


# Partitioning RSFQ Circuits for Current Recycling

Gleb Krylov, *Student Member, IEEE*, and Eby G. Friedman , *Fellow, IEEE*

**Abstract**—RSFQ circuits require a DC bias current to operate properly. The bias current in conventional RSFQ circuits is supplied to each gate, resulting in large current requirements in VLSI complexity SFQ systems, on the order of tens to hundreds of amperes. These high currents are difficult to supply and distribute. Superconductive input/output pins and bias lines support this limited current. Large currents however require significant metal and input pin resources. In addition, large currents can inductively couple to sensitive superconductive inductors, degrading circuit operation and producing errors. Current recycling is a well known technique to reduce these bias currents. RSFQ circuits with similar bias current requirements can be placed on separate ground planes and serially biased. The inputs and outputs of these circuits are galvanically decoupled and require drivers and receivers between connections. In this paper, a methodology for automated partitioning of complex RSFQ circuits into blocks with similar bias currents is described, where the number of connections among the blocks is minimized. Blocks with a significant difference in bias current are balanced using dummy padding structures. These blocks are biased in series, reducing the total bias current by the number of partitions. The Fiduccia-Mattheyses algorithm is modified with RSFQ specific enhancements to partition the system. A geometric partitioning approach, optimized by simulated annealing, is also proposed. These algorithms are integrated into the circuit placement process, and the methodology is evaluated using several modified ISCAS'89 sequential benchmark circuits and the AMD2901. The proposed partitioning methodology is intended for use within an automated EDA flow to enable current recycling for arbitrary (non-uniform, irregular) VLSI complexity RSFQ circuits, drastically reducing overall bias current and input requirements

**Index Terms**—Automated place and route, current recycling, single flux quantum, superconducting integrated circuits, superconducting logic circuits.

## I. INTRODUCTION

ADVANCES in the fabrication [1] and electronic design automation [2] of superconductive electronics have resulted in increasing complexity of rapid single flux quantum (RSFQ) [3] circuits. Current research efforts in this area are primarily aimed at developing EDA tools to enable the automated design of RSFQ circuits.

One of the distinctive properties of RSFQ circuits, as compared to other superconductive logic families, is the DC bias

Manuscript received December 1, 2020; accepted February 10, 2021. Date of publication March 11, 2021; date of current version April 2, 2021. This work was supported by the Department of Defense (DoD) Agency—Intelligence Advanced Research Projects Activity (IARPA) through the U.S. Army Research Office under Grant W911NF-17-9-0001. (Corresponding author: Gleb Krylov).

The authors are with the Department of Electrical and Computer Engineering, University of Rochester, Rochester, New York 14627 USA (e-mail: gkrylov@ur.rochester.edu; friedman@ece.rochester.edu).

Color versions of one or more figures in this article are available at <https://doi.org/10.1109/TASC.2021.3065287>.

Digital Object Identifier 10.1109/TASC.2021.3065287

supply. Each Josephson junction (JJ) is biased at a specific level by a DC current. A bias current is supplied to each individual cell within a circuit. These DC bias currents are sourced from off-chip. Multiple techniques exist to distribute these currents throughout a superconductive IC to minimize bias variations and power dissipation [4].

Existing and prospective fabrication technologies enable VLSI complexity SFQ circuits with a complexity of over a million JJs per die, approximately several hundred thousand gates [5]. Assuming each gate requires a bias current of several hundred microamperes, the total estimated bias current is on the order of tens to hundreds of amperes. These high currents are difficult to efficiently supply and distribute [6]. Supplied from off-chip, these currents also require a large number of input pins. As each input has a current limit of 200 to 300 mA, the total number of inputs required to supply the bias system can exceed hundreds. In addition, the bias current is distributed on-chip though thin superconductive lines, which are necessarily wide to support these high currents, expending limited metal resources. Furthermore, the high currents produce large magnetic fields which can couple to sensitive RSFQ gates, introducing errors. It is therefore imperative to reduce on-chip bias currents within complex RSFQ circuits.

In this paper, a methodology to apply current recycling to arbitrary (non-uniform or irregular) large scale RSFQ circuits is proposed. This paper is organized as follows. In Section II, current recycling (serial biasing) in RSFQ circuits is briefly introduced, and existing challenges are described. In Section III, a methodology for partitioning RSFQ circuits into blocks with similar bias current requirements during the placement process is presented. Some conclusions are offered in Section IV.

## II. CURRENT RECYCLING

A well known technique to reduce the total bias current is current recycling, also known as serial biasing [7], [8]. In this technique, the circuit is partitioned into multiple segments (or islands) with approximately the same bias current. These segments are galvanically isolated from each other and the rest of the circuit. The partitions use different ground planes and are serially biased, with the ground of each partition acting as a bias supply for the next partition. The inputs and outputs of these serially biased islands are connected to the rest of the system through special pulse transfer circuits, requiring inductive or capacitive coupling [9]. These couplers (driver-receiver pairs) typically consist of a few JJs and coupling elements, and occupy significant area [10]. It is therefore desirable to reduce the number of these couplers.

Serial biasing was first proposed in 1989 [11] and is widely used in complex RSFQ circuits [12]. This approach however requires manual *ad hoc* designation of repetitive blocks within a circuit – a process difficult to automate and reuse. Unlike highly regular, specialized circuits typically used to demonstrate current recycling (*e.g.*, shift registers), a limited number of repetitive structures exists in general VLSI circuits. The benefits of current recycling in this case are applicable to only a small portion of a system. Moreover, automated design tools for RSFQ circuits, in particular, automated place and route (APAR) tools, need to be aware of these current recycling features, as the placement and routing process changes with the introduction of separate ground planes and driver-receiver pairs.

To date, current recycling has not been applied to general RSFQ circuits. In this paper, a methodology is proposed to mitigate these issues by automatically partitioning arbitrary logic to support current recycling [13]. A similar methodology has been simultaneously and independently proposed in [14], where the ground plane is partitioned after the circuits have been placed (post-placement). The primary distinctive feature of this work is that partitioning is performed during the first steps of the coarse placement process, when the location of the gates has yet to be finalized. In addition, different partitioning and optimization algorithms are used.

### III. PARTITIONING OF ARBITRARY RSFQ CIRCUITS DURING PLACEMENT

In this section, a methodology for partitioning RSFQ circuits during placement is proposed and demonstrated. In Section III-A, serial biasing of partitions with slightly different bias currents is reviewed. In Section III-B, partitioning during placement is introduced and the advantages and disadvantages of this approach are discussed. In Section III-C, the quadratic placement algorithm used here for coarse placement is briefly introduced. In Section III-D, partitioning RSFQ circuits with the Fiduccia-Mattheyses (FM) algorithm is discussed, and results are presented for a number of benchmark circuits. In Section III-E, geometric partitioning with simulated annealing is described, and results are presented and compared to partitioning with the FM algorithm.

#### A. Unbalanced Partitioning of RSFQ Circuits With Padding

In existing RSFQ circuits utilizing serial biasing, all of the partitions exhibit identical bias currents [12]. This condition, however, is not a strict requirement. While different bias currents for each island degrades the bias margins, resulting in over- or underbiasing of the entire island, certain small differences in the bias current can be tolerated. The balance conditions for the partitioning algorithm therefore depend upon the robustness of the circuit to changes in the bias current.

RSFQ technology provides a means to mitigate any imbalance among the different islands. Dummy gates, for example, a chain of JTL stages or individual JJs, can be added to those islands with a smaller bias to equalize the bias current among islands. A padding JTL can be placed anywhere within an island and only needs to be connected to a bias line within this island, not a signal

line. This JTL does not need to be operational or efficient. The total bias current of these elements can therefore be arbitrarily chosen to equalize the bias currents among the islands while reducing the added area.

In ERSFQ circuits [15], it is desirable to connect a dummy JTL to a clock line. In this case, due to the properties of ERSFQ bias networks [16], padding elements do not dissipate static power. Furthermore, ERSFQ circuits utilize a feeding JTL (FJTL) as a voltage regulator [17]. In an ERSFQ circuit with current recycling, the FJTL sets the bias voltage for each island. Each island therefore requires a separate FJTL [18]. Alternatively, a single FJTL can be located within the island with the highest ground voltage. In this way, the FJTL provides a reference for the highest voltage in the circuit, although dissipating additional dynamic power. The current regulation capability of a FJTL also provides additional robustness to small variations in bias current. If the circuit is slightly over- or underbiased, the FJTL compensates for these small changes, maintaining the bias current in the corresponding circuit close to the design target [4].

#### B. Partitioning During Placement

Any complex circuit can be represented as a hypergraph, with vertices corresponding to the gates and hyperedges corresponding to the connections between the gates. Hypergraph partitioning algorithms and heuristics are both widely used in CMOS EDA placement tools [19]. These algorithms are included within the coarse placement process to minimize the number of connections to the different parts of a circuit, thereby reducing the total wire length and overall wiring congestion. Balanced partitioning of a hypergraph is an NP-hard problem [19]. A variety of heuristics have therefore been developed to enable the partitioning and placement of VLSI circuits.

The primary advantage of partitioning during placement is integration into existing EDA flows. The placement tools perform partitioning and can be modified to consider the bias current of the cells during the placement process. In addition, empty space can be reserved by the placement tools to place and route driver-receiver pairs between islands. The number and area of these structures depend upon the number of connections between different partitions, and is not known in advance. This approach also avoids complex irregular shapes for the ground planes, which can be difficult to characterize. With the additional constraints of balancing the bias currents among partitions, partitioning during placement can increase the average wire length [20].

Unlike partitioning during placement of large scale CMOS circuits, a small number of partitions for current recycling in RSFQ provides significant benefits. The total bias current of a circuit separated into  $N$  partitions is reduced by  $N$ . The number of connections between islands also increases with the number of islands. The total number of partitions for current recycling can therefore be small (up to a few tens of partitions). An advantage of recursive bipartitioning is that each pair of partitions is balanced by the bias current, enabling reuse of the same padding elements.

Multiple modifications are necessary to apply existing partitioning algorithms to the problem of RSFQ current recycling. The reduced number of connections between partitions produces fewer driver-receiver pairs. Moreover, due to the limited fanout, RSFQ circuits can be represented as a regular graph rather than as a hypergraph, where each net (edge) is a one-to-one connection between two gates. In CMOS, the balance optimization condition is typically related to the area of the gates [21]. For current recycling in RSFQ circuits, the bias currents are the primary issue within the balance condition, as each gate requires a specific bias current.

### C. Coarse Placement

Many algorithms exist for the automated placement of standard cells [22]. The partitioning methodology presented here is embedded into the first stages of the coarse placement process. During this stage, an approximate location is determined for each cell based on wire length constraints. These locations often significantly overlap. This issue is later resolved in the placement process during the legalization step.

The quadratic placement (QP) algorithm is used here to produce a coarse placement for the proposed methodology. QP is a well known and relatively old algorithm for analytic placement [23]. Among the primary advantages of QP is low computational complexity and global minimization of the wire length. Although many cells are frequently placed at the same coordinates, this placement procedure produces a high fidelity representation of the approximate gate locations. In this section, the QP algorithm is briefly described.

The primary objective of quadratic placement is to minimize the wire length. A connectivity matrix  $C = [c_{ij}]$  is produced, where  $c_{ii} = 0$  and  $c_{ij}$  represents the weight of the connection between different nodes (gates or standard cells) [24]. Different models exist for the nets connecting multiple nodes [23]; for RSFQ logic, all nets can be treated as one-to-one nets. The cost function to minimize the wire length is

$$F = \frac{1}{2} \sum_{i,j=1}^n c_{ij} ((x_i - x_j)^2 + (y_i - y_j)^2). \quad (1)$$

An auxiliary diagonal matrix  $D$  is introduced, where  $d_{ii} = \sum_{j=1}^n c_{ij}$  is the sum of all of the connection weights for a target node [24]. The objective function (1) is rewritten in the form,

$$F = x^T Q x + y^T Q y, \quad (2)$$

where  $Q = D - C$ , and  $x$  and  $y$  are the coordinates for each gate. This problem is reformulated as two linear systems in  $x$  and  $y$  coordinates (only  $x$  is shown here for brevity) [24],

$$Q_{cc} x_c = -Q_{cf} x_f, \quad (3)$$

where  $Q_{cc}$  and  $Q_{cf}$  are submatrices of  $Q$  corresponding to the connections, respectively, between the nodes and between the nodes and immovable pads.  $x_c$  and  $x_f$  are, respectively, the coordinate of the nodes and pads [23]. The coordinates can be determined using any method for solving a system of linear equations. As each node is connected to only a few other nodes,  $Q$  is sparse, resulting in high computational efficiency of QP

even for a large number of nodes. The QP algorithm is used here to produce a high fidelity model of the initial coarse placement.

### D. Partitioning Using Fiduccia-Mattheyses Heuristic

In this section, the Fiduccia-Mattheyses algorithm [25], an improvement on the earlier Kernighan-Lin algorithm [26], is used for bipartitioning. The number of connections between partitions is minimized, while the balance conditions for the partitions are modified to consider the bias currents of different nodes (gates/cells).

For partitioning using FM, a few important terms need to be introduced. A cut is an edge connecting vertices in different partitions. In RSFQ circuits with current recycling, each cut corresponds to a connection between different serially biased blocks. Each cut therefore requires a coupler (driver-receiver pair) circuit. For each vertex, a gain corresponds to a change in the number of connections, resulting from the movement of this vertex into a different partition. If all neighboring vertices of a given vertex are already in the updated partition, the gain is maximum. The vertex with the highest gain represents the best possible move.

The FM algorithm consists of multiple passes. Each pass consists of multiple iterations (moves) and operates as follows. The graph is initially separated into two random partitions. The gain is calculated for each vertex. The move with the highest gain is selected. The corresponding vertex is moved to the other partition. The gain values for the neighboring nodes are updated, and the vertex is locked – it can not be moved again during this pass. This procedure is repeated until all of the vertices are locked. The best partition during a specific pass is used as the initial partition for the following pass, and all nodes are once again unlocked. The algorithm terminates when no further improvement in the solution is produced. The balance conditions are introduced to avoid moving all gates into one partition. The vertex is only moved if the resulting partitions are balanced.

RSFQ technology can also provide certain benefits to the FM algorithm. Most RSFQ gates exhibit a fanout of one, and most cell libraries contain only two input logic gates. Including the clock signal, the highest/lowest gain is bounded by  $\pm 4$  (number of inputs and outputs). Unlike CMOS, no nodes exist with a larger gain including the clock input.

Partitioning during the placement process is illustrated in Fig. 1. The initial circuit netlist, in graph representation, is placed using the QP algorithm, as shown in Fig. 1(a). The circuit is bipartitioned using the FM algorithm. When partitioning RSFQ netlists for current recycling, the primary balance condition is the bias current. In addition, the gain of the node can be modified to consider the relative imbalance among partitions and the approximate coordinates determined by the coarse placement algorithm. Coarse placement is performed on each of the resulting partitions, where the connections to the nodes in another partition are represented by immovable pads, as shown in Fig. 1(b). As the QP algorithm aims to minimize wire length, the connected nodes within different islands are visibly “pulled” toward each other. The space between the partitions in the layouts, shown in Fig. 1(b) and (c), is exaggerated. As discussed in Section III-B, this spacing can be adjusted based on



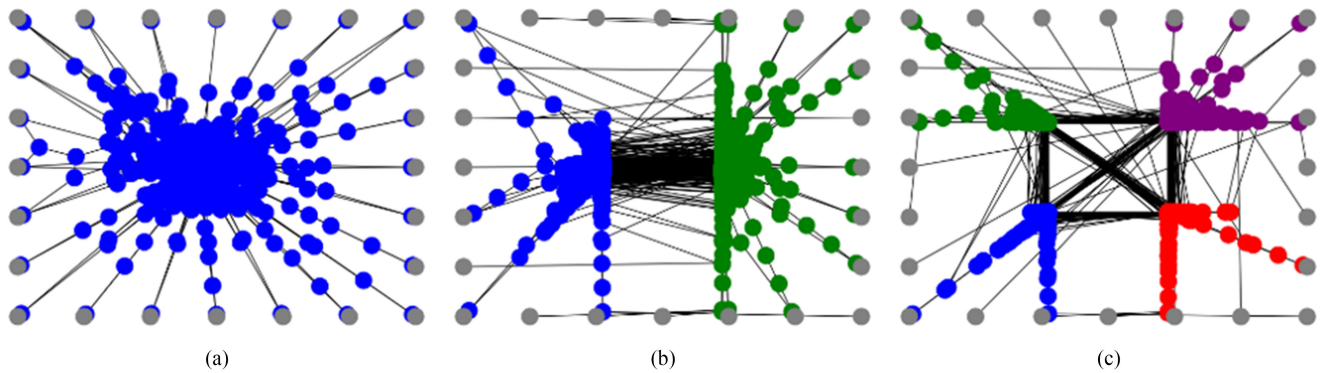


Fig. 1. Coarse placement and partitioning with FM heuristic of a benchmark circuit, (a) initial placement, (b) bipartitioning and placement, and (c) four way partitioning and placement. Different shaded circles (nodes) correspond to individual cells in different partitions, while the lines (edges) represent signal connections.

TABLE I  
RESULTS OF FM PARTITIONING ON MODIFIED ISCAS'89 BENCHMARK CIRCUITS AND THE AMD2901 ALU

| Benchmark circuit | Bias current, mA | Vertices (gates) | Edges (nets) | Two partitions |                |                     |                          | Four partitions |                          |                               |                          |
|-------------------|------------------|------------------|--------------|----------------|----------------|---------------------|--------------------------|-----------------|--------------------------|-------------------------------|--------------------------|
|                   |                  |                  |              | Cuts           | Bias imbalance | Bias difference, mA | Maximum bias current, mA | Cuts (total)    | Bias imbalance (maximum) | Bias difference (maximum), mA | Maximum bias current, mA |
| s27               | 27               | 49               | 70           | 8              | 4.8%           | 0.6                 | 13.7                     | 15              | 5.7%                     | 0.4                           | 6.9                      |
| s298              | 382              | 626              | 942          | 70             | 1.3%           | 2.5                 | 192                      | 125             | 1.4%                     | 1.3                           | 96                       |
| s344              | 379              | 620              | 950          | 75             | 1.5%           | 2.8                 | 191                      | 125             | 1.7%                     | 1.6                           | 96                       |
| s420              | 552              | 889              | 1,354        | 96             | 1.1%           | 2.9                 | 278                      | 180             | 1.3%                     | 1.8                           | 139                      |
| s1238             | 1,516            | 2,450            | 3,776        | 390            | 0.02%          | 0.2                 | 758                      | 657             | 0.1%                     | 0.6                           | 379                      |
| amd2901s          | 2,218            | 3,021            | 4,799        | 459            | 0.01%          | 0.3                 | 1,109                    | 650             | 1.2%                     | 6.7                           | 558                      |
| amd2901f          | 3,313            | 9,349            | 11,127       | 205            | 1.8%           | 30                  | 1,671                    | 669             | 3.4%                     | 27                            | 842                      |

the number of driver-receiver pairs. This procedure is recursively repeated on subsequent partitions, as shown in Fig. 1(c), until the target number of partitions is reached. In a standard cell design flow, the nodes are placed within the corresponding cell rows at later stages of the placement process.

This partitioning methodology is implemented in Python and has been evaluated on CMOS industry standard ISCAS'89 benchmark circuits [27] and the AMD2901 – a 4 bit slice ALU. The benchmark circuits have been modified to better consider RSFQ circuits. The netlist characteristics for these circuits – the bias current and the number of nodes (gates) and edges (nets) – are listed in Table I. Splitters are included for multiple fanout [28]. Multiple fanin gates are replaced by gates with only two inputs. NAND/NOR gates are divided into AND/OR and NOT gates, and the clock inputs are included in each logic gate. A clock distribution network composed of a large splitter tree is also introduced [29], as clock splitters comprise a significant fraction of the total bias current. Two different AMD2901 ALUs are used – the AMD2901f which includes the necessary path balancing D flip flops, and the AMD2901s which does not include these extra flip flops.

The results of applying recursive bipartitioning and placement to these benchmark circuits using this methodology are listed in Table I. Note that the FM algorithm utilizes an initial random or semi-random partitioning, and is therefore nondeterministic – each run produces slightly different results. This methodology minimizes the number of connections among islands while satisfying the required balance constraints. These results emphasize

balancing the bias current among the islands. The number of connections among the islands can be further reduced if this objective is prioritized within the partitioning algorithm. As listed in Table I, this methodology produces a reasonable number of connections between partitions for complex, highly irregular circuits. For two way partitioning, the fraction of cut nets is approximately 5% to 15%, depending upon the structure of a particular circuit. With a larger number of partitions, the number of cut nets increases. Approximately 20% of all connections are cut for four way partitioning. The resulting partitions can be serially biased, reducing the total bias current required by the system. For small differences in bias current, as listed in Table I, dummy padding structures for bias balancing may not be necessary, particularly in ERSFQ circuits.

Although this FM heuristic produces highly balanced partitions with a small number of cut nets, it is computationally hard. The time required to partition a large circuit using the FM algorithm can become impractical. Clustering steps are typically added before partitioning to reduce the complexity [19].

#### E. Geometric Partitioning With Simulated Annealing

A major drawback of this partitioning process using the FM algorithm is the introduction of multiple steps into the design flow. Although APAR tools include multiple partitioning steps as part of the placement process, integration of additional constraints related to equalizing the bias current can increase the total wire length and runtime of the placement process. A

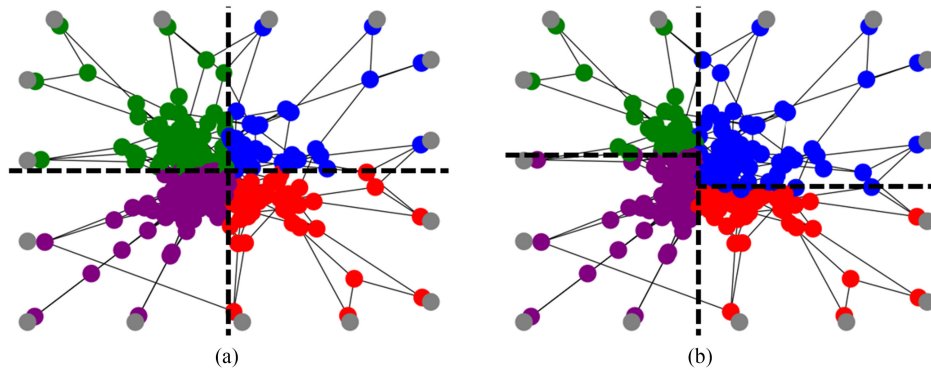


Fig. 2. Coarse placement and geometric partitioning of a benchmark circuit, (a) simple geometric partitioning with equal area, and (b) geometric partitioning with partition boundaries optimized by simulated annealing.

TABLE II  
RESULTS OF GEOMETRIC PARTITIONING ON MODIFIED ISCAS'89 BENCHMARK CIRCUITS AND THE AMD2901 ALU

| Benchmark circuit | Two partitions |                | Two partitions with optimization |                |                     |                          | Four partitions with optimization |                              |                               |                          |
|-------------------|----------------|----------------|----------------------------------|----------------|---------------------|--------------------------|-----------------------------------|------------------------------|-------------------------------|--------------------------|
|                   | Cuts           | Bias imbalance | Cuts                             | Bias imbalance | Bias difference, mA | Maximum bias current, mA | Cuts (total)                      | Bias imbalance (maximum), mA | Bias difference (maximum), mA | Maximum bias current, mA |
| s27               | 6              | 158%           | 16                               | 0.2%           | 0.03                | 13.4                     | 25                                | 14%                          | 0.9                           | 7.2                      |
| s298              | 27             | 1,198%         | 104                              | 0.4%           | 0.8                 | 191                      | 185                               | 57%                          | 42                            | 114                      |
| s344              | 90             | 11%            | 90                               | 0.4%           | 0.7                 | 190                      | 185                               | 13%                          | 11                            | 100                      |
| s420              | 48             | 786%           | 117                              | 1.4%           | 3.8                 | 278                      | 243                               | 37%                          | 46                            | 170                      |
| s1238             | 194            | 739%           | 386                              | 1.5%           | 12                  | 764                      | 738                               | 25%                          | 80                            | 405                      |
| amd2901s          | 177            | 109%           | 318                              | 1.0%           | 11                  | 1,114                    | 667                               | 5.1%                         | 27                            | 560                      |
| amd2901f          | 77             | 2,365%         | 273                              | 0.5%           | 8.1                 | 1,660                    | 692                               | 11%                          | 90                            | 881                      |

simpler technique, less intrusive within established EDA flows, is geometric partitioning [30]. In this partitioning approach, coarse placement of a circuit is divided into multiple blocks based only on the cell (node) coordinates, as shown in Fig. 2(a).

As described in Section III-C, the initial placement process aims to minimize the total wire length. Dividing this placement into partitions of equal area based only on the coordinates can produce severely unbalanced partitions with a large number of cut nets, as listed in Table II for the same benchmark circuits (as described in Section III-D). If the slightly modified coordinates of the partition boundaries consider the bias currents of the resulting islands, as shown in Fig. 2(b), the balance characteristics are improved. Any optimization technique can be used to select these modified coordinates.

Simulated annealing is used here to select the preferable boundary for the geometric partition, producing a smaller bias imbalance and fewer cut nets. For bipartitioning, only one boundary is considered, while for four way partitioning, three boundaries are considered (one for the initial bipartition and one for each resulting partition). The results of geometric partitioning with simulated annealing are also listed in Table II. As with FM partitioning, described in Section III-D, the cost function for simulated annealing emphasizes a balance of the bias currents among different islands. Fewer cut nets with a higher bias imbalance can be produced by changing the cost function of the optimization process.

As seen from a comparison between Tables I and II, geometric partitioning generally produces a greater bias imbalance between islands with more cut nets as compared to FM partitioning.

This approach, however, is computationally more efficient, and is less difficult to integrate into an established design flow. This approach also greatly reduces the maximum bias current of a system due to serial biasing, while requiring minimal changes to the circuit at the cost of increased area.

#### IV. CONCLUSIONS

A methodology for current recycling in large scale RSFQ circuits is described in this paper. The methodology enables current recycling in complex irregular RSFQ circuits by partitioning these circuits into islands with a similar total bias current. The partitioning step is integrated into the first stages of the coarse placement process. Islands with a significantly different bias current are balanced using dummy padding structures. These structures may not be necessary for highly balanced partitions. A modified Fiduccia-Mattheyses algorithm is used for balanced partitioning to reduce the number of galvanically isolated drivers/receivers in an RSFQ circuit with serial biasing. Approximately 5% to 15% for bipartitioning and approximately 20% for four way partitioning of the nets require a driver-receiver pair, while the differences in bias current are on the order of a few per cent. An alternative partitioning technique, geometric partitioning with optimization, is also described. This approach produces more unbalanced partitions, requiring additional driver-receiver pairs. It is however computationally more efficient and introduces fewer modifications into the placement process while enjoying the benefits of current recycling. This partitioning methodology can be integrated into standard

CMOS-like EDA design flows for RSFQ circuits by including bias weighted partitioning during the placement process.

#### ACKNOWLEDGMENT

The authors thank Synopsys Inc. for useful discussions and the AMD2901 netlists. The content of the information does not necessarily reflect the position or the policy of the U.S. Government, and no official endorsement should be inferred.

#### REFERENCES

- [1] S. K. Tolpygo *et al.*, "Advanced fabrication processes for superconducting very large-scale integrated circuits," *IEEE Trans. Appl. Supercond.*, vol. 26, no. 3, Apr. 2016, Art. no. 1100110.
- [2] K. Gaj, Q. P. Herr, V. Adler, A. Krasniewski, E. G. Friedman, and M. J. Feldman, "Tools for the computer-aided design of multigigahertz superconducting digital circuits," *IEEE Trans. Appl. Supercond.*, vol. 9, no. 1, pp. 18–38, Mar. 1999.
- [3] K. K. Likharev and V. K. Semenov, "RSFQ logic/memory family: A new Josephson-junction technology for sub-terahertz-clock-frequency digital systems," *IEEE Trans. Appl. Supercond.*, vol. 1, no. 1, pp. 3–28, Mar. 1991.
- [4] G. Krylov and E. G. Friedman, "Design methodology for distributed large scale ERSFQ bias networks," *IEEE Trans. Very Large Scale Integration (VLSI) Syst.*, vol. 28, no. 11, pp. 2438–2447, Nov. 2020.
- [5] V. K. Semenov, Y. A. Polyakov, and S. K. Tolpygo, "Very large scale integration of Josephson-junction-based superconductor random access memories," *IEEE Trans. Appl. Supercond.*, vol. 29, no. 5, Aug. 2019, Art. no. 1302809.
- [6] S. K. Tolpygo, "Superconductor digital electronics: Scalability and energy efficiency issues," *Low Temp. Phys.*, vol. 42, no. 5, pp. 361–379, Jun. 2016.
- [7] V. K. Semenov and Y. Polyakov, "Current recycling: New results," *IEEE Trans. Appl. Supercond.*, vol. 29, no. 5, Aug. 2019, Art. no. 1302304.
- [8] J. H. Kang and S. B. Kaplan, "Current recycling and SFQ signal transfer in large scale RSFQ circuits," *IEEE Trans. Appl. Supercond.*, vol. 13, no. 2, pp. 547–550, Jun. 2003.
- [9] M. W. Johnson, Q. P. Herr, D. J. Durand, and L. A. Abelson, "Differential SFQ transmission using either inductive or capacitive coupling," *IEEE Trans. Appl. Supercond.*, vol. 13, no. 2, pp. 507–510, Jun. 2003.
- [10] K. Sano *et al.*, "Reduction of the supply current of single-flux-quantum time-to-digital converters by current recycling techniques," *IEEE Trans. Appl. Supercond.*, vol. 27, no. 4, Jun. 2017, Art. no. 1300305.
- [11] V. K. Semenov and M. A. Voronova, "DC voltage multipliers: A novel application of synchronization in Josephson junction arrays," *IEEE Trans. Magn.*, vol. 25, no. 2, pp. 1432–1435, Mar. 1989.
- [12] T. V. Filippov, A. Sahu, S. Sarwana, D. Gupta, and V. K. Semenov, "Serially biased components for Digital-RF receiver," *IEEE Trans. Appl. Supercond.*, vol. 19, no. 3, pp. 580–584, Jun. 2009.
- [13] G. Krylov and E. G. Friedman, "Partitioning of RSFQ circuits for current recycling," in *Proc. IEEE Appl. Supercond. Conf.*, Nov. 2020.
- [14] N. K. Katam, B. Zhang, and M. Pedram, "Ground plane partitioning for current recycling of superconducting circuits," in *Proc. ACM/IEEE Design, Automat. Test Europe Conf.*, Mar. 2020, pp. 478–483.
- [15] O. A. Mukhanov, "Energy-efficient single flux quantum technology," *IEEE Trans. Appl. Supercond.*, vol. 21, no. 3, pp. 760–769, Jun. 2011.
- [16] G. Krylov and E. G. Friedman, "Bias networks for high complexity energy efficient single flux quantum circuits," in *Proc. Government Microcircuit Appl. Crit. Technol. Conf.*, Mar. 2020.
- [17] G. Krylov and E. G. Friedman, "Bias distribution in ERSFQ VLSI circuits," in *Proc. IEEE Int. Symp. Circuits Syst.*, Oct. 2020, pp. 1–5.
- [18] N. K. Katam, O. Mukhanov, and M. Pedram, "Simulation analysis and energy-saving techniques for ERSFQ circuits," *IEEE Trans. Appl. Supercond.*, vol. 29, no. 5, Aug. 2019, Art. no. 1302607.
- [19] D. A. Papa and I. L. Markov, "Hypergraph partitioning and clustering," in *Handbook of Approximation Algorithms and Metaheuristics*. Chapman & Hall/CRC, 2007.
- [20] T. Manikas and G. R. Kane, "Partitioning effects on estimated wire length for mixed macro and standard cell placement," in *Proc. ACM/IEEE Int. Workshop Log. Synth.*, Jun. 2002.
- [21] C. J. Alpert, A. E. Caldwell, A. B. Kahng, and I. L. Markov, "Hypergraph partitioning with fixed vertices," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 19, no. 2, pp. 267–272, Feb. 2000.
- [22] A. B. Kahng, J. Lienig, I. L. Markov, and J. Hu, *VLSI Physical Design: From Graph Partitioning to Timing Closure*, Springer, Netherlands, 2011.
- [23] C. J. Alpert, T. Chan, D.-H. Huang, I. L. Markov, and K. Yan, "Quadratic placement revisited," in *Proc. ACM/IEEE Des. Automat. Conf.*, Jun. 1997, pp. 752–757.
- [24] R.-S. Tsay, E. S. Kuh, and C.-P. Hsu, "PROUD: A sea-of-gates placement algorithm," *IEEE Des. Test of Comput.*, vol. 5, no. 6, pp. 44–56, Dec. 1988.
- [25] C. M. Fiduccia and R. M. Mattheyses, "A linear-time heuristic for improving network partitions," in *Proc. ACM/IEEE Des. Automat. Conf.*, Jun. 1982, pp. 175–181.
- [26] B. W. Kernighan and S. Lin, "An efficient heuristic procedure for partitioning graphs," *The Bell System Tech. J.*, vol. 49, no. 2, pp. 291–307, Feb. 1970.
- [27] F. Brglez, D. Bryan, and K. Kozminski, "Combinational profiles of sequential benchmark circuits," *Proc. IEEE Int. Symp. Circuits Syst.*, vol. 3, pp. 1929–1934, May 1989.
- [28] T. Jabbari, G. Krylov, J. Kawa, and E. G. Friedman, "Efficient splitter trees in SFQ circuits," *IEEE Trans. Appl. Supercond.*, 2021, submitted for publication.
- [29] T. Jabbari, E. G. Friedma, and J. Kawa, "H-tree clock synthesis in RSFQ circuits," in *Proc. IEEE Baltic Electron. Conf.*, Oct. 2020, pp. 1–5.
- [30] F. M. Johannes, "Partitioning of VLSI circuits and systems," in *Proc. ACM/IEEE Des. Automat. Conf.*, Jun. 1996, pp. 83–87.