Towards AQFP-Capable Physical Design Automation

Hongjia Li[1], Mengshu Sun[1], Tianyun Zhang[2],
Olivia Chen[3], Nobuyuki Yoshikawa[3], Bei Yu[4], Yanzhi Wang[1], Yibo Lin[5]
[1]*Northeastern University,* [2]*Syracuse University,*
[3]*Yokohama National University,* [4]*Chinese University of Hong Kong,* [5]*Peking University*
[1]{li.hongjia, sun.meng, yanz.wang}@northeastern.edu, [2]tzhan120@syr.edu,
[3]{chen-olivia-pg, yoshikawa-nobuyuki-gt}@ynu.ac.jp, [4]byu@cse.cuhk.edu.hk, [5]yibolin@pku.edu.cn

***Abstract*—Adiabatic Quantum-Flux-Parametron (AQFP) superconducting technology exhibits a high energy efficiency among superconducting electronics, however lacks effective design automation tools. In this work, we develop the first, efficient placement and routing framework for AQFP circuits considering the unique features and constraints, using MIT-LL technology as an example. Our proposed placement framework iteratively executes a fixed-order, row-wise placement algorithm, where the row-wise algorithm derives optimal solution with polynomial-time complexity. To address the maximum wirelength constraint issue in AQFP circuits, a whole row of buffers (or even more rows) is inserted. A* routing algorithm is adopted as the backbone algorithm, incorporating dynamic step size and net negotiation process to reduce the computational complexity accounting for AQFP characteristics, improving overall routability. Extensive experimental results demonstrate the effectiveness of our proposed framework.**

I. Introduction

Superconducting computing technology has significantly enhanced power efficiency over state-of-the-art CMOS. Adiabatic quantum-flux-parametron (AQFP) logic achieves a significant reduction in both static and dynamic power consumption [1] by adopting adiabatic switching [2], in which the potential energy profile evolves from a single well to a double-well so that the logic state can change quasi-statically. By using processes such as the AIST standard process 2 (STP2) [3] or the MIT-LL SFQ process [4] of AQFP circuits fabrication, it can potentially achieve $10^4 - 10^5$ energy efficiency gain compared with state-of-the-art CMOS with a clock frequency of several GHz [5].

To deliver the extraordinary promise in energy efficiency, it is critical to investigate the choice of logic gates, circuits and architectures and develop effective design automation techniques for AQFP circuits. The currently mature design automation tools for CMOS cannot be directly applied to the design of superconducting electronics, due to different active components (transistor in CMOS vs. JJ in AQFP), different passive components, different suites of basic logic gates (AQFP relies on the efficient realization of majority/minority gates), different clocking schemes (e.g., 4-phase clocking in AQFP), and the cost of buffers, splitters, and AC biasing. This work focuses on the physical design of AQFP and presents the first systematic, AQFP-specific placement, and routing framework.

AQFP circuits exhibit some unique characteristics with implications on physical design. After logic synthesis for the AQFP circuit, the following procedure inserts splitters for fanouts and buffers for balancing the path delay (clock phase) to each gate [6]. As a result, each logic cell gets assigned in a specific, fixed clock phase, which corresponds to a specific row. Additionally, some specific properties can be advantageous for the routing procedure. Due to the design requirements, wire connection only exits from row i to row $i+1$ and is 1-to-1 in AQFP circuits. And the distance between two nearby rows of logic cells can be flexible. At the downside, AQFP circuits contain some restrictions, including spacing constraint between adjacent cells (either abut or keeping a minimum spacing), zigzag spacing constraint due to the minimum wire segment length, the maximum wirelength constraint WL_{max} beyond which an additional row of buffers need to be inserted (in this case the corresponding placement and routing procedure needs to be performed again), etc. These various spacing constraints result in a larger number of combinatorial constraints than CMOS-based placement and routing problems, thereby restricting the direct utilization of CMOS-based placement and routing techniques to AQFP.

Our proposed AQFP placement and routing framework effectively overcome these challenges. In AQFP placement, we have the key observation: *There exists a polynomial-time optimal solution for the row-wise placement problem with cell order fixed*, or more precisely, the solution is proved to be arbitrarily close to the optimal one within polynomial time. We formulate the fixed-order, row-wise placement algorithm as a discrete optimization problem with a large number of combinatorial constraints, which is difficult to solve directly. We provide the optimal solution through (i) performing Lagrangian relaxation concerning the maximum wirelength constraint, (ii) solving the Lagrangian subproblem by transforming it to an equivalent shortest path problem in a graph and solving using dynamic programming, and (iii) solving the row-wise placement problem using the subgradient method. The overall AQFP placement is based on the row-wise algorithm starting from a cell order determination process, effectively accounting for the possible violation of the maximum wirelength constraint, in which case row(s) of buffers will be inserted. Compatible with logic cell row placement, the placement of buffer row can be solved using our placement technique. The row of buffers will be co-placed together with logic cell rows for facilitating routability and satisfying the wirelength constraint. For AQFP routing, we adopt the A* routing algorithm [7] as the backbone algorithm, incorporating dynamic step size to reduce the computational complexity accounting for AQFP characteristics. We further

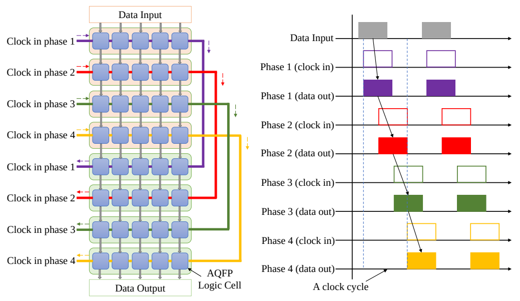

Fig. 1: Four phase clocking scheme and data propagation for AQFP circuits.

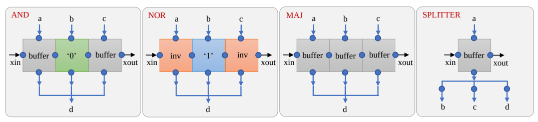

Fig. 2: Examples of AQFP logic gates (AND, NOR, MAJ, and splitter).

incorporate the net negotiation process [8] to avoid the time-consuming rip-up and re-route and improve routability, resulting in a fast overall routing speed.

Experimental results demonstrate the effectiveness of the proposed framework. On representative benchmark circuits, the proposed framework achieves up to 40.2% reduction in total routed wirelength compared with baselines, with fast speed and satisfying all constraints. Moreover, the proposed framework enables the efficient physical design of a full 32-bit ALU for the RISC-V processor, beyond the prior design capability of AQFP which cannot satisfy design constraints simultaneously, illustrating the benefit of design automation.

II. Background and Motivation

A. AQFP Superconducting Logic

The standard AQFP cell library [9] is built via the minimalist design approach [9], including basic logic gates such as AND, OR, NOT, MAJORITY, BUFFER, and SPLITTER. An AQFP logic gate is driven by AC-power, which serves both as the excitation current and synchronization mechanism [10]. The AQFP buffer is based on a double-Josephson-Junction SQUID [11], which is the basic structure in AQFP circuits. The AQFP inverter and constant cell are designed based on the AQFP buffer [6]. 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. Additionally, the splitter in AQFP is also implemented from the AQFP buffer. Unlike the direct fan-out in CMOS gates, in the AQFP circuit, splitters are utilized to fan-out for all gates (when the number of fan-outs is 2 or more). Fig. 2 shows examples of AQFP logic gates.

In AQFP circuits, both combinational and sequential AQFP logic cells are driven by AC-power compared with the conventional CMOS technology. Other than the excitation current, the AC power also serves as the synchronization mechanism, i.e., a clock signal to synchronize the outputs of all gates in the same clock phase. Hence, data propagation in AQFP circuits requires overlapping of clock signals from nearby phases. An example of a four-phase clocking scheme of AQFP circuits and corresponding data flow is shown in Fig. 1. As shown in Fig. 1, 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.

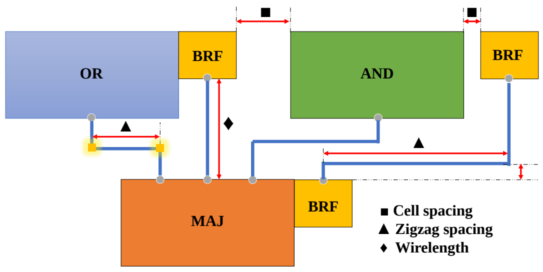

Fig. 3: The Illustration of spacing constraints in the AQFP placement with the detail of interconnect via.

B. Uniqueness in the AQFP Placement and Routing Problem

As shown in Fig. 1, after logic synthesis and buffer/splitter insertion for path balancing, each logic cell is assigned in a specific, fixed clock phase, which corresponds to a specific row in the figures. The placement procedure will not change the specific row index that a logic cell resides in. It will only change the relative, horizontal locations of logic cells within each row.

There are some desirable properties that the routing procedure could make use of as well. (i).The wire connection is only from row i to row $i+1$ in AQFP circuits. No connection traverses back or skips certain intermediate rows. (ii). Because of the requirement of splitter and splitter design (of course there is a requirement to choose the appropriate connection for each splitter output), each wire connection is 1-to-1. There is no wire connection with more than one fanouts. (iii). The distance between two nearby rows of logic cells can be flexible. These properties enable separate routing between every two consecutive rows, and the routing process can be potentially very fast.

C. Challenges in the AQFP Placement and Routing Problem

On the other hand, some characteristics may complicate the placement and routing procedure. As shown in Fig. 3, the first category is a large number of spacing constraints, including cell spacing and zigzag spacing, which lead to a large number of combinatorial constraints in the AQFP placement problem. Firstly, two horizontally neighboring cells in a row can be either abutting or keeping a minimum spacing (i.e., cell spacing). Secondly, specific vias are adopted to perform the wire direction shift in AQFP circuits as shown in Fig. 3. Due to the unique vias structure, these zigzags need to have at least certain pre-defined spacing (e.g. $10\mu m$ for MIT-LL process). The second category is that the maximum length of a single connection is also limited to WL_{max} (e.g., $1mm$ in the MIT-LL process), beyond which additional buffer will be required. Please note that an entire row of buffers needs to be inserted for path logic balancing in AQFP.

These characteristics need to be accounted for to achieve functional and efficient placement and routing procedures. Due to the various spacing constraints, there exists a larger number of combinatorial constraints than CMOS-based placement problems, especially row-based placement ones (e.g., [12], [13]). Also, the maximum wirelength constraint is absent (or as a soft constraint) in CMOS placement and routing, not to mention the associated buffer insertions. These distinctions make it difficult to directly apply CMOS-based placement solutions to AQFP circuits.

III. Proposed AQFP Physical Design Framework

The overall AQFP placement framework starts from a cell order determination process, mainly executes our proposed fixed-order row-wise placement algorithm, and finally our AQFP routing solution.

A. Fixed-Order Row-Wise Placement Algorithm

In this section, we refine the locations of AQFP cells *within a placement row* by assuming cells in other rows are fixed. Assume we are given an ordered sequence of cells from left to right $1, 2, \ldots, N$. The optimization will keep the order of cells within the row. Suppose there are N cells in the row and each cell i has a set of M discrete candidate locations $L_i = \{x_i^1, x_i^2, \ldots, x_i^M\}$ (corresponding to grids in AQFP fabrication). The location here refers to the lower-left corner of the cell. Later we will see that the selection of candidate locations can be utilized for cell legalization (satisfying various spacing constraints for AQFP).

Let w_i be the width of cell i. Let $P_i = \{p_i^1, p_i^2, \ldots, p_i^{K_i}\}$ be the set of pins for cell i. Without abuse of symbols, we also use them to represent offsets of pins w.r.t the lower-left corner of the cell. Let E be the set of nets that the cells in the row are incident to. For each net e, we use $p_i^k \in e$ to denote the pin of a cell i incident to the net. Then, we use $WL(e)$, which is short for $WL(e; x_i, \forall p_i^k \in e)$, to denote the wirelength cost of net e. For example, if we consider half-perimeter wirelength (HPWL), $WL(e)$ can be written as,

$$WL(e) = \max_{p_i^k \in e}(x_i + p_i^k) - \min_{p_i^k \in e}(x_i + p_i^k), e \in E. \quad (1)$$

This is a general problem formulation that applies to multi-pin nets and will work of course for AQFP nets which are restricted to two pins. For a set of nets $E_m \subseteq E$, there is a maximum limit of wirelength, WL_{max}. This is a general framework and it is possible that $E_m = E$.

The mathematical formulation for the fixed-order row-wise placement can be written as follows,

$$\begin{aligned}
\min_{x_i} \quad & \sum_{e \in E} WL(e), && && \text{(2a)}\\
\text{s.t.} \quad & WL(e) \le WL_{max}, && \forall e \in E_m, && \text{(2b)}\\
& x_{i-1} + w_{i-1} \ge x_i - Bz_i, && i = 2, \ldots, N, && \text{(2c)}\\
& x_{i-1} + w_{i-1} \le x_i - s_i^{min} z_i, && i = 2, \ldots, N, && \text{(2d)}\\
& x_i \in \{x_i^1, x_i^2, \ldots, x_i^M\}, && i = 1, 2, \ldots, N && \text{(2e)}\\
& z_i \in \{0, 1\}, && i = 1, 2, \ldots, N-1, && \text{(2f)}
\end{aligned}$$

where we introduce binary variable z_i, big positive constant B, and (2c), (2d) to ensure the placement is legal, as two horizontally neighboring cells in a row can either be abutting or keeping a minimum spacing s_i^{min}. If $z_i = 0$, then the two equations can be combined to $x_{i-1} + w_{i-1} = x_i$; if $z_i = 1$, then only (2d) will be activated as $x_{i-1} + w_{i-1} \le x_i - s_i^{min}$. Spacing requirements between pins can also be converted to the spacing between cell pairs because pin offsets are known given a cell. Thus, minimum spacing s_i^{min} varies from cell pair to cell pair, to generalize the spacing requirement between cells and pins. Equation (2b) guarantees special nets to satisfy the maximum wirelength constraint. Or we may incorporate parameter $\gamma < 1$ such that $WL(e) \le \gamma \cdot WL_{max}$, as $WL(e)$ only accounts for the horizonal distance, which is a lower bound of the wirelength estimate. Equation (2e) guarantees to select valid locations given fixed cells in other rows; this helps to satisfy the horizontal spacing requirement between pins in the target row and other fixed rows (i.e., zigzag spacing constraints).

As mentioned above, our target problem is a discrete optimization problem containing extensive combinatorial constraints, which is difficult to solve directly. On the other hand, if we relax (2b) into the objective with Lagrangian multiplier λ_e, we obtain the Lagrangian problem $\mathcal{L}(x, \lambda)$ as follows,

$$\begin{aligned}
\min_{x_i} \quad & \sum_{e \in E} WL(e) + \sum_{e \in E_m} \lambda_e (WL(e) - WL_{max}), && \text{(3a)}\\
\text{s.t.} \quad & \lambda_e \ge 0, && \text{(3b)}\\
& \text{Equation (2c)} \sim \text{(2f)}. && \text{(3c)}
\end{aligned}$$

Given a specific λ, the Lagrangian subproblem $\mathcal{L}(x; \lambda)$ tries to minimize a weighted wirelength. $\mathcal{L}(x; \lambda)$ is still a discrete optimization problem. However, when we effectively select the candidate locations of cells for cell legalization (satisfying the spacing constraints in AQFP), $\mathcal{L}(x; \lambda)$ essentially models a shortest path problem in a graph [14]–[16]. The cost of an edge in the graph can be written as,

$$c(x_{i-1}^u, x_i^v) = SP(x_{i-1}^u, x_i^v) + \sum_{\substack{\exists p_{i-1} \in e,\\ \forall e \in E}} w_e \cdot WL(e),$$

$$SP(x_{i-1}^u, x_i^v) = \begin{cases} 0, & \text{if } x_{i-1}^u + w_{i-1} = x_i^v \text{ or} \\ & x_{i-1}^u + w_{i-1} \le x_i^v - s_i^{min}, \\ \infty, & \text{others}, \end{cases} \quad (4)$$

$$w_e = \begin{cases} 1 + \lambda_e, & \text{if } e \in E_m, \\ 1, & \text{others}, \end{cases}$$

where the first term SP denotes the spacing cost between two neighboring cells, and the second term denotes the total weighted wirelength cost of cell $i-1$ for all the nets it incidents to. It needs to be noted that the formulation assumes a pin in the row will only connect to other pins outside the row, which is true in the AQFP circuits.

With the analysis so far, we demonstrate that the Lagrangian subproblem $\mathcal{L}(x;\lambda)$ is equivalent to the shortest path problem, which can be solved optimally in polynomial time. Then we can solve the original Lagrangian problem $\mathcal{L}(x,\lambda)$ by iteratively solving the subproblem and updating the multiplier λ_e. The *subgradient method* is adopted to update the multiplier at the kth iteration,

$$\lambda_e^{k+1} = \max(0, \lambda_e^k + t_k(WL(e) - WL_{max})), \forall e \in E_m, \quad (5)$$

where t_k controls the step size of updating. According to the convergence condition given by [17], i.e., $t_k \to 0, \sum_{i=1}^{k} t_i \to \infty$ when $k \to \infty$, we use $t_k = \frac{1}{k^\alpha}$, with constant $\alpha < 1$ to control the solution quality and convergence speed.

Complexity analysis. The shortest path problem for the Lagrangian subproblem $\mathcal{L}(x;\lambda)$ can be solved with topological traversal or dynamic programming in $\mathcal{O}(M^2N)$ steps, with M as the number of discrete locations for each cell and N is the number of cells in the row [15]. The subgradient method has a convergence rate of $\mathcal{O}(\frac{1}{k^\alpha}), \alpha < 1$. If we set $\alpha = 0.5$, then it is $\mathcal{O}(\frac{1}{\sqrt{k}})$. In other words, to get $\mathcal{L}(\lambda_{best}^k) - \mathcal{L}(\lambda^*) \le \epsilon$, it requires $\mathcal{O}(\frac{1}{\epsilon^2})$ iterations [17]. In practice, it converges much faster than the bound, e.g., it takes 2,247.8s using the bound on the 8-bit Kogge-Stone adder, while our method takes only 66.38s for convergence.

Algorithm 1: Fixed-Order Row-Wise Placement Algorithm

```
1 Given an ordered sequence of cells $1, 2, \ldots, N$;
2 Determine the optimal and legal locations of cells in
    this order;
3 $\lambda_e \leftarrow 1, \forall e \in E$;
4 while Not converged do
5     Solve Lagrangian subproblem $\mathcal{L}(x;\lambda)$ with the
        shortest path algorithm;
6     Update $\lambda$ according to (5);
7 end
8 return cell locations $x$;
```

B. Buffer Row(s) Insertion to Satisfy Maximum Wirelength Constraint

The above algorithm (Algorithm 1) provides a row-wise placement solution when the maximum wirelength constraint can be satisfied. On the other hand, through the subgradient method, it will indicate if WL_{max} cannot be satisfied simultaneously for all rows. In this case, certain row(s) of buffers need to be inserted, with more details to be described next. For large-scale AQFP circuits, it is possible that the maximum wirelength constraint cannot be satisfied in row-wise placement due to very wide rows. In this case, we need to insert a whole row of buffers (or even multiple rows) between these two consecutive rows. Moreover, the row of buffers needs to be co-placed together with logic cell rows for facilitating routability and satisfying wirelength constraint.

First, buffers are ordered using a greedy scheme. Next, the placement of the buffer row can be solved using the fixed-order row-wise placement algorithm presented in Algorithm 1 since both the top row and bottom row are fixed. Through the subgradient method, if the maximum wirelength constraint WL_{max} cannot be satisfied, more buffer rows need to be inserted. The procedure will be repeated until there is no violation of the maximum wirelength constraint.

C. Overall Placement Algorithm

The overall placement algorithm is based on the above fixed-order, row-wise placement solution. In the first step, we apply the existing cell order determination placement algorithm (e.g., the classical GORDIAN [18]) to determine the cell order in each row. Please note that the 1-D version of such an algorithm (which applies to our target problem) will be very sufficient. Next, we calculate the *estimated row width* as the summation of cell widths in one row, and this value for each row will not change after logic synthesis. We fix the cell location of the row with the largest estimated width, which will (approximately) determine the width of the whole circuit block. Then we execute the above row-wise placement algorithm for neighboring rows starting from the fixed one, thereby finishing the whole circuit block placement. In cases that violation of the maximum wirelength constraint WL_{max} exists, a buffer row will be inserted.

The overall AQFP placement algorithm is provided in Algorithm 2. One can observe that the overall placement algorithm has linear complexity with the total number of cell rows, including both logic cell rows and buffer rows.

Algorithm 2: Overall AQFP Placement Algorithm

```
1 Given the path balanced netlist and fixed $R$ clock
    phases (rows);
2 Determine the cell order in each row;
3 Fix the cell locations in row $Max\_row$ with the largest
    estimated width;
4 for row = Max_row-1 to 0, Max_row+1 to R do
5     Solve the row-wise placement using Algorithm 1 ;
6     Update cell locations in $row$;
7     while $WL_{max}$ violation exists do
8         Insert one buffer row;
9         Determine the buffer order;
10        Solve the buffer placement problem using
            Algorithm 1 (with modification discussed in
            Section 3.2);
11        Update buffer locations;
12    end
13 end
```

D. AQFP Routing based on Placement Results

We develop an efficient AQFP routing algorithm based on the derived placement results. Thanks to the unique property that only row-wise routing is needed in AQFP, an integrated routing (instead of separate global and detailed routing) framework

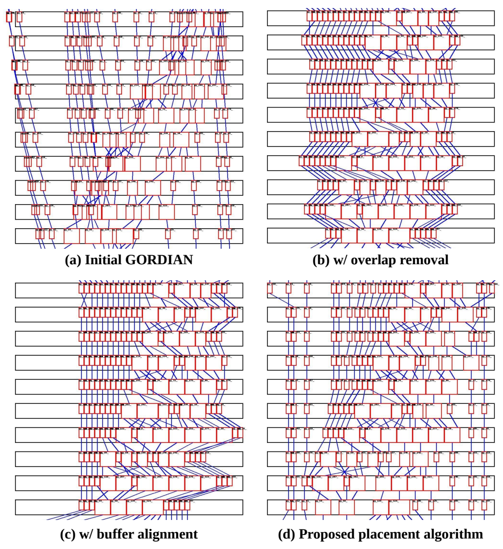

Fig. 4: The example comparison placement diagrams of an 8-bit Kogge-Stone adder (partial circuit).

is sufficient. We adopt the A* routing algorithm [7] as the backbone algorithm for a single net. This algorithm is based on a priority queue and a lower bound estimation of the wirelength and is optimal for a single AQFP net because these nets are 1-to-1. For more compatibility with AQFP, we incorporate dynamic step size to reduce the computational complexity accounting for AQFP characteristics, e.g., the zigzag spacing constraint. The step size becomes the minimum spacing (e.g., 10μm for MIT-LL process) whenever the wire routing takes turns. Furthermore, we incorporate the net negotiation process [8], which allows net overlap in an early phase and then gradually increases the overlapping cost (both the historical and current cost terms) to resolve such overlaps. The net order in the negotiation-based routing process will prioritize those nets with longer estimated wirelength, to better satisfy the maximum wirelength constraint. Such negotiation-based A* routing framework for AQFP can avoid the time-consuming rip-up and re-route process and improve routability, with a fast routing speed as shall be seen in experimental results.

IV. Experimental Results

In this work, we implement our framework using Python and the procedure runs on an AMD Ryzen Threadripper 2920X Processor with 12 high-performance cores and 24 parallel threads. We validate our proposed placement and routing framework using representative benchmark circuits for AQFP testing [5], [19], 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).

A. Placement Results and Comparisons

Because of (i) the lack of previous systematic research on AQFP physical design, and (ii) the difficulty of directly applying CMOS-based physical design framework, we develop our baseline systems. The classic GORDIAN is adopted as our initialization method for cell order determination. Compared with the conventional GORDIAN algorithm, we only place the horizontal location of logic cells, whereas the vertical location is pre-determined by the rows (clock phases). However, the result incurs logic cell overlapping issues and constraints violations for AQFP circuits, which calls for effective cell overlap removal operation or a more detailed placement process. Fig. 4(a) demonstrates the partial result of an 8-bit Kogge-Stone adder. As a result, we develop two enhanced baselines as follows:

1) GORDIAN w/ overlap removal: To remove the cells overlap with minimal wirelength increase, we place the cells in the order of the obtained horizontal values based on the initial GORDIAN results, with an example shown in Fig. 4(b).

2) GORDIAN w/ buffer alignment: It is observed that a considerable number of consecutive buffers are inserted to achieve path balancing. Therefore, we implement an enhanced placement algorithm by performing buffer alignment as shown in Fig. 4(c), resulting in straightly vertical connections. At the downside, consecutive buffer alignment can cause irregular (not in a rectangle shape) placement results.

3) Our proposed placement algorithm: As introduced above, our proposed algorithm starts from GORDIAN initialization and leverages the optimal fixed-order, row-wise placement solution. As shown in Fig. 4(d), our proposed algorithm can achieve a (near-)evenly distributed placement results with a (near-)optimal solution within polynomial time.

The detailed comparison results are shown in TABLE I. Total half-perimeter wirelength (HPWL) and routed wirelength (WL) are measured as our comparison criteria. The former is estimated in the placement phase, while the latter is the actual results derived after routing (negotiation-based A* routing tailored for AQFP circuits, for both proposed placement results and baselines). Compared with baselines, our proposed AQFP placement algorithm can reduce the total HPWL estimation up to 45.9% and the total routed WL up to 40.2% at a fast speed. Additionally, for the first time, we perform a complete placement for the AQFP 32-bit ALU circuit, resulting in a total of 3,536 mm HPWL estimation, finished in 4,238 seconds. We do not show the comparison with baseline systems on this ALU, because the baseline systems cannot satisfy the maximum wirelength constraints for this circuit, thus resulting in invalid placement/routing results.

B. Routing Results and Comparisons

To demonstrate the effectiveness of our proposed physical design framework, we compare the final results between the above two placement baselines using our proposed routing algorithm with our proposed framework. Due to the feature of MIT-LL technology, two metal layers can be utilized for routing, which means there are a certain number of routing tracks on top of the cells. Therefore, we compare the total number of additional tracks outside the cells, which has a direct effect on the total area cost of the AQFP circuits. Detailed results are shown in TABLE II. It shows that our proposed physical design framework can efficiently reduce the number of tracks outside cells with a faster routing speed

TABLE I: Comparison (on total HPWL and routed WL) between baselines and our proposed AQFP placement algorithm.

Circuits	Cells #	Rows #	GORDIAN w/ overlap removal		GORDIAN w/ buffer alignment		**Proposed Placement Algorithm**				
			HPWL	Routed WL	HPWL	Routed WL	HPWL	Reduction	Routed WL	Reduction	Time (s)
apc32	914	23	16,401	25,885	16,819	26,883	12,771	24.0%	22,884	14.5%	18.2
adder8	1,050	21	21,076	33,584	18,634	31,900	10,074	45.9%	21,994	31.1%	66.3
decoder	2,240	20	130,724	167,896	128,920	167,690	112,428	12.8%	148,905	11.2%	109.1
c432	2,538	39	71,280	105,610	75,515	116,267	50,989	32.5%	77,459	33.4%	180.2
sorter32	3,840	30	303,204	378,408	302,236	378,408	203,750	32.6%	226,369	40.2%	564.1
apc128	4,650	36	305,635	367,235	328,548	400,000	219,328	33.2%	267,944	33.0%	1,243

Note: *HPWL and Routed WL results are the summation of all nets measured using μm. *Since sorter32 is a perfectly balanced circuit containing no buffer, the routed WL results of two baselines are the same.

TABLE II: Comparison (on the number of tracks outside cells) between baselines and our proposed Physical Design Framework.

Circuits	Nets #	Rows #	GORDIAN w/ overlap removal		GORDIAN w/ buffer alignment		**Proposed Physical Design Framework**		
			Tracks #	Time (s)	Tracks #	Time (s)	Tracks #	Reduction	Time (s)
apc32	423	23	0	0.54	0	0.56	0	-	0.56
adder8	494	21	0	0.75	0	0.72	0	-	0.39
decoder	979	20	13	172.7	13	174	12	7.7%	141.7
c432	1,157	39	0	3.45	2	17.4	0	100%	1.84
sorter32	1,474	30	31	457.5	31	453.6	14	54.8%	362.7
apc128	2,163	36	28	534.0	45	1,072	3	93.3%	53.78

TABLE III: Comparison results (on the number of tracks outside cells) between baseline (LEA-based routing) and our proposed routing algorithm.

Circuits	Nets #	LEA-based routing		Our proposed routing	
		Tracks #	Time (s)	Tracks #	Time (s)
apc32	423	22	0.18	0	0.56
adder8	494	19	0.17	0	0.39
decoder	979	92	0.86	12	141.7
c432	1,157	64	0.93	0	1.84
sorter32	1,474	119	1.30	14	362.7
apc128	2,163	143	4.58	3	53.78

compared with baselines. When there are no additional tracks outside cells (between rows) and the maximum wirelength constraint can be satisfied, the routing result can be considered to be the best possible.

Additionally, to evaluate our proposed routing algorithm, we adopt *channel routing* as our baseline. Different from the conventional constrained left edge algorithm (LEA), we implement an enhanced constrained left edge algorithm, which can handle the cases with constraint cycles. The comparison results are shown in TABLE III. Our proposed routing can significantly reduce the number of tracks outside the cells, resulting in a considerable area saving.

V. Conclusion

In this work, we develop the first, efficient placement and routing framework for AQFP circuits considering the unique features and constraints. Our proposed fixed-order row-wise placement algorithm is performed iteratively with a fast converge speed. A* routing algorithm is adopted incorporating dynamic step size, achieving reduced computational complexity. The efficiency and effectiveness of our proposed framework are demonstrated through extensive experimental results.

Acknowledgment

The research is based on work supported by the Office of the Director of National Intelligence, Intelligence Advanced Research Projects Activity (IARPA), via the U.S. Army Research Office grant W911NF-17-1-0120, and by the National Science Foundation CCF-2008514 and OISE-1854213.

References

[1] N. Takeuchi, Y. Yamanashi, and N. Yoshikawa, "Energy efficiency of adiabatic superconductor logic," *Superconductor Science and Technology*, 2014.

[2] K. K. Likharev and V. K. Semenov, "Rsfq logic/memory family: A new josephson-junction technology for sub-terahertz-clock-frequency digital systems," *IEEE Transactions on Applied Superconductivity*, 1991.

[3] S. Nagasawa, Y. Hashimoto, H. Numata, and S. Tahara, "A 380 ps, 9.5 mw josephson 4-kbit ram operated at a high bit yield," *IEEE Transactions on Applied Superconductivity*, 1995.

[4] S. K. Tolpygo *et al.*, "Advanced fabrication processes for superconducting very large-scale integrated circuits," *IEEE Transactions on Applied Superconductivity*, 2016.

[5] O. Chen *et al.*, "Adiabatic quantum-flux-parametron: Towards building extremely energy-efficient circuits and systems," *Scientific reports*, 2019.

[6] R. Cai *et al.*, "A majority logic synthesis framework for adiabatic quantum-flux-parametron superconducting circuits," in *GLSVLSI*, 2019.

[7] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE transactions on Systems Science and Cybernetics*, 1968.

[8] L. McMurchie and C. Ebeling, "Pathfinder: a negotiation-based performance-driven router for fpgas," in *Reconfigurable Computing*, 2008.

[9] N. Takeuchi, Y. Yamanashi, and N. Yoshikawa, "Adiabatic quantum-flux-parametron cell library adopting minimalist design," *Journal of Applied Physics*, 2015.

[10] Q. Xu *et al.*, "Synthesis flow for cell-based adiabatic quantum-flux-parametron structural circuit generation with hdl back-end verification," *IEEE Transactions on Applied Superconductivity*, 2017.

[11] J. Clarke and A. Braginski, "The squid handbook: Applications of squids and squid systems. vol. 2," 2006.

[12] A. B. Kahng, I. L. Markov, and S. Reda, "On legalization of row-based placements," in *GLSVLSI*, 2004.

[13] Y. Du and M. D. Wong, "Optimization of standard cell based detailed placement for 16 nm finfet process," in *DATE*, 2014.

[14] T. Taghavi *et al.*, "New placement prediction and mitigation techniques for local routing congestion," in *ICCAD*, 2010.

[15] B. Yu *et al.*, "Methodology for standard cell compliance and detailed placement for triple patterning lithography," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2015.

[16] Y. Lin *et al.*, "Stitch aware detailed placement for multiple e-beam lithography," *Integration*, 2017.

[17] M. Held, P. Wolfe, and H. P. Crowder, "Validation of subgradient optimization," *Mathematical programming*, 1974.

[18] J. M. Kleinhans, G. Sigl, F. M. Johannes, and K. J. Antreich, "Gordian: Vlsi placement by quadratic programming and slicing optimization," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 1991.

[19] Iscas85 combinational benchmark circuits. [Online]. Available: https://filebox.ece.vt.edu/ mhsiao/iscas85.html