# A Global Optimization Algorithm for Buffer and Splitter Insertion in Adiabatic Quantum-Flux-Parametron Circuits

Rongliang Fu
The Chinese University of Hong Kong
rlfu@cse.cuhk.edu.hk

Mengmeng Wang
Yokohama National University
wang-mengmeng-kj@ynu.jp

Yirong Kan
Nara Institute of Science and Technology
kan.yirong@is.naist.jp

Nobuyuki Yoshikawa
Yokohama National University
nyoshi@ynu.ac.jp

Tsung-Yi Ho
The Chinese University of Hong Kong
tyho@cse.cuhk.edu.hk

Olivia Chen
Tokyo City University
olivia.chen@ieee.org

## ABSTRACT

As a highly energy-efficient application of low-temperature superconductivity, the adiabatic quantum-flux-parametron (AQFP) logic circuit has characteristics of extremely low-power consumption, making it an attractive candidate for extremely energy-efficient computing systems. Since logic gates are driven by the alternating current (AC) serving as the clock signal in AQFP circuits, plenty of AQFP buffers are required to ensure that the dataflow is synchronized at all logic levels of the circuit. Meanwhile, since the currently developed AQFP logic gates can only drive a single output, splitters are required by logic gates to drive multiple fan-outs. These gates take up a significant amount of the circuit's area and delay. This paper proposes a global optimization algorithm for buffer and splitter (B/S) insertion to address the issues above. The B/S insertion is first identified as a combinational optimization problem, and a dynamic programming formulation is presented to find the global optimal solution. Due to the limitation of its impractical search space, an integer linear programming formulation is proposed to explore the global optimization of B/S insertion approximately. Experimental results on the ISCAS'85 and simple arithmetic benchmark circuits show the effectiveness of the proposed method, with an average reduction of 8.22% and 7.37% in the number of buffers and splitters inserted compared to the state-of-the-art methods from ICCAD'21 and DAC'22, respectively.

## KEYWORDS

superconducting electronics, AQFP, buffer and splitter insertion

**ACM Reference Format:**
Rongliang Fu, Mengmeng Wang, Yirong Kan, Nobuyuki Yoshikawa, Tsung-Yi Ho, and Olivia Chen. 2023. A Global Optimization Algorithm for Buffer and Splitter Insertion in Adiabatic Quantum-Flux-Parametron Circuits. In *28th Asia and South Pacific Design Automation Conference (ASPDAC '23), January 16–19, 2023, Tokyo, Japan.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3566097.3567936

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
*ASPDAC '23, January 16–19, 2023, Tokyo, Japan*
© 2023 Association for Computing Machinery.
ACM ISBN 978-1-4503-9783-4/23/01…$15.00
https://doi.org/10.1145/3566097.3567936

**Table 1: The comparison between CMOS and AQFP.**

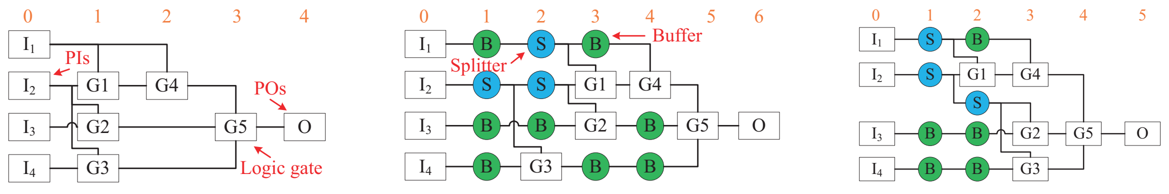
| Circuit | AQFP | CMOS |
|---|---|---|
| **Active component** | Josephson junction (JJ) | Transistor |
| **Passive component** | Inductor | Capacitor |
| **Information** | Current pulse | Voltage level |
| **Clocking Scheme** | Synchronize (Clock signal) | Asynchronous |
| **Fan-out** | 1 (Splitter for multiple fan-outs) | $\geq 1$ |
| **Power** | AC | DC |

## 1 INTRODUCTION

Computation speed and power consumption have become the ultimate challenge of semiconductor integrated circuits. As an application of low-temperature superconductivity, the superconducting logic circuit has high speed and low power consumption characteristics, making it an attractive alternative to CMOS for future computing systems. Josephson junction (JJ), the superconducting switching device, switches rapidly ($\sim$ 1 ps) with extremely low energy per switch ($< 10^{-19}$ J) and communicates information by voltage pulses that propagate over superconducting transmission lines almost without loss. As a member of the superconducting logic families, adiabatic quantum-flux-parametron (AQFP) logic has attracted much attention recently due to its extremely energy-efficient technology, where adiabatic switching operations can significantly reduce energy dissipation. In AQFP logic, alternating current (AC) serves as both clock signals and power supplies, lightening power consumption overhead of direct current (DC) bias while operating at gigahertz-level frequencies. It has been demonstrated that the switching energy (energy dissipation per switching event) of an AQFP gate ranges from $10^{-20}$ J to $10^{-21}$ J at 5GHz operation [10].

Although AQFP logic has these superior advantages, the underlying differences with conventional CMOS logic, as shown in Table 1, make electronic design automation (EDA) tools of CMOS logic not applicable to AQFP logic. Firstly, each AQFP logic gate requires a clock signal to release its signal to its successors and reset its state. Thus, numerous buffers must be inserted to ensure all inputs to each logic gate have the same delay (clock phases), making the circuit work correctly. In other words, all the input data paths of a logic gate must have the same number of logic gates, namely the logic level. Secondly, due to output current limitations, an AQFP logic gate can only drive one logic gate. To resolve this issue, a specially designed gate, namely a splitter, must be inserted to achieve multiple fan-outs. Therefore, buffer and splitter insertion is critical to satisfying the above two requirements of AQFP circuits.

With the rapid development of physical-level research and the growth of fabrication capabilities for larger and more complex



(a) The input circuit. (b) The result of [5]. (c) The optimal result.

**Figure 1: An example of B/S insertion where the maximum fan-out of the splitter is 2. For the B/S insertion of the input circuit (a), where orange digitals represent the logic level of gates, (b) has greater circuit depth and inserts three more buffers than (c).**

circuits, the research on the optimization of buffer and splitter insertion becomes vital for AQFP logic. Although several related works [1, 2, 5–7] have been proposed to optimize the buffer and splitter (B/S) insertion for AQFP circuits, inserted buffers and splitters still account for over than half of the JJ number, with the latest results [7] averaging 52.06%. In [1] and [2], splitters and buffers are inserted separately after conventional logic synthesis, then timing-like heuristic algorithms are presented to optimize the number of buffers and splitters. In [6], a heuristic method is applied to implement irredundant buffer and splitter insertion by scheduling and moving groups of gates, called chunks, together. Although this method can reduce the number of buffers and splitters, the process of chunked movement may be endless due to alternating up or down moves. Besides, [5] focuses on the buffer and splitter insertion problem on a single net and presents a dynamic programming-based algorithm that can provide an optimal buffer and splitter insertion solution for each net of the input circuit. However, the interaction among nets can affect the number of buffers and splitters inserted. As shown in Fig. 1, there are two B/S insertion solutions for the same input circuit. [5] heuristically determines the logic level of each logic gate, which may lead to a suboptimal result, that is, three more buffers are inserted in Fig. 1(b) than the optimal result in Fig. 1(c). Besides, [7] identifies B/S insertion as a scheduling problem and proposes an SMT formulation to find the global optimum, which is impractical for large AQFP circuits. It also provides an efficient heuristic for B/S insertion, which reduces the number of inserted B/S by only near 4% compared to [5], although it is faster.

This paper focuses on how to further reduce the cost caused by B/S insertion after logic optimization. Since the splitter requires a clock signal, it can also be used to increase the delay of the data path so that the structure of the splitter tree has a significant impact on the number of buffers and splitters inserted. This paper identifies the B/S insertion as a combinational optimization problem and describes a dynamic programming formulation to find the global optimum, which is impractical due to the limitation of its enormous search space. To practically find a high-quality solution for B/S insertion, the problem is approximately identified as an integer linear programming formulation. The proposed algorithm divides the B/S insertion process into three steps: (i) logic level optimization aiming at globally minimizing the cost of B/S insertion, (ii) splitter tree generation based on the optimal multi-way search tree, and (iii) buffer insertion. The proposed algorithm shows effective results on the ISCAS'85 and arithmetic benchmark circuits, with an average reduction of 8.22% and 7.37% in the number of buffers and splitters inserted compared to the methods in ICCAD'21 [5] and DAC'22 [7], respectively. Furthermore, for circuits with a maximum fan-out number over 30, the proposed method can reduce the B/S number by an average of 10.43% and 12.79% than these two methods, especially by 15.83% and 22.72% for c7552. Moreover, all generated circuits generated by the proposed method have less circuit depth.

The rest of the paper is organized as follows. Section 2 introduces AQFP logic and defines some terminologies. Section 3 presents our proposed method and describes the logic level optimization and the splitter tree generation in detail. Section 4 shows experimental results by comparing the proposed algorithm to the state-of-the-art methods. The paper is summarized in Section 5.

## 2 PRELIMINARIES

### 2.1 AQFP Logic

Adiabatic quantum-flux-parametron logic is a digital logic implementation technology based on superconducting Josephson junctions. The power consumption of AQFP circuits is relatively low compared to that of other superconductor logic families. An AQFP circuit usually consists of buffers, splitters, read-out interfaces, and other logic gates. As the basic component in AQFP logic, the buffer consists of two superconductor-inductor loops and can construct other logic gates. When the AC is supplied for an AQFP buffer, a single flux quantum is stored in either its left or right loop, and the direction of its output current can represent its logic state. The splitter consists of a buffer and a 1-to-$n$ branch circuit (usually, $2 \leq n \leq 4$) and is clocked. In an AQFP circuit, if the output signal of one logic gate needs to be transmitted to multiple logic gates, splitters must be inserted into its output to realize multiple fan-outs.

In AQFP logic, the multi-phase clocking scheme is usually applied, such as the 4-phase clocking scheme [9]. Both Fig.2(a) and Fig.2(b) use the 4-phase clocking scheme, where $I_{x1}$ and $I_{x2}$ are AC-based clock signals with a phase separation of 90°, and $I_d$ is a DC input, which applies an offset flux of half single flux quantum to each gate. Fig. 2 shows the necessity of buffer insertion for correct operation in an AQFP circuit with the function $q = a\&b\&c$. Compared to Fig. 2(a), all inputs to each logic gate in Fig. 2(b) are in the same clock phase by the buffer insertion, i.e., they have an equal delay. According to the timing schematic in Fig. 2(c), when the data input $a$ arrives at the logic gate $g2$, the signal of $w1$ does not arrive, making that the logic gate $g2$ has no output when the clock signal $I_{x2}$ arrives. At the next clock phase, the signal of $w1$ arrives at $g2$, then $g2$ has the output 0 after $I_{x2}$ arrives. By contrast, the circuit in Fig. 2(b) can correctly present the function $q = a\&b\&c$

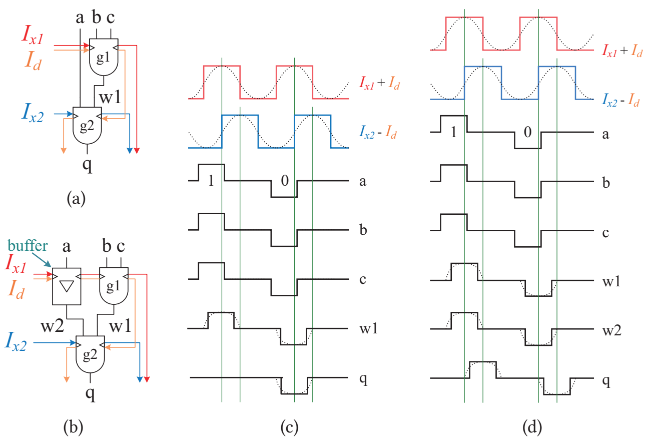


**Figure 2: An example shows the necessity of buffer insertion for correct operation in an AQFP with the function $q = a\&b\&c$, where a 4-phase clocking scheme [9] is applied. (a) and (b) are gate-level schematics before and after buffer insertion, respectively, where $I_{x1}$ and $I_{x2}$ are AC-based clock signals with a phase separation of 90°, and $I_d$ is the DC input, which applies an offset flux to each logic gate. Besides, $a$, $b$, and $c$ are three data inputs, and $q$ is a data output; (c) and (d) are timing schematics of (a) and (b), respectively.**

due to all inputs to each gate with the same logic level by the buffer insertion, whose timing schematic is shown in Fig. 2(d).

Meanwhile, all logic gates of an AQFP circuit are divided into multiple levels as the arrival order of input data. All primary inputs (PIs) are usually aligned at the first level, and all primary outputs (POs) are usually aligned at the last level. Therefore, the logic level of all PIs is set to 0, and the logic level of all POs is the maximum logic level, equal to the circuit depth. To reduce the circuit depth, the logic level of POs should be as small as possible.

## 2.2 Terminology

An AQFP circuit can be represented by a network $N(V, E)$, where $V$ is the set of logic gates, and $E$ is the set of nets. The node set $V = I \cup O \cup G$ consists of the set $I$ of PIs, the set $O$ of POs, and the set $G$ of logic gates. For a node $v \in V$, $E_i(v)$ is the set of its input edges, and $E_o(v)$ is the set of its output edges. The edge set $E = E_u \cup E_m$ consists of the set $E_u$ of 2-pin nets and the set $E_m$ of over-2-pin nets. For a edge $e \in E$, $e_s$ is the source of the edge $e$, and $e_t$ is the set of sinks of the edge $e$. If the edge $e$ is a 2-pin net, i.e., $e \in E_u$, then $|e_t| = 1$, otherwise $|e_t| > 1, e \in E_m$. After buffer and splitter insertion, an extended network $N'(V', E')$ can be obtained, and $V' = V \cup B \cup S$, where $B$ and $S$ represent the set of buffers and the set of splitters, respectively. The maximum fan-out of the splitter is denoted as $X$. For a node $v \in V$, $\mathrm{FI}(v)$, $\mathrm{FO}(v)$, and $\mathrm{L}(v)$ represent the set of its fan-in nodes, the set of its fan-out nodes, and its logic level, respectively.

## 2.3 Problem Formulation

To meet the delay and fan-out requirements, buffers and splitters must be inserted into the AQFP circuit, dramatically impacting the circuit's area and delay. This paper focuses on the buffer and splitter insertion problem for AQFP gate-level circuits after logic synthesis, formulated as follows:

- Input:
  (1) A given circuit network $N(V, E)$.
  (2) The maximum fan-out $X$ of the splitter.
- Output:
  An extended network $N'(V', E')$ with an equal delay for all inputs to any gate and a single fan-out for all gate outputs.
- Constraints:
  (1) Fan-out constraint: $\forall e \in E', |e_t| = 1$.
  (2) Delay constraint: $\forall a, b \in \mathrm{FI}(v), \mathrm{L}(a) = \mathrm{L}(b)$.
  (3) PI alignment: $\forall i \in I, \mathrm{L}(i) = 0$.
  (4) PO alignment: $\forall o \in O, \mathrm{L}(o) = \max_{v \in V'} L(o)$.
- Goal:
  Minimize the number of buffers and splitters inserted without changing the functional structure of the original network $N$, formulated as follows:

$$\min_{S,B \in G'} |S \cup B| \tag{1}$$

## 3 A GLOBAL OPTIMIZATION ALGORITHM FOR BUFFER AND SPLITTER INSERTION

For buffer insertion, the number of inserted buffers depends on the clock phase difference of all inputs to the logic gate, which can be calculated as the logic level difference between the source and the sink of the net. Splitter insertion is actually to solve the multi-fan-out of the net, and the number of inserted splitters depends on the fan-out number of the net. Therefore, the objective function in Equation 1 can be reformulated as:

$$\min_{e \in E} SplitterTreeCost(e) \tag{2}$$

where the function *SplitterTreeCost* can calculate the number of buffers and splitters inserted in the net $e$ in terms of the logic level of the source and sinks of the net $e$. Buffer and splitter insertion for an over-2-pin net can be converted into an optimal multi-way search tree generation problem, as specifically described in Section 3.2. After assigning logic levels to all nodes, the number of buffers and splitters inserted can be obtained. Since the upper boundary $ub$ and the lower boundary $lb$ of the logic level of each node can be obtained by topological sorting, an optimal logic level assignment can be found to obtain the global optimal solution of buffer and splitter insertion. Therefore, the buffer and splitter insertion can be regarded as a combinational optimization problem, formulated as:

$$\min_{\substack{L \\ v \in V, lb(v) \le L(v) \le ub(v) \\ \forall u \in FI(v), L(v) - L(u) \ge 1}} SplitterTreesCost(L) \tag{3}$$

where the function *SplitterTreesCost* can calculate the number of buffers and splitters inserted in the circuit in terms of the set $L$.

Now the key to the buffer and splitter insertion problem is how to determine the logic level of each node. Actually, it can be converted to the shortest path problem. Firstly, all nodes are divided into $m$ groups $G$ by the topological sorting from PIs to POs, where the distance $D(\cdot)$ between two groups is the number of buffers and splitters inserted between nodes from these two groups. So, for $k^{th}$

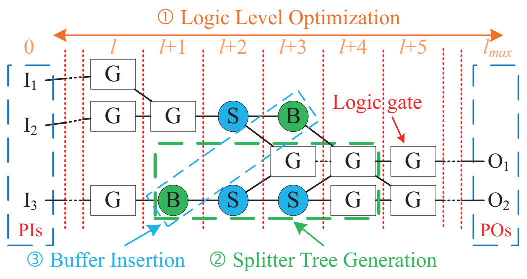


**Figure 3: Flow of the proposed method contained logic level optimization, splitter tree generation, and buffer insertion.**

group, its minimum cost $BS\,(L(G_k))$ under a logic level assignment $L\,(G_k)$ can be calculated as:

$$BS\,(L(G_k)) = min \begin{cases} BS\,(L(G_k))\,, \\ BS\,(L(G_{k-1})) + D\,(L(G_k), L(G_{k-1})) \end{cases} \tag{4}$$

However, it is impractical due to the very large search space whose size is $\sum_{k=1}^{|G|} \prod_{v\in G_k} \{ub(v) - lb(v) + 1\}$. Hence, to reduce the complexity and shrink the search space, a global optimization algorithm for B/S insertion is proposed, as shown in Fig. 3. Firstly, the complete multi-way tree is used to approximately evaluate the cost of an over-2-pin net's B/S insertion, which makes the problem converted into an integer linear programming problem to further determine the logic level of each gate. Then, optimal splitter trees are generated for each over-2-pin net via the optimal multi-way search tree generation method. Finally, buffers are inserted for all 2-pin nets.

## 3.1 Logic Level Optimization

The logic level of each gate can be calculated by minimizing the number of all buffers and splitters required by the circuit. The delay between two connected gates can be calculated from their logic level difference. So, the total delay of a net $e$ can be calculated as $p(e) = \sum_{t\in e_t} (\mathrm{L}(t) - \mathrm{L}(e_s) - 1)$. For a 2-pin net, the number of buffers inserted is equal to its delay, while for an over-2-pin net, the number of buffers inserted is less than its delay due to the delay of the splitter. To evaluate the number of buffers and splitters required by an over-2-pin net $e$, the complete multi-way tree is applied in the logic level optimization. Firstly, according to the sink number $|e_t|$ of the net $e$, a complete multi-way tree with $|e_t|$ leaves is built, whose all path sum $f(|e_t|)$ can be calculated. Then the remained delay can be fulfilled by a buffer chain. Therefore, the number of buffers and splitters required by the circuit can be formulated as:

$$\min \quad \sum_{e\in E_u} p(e) + \sum_{e\in E_m} \left( \frac{p(e) - f(|e_t|)}{|e_t|} + \beta(|e_t|) \right) \tag{5}$$

$$\text{s.t.} \quad \forall v \in I, \mathrm{L}(v) = 0 \tag{6}$$

$$\forall v, u \in O, \mathrm{L}(v) = \mathrm{L}(u) \tag{7}$$

$$\forall v \in O \cup G, \forall u \in \mathrm{FI}(v), \mathrm{L}(v) \geq \mathrm{L}(u) + 1 \tag{8}$$

$$\forall e \in E_m, \sum_{\substack{t\in C \\ \forall C\subseteq e_t, C\neq\emptyset}} (\mathrm{L}(t) - \mathrm{L}(e_s) - 1) \geq f\,(|C|) \tag{9}$$

where $f(x)$ and $\beta(x)$ are all path sum and the number of non-leaf nodes of the complete $X$-way tree with $x$ leaves, respectively.

Since $f(|e_t|)$ and $\beta(|e_t|)$ are constants for an edge $e$, the problem of minimizing the number of splitters and buffers can be abstracted as an integer linear programming problem as follows:

$$\min \quad \sum_{v\in V} \left( \sum_{e\in E_i(v)} \frac{1}{|e_t|} - \sum_{e\in E_o(v)} \frac{1}{|e_t|} \right) * \mathrm{L}(v) \tag{10}$$

$$\text{s.t.} \quad Equations\ 6 - 9 \tag{11}$$

Equation 6 and Equation 7 align PIs and POs so that all PIs and all POs are on the first and last levels, respectively. To limit the fan-out of the node, as shown in Equation 9, all path sum of the multi-way tree constructed from any subset $C$ of its sinks $e_t$ for any edge $e$ must be greater than or equal to that of the complete multi-way tree with $|C|$ leaves, that is because the complete multi-way tree has the smallest all path sum and can meet the fan-out requirement. In Equation 9, all subsets of sinks of each edge need to be enumerated. To list all subsets of sinks $e_t = \{s_1, s_2, ..., s_{|e_t|}\}$ of the edge $e$, a characteristic function $\chi_A(x)$ is used to identify each subset $A$ of $e_t$.

$$\chi_A(x) = \begin{cases} 1 & \text{if } x \in A \\ 0 & \text{if } x \notin A \end{cases} \tag{12}$$

$\chi_A(x)$ can be represented by a 0-1 sequence. For example, for $e_t = \{s_0, s_1, s_2, s_3, s_4\}$,

| | |
|---|---|
| $\{s_4\}$ | 00001 |
| $\{s_2, s_4\}$ | 00101 |
| $\{s_0, s_1, s_3\}$ | 11010 |
| $\{s_0, s_1, s_2, s_3, s_4\}$ | 11111 |

So all combinations of sinks $e_t$, i.e., all subsets of sinks $e_t$, can be generated by full permutations of the 0-1 sequence of length $|E|$. Then the sinks corresponding to the index of the '1' element in each permutation form a subset. However, there are $2^{|e_t|}$ subsets for the edge $e$, which makes enumerating all subsets infeasible for the edge with a large number of sinks. So, for the edge with over 30 sinks, $n$ sequences are first generated by the random shuffle algorithm. Then, $(|e_t| - 1) * (|e_t| - 2)/2$ subsets of the length from 3 to $|e_t|$ are selected starting from the left side. In this way, there are $n * (|e_t| - 1) * (|e_t| - 2)/2$ subsets selected to constrain the maximum number of subsets of nodes in the multi-way tree. Finally, due to numerous decision variables and constraints, making the range of decision variables not easy to judge, an integer linear programming solver using a linear programming-based branch-and-bound algorithm is used to solve the integer linear programming problem in Equation 10 to obtain the logic level of each gate.

## 3.2 Splitter Tree Generation

After obtaining the logic level of each gate, the delay between any two connected gates can be determined. To satisfy the delay and fan-out requirements, buffers and splitters must be inserted. For a 2-pin net, $d$ buffers can directly be inserted according to its delay $d$. For an over-2-pin net, a splitter tree composed of buffers and splitters needs to be generated, minimizing the number of buffers and splitters inserted.

To solve the splitter tree generation problem of the $(n + 1)$-pin net $e$, several terms need to be defined.

- The nodes of the splitter tree: The leaf nodes of the splitter tree is the sinks $e_t$ of the net $e$, where $n = |e_t|, n \geq 1$. The

**Algorithm 1:** Optimal splitter tree generation algorithm

```
Input: An over-2-pin net $e$ with a source $s$ and $n$ sinks $\boldsymbol{t}$,
       and the maximum fan-out $X$ of the splitter
Output: A generated split tree
Calculate the delay: $\boldsymbol{t}_i.delay = \mathrm{L}(\boldsymbol{t}_i) - \mathrm{L}(s) - 1, i \in [1, n]$
Reorder sinks $\boldsymbol{t}$ in ascending order in terms of their delays
$D = \max_{i \in [1,n]} t_i.delay + \lceil \log_X n \rceil + 1$
$\mathbf{dp}[n][n][X][D+1] = \{+\infty, +\infty, +\infty\}$
$\mathbf{pt}[n][n][X][D+1] = \{-1, -1\}$
// Initialization
for $s \in [1, \min\{n, X\}]; l \in [1, n-s+1]$ do
    $r = l + s$
    if $s == 1$ then
        for $d = 0$ to $D$ do
            $\Delta = |d - \boldsymbol{t}_l.delay|$
            $\mathbf{dp}_{l,l,1,d} = d > \boldsymbol{t}_l.delay? \{\Delta, \Delta, 0\} : \{0, 0, \Delta\}$
    else
        $\mathbf{dp}_{l,r,s,D} = \mathbf{dp}_{l,r-1,s-1,D} + \mathbf{dp}_{r,r,1,D}$
// Calculate the minimum cost
for $len \in [2, n]; l \in [1, n-len+1]$ do
    $r = l + len$
    for $d = D-1$ to $0$; $s \in [1, \min\{len, X\}]$ do
        if $s == 1$ then
            $\mathbf{dp}_{l,r,s,d} = \min_{k \in [1, \min(len, X)]} \mathbf{dp}_{l,r,k,d+1} + \{0, 0, 1\}$
            $\mathbf{pt}_{l,r,s,d} = \{-1, k\}$
        else
            $\mathbf{dp}_{l,r,s,d} = \min_{k \in [l+u, r-v], s=u+v} \mathbf{dp}_{l,k,u,d} + \mathbf{dp}_{k+1,r,v,d}$
            $\mathbf{pt}_{l,r,s,d} = \{k, u\}$
return a splitter tree built by backtracking method using pt
```

out-degree of each non-leaf node must be in the range $[1, X]$. The non-leaf node is a buffer if its out-degree equals one, and a splitter otherwise. Besides, the depth of each node $v$ in the splitter tree is denoted as $d(v)$, and the delay of the leaf node $v$ is denoted as $v.delay$.

- The cost of generated splitter tree: For a generated splitter tree, its cost [5] consists of three parts, including maximum extra delay $ed = \max_{t \in e_t}\{d(t) - t.delay\}$ of leaf nodes, total extra delay $ted = \sum_{t \in e_t} \{d(t) - t.delay\}$ of leaf nodes, and the number $nn$ of non-leaf nodes of the generated tree. Besides, the addition of the two costs $\{ed_1, ted_1, nn_1\}, \{ed_2, ted_2, nn_2\}$ is defined as $\{\max\{ed_1, ed_2\}, ted_1 + ted_2, nn_1 + nn_2\}$.

Referring to the optimal multi-way search tree [4], an optimal splitter tree generation algorithm based on dynamic programming is proposed, as shown in Algorithm 1. Firstly, the delay of each leaf node is calculated (line 1). Then, leaf nodes are reordered in ascending order in terms of their delays, and the maximum depth is calculated. Since the complete X-way tree with $|e_t|$ leaf nodes has the minimum depth $d_{min} = \lceil \log_X |e_t| \rceil + 1$, there is a splitter tree with $|e_t|$ leaf nodes, where the extra delay of its leaf node with the maximum delay is $d_{min}$, meaning that the $ed$ of the optimal splitter tree is not greater than $d_{min}$. $\mathbf{dp}_{l,r,s,d}$ records the cost of $s$ splitter trees with leaf nodes $e_l, e_{l+1}, ..., e_r$, whose root nodes' depth is $d$. $\mathbf{pt}_{l,r,s,d} = \{m, s'\}$ records two splitters: one with $s'$ fan-outs and leaf nodes $e_l, e_{l+1}, ..., e_m$, and another one with $s - s'$ fan-outs and leaf nodes $e_{m+1}, e_{m+2}, ..., e_r$. Lines 4-14 initialize **dp** and **pt**. When the fan-out of the root node of the substructure $\{l, r, s, d\}$ with leaf nodes $e_l, e_{l+1}, ..., e_r$ is $s = 1$ in the depth, a buffer is inserted, otherwise it is divided into two parts, and a splitter is inserted (line 19-24). After obtaining the cost $\{1, n, 1, 0\}$ of the root node, the optimal splitter tree can be constructed by the backtracking algorithm in terms of **pt**. The time complexity and the space complexity of Algorithm 1 are $O(Xn^3 \lceil \log_X n \rceil)$, $O(Xn^2 \lceil \log_X n \rceil)$, respectively.

## 4 EXPERIMENTAL RESULTS

The proposed global optimization algorithm for buffer and splitter insertion is implemented in C++-17 language. The experiments were executed on Intel(R) Xeon(R) W-2223 3.60GHz CPU machine with 32 GB memory. The Gurobi [8] is selected as the integer linear programming solver due to its significant performance. The intrinsic logic gate in AQFP logic is the 3-input majority (MAJ) gate. A 2-input AND gate can be obtained from a 3-input MAJ gate, one of which data input is a constant '0'. A 2-input OR gate can be obtained from a 3-input MAJ gate, one of which data input is a constant '1'. In current AQFP circuits, buffers, splitters, and inverters all have 2 JJs, 3-input MAJ, 2-input AND, and 2-input OR all have 6 JJs. The maximum fan-out of the splitter is 4.

The performance of the proposed algorithm was evaluated with ISCAS'85 and simple arithmetic benchmark circuits without any splitters and buffers obtained from [3]. To obtain the experimental results more accurately, several modifications have been made to the benchmark circuits. Since the inverter can be integrated into the input of its connected gate, inverters connected to POs are transmitted to the input ports of the predecessors of inverters, such as $\overline{<x, y, z>} = <\overline{x}, \overline{y}, \overline{z}>$ for a MAJ gate, $\overline{x \& y} = \overline{x} | \overline{y}$ for an AND gate, and $\overline{x | y} = \overline{x} \& \overline{y}$ for an OR gate. If the inverter is directly connected to both the PI and the PO, it can be reserved. In this way, the number of inverters (INV), other logic gates, JJs, and maximum fan-out (MaxFO) are counted in Table 2.

The proposed method was compared with two methods proposed in ICCAD'21 [5] and DAC'22 [7], respectively, in terms of the number (BS) of inserted B/S, the circuit's JJ number (JJs) and the circuit depth (Depth), as shown in Table 2. Since the JJ amount directly determines the area and power consumption of the AQFP circuit, it is an important metric in the AQFP circuit. The circuit depth denotes the maximum logic level of POs in the circuits. Besides, to ensure consistency in the delay and fan-out processing for PIs and POs of the circuit by each experimental method, PIs and POs also require buffer and splitter insertion so that all PIs have a logic level of 0 and all POs have the same logic level.

Compared with ICCAD'21, the proposed method has an average of 8.22% and 4.54% reduction in the number of inserted B/S and the JJ number, respectively. Due to the incomplete data of the global optimum method from DAC'22, the proposed method is compared

**Table 2: The experimental result comparing the proposed method with the state-of-the-art methods**

| Testcase | Original Circuit | | | | | ICCAD'21 | | | DAC'22 | | | Ours | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | INV | Gates | JJs | Depth | MaxFO | BS | JJs | Depth | BS | JJs | Depth | BS | JJs | Depth |
| alu32 | 0 | 1513 | 9078 | 100 | 128 | 15087 | 39252 | 171 | 15040 | 39158 | 173 | 13976 | 37030 | 169 |
| mult8 | 0 | 439 | 2634 | 35 | 9 | 1836 | 6306 | 70 | 1869 | 6372 | 71 | 1681 | 5996 | 70 |
| counter16 | 0 | 29 | 174 | 9 | 4 | 82 | 338 | 17 | 65 | 304 | 17 | 66 | 306 | 17 |
| counter32 | 0 | 82 | 492 | 13 | 4 | 178 | 848 | 23 | 155 | 802 | 23 | 156 | 804 | 23 |
| counter64 | 0 | 195 | 1170 | 17 | 4 | 382 | 1934 | 30 | 352 | 1874 | 30 | 351 | 1872 | 30 |
| counter128 | 0 | 428 | 2568 | 22 | 4 | 798 | 4164 | 38 | 760 | 4088 | 38 | 755 | 4078 | 38 |
| c432 | 0 | 121 | 726 | 26 | 10 | 852 | 2430 | 37 | 874 | 2474 | 38 | 829 | 2384 | 37 |
| c499 | 0 | 387 | 2322 | 18 | 8 | 1210 | 4742 | 29 | 1275 | 4872 | 31 | 1173 | 4668 | 29 |
| c880 | 0 | 306 | 1836 | 27 | 9 | 1661 | 5158 | 40 | 1703 | 5242 | 41 | 1536 | 4908 | 40 |
| c1355 | 0 | 389 | 2334 | 18 | 9 | 1203 | 4740 | 29 | 1290 | 4914 | 31 | 1186 | 4706 | 29 |
| c1908 | 0 | 289 | 1734 | 21 | 14 | 1332 | 4398 | 34 | 1298 | 4330 | 35 | 1253 | 4240 | 34 |
| c2670 | 1 | 369 | 2216 | 21 | 32 | 1988 | 6192 | 28 | 2131 | 6478 | 30 | 1869 | 5954 | 28 |
| c3540 | 0 | 794 | 4764 | 32 | 38 | 2303 | 9370 | 53 | 2266 | 9296 | 55 | 1963 | 8690 | 52 |
| c5315 | 15 | 1317 | 7932 | 26 | 41 | 5997 | 19926 | 40 | 6014 | 19960 | 42 | 5505 | 18942 | 40 |
| c6288 | 0 | 1870 | 11220 | 89 | 17 | 9297 | 29814 | 179 | 9893 | 31006 | 180 | 8832 | 28884 | 179 |
| c7552 | 1 | 1395 | 8372 | 33 | 170 | 8041 | 24454 | 58 | 8758 | 25888 | 66 | 6768 | 21908 | 58 |
| Average | | | | | | 1.0928 | 1.0484 | 1.0019 | 1.0844 | 1.0498 | 1.0362 | 1 | 1 | 1 |

with the DAC'22 heuristic method. Compared with the results of DAC'22, the proposed method can obtain 7.37% fewer buffers and splitters and 4.58% fewer JJs on average. In addition, it is seen that the proposed algorithm has better results than the state-of-the-art methods on the circuits with large fan-out. For instance, for circuits with a maximum fan-out number over 30, the proposed method has 10.43% and 12.79% fewer B/S on average than the state-of-the-art methods. In the c7552 circuit with a maximum fan-out of 170, the proposed algorithm has a significant improvement over other methods, up to 15.83% and 22.72%. In addition, the depth of each generated circuit is less than or equal to that of the ICCAD'21 method and the DAC'22 method.

## 5 CONCLUSION

This paper introduces the buffer and splitter insertion method to make the circuit satisfy the delay requirement and fan-out limitation of AQFP logic, which is a vital step in the design flow of AQFP circuits. For the buffer and splitter insertion problem, the interaction among nets makes it challenging to find an optimal solution. To minimize the number of buffers and splitters inserted, a global optimization algorithm for buffer and splitter insertion is proposed. Firstly, an integer linear programming model is built to obtain the logic level of each gate so that the delay between connected gates can be determined. Then, the sinks of each net are sorted in non-decreasing order according to the delay of its sinks. Finally, the optimal splitter tree can be generated for each net by the optimal multi-way search tree method based on dynamic programming. On the ISCAS'85 and simple arithmetic benchmark circuits, the proposed method achieves an 8.22% and 7.37% reduction in the inserted B/S number compared with the methods in ICCAD'21 and DAC'22, respectively. Furthermore, the proposed method can reduce the inserted B/S number by 10.43% and 12.79% on average for circuits with a maximum fan-out number over 30. Experimental results also show our improvements in the circuit depth.

## ACKNOWLEDGMENTS

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

## REFERENCES

[1] Christopher L Ayala, Ro Saito, Tomoyuki Tanaka, Olivia Chen, Naoki Takeuchi, Yuxing He, and Nobuyuki Yoshikawa. 2020. A semi-custom design methodology and environment for implementing superconductor adiabatic quantum-flux-parametron microprocessors. *Superconductor Science and Technology* 33, 5 (Mar. 2020), 054006. https://doi.org/10.1088/1361-6668/ab7ec3

[2] Ruizhe Cai, Olivia Chen, Ao Ren, Ning Liu, Nobuyuki Yoshikawa, and Yanzhi Wang. 2019. A Buffer and Splitter Insertion Framework for Adiabatic Quantum-Flux-Parametron Superconducting Circuits. In *2019 IEEE 37th International Conference on Computer Design (ICCD)*. 429–436. https://doi.org/10.1109/ICCD46524.2019.00067

[3] EPFL. 2021. *ISCAS'85 and simple arithmetic benchmarks*. https://github.com/lsils/SCE-benchmarks/tree/main/ISCAS

[4] L. Gotlieb. 1981. Optimal Multi-Way Search Trees. *SIAM J. Comput.* 10, 3 (1981), 422–433. https://doi.org/10.1137/0210031

[5] Chao-Yuan Huang, Yi-Chen Chang, Ming-Jer Tsai, and Tsung-Yi Ho. 2021. An Optimal Algorithm for Splitter and Buffer Insertion in Adiabatic Quantum-Flux-Parametron Circuits. In *2021 IEEE/ACM International Conference On Computer Aided Design (ICCAD)*. 1–8. https://doi.org/10.1109/ICCAD51958.2021.9643456

[6] Siang-Yun Lee, Heinz Riener, and Giovanni De Micheli. 2021. Irredundant Buffer and Splitter Insertion and Scheduling-Based Optimization for AQFP Circuits. https://doi.org/10.48550/ARXIV.2109.00291

[7] Siang-Yun Lee, Heinz Riener, and Giovanni De Micheli. 2022. Beyond Local Optimality of Buffer and Splitter Insertion for AQFP Circuits. In *Proceedings of the 59th ACM/IEEE Design Automation Conference (DAC '22)*. 445–450. https://doi.org/10.1145/3489517.3530661

[8] Gurobi Optimization LLC. 2022. *Gurobi - The Fastest Solver - Gurobi*. https://www.gurobi.com/

[9] Naoki Takeuchi, Shuichi Nagasawa, Fumihiro China, Takumi Ando, Mutsuo Hidaka, Yuki Yamanashi, and Nobuyuki Yoshikawa. 2017. Adiabatic quantum-flux-parametron cell library designed using a 10 kA cm$^{-2}$ niobium fabrication process. *Superconductor Science and Technology* 30, 3 (Jan. 2017), 035002. https://doi.org/10.1088/1361-6668/aa52f3

[10] Naoki Takeuchi, Taiki Yamae, Christopher L. Ayala, Hideo Suzuki, and Nobuyuki Yoshikawa. 2019. An adiabatic superconductor 8-bit adder with 24kBT energy dissipation per junction. *Applied Physics Letters* 114, 4 (2019), 042602. https://doi.org/10.1063/1.5080753 arXiv:https://doi.org/10.1063/1.5080753