A Timing Uncertainty-Aware Clock Tree Topology Generation Algorithm for Single Flux Quantum Circuits

Soheil Nazar Shahsavani, Bo Zhang, and Massoud Pedram
Department of Electrical and Computer Engineering, University of Southern California, Los Angeles, CA, USA
{nazarsha, zhan254, pedram}@usc.edu

***Abstract*—This paper presents a low-cost, timing uncertainty-aware synchronous clock tree topology generation algorithm for *single flux quantum* (SFQ) logic circuits. The proposed method considers the criticality of the data paths in terms of timing slacks as well as the total wirelength of the clock tree and generates a (height-) balanced binary clock tree using a bottom-up approach and an *integer linear programming* (ILP) formulation. The statistical timing analysis results for ten benchmark circuits show that the proposed method improves the total wirelength and the total negative hold slack by 4.2% and 64.6%, respectively, on average, compared with a wirelength-driven state-of-the-art balanced topology generation approach.**

***Index Terms*—single flux quantum, clock tree synthesis, clock topology generation, timing uncertainty, mathematical programming.**

I. Introduction

Superconducting electronics is a promising replacement for CMOS technology, aimed at high-performance and low-power computing. Phenomenal characteristics of *Josephson junctions* (JJs), basic circuit elements in *single flux quantum* (SFQ) technology, such as ultra-fast switching speed and ultra-low energy consumption, promise enhancements of at least two orders of magnitude in terms of the product of the energy-efficiency and operation frequency, compared with CMOS technology [1].

Physical design of logic circuits, especially the synthesis of a robust *clock distribution network* (CDN), plays an important role in designing high-performance circuits which are resilient against process-induced sources of variability. A CDN should be designed to not only eliminate timing violations (by controlling clock skews at various launch and capture flops under nominal conditions) but also to reduce mismatches between the target clock skews under nominal conditions and those achieved under process variations.

Clock network synthesis for SFQ circuits is more challenging than it is for state-of-the-art CMOS circuits because of the following reasons: (i) Clock frequencies in SFQ circuits are in 25-50GHz range, which call for much tighter control of the clock skew; (ii) Nearly all SFQ combinational gates (exceptions being splitter cells and *Josephson junction* (JJ) transmission line cells) receive a clock signal in order to pass out their internal state information as a voltage pulse to the output; (iii) Splitter cells are required to distribute the clock signal from the clock source to all combinational gates that receive the clock signal as an input. (iv) Due to layout constraints (e.g., limited number of routing layers and the need to match driver/receiver impedances to the line impedances to avoid signal bounce and oscillations) and SFQ process design rules (e.g., lack of a stacked via), designing a CDN that would simultaneously minimize various timing metrics such as the maximum clock skew, the insertion delay, and the total negative slack is a difficult undertaking. (v) As a result of ultra-deep (gate-level) pipelining in SFQ circuits, the ratio of the clock insertion delay to the combinational path delay is large. Process-induced variations in SFQ circuits include variations in: (a) inductance values inside the gates and those associated with the passive transmission lines, (b) biasing current levels that affect both gate and clock splitter delays, and (c) critical current levels of JJs which in turn affects the JJs' switching characteristics. These variations can thus change the delay of clock splitters employed in the CDN as well as delays of combinational gates in the circuit, which can in turn result in a large number of setup or hold time violations. Consequently, a large portion of timing margins should be dedicated to process-induced clock skews. These constraints translate into large timing uncertainty in the clock insertion delays and a low timing yield.

This paper presents a timing-aware clock tree topology generation algorithm which considers the signal flow and timing in the data path and the worst-case skew to any clock sink considering both wire and splitter cell delays and the total wirelength of the clock tree, while also accounting for the process variations and timing uncertainties. More precisely, the proposed algorithm generates a balanced binary *clock tree topology* (CTT), level by level, and in a bottom-up manner. Targeting a zero clock skew, generated clock trees are height balanced, i.e, there is an equal number of splitters from the root of the tree to any of the leaf nodes (note that in some cases a single output of an splitter cell is used i.e., the splitter will simply function as a delay element). At each level of the tree, the proposed algorithm solves an *integer linear programming* (ILP) problem to determine which nodes in the clock tree should be paired up (i.e., become siblings by assigning them the same parent node). The objective function of this ILP formulation is the weighted sum of the total wire length and total negative slack of the clock tree. By minimizing this objective function, we simultaneously minimize the routing cost of the clock tree and optimize the assignment of sink nodes to appropriate branches of the clock tree to increase the efficacy of *common path pessimism removal* (CPPR) techniques [2], which in turn help control the adverse and uncertain effects of process-induced variations on the worst-case clock skews.

Experimental results using statistical timing analysis show that the proposed approach improves the total negative slack and the overhead of timing-closure in terms of the number of timing-fix buffers by 64.6% and 52.2%, respectively, compared with an state-of-the-art wirelength-driven approach. Additionally, the total wirelength of the clock tree is reduced by 4.2%.

II. Background

A. Definitions

In this section, we summarize some notations and definitions used throughout this paper. We define the *clock tree topology* (CTT) to be a directed binary tree G (each node has a maximum of 1 incoming edge and 2 outgoing edges). The set of nodes for G consists of the root of the tree S_0 (corresponding to the clock source) and a set of leaf nodes $S = \{S_1, \ldots, S_n\}$, representing the clock sinks. In G, an edge $e_{j,k}$ corresponds to a connection from a parent node S_j to a child node S_k. The **height** of a node is the number of nodes on the path from that node to a leaf node, denoted by H_i. The height of a tree is defined as the height of the root node S_0. The **depth** of a node i is defined as the number of nodes in the (unique) path from the root of the tree to node S_i, denoted by D_i. We assume that leaf nodes are at level 0 of the tree and the root node is at the highest level of the tree. **Common clock path** to a pair of leaf nodes refers to the portion of the clock tree (nodes and edges) that is common between those two nodes. Similarly, the non-common clock path refers to nodes and edges that are only on the clock path to one of the leaf nodes.

Consider a pair of sequentially adjacent flip-flops (FF) connected by combinational gates and interconnects where the signal flows from FF_i (launching FF) to FF_j (capturing FF). The clock skew between FFs i and j is defined as the difference between clock arrival times (phase delay values) at these two nodes and calculated as $t_{skew_{i,j}} = T_i - T_j$, where T_i and T_j denote the clock arrival time at sinks i

The research is based upon work supported by the Office of the Director of National Intelligence (ODNI), Intelligence Advanced Research Projects Activity (IARPA), via the U.S. Army Research Office grant W911NF-17-1-0120.

and j, respectively. In this paper, the max clock skew for a circuit is defined as the maximum absolute value of clock skew between any two FFs. The insertion delay at a node S_i is calculated as the sum of the delay of the nodes and edges on the path from the root of the tree to S_i, as defined below.

$$T_i = D_i \times t_{spl} + \sum_{\mathbf{e_{k,j} \in path(R, S_i)}} \delta_{k,j} \quad (1)$$

where t_{spl} and $\delta_{k,j}$ denote the delay of a splitter cell and the wire delay associated with edge $e_{k,j}$, respectively.

For each path between two sequentially adjacent FFs, two timing constraints are defined; **Setup time** constraint specifies the amount of time the input data to the capturing FF should stay steady before the arrival of the next clock edge [3]. The following inequality summarizes the relation between maximum clock-Q delay of a FF (t^{max}_{c2Q}), the maximum combinational path delay (t^{max}_{comb}), the maximum clock skew ($t^{max}_{skew_{ij}}$), and the maximum setup time of the capturing FF (t^{max}_{setup}).

$$T_p \geq t^{max}_{c2Q_i} + t^{max}_{comb} + t^{max}_{skew_{ij}} + t^{max}_{setup_j} \quad (2)$$

The **hold time** of the capturing FF, defined as the amount of time the input data signal should stay valid after the clock edge, imposes an additional constraint on clock insertion delays and the total combinational path delay over the data path, as follows:

$$t^{min}_{c2Q_i} + t^{min}_{comb} + t_{skew_{ij}} \geq t^{max}_{hold_j} \quad (3)$$

In the worst case, the input signal at the capturing FF should remain stable for t^{max}_{hold} after the clock edge of the same clock signal arrives.

Timing criticality of a path in terms of setup or hold constraints can be defined as the difference between the actual and the required arrival times at the capturing FF. For setup and hold time constraints, setup and hold slacks at a node (e.g. FF_j) corresponding to a path (e.g. $FF_i \rightarrow FF_j$) are defined as follows:

$$slack^{setup}_j = T_p - t^{max}_{c2Q_i} - t^{max}_{comb} - t^{max}_{skew_{ij}} - t^{max}_{setup_j}$$
$$slack^{hold}_j = t^{min}_{c2Q_i} + t^{min}_{comb} + t^{min}_{skew_{ij}} - t^{max}_{hold_j} \quad (4)$$

We define $slack_{i \rightarrow j}$ to be equal to $slack_j$ corresponding to a path from node i to node j. Given the above definitions, a positive slack means the timing constraint is satisfied (i.e., the signal arrives earlier than it is required) while a negative slack is an indicator of a (setup or hold) time violation (i.e., the signal arrives later than it is required) [3].

As a consequence of variations in the fabrication process, the clock-Q delay, setup, and hold time of the FFs, and the propagation delay of combinational gates (e.g., splitters) both in data and clock paths deviate from their nominal values. Accordingly, the setup and hold slacks at different nodes may become negative, resulting in timing violations. As observed, hold time violations are generally more important as they can not be fixed by reducing the clock frequency.

III. Proposed Method

The timing uncertainty-aware topology generation algorithm is defined as follows. **Objective:** Minimize the weighted sum of (i) the total negative slack induced by timing uncertainty in a clock tree and (ii) the total wirelength of a clock tree. **Input:** (i) a placed netlist, (ii) a list of sink nodes, (iii) timing characteristics of logic gates in terms of clock-Q delay, setup time, hold time, combinational delay. **Output:** A CTT connecting the clock source to all the clock sinks. **Constraints:** The clock topology should be a binary height-balanced tree; each node should have at most two child nodes as the maximum fan-out for splitters in the current SFQ cell library is two [4]. Additionally, the total number of clock splitters on each source-sink path should be identical, i.e., all the sinks should have the same depth.

A. Overall Flow

The overall flow of the proposed CTT generation algorithm is shown in Fig. 1. The *clock topology generation algorithm* (ILPTopGen) starts by performing timing analysis on the circuit to calculate the timing criticality of the paths (in terms of setup and hold time slacks), assuming an ideal zero-skew clock network. In the following step, the hierarchy of the clock tree is formed; the number of nodes in each level of the clock tree is calculated. Next, the connections between parent and child nodes are formed level by level, in a bottom-up manner. In each step of the clock topology generation, an *integer linear programming* (ILP) problem is solved that determines which nodes in the clock tree should be paired up. Once the clock topology is generated, the locations of all internal nodes of the tree on the Manhattan plane are determined. Subsequently, timing analysis is performed to calculate the timing slacks after the clock tree synthesis, considering the effect of clock skew and timing uncertainties. At this step, the overhead of fixing all timing violations is estimated. In the following, each step of the ILPTopGen algorithm is detailed.

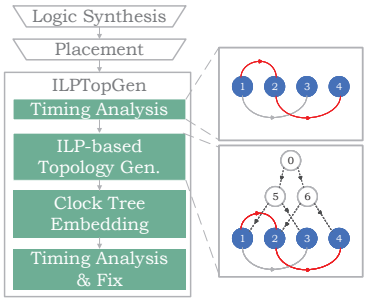

Fig. 1: The overall flow of the proposed clock tree topology generation algorithm (ILPTopGen).

In the timing analysis step, we consider the effect of process variations on the timing metrics of the logic gates and calculate the worst-case timing slack for each path considering the maximum and minimum values of clock-Q delay, setup time, hold time, and combinational gate delay using equation (4). Consequently, we identify the paths that are inherently prone to timing violations. As this analysis is performed after the placement and prior to synthesizing a clock tree, we assume an ideal zero-skew clock tree.

The timing critical paths (i.e., the ones with small positive or negative slacks at the capturing FF) may easily violate the timing constraints when accounting for the process-induced clock skews. Consequently, to minimize the overhead of timing-closure and maximize the timing yield, the timing uncertainty on the clock path to FFs connected by the timing critical paths should be minimized.

Assuming a circuit has n sink nodes, we design the clock tree to have a height of $h^{max} = \lceil \log_2^n \rceil$. Additionally, assuming the sinks to be at level 0 of the tree, the number of nodes at level i ($0 < i$) of the tree will be $2^{h^{max}-i}$. As a result of this topology, single output nodes in the tree only appear at level 1. Nodes at higher levels have exactly one parent and two children. Therefore, ignoring the connections between level 1 and level 0 of the tree, the topology is a full binary tree. Note that although different balanced tree topologies with fewer number of nodes can be utilized, using a larger number of splitters and singleton nodes in the last level of the tree provides more flexibility in terms of placement of the clock splitters, which in turn reduces the total wirelength of the clock tree and the maximum clock skew. In this work, we index leaf nodes based on the topological ordering (i.e., logic level) of the netlist such that there are only data paths from nodes with an smaller index to nodes with a larger index.

Subsequently, we add edges between the parents and child nodes in the tree topology in a bottom-up manner, level by level. The pseudo code for the proposed method for adding edges to the clock tree is shown in Algorithm 1. The inputs to this algorithm are the number of nodes at each level of the tree N, the distances between every pair of nodes at level 0 of the tree (i.e., clock sink nodes) $Dist^0$ and the timing slack at each node denoted by $Slack^0$. Algorithm 1 runs for h^{max} iterations. In each iteration i, by solving an ILP problem, nodes that should be paired up and assigned to the same parent are determined, a parent node is assigned to each pair, and edges are

Algorithm 1: Pair-up nodes and add edges to clock tree

Input: $N = \{card(V^i), i = 1 \dots h^{max}\}$, $E = \{\}$, $V^0 = \{S_1 \dots S_n\}$, $Dist^0$, $Slack^0 = \{slack_{j\to k} \mid V_j, V_k \in V_0, \exists E_{j\to k}\}$
Output: $G(V, E)$

1 **function** pairUpAndAddEdgesToTree()
2 **foreach** $i = 0 \dots h^{max}$ **do**
3 $nDist^i$ = **normalizeDist**($Dist^i$)
4 $nSlack^i$ **normalizeSlack**($Slack^i$)
5 $Pairs^i$ = **solveILP**($i, nDist^i, nSlack^i$)
6 $m = 0$
7 **foreach** $j = 0 \dots card(V^i)$ **do**
8 **foreach** $k = j + 1 \dots card(V^i)$ **do**
9 **if** $Pairs^i_{j,k} == 1$ **then**
10 V^{i+1}_m = **parent**$\left(V^i_j, V^i_k\right)$
11 $V^{i+1} \leftarrow V^{i+1} \cup \{V^{i+1}_m\}$
12 $E^{i+1} \leftarrow E^{i+1} \cup \{e_{V^{i+1}_m, V^i_j}, e_{V^{i+1}_m, V^i_k}\}$
13 $loc\left[V^{i+1}_m\right]$ = **estLoc** $\left(loc(V^i_j), loc(V^i_k)\right)$
14 $m \leftarrow m + 1$
15 $V \leftarrow V \cup V^{i+1}$
16 $E \leftarrow E \cup E^{i+1}$
17 $Dist^{i+1}$ = **updateDistances**$\left(loc\left[V^i\right]\right)$
18 $Slack^{i+1}$ = **updateSlacks**$\left(V^i, V^{i+1}, Slack^i\right)$
19 **return** $G(V, E)$
20 **function** updateSlacks($V^i, V^{i+1}, Slack^i$)
21 **foreach** $j = 0 \dots card(V^i)$ **do**
22 **foreach** $k = j + 1 \dots card(V^i)$ **do**
23 **if** $slack^i_{j\to k} > 0$ **then**
24 V^{i+1}_m = parent(v^i_j)
25 V^{i+1}_n = parent(v^i_k)
26 **if** $V^{i+1}_m \neq V^{i+1}_n$ **then**
27 $slack^{i+1}_{V^{i+1}_m \to V^{i+1}_n}$ += $\gamma + slack^i_{j\to k}$
28 **return** $Slacks^{i+1}$

added from parent nodes at level $i + 1$ to child nodes at level i (cf. lines 5-12).

$Dist^i$ and $Slack^i$ are both triangular matrices with size $n^i \times n^i$, where n^i is number of nodes at level i. For each two nodes j and k at level i, $dist^i_{j,k}$ denotes the Manhattan distance between nodes j and k. $nDist^i$ matrix is calculated by normalizing all the elements of $Dist^i$ by the max distance between any pair of nodes at level i of the tree (cf. line 3). An element $slack^i_{j\to k}$ in the $Slack^i$ matrix represents the slack at node k corresponding to the path from node j to node k. For each two nodes j and k, there may be three possibilities: (i) there are no data paths from $j \to k$, (ii) there is a path with a negative slack at node k, and (iii) there is a path with a positive slack at node k. For case (i) above we set $slack^i_{j\to k}$ to be zero. For the other two cases, we subtract the largest positive slack from all the slack values for each $j \to k$ data path connection such that all slacks become negative. $nSlack^i$ matrix is produced by normalizing all the elements of $Slack^i$ by the largest negative element (cf. line 4). Accordingly, for all nodes j and k, a normalized slack value ($0 \leq slack^i_{j\to k} \leq 1$) is obtained.

As mentioned earlier, to determine the topology of the tree, we need to determine which nodes are paired up at each level of the tree. The objective function for this problem is two-fold: (i) to assign the same parent to the two nodes that are on a timing critical path, (ii) to reduce the total wirelength of the edges connecting two levels of the clock tree. The decision variable is defined as $pair^i_{j,k}$ where j and k are indices of the nodes at level i of the tree ($j < k$). If $pair^i_{j,k}$ is one, nodes j and k become siblings and are assigned to the same parent node. Otherwise they will have different parents and the total number of splitters on the non-common clock path to them increases. The constraints are as follows: (i) each node can only be paired with one other node, (ii) based on the number of nodes at levels i and $i+1$ of the hierarchy, the total number of pairs is fixed. Accordingly, an integer linear programming problem is formulated to determine the connections between nodes at level $i + 1$ and i as follows:

$$
\begin{aligned}
\text{minimize} \quad & \sum_{j=1}^{n^i} \sum_{k=j+1}^{n^i} \alpha \cdot dist^i_{j,k} \cdot pair^i_{j,k} \quad + \\
& \sum_{j=1}^{n^i} \sum_{k=j+1}^{n^i} (1-\alpha) \cdot slack^i_{j\to k} \cdot (1 - pair^i_{j,k}) \\
\text{subject to} \quad & \sum_{k=j+1}^{n^i} pair^i_{j,k} \leq 1 \qquad \forall\, j \in \{1 \dots n^i\} \\
& \sum_{j=1}^{n^i} \sum_{k=j+1}^{n^i} pair^i_{j,k} = m^i \\
\text{variables} \quad & pair^i_{j,k} \in \{0,1\} \qquad \forall\, j,k \in \{1 \dots n^i\},\ j < k
\end{aligned} \tag{5}
$$

In this formulation, n^i denotes the number of nodes at level i of the clock tree. α is a constant ($0 < \alpha < 1$) setting the ratio of importance between the total wirelength and timing criticality. m^i represents the total number of pairs at each level of the tree. At level 0 of the tree (i.e., all nodes are leaf nodes), $m^i = n^0 - n^1$ as there may be single output nodes at level 1. For all other levels, in which each parent node has exactly two child nodes, $m^i = n^i/2$. $dist^i_{j,k}$ and $slack^i_{j,k}$ are constant values, elements of the $nDist^i$ and $nSlack^i$ matrices, respectively.

In the above formulation, the cost function comprises of the total cost of the edges from a level to another (in terms of wirelength) and the total (negative) slack for all the paths among nodes in a level. Since the clock tree is not embedded yet, the cost of each edge (in terms of wirelength) is unknown. In zero-skew embedding algorithms such as the deferred merge embedding (DME) [5], the location of the internal nodes of a clock tree are determined based on the observation that the sum of the cost of the edges between a parent node p and its two child nodes j and k is a constant value (ignoring the detour wires). If nodes j and k are at locations $loc(k)$ and $loc(j)$, respectively, the total cost of edges from a parent node p to j and k is equal to the Manhattan distance between $loc(k)$ and $loc(j)$, independent of the $loc(p)$. Consequently, the $dist^i_{j,k}$ value models the increase in the total wirelength of the clock tree if nodes j and k become siblings. Similarly, if nodes j and k are paired up, the number of splitters on the non-common clock path to nodes j to k becomes 0. Accordingly, slack at node k can not further decrease (become more negative) as a result of timing uncertainty in the clock network.

Intuitively, if nodes j, k are close to each other or there exists a large negative slack on the path from j to k, pairing up j and k reduces the cost function. Using this hierarchical approach, we jointly minimize the uncertainty on the clock skew and routing cost for each level of the tree.

After the ILP problem is solved, the locations of the nodes in level $i+1$ are estimated as a function of the locations of their child nodes (cf. Algorithm 1 line 13). It is important to note that these locations are not the final embeddings of the internal nodes, but estimations required for evaluating the total wirelength of the next level of the tree. We estimate the location of a parent node to be the center of the *merging segment* (MS) of the two child nodes. This is motivated by the well-known DME algorithm that creates a zero-skew MS for each parent node as a function of the merging segments of its child nodes, such that the total wirelength of the clock tree is minimized [5]. Once the location of nodes are estimated, the $Dist^{i+1}$ matrix is updated.

The slack on the data paths between non-leaf nodes is calculated using the *updateSlacks* function (cf. Algorithm 1 lines 20 - 28). In this function, γ represents a (negative) constant value, modeling the uncertainty on the clock path caused by not pairing up nodes j and k, calculated as $(t^{min}_{spl} - t^{max}_{spl})$. The intuition is that if two nodes with a data path connection are not paired in this level, the number of clock splitters on the non-common clock path to them increases by two. in the worst case, one of the splitter delays may decrease to

TABLE I: Simulation results for ten benchmarks from ISCAS85 and EPFL benchmark suites [6] [7]. WL, Avg., and Impr. stand for wirelength, average, and improvement, respectively. We report the average total negative slack and the average number of hold-fixing buffers for all SSTA samples.

	#Sinks	Max Clock Skew (Nominal)			Total Clock Tree WL			Avg. Total Negative Hold Slack (ps) (SSTA)			Avg. #Hold-fixing buffs (SSTA)		
Benchmark		**Baseline**	**ILPTopGen**	**Impr (%)**	**Baseline**	**ILPTopGen**	**Impr (%)**	**Baseline**	**ILPTopGen**	**Impr (%)**	**Baseline**	**ILPTopGen**	**Impr (%)**
EPFL/router	830	3.2	4	-25	331040	326940	1.2	-5.7	-5.2	8.8	7.1	5.3	25.4
EPFL/cavlc	1559	5.6	3.7	33.9	625100	673600	-7.8	-8.2	-2.2	73.2	8	3.4	57.5
EPFL/ctrl	253	3.1	3.7	-19.4	91370	86900	4.9	-2.8	-0.5	82.1	1.4	1.3	7.1
ISCAS/c432	992	4.6	4.1	10.9	370570	344570	7	-7.2	-1.5	79.2	7.7	2.3	70.1
ISCAS/c499	630	3.3	1.8	45.5	281200	298600	-6.2	-2.8	-1.3	53.6	3.5	1	71.4
ISCAS/c1908	1206	7.8	5.6	28.2	633300	654900	-3.4	-8.5	-2.4	71.8	6.8	3.4	50
ISCAS/c2670	3220	6.5	4.9	24.6	1332700	1285400	3.5	-17.9	-6.8	62	18.6	8.5	54.3
ISCAS/c3540	2679	4.2	5.1	-21.4	1275200	1239300	2.8	-17.4	-6.5	62.6	16.2	9.1	43.8
ISCAS/c5315	5286	21.2	12.5	41	2698000	2302000	14.7	-126.1	-47.9	62	76.6	34.7	54.7
ISCAS/c6288	5567	5.9	5.6	5.1	2511900	2516500	-0.2	-40.1	-9.8	75.6	33.9	12.8	62.2
Average		**6.5**	**5.1**	**21.5**	**1015038**	**972871**	**4.2**	**-23.7**	**-8.4**	**64.6**	**18.4**	**8.8**	**52.2**

the min possible value while the other one may increase to the max possible value. Therefore, the uncertainty on the clock skew between them increases by γ.

Solving the ILP for all the levels of the tree generates a height balanced binary tree topology. In the next section, we present the experimental setup, the metrics used for evaluating of the quality of results, and the simulation results.

IV. Experiments

We implemented the proposed clock topology generation algorithm in C++ and used the IBM CPLEX v. 12.8 package for solving the ILP problems [8]. We used the qPlace tool for placement and qSSTA tool for statistical static timing analysis [9]. We implemented the CTT generation algorithm proposed in [10], which uses a variation of MMM method for generating balanced tree topologies with low routing cost [11], as the baseline of our comparison. Note that well-known topology generation algorithms such as greedy deferred merge embedding [12], balanced bi-partitioning [13], and geometric matching [14] either result in unbalanced clock topologies or are wirelength-driven and do not consider the timing uncertainties. Therefore, are not suitable options for generating high-quality clock trees for SFQ circuits. We used the DME algorithm for clock tree embedding [5] and the approach proposed in [10] for clock splitter legalization. Note that although the target is a zero-skew clock tree, due to mapping splitters to routing channels and legalization of clock splitters, the final clock skew may be non-zero [10]. Based on experimental analysis, we chose parameter α to be 0.67.

We evaluated the quality of results using *statistical static timing analysis* (SSTA). For this purpose, in each simulation, we generated random values for all the parameters in a logic cell (i.e, critical current, and inductor values) assuming normal distribution and standard deviation values of 3% and 8% for inductors and JJ area, respectively. Then we calculated the clock-Q delay, setup, and hold time values, and propagation delays for both logic and clock gates and performed timing analysis based on the generated values. We used the RSFQ cell library from [4] and extracted the timing parameters for each gate. As a consequence of process variations, the delay of an splitter gate can deviate from the nominal value of $5.5ps$ to the minimum and maximum values of $4.1ps$ and $9.89ps$, respectively. We performed 6000 Monte-Carlo simulations for each circuit. For each design, we report the total negative slack and the total number of logic gates required to fix the timing violations on all paths, divided by the total number of SSTA samples. Due to space limitations, in this work we only considered the hold time slacks during clock tree generation and we report the quality of results considering the hold constraints only. Experimental results in terms of the the maximum clock skew (in the nominal condition), the total wirelength of the clock tree, the total negative slack, and the total number of required hold-fixing buffers are shown in Table I. As shown, the proposed method reduces the nominal clock skew and the total wirelength of the clock trees by 21.5% and 4.2%, respectively. As mentioned earlier, having more flexibility in terms of placement of clock splitters helps reducing the nominal clock skew and the total wirelength values for most of the benchmarks. Additionally, the ILPTopGen algorithm reduces the average negative hold slack and the average number of required hold buffers by 64.6% and 52.2%, respectively, compared with the baseline method. The proposed method consistently improves the total negative hold slack and the number of required hold-fixing buffers for all the benchmarks. The key factor enabling the success of this approach is considering the critically of the data paths and the total wirelength of the clock, simultaneously, during the clock topology generation. The proposed approach leads to low cost clock trees which require a small number of gates for fixing timing violations in presence of process variations, which in turn reduces the total area and power consumption of the circuits.

V. Conclusion

This paper presented a hierarchical bottom-up clock tree topology generation algorithm for superconducting electronic circuits that considers the timing criticality of the data paths as well as the total wirelength of the clock network using a novel integer linear programming formulation. The ILPTopGen algorithm generates a height-balanced binary clock tree and optimizes the assignment of sink nodes to appropriate branches of the clock tree, aimed at reducing the process induced total negative slack and the total wirelength of the clock tree, simultaneously.

References

[1] 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, 1701610, Jun 2013.

[2] V. Garg, "Common path pessimism removal: An industry perspective: Special session: Common path pessimism removal," in *2014 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, Nov 2014, pp. 592–595.

[3] A. B. Kahng, J. Lienig, I. L. Markov, and J. Hu, *VLSI Physical Design: From Graph Partitioning to Timing Closure*, 1st ed. Springer Publishing Company, Incorporated, 2011.

[4] C. J. Fourie, "Extraction of dc-biased sfq circuit verilog models," *IEEE Transactions on Applied Superconductivity*, vol. 28, no. 6, pp. 1–11, Sep. 2018.

[5] J. Cong, A. B. Kahng, C.-K. Koh, and C.-W. A. Tsao, "Bounded-skew clock and steiner routing," *ACM Trans. Des. Autom. Electron. Syst.*, vol. 3, no. 3, pp. 341–388, Jul. 1998.

[6] M. C. Hansen, H. Yalcin, and J. P. Hayes, "Unveiling the ISCAS-85 benchmarks: A case study in reverse engineering," *IEEE Des. Test*, vol. 16, no. 3, Jul. 1999.

[7] "The epfl combinational benchmark suite," https://lsi.epfl.ch/benchmarks, accessed: 2017.

[8] "IBM ILOG CPLEX." [Online]. Available: www.ilog.com/products/cplex/

[9] S. N. Shahsavani, T. Lin, A. Shafaei, C. J. Fourie, and M. Pedram, "An integrated row-based cell placement and interconnect synthesis tool for large sfq logic circuits," *IEEE Transactions on Applied Superconductivity*, vol. 27, no. 4, pp. 1–8, June 2017.

[10] S. N. Shahsavani and M. Pedram, "A minimum-skew clock tree synthesis algorithm for single flux quantum logic circuits," *IEEE Transactions on Applied Superconductivity*, vol. 29, no. 8, pp. 1–13, Dec 2019.

[11] M. A. Jackson, A. Srinivasan, and E. S. Kuh, "Clock routing for high-performance ics," in *Design Automation Conference, 1990. Proceedings., 27th ACM/IEEE*. IEEE, 1990, pp. 573–579.

[12] M. Edahiro, "A clustering-based optimization algorithm in zero-skew routings," in *30th ACM/IEEE Design Automation Conference*, June 1993, pp. 612–616.

[13] Ting-Hai Chao, Yu-Chin Hsu, Jan-Ming Ho, and A. B. Kahng, "Zero skew clock routing with minimum wirelength," *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, vol. 39, no. 11, pp. 799–814, Nov 1992.

[14] J. Cong, A. B. Kahng, and G. Robins, "Matching-based methods for high-performance clock routing," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 12, no. 8, pp. 1157–1169, Aug 1993.