (2020), 748–761.

[11] Sergey K Tolpygo, Vladimir Bolkhovsky, Terence J Weir, Alex Wynn, Daniel E Oates, Leonard M Johnson, and Mark A Gouker. 2016. Advanced fabrication processes for superconducting very large-scale integrated circuits. *IEEE Transactions on Applied Superconductivity* 26, 3 (2016), 1–10.