An Integrated Row-based Cell Placement and Interconnect Synthesis Tool for Large SFQ Logic Circuits

Soheil Nazar Shahsavani, Ting-Ru Lin, Alireza Shafaei,
Coenrad J. Fourie, *Member, IEEE*, and Massoud Pedram, *Fellow, IEEE*

***Abstract*—This paper presents a row-based design methodology covering cell placement, clock tree synthesis, and routing steps for large SFQ circuits. The proposed placement tool initiates by running a state-of-the-art CMOS placer, which places fixed-height but variable-width cells in rows on the chip. Cells in each row are then grouped together such that each group contains at most k cells with the same logic level. Next, for clock routing, this paper proposes HL-tree, which adopts an H-tree with PTL connections to distribute the clock to groups, and within each group, a linear path composed of splitters and JTLs provides the clock to cells. Increasing k reduces the chip area, but also may incur a performance loss. To evaluate the effectiveness of the proposed approach, place-and-route results of a 32-bit Kogge-Stone adder for different values of k are reported. By using this new design methodology, the overall chip area can be reduced by 27% compared with the results of a conventional CMOS placement accompanied by an H-tree clock network.**

***Index Terms*—Row-based design automation, single-flux quantum (SFQ), placement, routing, clock tree synthesis.**

I. Introduction

DEMAND for high performance and energy efficient computing has been driving the development of the semiconductor technology for decades [1]. Until recently, conventional computing technology based on CMOS devices and standard metal interconnects has been able to increase computing performance and energy efficiency fast enough to keep up with this increasing demand. Unfortunately, with increasing challenges to physical scaling of CMOS devices and the conclusive end of Moore's law in sight, there is a significant need to find new technologies and design methodologies that would allow continuation of performance and energy efficiency scaling to well beyond the end-of-scaling CMOS nodes (e.g., an all-around 5 nm gate-length transistor).

Superconducting computing based on the Josephson effect has the potential to be one such solution. This is because Josephson junctions (JJs) switch quickly ($\sim$ 1 ps) and dissipate little switching energy ($\sim 10^{-19}$ J) [2] at low temperatures. In particular, *rapid single flux quantum* (RSFQ) technology was introduced in the 1980s, which uses quantized voltage pulses in digital data generation, reproduction, amplification, memorization, and processing [3]. Further, it has been demonstrated that RSFQ circuits are functional at operating frequencies of up to 770 GHz [4]. Recent developments introduce various approaches, such as new *single flux quantum* (SFQ) logic families, including dual-rail RSFQ [5], *self-clocked complementary logic* (SCCL) [6], *reciprocal quantum logic* (RQL) [7], redesign of the current biasing network for RSFQ [8][9][10], and application of low supply voltage for RSFQ circuits [11]. These techniques significantly reduce the power and energy consumption of SFQ logic circuit realizations [12].

Although extraordinary characteristics such as high frequency and low energy dissipation have been observed, many problems, including architectures, design automation methodologies and tools, and device fabrication require solutions in order for the SFQ logic to become a realistic option for realizing large-scale, high-performance, and energy-efficient computing systems of the future [13]. The increase in integration density of modern superconducting circuit processes allows the design of increasingly complex circuits. The layout of such large circuits requires automated placement and routing tools. A comprehensive study on the status and capabilities of software design tools for *superconductive electronics* (SCE) was published in 1999 [14], and some of the main shortcomings identified then were lack of uniform tool used, even within institutions, and lack of standardized data formats. However, design challenges of SFQ logic and tool development did not receive much attention in the decade thereafter, and a follow-up review published in 2013 [15] concluded that the status of SCE software tools was little better than in 1999.

The focus of this paper is to propose a new design methodology for large SFQ circuits. An earlier SFQ design methodology presented in [16] places cells on a rectangular grid using the min-cut placement algorithm. Moreover, each cell is surrounded by a reserved space for clock channels within which an H-tree synchronous clock is routed to construct a zero-skew clock scheme. Data signals are routed over the gates using *passive transmission lines* (PTLs). The major drawback of this design methodology is that a considerable portion of the chip area should be devoted to implementing the H-tree network, which subsequently increases the chip area.

To reduce the chip area for large SFQ circuits and enable automated placement and routing of such circuits, a row-based design methodology is presented in this paper. More specifically, we present a new design methodology based on fixed-height but variable-width logic cells, where these cells are placed in pre-defined rows on the chip to improve the packing efficiency and automation capabilities of layout tools. After running a state-of-the-art CMOS placer, cells in each row are grouped together such that each group contains at most $k \geq 1$ cells with the same logic level. Furthermore, cells in a group are horizontally abutted. The global clock is then routed using a novel clock network, called HL-tree, which is

Manuscript received September 6, 2016. This work was supported by the Safe and Secure Operations Office of the Intelligence Advanced Research Projects Activity (IARPA) under contract no. FA8750-15-C-0203-IARPA-BAA-14-03.

S. Nazar Shahsavani, T.-R. Lin, A. Shafaei, and M. Pedram are with the Department of Electrical Engineering, University of Southern California, Los Angeles, CA 90007 USA (e-mail: nazarsha@usc.edu; tingruli@usc.edu; shafaeib@usc.edu; pedram@usc.edu).

C. J. Fourie is with Stellenbosch University, 7602 Stellenbosch, South Africa (e-mail: coenrad@sun.ac.za).

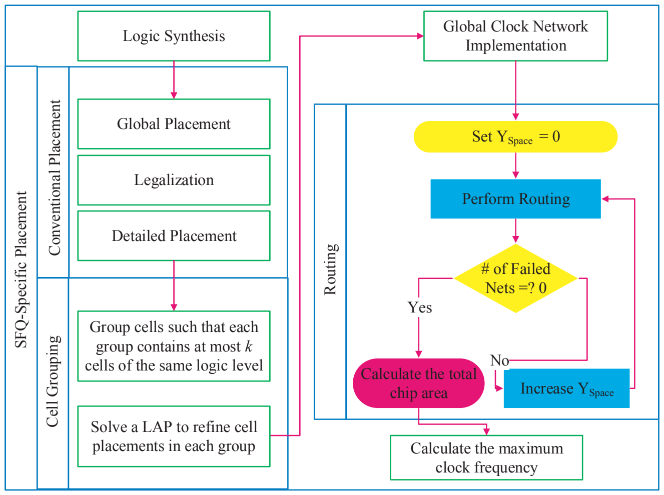
Logic Synthesis
Global Clock Network Implementation
SFQ-Specific Placement
Conventional Placement
Global Placement
Legalization
Detailed Placement
Cell Grouping
Group cells such that each group contains at most k cells of the same logic level
Solve a LAP to refine cell placements in each group
Routing
Set Y_{Space} = 0
Perform Routing
# of Failed Nets =? 0
Yes
No
Calculate the total chip area
Increase Y_{Space}
Calculate the maximum clock frequency

Fig. 1. Proposed place-and-route algorithm. LAP stands for linear assignment problem [25], and Y_{Space} denotes the vertical space between each two row.

a combination of H-tree (a zero-skew clocking scheme) and L-tree (a linear clock propagation mechanism) networks. In HL-tree, an H-tree is adopted to distribute the clock to groups, and within each group, a linear path composed of splitters and JTLs provides the clock to all cells in that group.

Larger values of k result in a smaller H-tree, which means that smaller reserved spaces are needed for the H-tree construction, reducing the total chip area. As a result of this area reduction, wirelengths may also decrease, which may in turn decrease the longest delay on a PTL. However, due to the sequential distribution of the clock in each group, a clock skew is introduced which can incur a performance loss. Hence, deriving the appropriate value of k is crucial in order to reduce the chip area without degrading the maximum clock frequency. As a special case, with $k = 1$, a complete H-tree solution, same as [16] is obtained, which offers the largest chip area among different values of k. To evaluate our method, we place and route a 32-bit Kogge-Stone [18] [19] adder for different values of k and discuss the effectiveness thereof.

The rest of the paper is organized as follows. Our SFQ design methodology including placement, different clock tree synthesis methods, and signal routing are discussed in Section II. Section III presents analytical models for calculating the maximum clock frequency of different clocking schemes. Simulation results of a 32-bit Kogge-Stone adder are reported in Section IV. Finally, the paper is concluded in Section V.

II. Proposed SFQ Design Methodology

Our proposed design methodology is comprised of three different phases: (i) cell placement, (ii) clock tree synthesis, and (iii) routing. Details of each phase are explained in this section. An overview of the place-and-route algorithm is also depicted in Fig. 1.

A. Cell Placement

The output of logic synthesis, which is a gate-level netlist, is input to the placement tool. In addition, the placement tool needs dimensions and pin locations of each cell (gate). The placement tool then assigns cells to positions on the chip such that no two cells overlap with each other and a cost function (e.g., chip area, total wirelength, or critical path delay) is minimized. An important consideration in the placement problem is that the placement solution must be routable. Hence, to avoid multiple costly iterations between placement and routing steps, routing-aware placement algorithms are of significant interest.

To simplify the design automation of the placement step, row-based placement techniques are widely used in the VLSI design community and semiconductor industry. These techniques place (fixed-height, but variable width) cells in rows on the chip. Furthermore, power interconnects run horizontally through the top and bottom of cells. Therefore, when cells are placed adjacent to each other, power interconnects form two continuous parallel tracks in each row. Clock signal is distributed through an H-tree network to all cells. Input and output pins of cells are available at the top and/or bottom sides of the cell, and are connected by interconnects routed in the routing channels between adjacent rows. Connections from one row to another are done either through the surrounding "ring" or by using feed-through cells. A cell that complies with these features will be referred to as a *standard cell*.

To efficiently place netlists with millions of cells, the following three-step placement algorithm is typically used. (i) **Global placement:** the non-overlapping cell constraint is ignored in this step and approximate locations of cells are obtained by placing cells in global bins. The main focus of the global placer is to optimize the cost function by iterating between the solution of some mathematical program (e.g., a constrained quadratic or linear programming problem) and a cell spreading (or bi-partitioning) step. (ii) **Legalization:** the output of the global placement must be legalized to remove any cell overlaps. (iii) **Detailed placement:** legalization is further refined by using local adjustments such as cell movement or swapping to reduce wirelength. The placement legality must be preserved during the detailed placement. Legalization and detailed placement are sometimes grouped as one step.

Modern global placement algorithms are based on analytical methods in which a mathematical analysis is adopted to optimize the cost function. For instance, SimPL [20] solves large and sparse systems of linear equations (formulated using force-directed placement) by Conjugate Gradient method [21]. More specifically, the force-directed method reduces the placement problem to that of solving a set of simultaneous linear equations to determine equilibrium (i.e., zero-force) locations for cells based on Hooke's law analogy. Next, cell spreading is performed by inserting additional forces that pull cells away from dense regions toward carefully placed anchors (pseudo-fixed pins). On the other hand, min-cut placement and simulated annealing-based placement are often used for performing the legalization and detailed placement steps.

For the SFQ placement problem, we assume that the input netlist is *path-balanced*[1] and all fan-outs are implemented with splitters. Our placement tool then initiates by running a global placement (using SimPL [20]) followed by detailed placement and legalization, which together generate a legal solution with the minimum total wirelength. Our proposed clock tree network can work with this initial placement solution (this is when we have $k = 1$), but results in a large H-tree for clock routing. Accordingly, in our proposed placement approach and when $k > 1$, this initial placement solution is refined to minimize the clock tree cost. For this purpose, the logic level of each cell in the netlist is needed, which can be generated by

[1]In a path-balanced netlist, all paths from any primary input to any primary output have the same logical depth [22]. Any netlist can be path-balanced by inserting D flip-flops (DFFs) to paths with a smaller depth.

TABLE I
WIDTHS OF STANDARD CELLS USING MIT-LL SFQEE5 [26]. ALL STANDARD CELLS HAVE A HEIGHT OF 120 µM, EXCEPT FOR SPLITTER CELLS USED IN THE H-TREE AND HL-TREE WHICH HAVE A HEIGHT OF 40 µM.

Cell	Splitter	NOT	DFF	AND	OR	XOR	NDRO
Width (µm)	30	30	40	50	50	50	50

the logic synthesis tool. The logic level of a cell captures the stage of the clock signal that is received by that cell. More precisely, if the longest path from any primary input of the circuit to a SFQ logic cell in a circuit has a logical depth (in terms of node count) of l, then the clock stage associated with that cell is l and the cell will be given level l.

After the initial placement solution is generated by SimPL [20], for each row, starting from left, we select the first cell. Assume that the logic level of this cell is i. We then move to the right and select the first $k-1$ level i cells. This creates our first group which is placed at the first column in that row. We continue the same process to create other groups. Moreover, cells in each group are horizontally abutted. In other words, horizontal spaces between cells in a group are removed to reduce the chip area. By this approach, a group can contain at most k cells; however, the width of each column is chosen based on the group with the largest width in that column and other groups in the column are left aligned.

B. Standard Cells

As stated above, our placement tool utilizes a row-based methodology. That is, the chip is partitioned into several rows for placing standard cells. Each SFQ standard cell in our library is composed of two parts (cf. Fig. 2): (i) A *logic design* part (logic part for short), which implements a Boolean function such as AND, OR, INV, etc. (ii) A *built-in clock distribution* part (clock part for short), which is placed above the logic part. The clock part contains a splitter which provides the clock signal to the corresponding logic part, and also passes the clock pulse to the next cell. When the logic part does not need a clock signal (e.g., splitter or merger cells), the clock part implements a JTL to pass the clock pulse to the next cell. As a side note, if cells in a row are sorted based on their logic level, the built-in clock part without requiring a PTL can distribute the clock signal to all cells in that row.

In our SFQ standard cell library, using the MIT-LL SFQ5ee process technology [26], all cells have the same height of 120 µm. More specifically, heights of logic and clock parts of each standard cell are 80 µm and 40 µm, respectively. However, widths of standard cells, as reported in Table I, vary from 30 µm to 50 µm. On the other hand, splitter cells that are used for implementing the H-tree clock network have a height of 40 µm and a width of 30 µm.

For the clock part, four different templates exist which are shown in Fig. 3. This built-in clock is used to linearly (sequentially) propagate the clock pulse to all cells in a group. Accordingly, in a group with k cells, from left to right, Fig. 3 (a), Fig. 3 (b), and Fig. 3 (c) are picked for clock parts of the first cell, $k-2$ intermediate cells, and the last cell, respectively. Fig. 3 (d) shows the clock part for a special case where $k = 1$ or $k > 1$ but a group has only one cell. Furthermore, we do not use JTLs for data signal routing, since (i) JTLs require JJs and hence occupy the active layer, which in turn complicates the placement problem, and (ii) JTLs are slower than PTLs

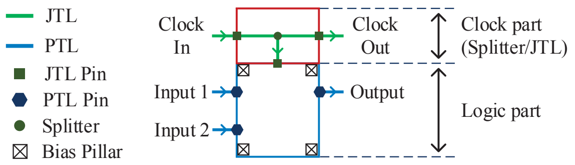

Fig. 2. A sample standard cell composed of clock and logic parts. Clock part has different templates which are shown in Fig. 3.

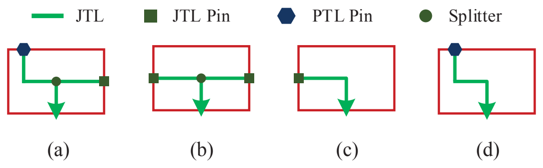

Fig. 3. Four templates for the clock part of our standard cells.

especially for long-distance communications. Therefore, data nets are routed using PTLs.

C. Clock Tree Synthesis

Clock tree synthesis is a process that makes sure that the clock signal is properly distributed to all sequential elements in a circuit. Clock routing is performed before signal routing to avoid competition for resources occupied by signal nets. In SFQ circuits, a D flip-flop (DFF) is included in each logic cell (except for splitters and mergers). As a result, the clock tree network is significantly larger in SFQ circuits compared with that of CMOS circuits. Accordingly, the clock tree synthesis must be given a very high priority in SFQ designs.

Assuming that a zero-skew clock scheme (i.e., the clock tick simultaneously arrives at all registers) is available, which can be done using the H-tree clock network, one may directly use the output of the CMOS placement tool for placing SFQ circuits. This approach has been adopted in [16], which uses the min-cut placement algorithm for deriving cell positions. Although the results can be improved by using state-of-the-art placement tools, such as SimPL [20], the main issue is the implementation of the H-tree clock in SFQ circuits. Since almost all SFQ cells require a clock signal, a huge H-tree network is needed especially in large circuits, which in turn decreases the chip density (i.e., portion of the chip used for logic cells).

To alleviate the high cost of the clock network in large SFQ circuits, we propagate the clock to all cells using a novel clock network called *HL-tree*. After grouping at most k cells of the same logic level in each row, the global clock is simultaneously transfered to the first cell from the left of each group using a partial H-tree. Inside each group, the clock signal is propagated from the initial cell to the rest of the cells in that group using the built-in clock part. By using our HL-tree, since horizontal spaces between cells in a group are removed, the chip area is reduced. Consequently, total wirelength may also decrease. This is especially important if the maximum wirelength that results in the longest PTL delay is decreased. However, due to the sequential distribution of the clock inside groups, a clock skew is introduced. Hence, we place and route the netlist for different values of k.

Two other clock networks have also been considered in this work. (i) *H-tree*: for $k = 1$, a large H-tree is implemented to transfer the clock to all cells in the circuit. In this case, the built-in clock part of standard cells is not needed and thus is removed. (ii) *L-tree*: after obtaining the initial placement solution, cells in each row are sorted in the increasing order of their logic level. The clock signal is provided from the left side of the chip, and a combination of PTLs and splitters distribute the clock signal to the first cell in each row. Clock signal is then transfered to all cells with the same logic level as the first cell using the built-in clock part. Moreover, the clock signal is propagated from the last cell of the group with the max number of cells of logic level i, among all rows, to all the first cells of the logic level $i+1$, in all the other rows, using a combination of PTLs and splitters. Further, the clock signal is propagated to the first cell of level $i+1$ in the same row using a splitter and JTL connection. In this way, clock is synchronized when each logic level has finished its computation. Although this method, compared with the H-tree and HL-tree networks, leads to a much lower chip area, it significantly degrades the performance. This is because of the large clock skew imposed by the large number of same level cells that may appear in a row (in HL-tree, we can control this undesired clock skew by choosing relatively small k values). The HL-tree and L-tree clocking networks are shown in Fig. 4.

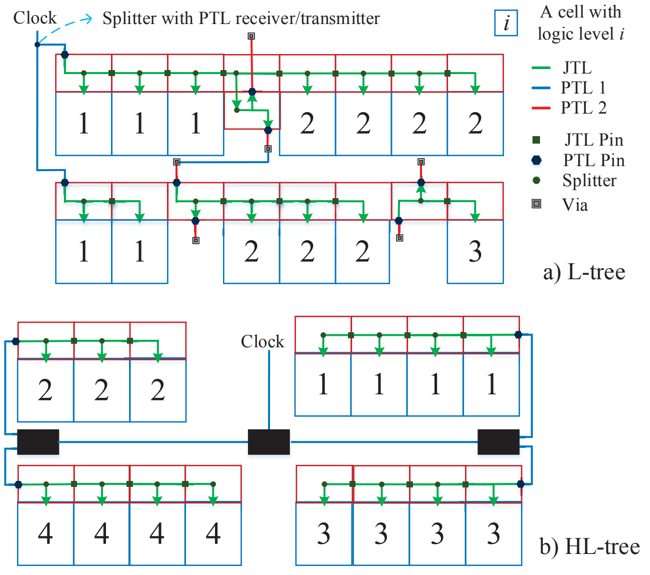

Fig. 4. (a) L-tree and (b) HL-tree ($k = 4$) clock networks. Numbers in each cell indicate the logic level of the cell. L-tree placement shown in (a) routes the clock signal inside each group without PTLs, whereas HL-tree placement requires PTLs to transfer the clock signal to the first cell of each group.

D. Routing

Given a placed netlist, a signal routing solution determines the necessary wiring, including net topologies and specific routing segments, to connect cells while meeting design rule constraints and routing resource capacities. Inputs to the routing problem are given in open standard *library exchange format* (LEF) and *design exchange format* (DEF) files, which together represent the complete physical layout of an integrated circuit in an ASCII format. Specifically, LEF includes design rules and abstract information about the cells, whereas DEF represents the netlist and circuit layout.

Routing of signal nets is performed in two steps. (i) **Global routing**, whereby wire segments are tentatively assigned to coarse-grain routing regions. If routing resources (e.g., number of metal layers) are insufficient, global routing may fail. Therefore, the global router can determine if a given placement is routable in the given technology. (ii) **Detailed routing**, whereby specific routing tracks, vias, and appropriate segments of metal layers are assigned to each net in a manner that is consistent with the given global route of the net in question. Accordingly, the detailed router must account for design rules.

Authors in [27], have proposed an advanced SFQ-specific routing methodology using PTLs, considering timing variabilities of active devices and PTLs, by adding additional PTLs for delay insertion using an integer linear programming method. On the other hand, our routing tool is built on top of the open-source Qrouter tool [23], which is developed based on the standard Lee maze router algorithm. Given LEF and DEF files, the routing area is partitioned into a two-dimensional grid of routing tracks. A wave propagation method connects the identified source and target nodes while avoiding obstructions during propagation and calculating the corresponding cost along multiple paths. The L-shaped or doglegged paths with the lowest cost are then traced back to the source node from the target and committed to memory. The grid positions occupied by routed paths become obstructions for future routes on other nets (or become additional source or target nodes for further routing of the same net).

There are two steps to complete the routing of a circuit. The first step seeks to find the routing solution for each net one at a time according to some net ordering, while keeping track of any failed net in a list. In the second step, each net that had previously failed is routed again, this time allowing the router to create shorts with other routed nets. Subsequently, all such shorted nets are removed and added to the list of failed routes. This process continues until all nets have been routed. To avoid an infinite loop in this iterative rip-up-and-re-route process, the same sequence of rip-up and re-route for two nets is not allowed to happen more than once. In addition, four corners of our standard cells are reserved for bias pillars, and clock and signal PTLs cannot be routed through these reserved zones. We handle such cases by introducing "no-route zones" in the Qrouter tool.

III. Performance Modeling

The minimum clock period (T_{CLK}) of the HL-tree clock network is calculated as follows (cf. Fig. 5):

$$T_{CLK} = \max_{\substack{u,v \in G, i,j \in C, (i,j) \in N \\ u \neq v, i \neq j}} \{skew_{u,v} + (p_{u,i} - 1)t_{split} + t_{c2q_i} + t_{pd_{i,j}} + t_{setup_j}\} + t_{margin}, \quad (1)$$

where G, C, and N denote the sets of all groups, cells, and nets, respectively, in the netlist, $skew_{u,v}$ is the difference between clock arrival times of the first cells in groups u and v, $p_{u,i}$ is the position of cell i in group u (assuming that the first cell from the left in a group is in position 1), t_{split} denotes the delay of propagating clock through the built-in clock part of cells which is equal to the splitter delay, t_{c2q_i} represents the clock to Q delay of cell i, $t_{pd_{i,j}}$ is the propagation delay (including PTL receiver and transmitter delays) from cell i to cell j, t_{setup_j} represents the setup time of cell j, and t_{margin} accounts for the required margin due to parameter spread. The aforesaid equation calculates propagation delays for all pairs of connecting cells and returns the maximum value as the maximum achievable clock period.

In Equation (1), t_{split} and t_{c2q_i} are much smaller than $t_{pd_{i,j}}$ due to the fact that $t_{pd_{i,j}}$ is composed of PTL transmiter

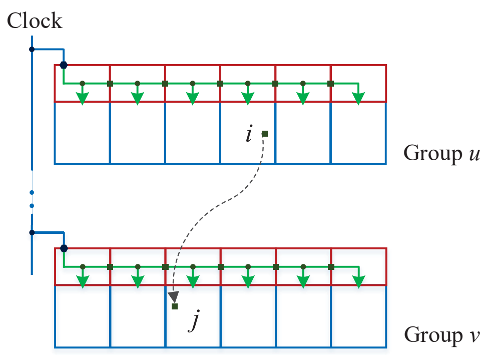

Fig. 5. The maximum clock frequency evaluation for HL-tree clock network.

and receiver delays as well as the propagation delay through the PTL wire, which is dependent on the longest path in the design; however, t_{split} is a constant value, i.e., the propagation delay through a splitter cell. Hence, the bottleneck for the maximum achievable frequency is the longest path in the design which should be reduced as much as possible. On the other hand, if a complete H-tree is used to propagate the clock signal to all cells in the design, $p_{u,i}$ is equal to 1 for all cells, eliminating the term $(p_{u,i}-1)t_{split}$ from Equation (1).

The maximum clock frequency for the case of L-tree clock network is mainly dependent on the maximum number of same level cells in a row. The minimum clock period for L-tree clock network is obtained as follows:

$$T_{CLK} = \max_{l \in L}\{(\max_{u \in G}(p_{u,l}) - 1) \cdot t_{split} + \max(t_{BT}, t_{pd_{l,l+1}})\}, \tag{2}$$

where L and G represent the set of all logic levels and the set of all cell groups, respectively. t_{BT} denotes the delay of the binary tree which is used to propagate clock from the last cell in level i to all the first cells in level $i+1$ among different rows, which consists of delays of splitters, PTL transmitters and receivers, and is logarithmically dependent on the the number of groups with logic level $i+1$. The maximum propagation delay between all cells of level l to all cells of level $l+1$ ($t_{pd_{l,l+1}}$) is also a function of the longest path in the design. Consequently, the maximum value of the signal propagation delay ($t_{pd_{l,l+1}}$) and the clock propagation delay between consecutive levels (t_{BT}) should be added to the minimum clock period. As it could be observed, the maximum clock frequency for the L-tree design is lower than those of H-tree and HL-tree designs because of the large overhead of linear clock propagation in cells of the same logic level, and also the binary tree delay to transfer the clock signal to next level logic. The maximum clock frequency in the L-tree design is achieved when all the cells of same logic level are distributed uniformly among the rows, and number of cells in different logic levels are close to each other. For instance, if there are n cells, l logic levels, and k rows, then the maximum possible clock frequency happens when there are $\frac{n}{l}$ cells in each logic level and each row consists of $\frac{n}{l \cdot k}$ cells.

IV. Simulation Results

To implement the global placement, we adopt a methodology similar to SimPL [20]. The main purpose of the global placement is to reduce the total interconnect length which could be approximated by *half-perimeter wirelength* (HPWL) denoted by:

$$HPWL = HPWL_x + HPWL_y \tag{3}$$

where

$$HPWL_x = \sum_{e \in E} |max(x_i)|_{i \in e} - |min(x_i)|_{i \in e} \tag{4}$$

$HPWL_y$ could be calculated in a similar manner. A set of cells with connections to each other represents a graph $G(V,E)$ with set of edges E, where e denotes a hyperedge (a multi-pin net), and set of vertices V denoting cells, with edge weights w_{ij} representing the cost of each net. Consequently, the total wirelength, θ, can be calculated as:

$$\theta = \sum_{i,j} w_{i,j}((x_i - x_j)^2 + (y_i - y_j)^2) \tag{5}$$

which could be rewritten in each of the x and y coordinates as follows [20]:

$$\theta_x = \frac{1}{2}\vec{x}^T A_x \vec{x} + B_x^T \vec{x} + const. \tag{6}$$

The Hessian matrix A represents the connection between movable cells and B denotes the connections between movable and fixed cells. Based on [20], (6) could be reduced to:

$$A_x \vec{x} = -B_x \tag{7}$$

For the placement problem, an initial random placement of cells is generated. Then, based on the Bound2Bound (B2B) net model [24] and the initial location of fixed and movable cells, A and B matrices are calculated and (7) is solved using Conjugate Gradient method [20][21].

Once the new locations are calculated, A and B matrices are updated and (7) is solved again. This step continues until the HPWL reduction decreases to less than a preset threshold value. This step is usually completed in 5-13 iterations depending on the number of cells in the design. Next, based on an algorithm similar to *look-ahead legalization* (LAL) [20], grids are formed throughout the chip and in each grid cell, based on the location of the fixed and movable cells, pseudo pins are located and matrices A and B are updated. Solving (7), generates the new location of the cells, and cells are moved toward the lower density areas of each grid cell to remove overlaps. LAL algorithm is performed until the ratio of cell area to whitespace area in each grid cell is less than a preset threshold value (called the overfill ratio). Legalization and detailed placement are then performed to generate the solution with the minimum wirelength. Furthermore, based on the logic level of the cells, placement algorithm modifies the cell locations based on our proposed SFQ-specific placement method.

In this method, cells with the same level are grouped together based on their initial ordering, produced by the global placement. We start by the first cell with logic level i, and group that cell with the first $k-1$ cells of the same logic level in the same row. We repeat this process for all cells in each row until all cells are grouped. Furthermore, in each cell group, a *linear assignment problem* (LAP) [25] is formulated as follows. The area dedicated to the placement of n cells is subdivided into n slots with an equal size, and the cost of assigning each cell to each slot is calculated in terms of the HPWL. The corresponding LAP is then solved to come

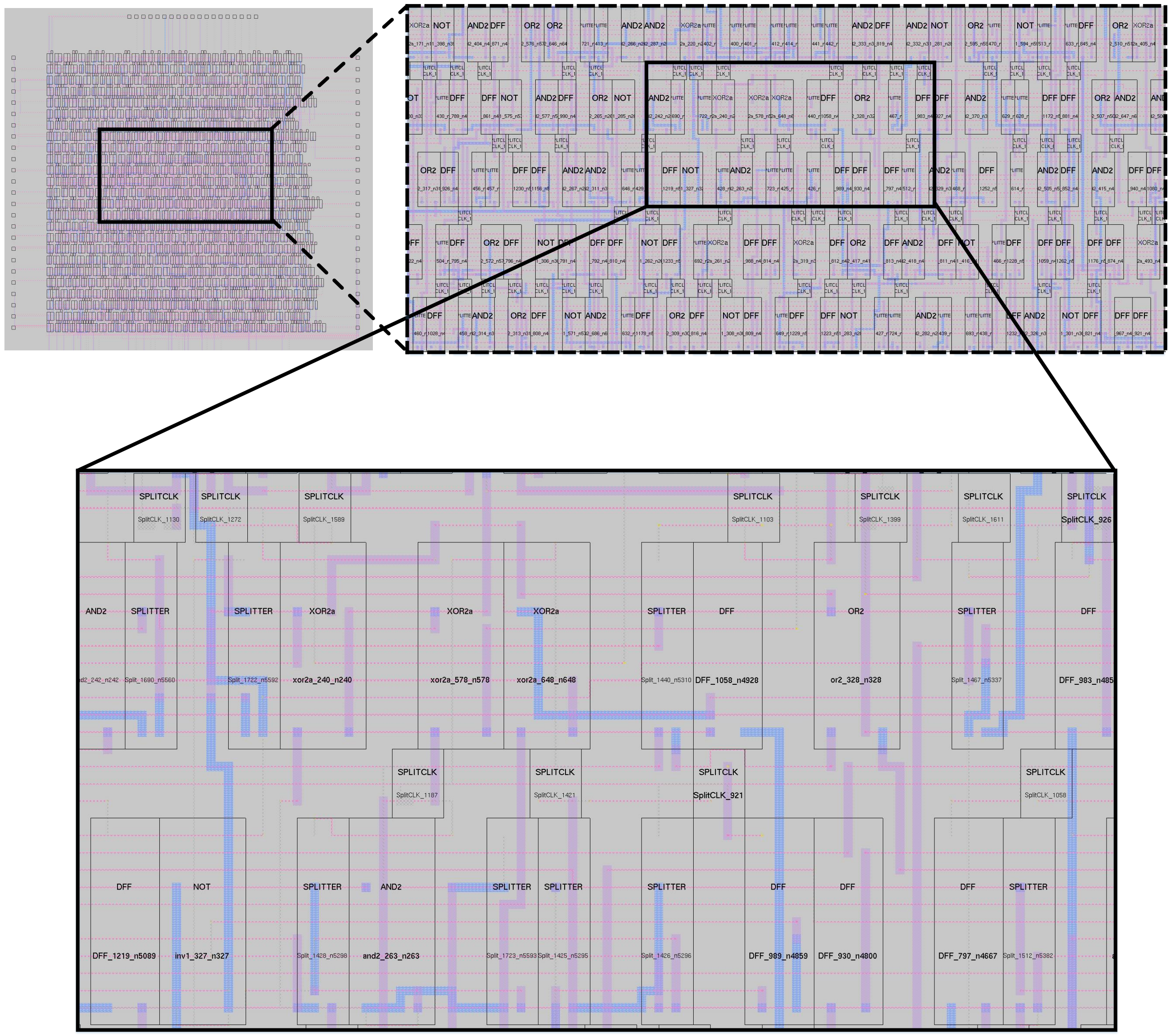

Fig. 6. Part of the layout of the 32-bit Kogge-Stone adder after SFQ-specific placement and routing.

TABLE II
COMPARISON OF DIFFERENT CLOCKING SCHEMES IN TERMS OF THE TOTAL CHIP AREA AND THE MAXIMUM CLOCK FREQUENCY FOR A 32-BIT KOGGE-STONE ADDER. M DENOTES THE NUMBER OF PTL LAYERS USED FOR ROUTING.

Clock Network	**Chip Area (mm^2)**		**Clock Frequency (GHz)**	
	$M = 4$	$M = 5$	$M = 4$	$M = 5$
H-Tree	29	29	17.9	17.9
HL-Tree ($k = 2$)	24	24	17.4	18.5
HL-Tree ($k = 3$)	22	22	10.6	18.1
HL-Tree ($k = 4$)	21	21	10.4	16.7

up with the best ordering of the cells in the group to further refine the placement and improve HPWL. Note that for the case of L-tree placement, all groups of the same logic level are attached to each other, and groups are sorted in the increasing order of the logic level.

Clock nets are then connected from the global clock pin (denoted by GCLK) to all cells in the circuit using H-tree, HL-tree, or L-tree clock networks. These new clock nets are also added to the netlist, and the updated netlist is passed to the Qrouter tool [23] to complete clock and signal routings. The router algorithm starts with the original placement and tries to connect all the nets. If there are no failed nets, the routing is finished. However, if some nets cannot be connected due to high congestion in the routing tracks, the vertical space between each two row (Y_{Space}) is increased. We use a binary search algorithm to find the minimum total area for which there are no failed nets after the routing is performed.

We have tested our placement and routing methodology on a 32-bit Kogge-Stone [19] adder, and generated final routing results for H-tree, L-tree and HL-tree clocking schemes. For the adopted adder, the total number of cells and data nets (signals) are $1,645$ and $2,147$, respectively, and the maximum logic level of a cell is 13. The total number of nets added for clock routing in the case of L-tree, HL-tree ($k = 4$), and H-tree are equal to 296, $1,045$, and $2,415$, respectively.

Results of the total chip area after routing with 4 and 5 PTL layers and different clocking schemes (H-tree, HL-tree with group sizes of 2, 3, and 4) are shown in Table II. The minimum chip area which could be achieved using H-tree is $1.38\times$ larger than that of HL-tree. Based on our simulations, performing cell reordering based on logic level after the placement makes the routing of the signals easier, and results in a lower chip area compared with the case where conventional placement is performed. Part of the final layout of HL-tree with (K = 2) placement and routing is depicted in Fig. 6. Using (1), maximum achievable frequencies for H-tree, HL-tree using groups of size 2, 3, and 4 approaches are reported in Table II.

In our current designs, the maximum wirelength due to the large size of current cells is significantly large. For instance, in the case of H-tree design, the longest path length is $4,170$ μm, which is propagated using PTLs. Assuming a propagation velocity of 100 μm/ps for PTLs, the longest path delay is 41.7 ps, which significantly affects the maximum clock frequency. The same issue degrades the maximum clock frequency of HL-tree networks for group size of 3 and 4 using 4 metal layers. By increasing the number of metal layers, this issue can be resolved and frequency can be increased by more than $1.7\times$. To further address this issue, we are redesigning our logic cells to reduce the overall chip area and hence the longest wirelength. Another approach to alleviate the longest path problem is to use repeaters for long wires.

V. Conclusion

We presented a row-based design methodology, where fixed-height cells are placed in rows on the chip. To obtain a low cost and more practical clock network, a space is reserved above each row which uses a combination of splitters/JTLs and PTLs to distribute the clock signal to all cells in a row. After obtaining the output of a state-of-the-art conventional placer, cells of the same logic level in each row are divided to subgroups, and clock is propagated to first cell of each subgroup using a partial H-tree. Furthermore, to reduce the use of PTL wires for clock routing within each row, clock is propagated from each cell of subgroup to the next cells, using a combination of JTLs/Splitters. Therefore, most of the PTL resources are available for signal routing which results in smaller number of PTL layers and/or smaller chip area to complete the signal routing by using HL-tree approach compared with a H-tree clocking methodology. Simulation results of a 32-bit Kogge-Stone adder also point to the effectiveness of our proposed design methodology in reducing the overall chip area. Also, it is observed that by increasing the number of PTL layers in the process fabrication, chips with smaller area can be achieved. For instance, by increasing the PTL layers from three to four and using our HL-tree placement approach, we can reduce the chip area by 23%. As a future work, we will integrate new constraints that capture the logic levels into the mathematical formulation of the force-directed placement in order to come up with the minimum total HPWL while maintaining the ascending order of levels in each row.

Acknowledgment

The authors would like to thank Naveen Katam (University of Southern California) for helpful discussion and providing the 32-bit Kogge-Stone adder netlist used in this paper.

References

[1] J. Koomey, "Worldwide electricity used in data centers," *Environmental Research Letters*, vol. 3, no. 3, pp. 034008–1–034008–8, Jul 2008.

[2] T. V. Duzer and C. W. Turner, *Principle of Superconducting Circuits*. New York: Elsevier, 1981.

[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 Transaction on Applied Superconductivity*, vol. 1, no. 1, pp. 3–28, Mar 1991.

[4] W. Chen, A. V. Rylyakov, V. Patel, J. E. Lukens and K. K. Likharev, "Rapid single flux quantum T-flip flop operating up to 770 GHz," *IEEE Transaction on Applied Superconductivity*, vol. 9, no. 2, pp. 3212–3215, Jun 1999.

[5] S. Polonsky, "Delay insensitive RSFQ circuits with zero static power dissipation," *IEEE Transaction on Applied Superconductivity*, vol. 9, pp. 3535–3538, Jun 1999.

[6] A. H. Silver and Q. P. Herr, "A new concept for ultra-low power and ultra-high clock rate circuits," *IEEE Transaction on Applied Superconductivity*, vol. 11, pp. 333–336, Jun 2001.

[7] O. T. Oberg, Q. P. Herr, A. G. Ioannidis, and A. Y. Herr, "Integrated power divider for superconducting digital circuits," *IEEE Transaction on Applied Superconductivity*, vol. 21, pp. 571–574, Jun 2011.

[8] Y. Yamanashi, T. Nishigai, and N. Yoshikawa, "Study of LR-loading technique for low-power single flux quantum circuits," *IEEE Transaction on Applied Superconductivity*, vol. 17, no. 2, pp. 150–153, Jun 2007.

[9] L. R. Eaton and M. W. Johnson, "Superconducting constant current source," 2009. [Online]. Available: U.S. Patent 7 002 366 B2

[10] D. E. Kirichenko, A. F. Kirichenko, and S. Sarwana, "No static power dissipation biasing of RSFQ circuits," *IEEE Transaction on Applied Superconductivity*, vol. 21, pp. 776–779, Jun 2011.

[11] M. Tanaka, M. Ito, A. Kitayama, T. Kouketsu, and A. Fujimaki, "18-GHz, 4.0-aJ/bit operation of ultra-low-energy rapid single-flux-quantum shift registers," *Japan Journal of Applied Physics*, vol. 51, no. 5, pp. 053102–1–053102–4, May 2012.

[12] O. A. Mukhanov, "Energy-efficient single flux quantum technology," *IEEE Transaction on Applied Superconductivity*, vol. 21, no. 3, pp. 760–769, Jun 2011.

[13] D. S. Holmes, A. L. Ripple, and M. A. Manheimer, "Energy-Efficient Superconducting Computing Power Budgets and Requirements," *IEEE Transaction on Applied Superconductivity*, vol. 23, no. 3, Jun 2013.

[14] K. Gaj, Q. P. Herr, V. Adler, A. Karniewski, E. G. Friedman and M. J. Feldman, "Tools or the computer-aided design of multigigahertz superconducting digital circuits," *IEEE Transaction on Applied Superconductivity*, vol. 9, pp. 18–38, Nov 1999.

[15] C. J. Fourie and M. H. Volkmann, "Status of Superconductor Electronic Circuit Design Software," *IEEE Transaction on Applied Superconductivity*, vol. 23, no. 3, 1300205, Jun 2013.

[16] Y. Kameda, S. Yorozu, and Y. Hashimoto, "A New Design Methodology for Single-Flux-Quantum (SFQ) Logic Circuits Using Passive-Transmission-Line (PTL) Wiring," *IEEE Transaction on Applied Superconductivity*, vol. 17, no. 2, pp. 508–511, Jun 2007.

[17] K. Gaj, E. G. Friedman, Marc, and M. J. Feldman, "Timing of multigigahertz rapid single flux quantum digital circuits," *Journal of VLSI Signal Processing*, vol. 16, pp. 247–276, 1997.

[18] N. H. E. Weste and D. M. Harris, *CMOS VLSI Design: A Circuits and Systems Perspective*, 4th ed. San fransisco, CA: Addison-Wesley, 2005.

[19] N. Katam *et al.*, "SPORT Lab SFQ Logic Circuit Benchmark Suite," Technical Report, University of Southern California, Los Angeles, CA, USA, 2017 [Online]. Available: http://sportlab.usc.edu/downloads/sfq_benchmarks/

[20] M.-C. Kim, D.-J. Lee, I. L. Markov, "Simpl: An effective placement algorithm," in *International Conference on Computer-Aided Design (ICCAD)*, pp. 649–656, Nov 2010.

[21] M. R. Hestenes and E. Stiefel, "Methods of Conjugate Gradients for Solving Linear Systems," *Journal of Research of the National Bureau of Standards*, vol. 49, no. 6, pp. 409–436, Dec 1952.

[22] W. P. Burleson, M. Ciesielski, F. Klass, and W. Liu, "Wave-pipelining: A tutorial and research survey," *IEEE Transaction on Very Large Scale Integration(VLSI) Systems*, vol. 6, pp. 464–474, 1998.

[23] "Open Circuit Design" [Online]. Available: http://opencircuitdesign.com /qrouter/index.html

[24] P. Spindler, U. Schlichtmann, and F. M. Johannes, "Kraftwerk2: A fast force-directed quadratic placement approach using an accurate net model," *IEEE Transaction on Computer-Aided Design*, vol. 27, Aug 2008.

[25] R. E. Burkard and E. Çela, *Linear Assignment Problems and Extensions*. Boston, MA: Springer US, pp. 75–149, 1999.

[26] S K. Tolpygo *et al.*, "Fabrication Process and Properties of Fully-Planarized Deep-Submicron Nb/AlAlOx/Nb Josephson Junctions for VLSI Circuits ," *IEEE Transaction on Applied Superconductivity*, vol. 25, June 2015.

[27] N. Kito, K. Takagi and N. Takagi "Automatic Wire-Routing of SFQ Digital Circuits Considering Wire-Length Matching," *IEEE Transactions on Applied Superconductivity*, vol. 26, 2016.