Unleashing the Potential of AQFP Logic Placement via Entanglement Entropy and Projection

Yinuo Bai
SSSLab, China University of Petroleum-Beijing

Enxin Yi
SSSLab, China University of Petroleum-Beijing

Wei Xing
The University of Sheffield

Bei Yu
The Chinese University of Hong Kong

Zhou Jin
SSSLab, China University of Petroleum-Beijing

Abstract

Adiabatic quantum-flux-parametron (AQFP) logic, known for its energy efficiency, has emerged as a prominent superconductor-based logic family, surpassing traditional rapid single flux quantum (RSFQ) logic. In AQFP circuits, each cell operates on AC power, serving as both a power supply and clock signal to drive data flow across clock phases. However, signal attenuation with increasing wirelength may result in more potential data errors. To address this, rows of buffers are inserted as repeaters to ensure data synchronization and avoid wirelength violations. However, these inserted buffer rows in the AQFP placement significantly amplifies power consumption and circuit delay. To address these challenges, in this paper, we propose an innovative and analytical method for the placement of AQFP. The proposed method aims at minimizing the need for additional buffers. The framework incorporates two key features: (1) entanglement entropy for topology initialization and (2) projection for placement and buffering. These features offer advantages such as avoiding intensive computations, including fix-order Lagrangian optimization in large-scale scenarios, while significantly reducing the required number of buffer rows. The experimental results validate the efficiency of the proposed framework, demonstrating an average reduction of 81% in the required number of buffers and acceleration of 1.88x in the processing time compared with the state-of-the-art method.

Keywords

Superconductor-based logic, adiabatic quantum-flux-parametron (AQFP), placement.

1 Introduction

The adiabatic quantum-flux-parameter (AQFP) is a highly energy-efficient Josephson Junctions (JJ)-based superconducting logic technology [1]. It operates at clock frequencies up to 10GHz, consuming significantly less energy than conventional superconducting logics such as rapid single flux quantum (RSFQ) and single flux quantum (SFQ) [1]. The exceptional performance of AQFP demonstrates its tremendous potential as a highly efficient alternative to CMOS technology. Using fabrication processes such as the MIT-LL SFQ process [2], AQFP circuits can achieve gains in energy efficiency ranging from 10^4 to 10^5 compared with state-of-the-art CMOS, even at high clock frequencies.

As the size of the circuit increases, the utilization of design automation (EDA) tools becomes essential to facilitate the development of AQFP circuits. However, the application of existing EDA tools developed for CMOS technology is not directly feasible in the realm of superconducting electronics. This is primarily due to some fundamental differences in active components, passive components, basic logic gate sets, clocking schemes, as well as the inclusion of buffers, splitters, and AC biasing, which contribute to additional complexities and costs.

Recent research in EDA tools for AQFP has seen significant progress [3–9]. Several frameworks have been proposed to automate different stages of the AQFP design flow. Among these stages, placement holds crucial importance as it profoundly impacts circuit efficiency [10], taking into account two key constraints: 1) data synchronization and 2) maximum wirelength constraint. Meeting these constraints necessitates the insertion of numerous buffers, which fill a significant portion of the circuit, resulting in increased power consumption and longer delay. The initial placement method, first presented in [4] and further enhanced in [5], employs a genetic algorithm (GA) to relocate cells within each row, in order to mitigate wirelength violations and minimize buffer insertion. However, these GA-based methods are time-intensive and may still yield solutions with considerable inserted buffers. Among the placement algorithms proposed in [6] and [7], ASAP stands out by utilizing fix-order Lagrangian relaxation and the directed-force method for cell balancing in the solution from topology initialization. ASAP transforms the Lagrangian relaxation solution into a polynomial-time shortest-path problem solution to accelerate large-scale circuit placements. However, neither the Lagrangian relaxation nor the directed force method specifically targets minimizing the maximum wirelength. Their objective functions focus on minimizing total half-perimeter wirelength [11] in the entire circuits or total quadratic wirelength in local circuits, respectively, thus causing ample room for further optimization.

In this paper, to tackle these challenges, we propose a novel placement framework for AQFP Logic circuits. The algorithm comprises two main steps: (1)Topology initialization and (2) Placement and buffering. In the topology initialization phase, we introduce the entanglement entropy of circuits as an objective function and minimize it by exchanging cells in the same row to approach a better circuit

DAC '24, June 23–27, 2024, San Francisco, CA, USA

ACM ISBN 979-8-4007-0601-1/24/06…$15.00
https://doi.org/10.1145/3649329.3658467

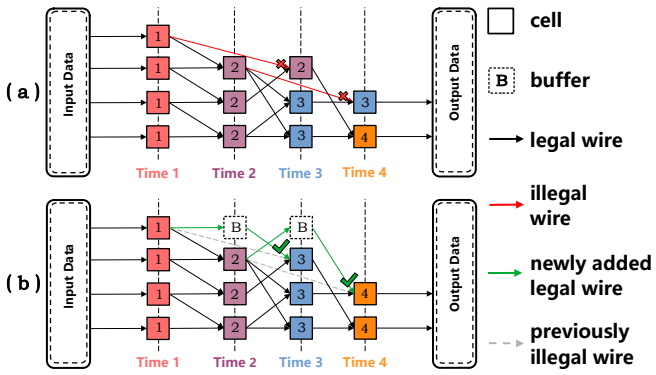

Figure 1: Path balancing process. (a) The circuit before path balancing. Two wires (red) connect cells that are not located between adjacent rows. (b) The circuit after path balancing. Two wires with excessive length are cut in the middle using buffers.

topology. For the placement and buffering phase, we position cells on their ideal position determined by the proposed projection-based strategy in iterations and incorporate the buffer insertion within the iteration to eradicate the persisted wirelength violation. Our approach achieves remarkable reductions of 98% and 81% in inserted buffers and substantial improvements of 81.82x and 1.88x in runtime on benchmark circuits [12] compared with the GA method and ASAP, respectively. Our contributions can be highlighted as follows:

(1) We propose the entanglement entropy in the topology initialization phase as the objective function. A superior topology exhibits a lower entanglement entropy.
(2) In the placement and buffering phase, we utilize a novel projection-based strategy to determine locally wire-saving positions for each cell. Through iterations, we achieve a globally optimized solution.
(3) To further enhance efficiency, we emphasize the topology with relatively low entanglement entropy rather than the minimum one in the topology initialization phase. Furthermore, we proposed a coordinate adjustment method to eradicate overlaps among cells in the placement and buffering phase.

2 Background

2.1 AQFP Logic and Circuit Topology

The pivotal constituents of an AQFP circuit encompass diverse AQFP logic cells and interconnecting wires. The output ports of each logic cell device are linked to the signal input ports of one or more logic cells via wires. Each logic cell features a pair of JJs[13]. The minimalist design approach forms the foundation for the development of a comprehensive cell library. The library encompasses essential logic gates, including AND, OR, NOT, MAJORITY, BUFFER, and SPLITTER [14].

By incorporating AC power as the power source and clock phase indicator, AQFP circuits have effectively overcome the energy dissipation limitations faced by conventional superconductor logic families such as RSFQ. To facilitate data transmission, cells in the circuit sharing the same clock phase are organized into the same row, as illustrated in Fig. 1.

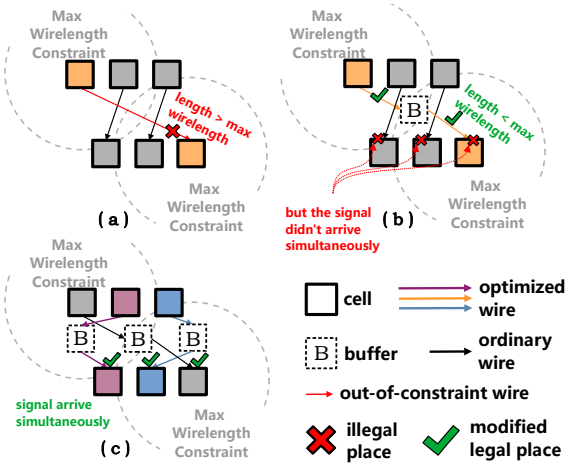

Figure 2: Buffer insertion to eliminate wirelength violations in fixed-order circumstance. (a) The red wire exceeds the maximum wirelength constraint. (b) Inserting a buffer to address the issue of wirelength, but encountering problems with path balancing. (c) A row of buffers ought to be inserted to address the issue of path balance.

2.2 Physical Constraints

In this subsection, we will provide a detailed explanation and solution of the two primary physical constraints : data synchronization and maximum wirelength constraints, which need to be thoroughly considered throughout the entire placement process.

Data Synchronization. For the circuit topology in Fig. 1, signal delay or advance that occurs in any row can potentially result in significant data errors in the final output. This necessitates the simultaneous arrival of input data to any given layer. Therefore, all cells in the same phase can only receive data from that in the above adjacent phase. By disregarding time delays along the data wires, it effectively requires that any wire can only connect cells located in adjacent rows. Such a constraint is referred to as Data Synchronization. A circuit that satisfies this constraint is termed path balanced. For all non-path-balancing circuits, buffers are necessary to equalize the input delay of all cells, as illustrated in Fig. 1.

Maximum Wirelength Constrain. Due to signal current attenuation along the wire, it is imperative to ensure that the length of each wire remains within the permissible limits, such as the one prescribed for the MIT-LL process, which is approximately 1mm. Whenever a wire exceeds this maximum length, it becomes necessary to incorporate buffer insertion as repeaters along the wire to effectively reduce its length. It should be emphasized that the addition of a single buffer necessitates the insertion of an entire row, even multiple rows in certain case, of buffers to maintain the path balance. For reference examples that demonstrate buffer insertion between two neighboring rows, please consult Fig. 2. In this paper, it is worth noting that the clock wire of the phase-aligned cell, responsible for delivering power and clock phase signals, is not taken into consideration [15].

Explicating Tasks. The legalized placement process must observe the mentioned constraints. With the aim of minimizing the number

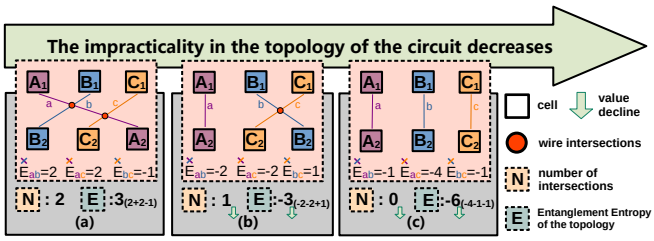

Figure 3: Rearranging the order to eliminate wirelength violations. The intersections and the entanglement entropy continuously decrease throughout the process. Entanglement entropy in (a), (b) and (c) reaches 3, -3 and -6, respectively.

of inserted buffers, considering the specified size, phases, and connections of cells and buffers, the outputs of the placement process are as follows: (1) Coordinates of each cell or buffer. These coordinates must ensure no overlaps among cells and buffers. (2) Connections among cells and buffer including any additions made later.

It is worth noting that the number of buffers inserted to achieve path balance in the original circuit does not change with the order of cells within each row. This implies that these buffers are necessary. Furthermore, the buffer insertion methods employed are both straightforward and unequivocal. Hence, in this paper, the consideration of path balancing for the original circuit will be omitted. It is assumed that the circuits in the context have already undergone the path balancing process by default. In the subsequent paper, we will provide a detailed explanation of our proposed method.

3 Proposed Framework

Our work consists of two stages: Topology Initialization and Placement and Buffering. This section would comprehensively explain the implementation specifics for both stages.

3.1 Topology Initialization

In topology initialization, our objective is to optimize the order of cells in each row. The placement based on this can serve as an initialization for further placement and buffering, significantly accelerating its iterations. The algorithm is presented in Algorithm 1. Some details of the algorithm will be further explained in the following.

Entanglement Entropy. We examine wirelength violation in a topological scenario, as depicted in Fig. 3, to illustrate the entanglement entropy. By following the steps shown, we can easily eliminate the violation. A noticeable geometric change is the gradual reduction of wire intersections. This observation suggests that a better-designed circuit topology may have fewer intersections. However, solely counting intersections is inadequate. First, finding a simple function accurately measuring the quantity of intersections is challenging. Second, the perception of intersection count often fails in complex scenarios where a cell emits or receives multiple wires, as shown in Fig. 4.

Building upon the preceding analysis, we introduce a novel term, called entanglement entropy, as a comprehensive definition for the entire circuit. To formulate this definition, let us denote the set of all wires l in the actual circuit as S. We consider a partition of S into subsets S_1, S_2, ..., S_i, where S_i represents the set of wires in the gap between the i-th and (i + 1)-th rows. Consequently, the entanglement

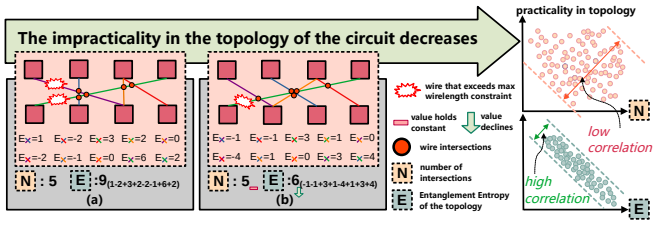

Figure 4: Two topologies exhibits the same 5 intersections but different entanglement entropy. Entanglement entropy in (a) and (b) reach 9 and 6, respectively. Wirelength violations are fewer in (b).

entropy (E) of the entire circuit can be expressed as follows:

$$E = \sum_i \sum_{l_1, l_2 \in S_i} -(X_{l_1} - X_{l_2}) \times (Y_{l_1} - Y_{l_2}). \tag{1}$$

Here, X_{l_1} represents the order of the endpoints of wire l_1 in the previous row, while Y_{l_1} denotes the order of endpoints in subsequent rows. The equation captures the interplay between the horizontal and vertical positions of the wire endpoints within each subset S_i. Upon examining Fig. 3, it becomes evident that E exhibits a gradual decrease in steps, resembling intersections. This characteristic suggests that the entanglement entropy consistently decreases, effectively minimizing the number of intersections within the circuit placement. The visual representation of E in Fig. 4 further highlights its ability to indicate a relatively better topology, even when the conventional intersection-counting perception fails to do so.

Reduction of Entanglement Entropy. The key point of this algorithm lies in efficiently computing the reduction of entanglement entropy (ΔE) to optimize the process. A crucial aspect is avoiding the inefficient recalculation of E and taking the difference for each exchange of two cells within the same row. Therefore, it becomes imperative to derive an explicit expression for ΔE to streamline the computation.

To derive the expression for ΔE, let us consider E_i as the entanglement entropy in the gap between the i-th and (i+1)-th row. Suppose the m-th and n-th cells in the previous row are exchanged. As a result, S_i can be divided into subsets: $S_{i,1}$ and $S_{i,2}$ representing the wires connected to the two cells, respectively, while the remaining wires form the set $S_{i,3}$. These subsets contain elements t_1, t_2, and $N_i - t_1 - t_2$, respectively, where N_i represents the number of elements in S_i. Before the exchange, E_i can be expressed as in Eq. (2). Here W

$$\begin{aligned} E_i = &- \sum_{l_k \in S_{i,3}} \sum_{l_f \in S_{i,1}} (m - X_{l_k})(Y_{l_f} - Y_{l_k}) \\ &- \sum_{l_k \in S_{i,3}} \sum_{l_g \in S_{i,2}} (n - X_{l_k})(Y_{l_g} - Y_{l_k}) \\ &- \sum_{l_f \in S_{i,1}} \sum_{l_g \in S_{i,2}} (m - n)(Y_{l_f} - Y_{l_g}) + W. \end{aligned} \tag{2}$$

represents the terms that do not include m and n. After the exchange, E_i transforms into E_i' with exchange m and n in Eq. (2). Hence, ΔE_i can be obtained by taking the difference as in Eq. (3). In fact, it is not difficult to deduce that:

$$\Delta E = \Delta E_{i-1} + \Delta E_i. \tag{4}$$

$$
\begin{aligned}
\Delta E_i &= E_i - E_i' \\
&= -\sum_{l_k \in S_{i,3}} \sum_{l_f \in S_{i,1}} (m-n)(Y_{l_f} - Y_{l_k}) \\
&- \sum_{l_k \in S_{i,3}} \sum_{l_g \in S_{i,2}} (n-m)(Y_{l_g} - Y_{l_k}) \\
&- \sum_{l_f \in S_{i,1}} \sum_{l_g \in S_{i,2}} (m-n)(Y_{l_f} - Y_{l_g}) \\
&= (m-n)\{(t_1 - t_2)(\sum_{l_f \in S_{i,1}} Y_f + \sum_{l_k \in S_{i,3}} Y_k + \sum_{l_g \in S_{i,2}} Y_g) \\
&+ N_i(\sum_{l_g \in S_{i,2}} Y_g - \sum_{l_f \in S_{i,1}} Y_f)\}.
\end{aligned} \tag{3}
$$

Thus, Eq. (3) and Eq. (4) will assist in circumventing redundant computations of ΔE, thereby greatly enhancing the efficiency of topology initialization.

Order Initialization. Assuming $m > n$, then the polarity of ΔE would be determined by the following two terms:

$$
N_i(\sum_{l_g \in S_{i,2}} Y_g - \sum_{l_f \in S_{i,1}} Y_f), \tag{5}
$$

$$
(t_1 - t_2)(\sum_{l_f \in S_{i,1}} Y_f + \sum_{l_k \in S_{i,3}} Y_k + \sum_{l_g \in S_{i,2}} Y_g). \tag{6}
$$

In many cases, Eq. (5) could be neglected compared to Eq. (6). Thus, when $t_1 > t_2$, there is a high probability that Eq. (4) will be greater than 0. This means that in the topology with lower E, in each row, cells connected to more wires are more likely to be located on the left end. Inspired by this fact, in topology initialization, we arrange the cells in each row in descending order according to the number of connected wires.

3.2 Placement and Buffering

In this section, we present an analytical method for placement and buffering, as outlined in Algorithm 2. It constitutes the complete placement work flow in conjunction with the Topology Initialization. Some details on specific steps within the algorithm will be elaborated in subsequent subsections.

Coordinate Initialization. In this module, we precisely position each cell at specific coordinates based on the circuit topology derived from the Algorithm 1, serving as the initialization for subsequent iterations. Our convention assumes that the data flow in Fig. 1 occurs along the positive y-axis, with the positive x-axis representing the perpendicular direction to the right.

To reduce the wire cost while avoiding cell overlap, for y-coordinate, we ensure that the minimum coordinate of the top-left corner of cells in a row equals the maximum coordinate of the bottom-left corner of cells in the row above. Additionally, we align the center of all cells within each row with the same y-coordinate. For the first row, we set the minimum y-coordinate of the top-left corner of the cell as 0. Due to analogous considerations, cells are arranged tremendously closely within each row in accordance with the sequence established by topology initialization. The upper left corners of the leftmost cells in each row are aligned, and the x-coordinate is defined as 0. Throughout the iteration, we only adjust the x-coordinate of each cell.

Algorithm 1 Topology Initialization

Input: Netlist including size, phase, and connection of all cells.
Output: Circuit topology with relatively low entanglement entropy.

```
Order initialization;
for iter = 1 to MaxIter do
    Exchange two cells in the same row;
    Calculate the entanglement entropy reduction ΔE;
    if ΔE > 0 then
        Update the topology;
    end if
    if convergence then
        break;
    end if
end for
```

Ideal Position for Sole Cell. Ideal location plays a pivotal role in minimizing the maximum length of the wires connected with the cell. Therefore, determining the ideal position is the core of this algorithm. In our following analysis, we assume that the distribution of cells connected to a given cell follows the pattern depicted in Fig. 5.

Considering the relatively small size of cells within the scope of this paper, we can disregard variations in the y-axis projection of each wire. Consequently, the connections among the cells and buffer centers in each row can be approximated as a set of equidistant parallel lines. Due to the symmetry, for simplicity of analysis, we can project cells located on different lines onto a single line, as illustrated in Fig. 5.

After the projection, by observing that the lengths of all wires cannot simultaneously exceed the lengths of the outer two wires, the ideal position of each cell becomes apparent: it is the average of the x-coordinates of the leftmost and rightmost connected cells.

Buffer Insertion. If wirelength violations persist even after the iteration process has converged, buffer insertion should be employed to eliminate them. We insert one buffer row at a time, instead of multiple rows, as the additional freedom introduced by each newly inserted buffer may potentially resolve remaining violations during the algorithm iteration. The y-coordinate of the inserted buffer is determined using the same approach as in Coordinate Initialization. As for the x-coordinate, it is set in the middle of the dual-ended cells of the truncated wire.

X-Coordinate Adjustment. Adjusting cell positions and inserting buffers may result in overlaps among cells and buffers, requiring a secondary adjustment to their x-coordinate. In the secondary adjustment, the x-coordinates of all cells require the following computation: $x_j^i = \max(x_j^i, x_{j-1}^i + w(x_{j-1}^i))$, where x_j^i represents the x-coordinate of the top-left corner of the cell in the i-th row and j-th position, and $w(x_j^i)$ represents its width.

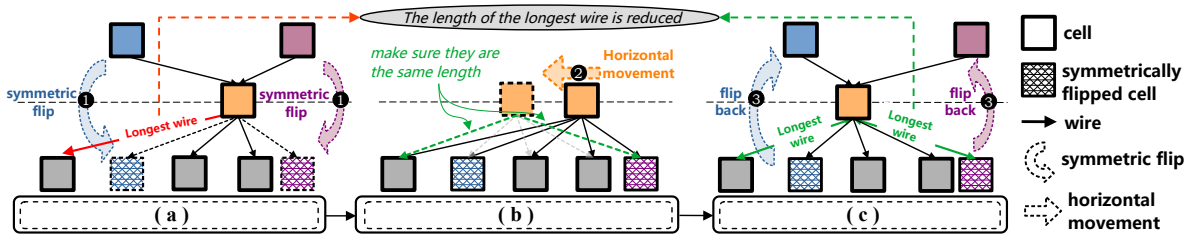

Figure 5: Move cells to the ideal position: (a) Flip two cells onto a different row while preserving the wirelength through symmetry. (b) Move the cell to the average of rightmost and leftmost connected cells. (c) Positioning the cell on this ideal position would minimize the maximum wirelength. Flip the two cells back.

Algorithm 2 Placement and Buffering

Input: The circuit topology generated by topology initialization
Output: Coordinate of each cell;

```
Coordinate Initialization;
Place the cells with wirelength violations on the ideal position;
X-coordinate adjustment;
Record the adjacent rows of the adjusted cell.
for row in recorded ones do
    if row is recorded a set times then
        Buffer insertion
        Recording the inserted row
        X-coordinate adjustment
    else
        Place the cells in the row with wirelength violations on the ideal position
        Recording adjacent row
        X-coordinate adjustment
    end if
end for
Row distance adjustment;
```

Row distance Adjustment. In actual circuit layouts, it is undesirable to have adjacent cells abutting each other in adjacent rows. Therefore, it is necessary to adjust the distance between rows after the iteration. We increment the distance until the first wirelength violation occurs, ensuring that the wirelength violation do not arise while maintaining a reasonable gap between adjacent rows.

4 Experimental Result

In order to assess the performance of our proposed algorithm, we implement it using the C++ programming language and conduct tests on an AMD EPYC 7702 CPU operating at a clock speed of 3.6 GHz, with 512 GB of RAM. To optimize its runtime speed and efficiency, we compile the implementation using the -O3 option of g++ 9.4.0. To evaluate the effectiveness of our algorithm, we test it on benchmark circuits shown in Table 1 [12]. "#Cell" and "#Buffer" represent the quantities of cells and buffers contained in the path-balanced cases, respectively. In these test cases, cells range from 7 to 3256, and buffers range from 6 to 17701. This set of benchmark circuits has been carefully selected to ensure a wide coverage in terms of structure, scale, and characteristics, thereby representing diverse layout requirements encountered in practical applications.

Table 1: Test cases and detailed information.

Index	Case	#Cell	#Buffer	Index	Case	#Cell	#Buffer
1	c17	7	6	7	c2670	718	1839
2	c432	183	812	8	c3540	1183	3639
3	c499	588	1151	9	c5315	1915	8762
4	c880	444	2184	10	c6288	3256	17701
5	c1355	584	1091	11	c7552	2129	7666
6	c1908	468	1782				

We conduct experiments to compare the proposed algorithm with two state-of-the-art placement algorithms, namely ASAP [6] and the GA-based method [4, 5]. Specifically, we compare their generated circuits in two primary dimensions, including time overhead and buffer insertion. The results are shown in Table 2. "#Buffer" and "#Row" represent the quantities of additional buffers and buffer rows inserted to eliminate all wirelength violations while ensuring path balance. The "Per1" and "Per2" represent ratio of the performance of the proposed method and that of the GA method and ASAP, respectively. The "Speedup1" and "Speedup2" stand for the inverse ratio.

4.1 Comparison on Buffer Insertion

According to Table 2, the GA method is only suitable for small benchmark cases such as c17 and c432. As the size of the circuit increases, the number of inserted rows and buffers increases significantly, even achieving 83 in c3540 and 13657 in c5315, respectively. ASAP performs better in larger benchmark tests, with more than half of the benchmark cases requiring no additional inserted buffer and row, especially ranging from c432 to c1908. The proposed algorithm surpasses all previous algorithms, with no buffer insertion in the majority of cases, particularly in c2670 and c5315. Moreover, in some cases where the insertion of buffers is unavoidable, our proposed algorithm can reduce the required number of inserted buffers and rows compared to ASAP. Projection-based methods play a crucial role in minimizing the maximum length of wires.

4.2 Comparison on Time Overhead

In Table 2, ASAP achieves a substantial acceleration over the GA method due to the transformation from Lagrangian subproblem to the shortest path problem. The average time overhead has been reduced by at least 95%. Breakthroughs have been achieved even in numerous cases ranging from c880 to c7552, where GA methods are highly likely to fail to converge. Our algorithm has taken a significant step forward on this basis, which achieves an average runtime that

Table 2: Comparisons among algorithms.

Case	GA Method [5]			ASAP [6]			Proposed Algorithm								
	#Buffer	#Row	Runtime(s)	#Buffer	#Row	Runtime(s)	#Buffer	Per1	Per2	#Row	Per1	Per2	Runtime(s)	Speedup1	Speedup2
c17	0	0	141.9	0	0	8.55	0	-	-	0	-	-	3.78	37.54x	2.26x
c432	37	1	1472.09	0	0	25.71	0	100%	-	0	100%	-	9.86	149.30x	2.61x
c499	1848	24	2983.3	0	0	40.99	0	100%	-	0	100%	-	22.56	132.24x	1.82x
c880	2828	38	>3600	0	0	54.85	0	100%	-	0	100%	-	31.92	>112.78x	1.72x
c1355	2126	28	>3600	0	0	41.64	0	100%	-	0	100%	-	19.62	>183.49x	2.12x
c1908	1732	26	>3600	0	0	60.46	0	100%	-	0	100%	-	43.56	>82.64x	1.39x
c2670	6376	48	>3600	315	2	81.29	0	100%	100%	0	100%	100%	52.97	>67.96x	1.53x
c3540	10600	83	>3600	2294	13	94.96	834	92%	64%	5	94%	62%	42.15	>85.41x	2.25x
c5315	13657	45	>3600	1426	5	374.04	0	100%	100%	0	100%	100%	283.64	>12.69x	1.31x
c6288	2838	12	>3600	0	0	158.48	0	100%	-	0	100%	-	192.53	>18.70x	0.82x
c7552	11594	48	>3600	4170	17	590.86	1766	85%	58%	9	81%	47%	208.72	>17.25x	2.83x
Average	-	-	-	-	-	-	-	98%	81%	-	98%	77%	-	>81.82x	1.88x

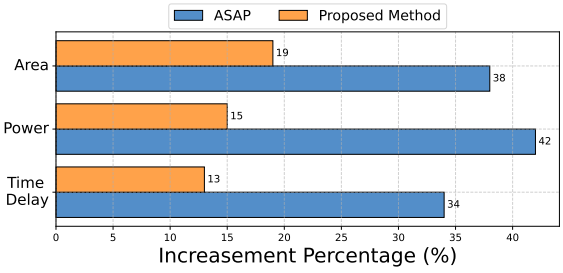

Figure 6: Comparison on increasement percentage of different metrics, including time delay, power and area, for c3540 before and after different placement strategies.

is at least 81.82x of ASAP's and 1.88x of GA method's, even with the ability to achieve up to 2x acceleration across multiple cases compared to the latter. It demonstrates that the circuit layout based on the topology with lower entanglement entropy could facilitate a faster convergence of detailed placement. The failure of c6288 could potentially be attributed to the absence of generating such a circuit topology during the global placement stage.

4.3 Comparison on Circuit Metrics

Time delay, power consumption, and area serve as crucial metrics to evaluate the overall performance of practical circuits and are significantly influenced by the placement phase. Therefore, for a more comprehensive assessment of the effectiveness of the proposed method, in addition to the previous direct comparison, it is imperative to simulate and measure the practical circuits obtained from the placement process and subsequently compare their parameters. Taking c3540 as an example, the circuits processed by ASAP and the proposed method were evaluated, and the corresponding parameters are illustrated in Fig. 6. Remarkably, the circuits processed by the proposed method exhibit a substantial reduction in time delay, power consumption and area simultaneous, indicating greater promise of the proposed method for industrial applications in comparison to previously proposed placement strategies.

5 Conclusion

In this paper, we propose a novel analytical algorithm for the efficient placement of AQFP logic circuits. Our primary objective is to minimize the additional buffers inserted in the AQFP circuits to improve the power, performance, and area. By introducing the entanglement entropy, we successfully acquiring the topology that could make detailed placement convergence fast. We propose a detailed placement method that could finally decide the specific coordinate of each cell. Additionally, we incorporate buffer insertion with refined cell positions to achieve a legalized placement solution. The experimental results demonstrate the superiority of the proposed method over previous methods, with an average reduction of 98% and 81% in buffer requirements and 81.82x and 1.88x in time requirements compared with two advanced algorithm.

Acknowledgment

This work was supported by NSFC (Grant No.62204265, 62234010, U23A20301). Zhou Jin is the corresponding author.

References

[1] N. Takeuchi, Y. Yamanashi, and N. Yoshikawa. Energy efficiency of adiabatic superconductor logic. *Supercond Sci Technol*, 2014.
[2] S. K. Tolpygo, V. Bolkhovsky, T. J. Weir, A. Wynn, D. E. Oates, L. M. Johnson, and M. A. Gouker. Advanced fabrication processes for superconducting very large-scale integrated circuits. *IEEE T APPL SUPERCON*, 2016.
[3] K. Inoue, N. Takeuchi, K. Ehara, Y. Yamanashi, and N. Yoshikawa. Simulation and experimental demonstration of logic circuits using an ultralow-power adiabatic quantum-flux-parametron. *IEEE T APPL SUPERCON*, 2013.
[4] 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 T APPL SUPERCON*, 2017.
[5] T. Tanaka, C. L. Ayala, Q. Xu, R. Saito, and N. Yoshikawa. Fabrication of adiabatic quantum-flux-parametron integrated circuits using an automatic placement tool based on genetic algorithms. *IEEE T APPL SUPERCON*, 2019.
[6] Y. Chang, H. Li, O. Chen, Y. Wang, N. Yoshikawa, and T. Ho. Asap: An analytical strategy for aqfp placement. In *ICCAD*, 2020.
[7] 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*, 2021.
[8] S. Lee, H. Riener, and G. De Micheli. Beyond local optimality of buffer and splitter insertion for aqfp circuits. In *DAC*, 2022.
[9] H. Fan, C. Guo, and W. Luk. Optimizing quantum circuit placement via machine learning. In *DAC*, 2022.
[10] F. Chang, Y. Tseng, Y. Yu, S. Lee, A. Cioba, I. Tseng, D. Shiu, J. Hsu, C. Wang, C. Yang, R. Wang, Y. Chang, T. Chen, and T. Chen. Flexible chip placement via reinforcement learning: Late breaking results. In *DAC*, 2022.
[11] C. Cheng, A. B. Kahng, I. Kang, and L. Wang. REPLACE: Advancing solution quality and routability validation in global placement. *IEEE TCAD*, 2019.
[12] https://sportlab.usc.edu/~msabrishami/benchmarks.html, 2023.
[13] J. Clarke and A. Braginski. *The SQUID Handbook: Applications of SQUIDs and SQUID Systems.* Wiley-VCH Verlag GmbH & Co. KGaA, 2006.
[14] N. Takeuchi, Y. Yamanashi, and N. Yoshikawa. Adiabatic quantum-flux-parametron cell library adopting minimalist design. *J. Appl. Phys*, 2015.
[15] 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*, 2022.