Reducing Clock Skew Variability via Cross Links

Anand Rajaram
Electrical Engineering Dept.
Texas A&M University
College Station, TX 77843
anandr@ee.tamu.edu

Jiang Hu
Electrical Engineering Dept.
Texas A&M University
College Station, TX 77843
jianghu@ee.tamu.edu

Rabi Mahapatra
Computer Science Dept.
Texas A&M University
College Station, TX 77843
rabi@cs.tamu.edu

Abstract

Increasingly significant variational effects present a great challenge for delivering desired clock skew reliably. Non-tree clock network has been recognized as a promising approach to overcome the variation problem. Existing non-tree clock routing methods are restricted to a few simple or regular structures, and often consume excessive amount of wires. In this paper, we suggest to construct a low cost non-tree clock network by inserting cross links in a given clock tree. The effect of the link insertion on clock skew variability is analyzed. Based on the analysis, two link insertion schemes are proposed. These methods can quickly convert a clock tree to a non-tree with significantly lower skew variability and very small amount of extra wires. Further, they can be applied to the recently popular non-zero skew routing easily. Experimental results on benchmark circuits show that this approach can achieve significant skew variability reduction with less than 2% increase of wirelength.

1 Introduction

In synchronous VLSI designs, the quality of a clock network is vitally important, since the pace of almost every data/signal transfer is coordinated by clock signals (or clock skew which is the difference of clock signal delays among clock sinks) [1]. Moreover, as one of the largest nets and one of the most frequently switching nets at the same time, the clock network has paramount influence on both energy efficiency and power/ground noise [2, 3]. Therefore, the objective of clock network design has long been delivering zero clock skew [4] or useful non-zero skew [5] with a minimum size/wirelength [6–9].

When VLSI technology enters ultra-deep submicron era, many variation effects start to manifest strongly and may deviate clock skew from desired values. The variation factors include manufacturing process variations [10, 11], power/ground noise [12] and ambient temperature variations [13]. The unwanted skew variations are not only harmful to timing performance but also difficult to control, because reliable estimations on these variational conditions are generally not available at the stage of clock network design.

Aimed to solve the skew variation problem, numerous clock routing works have been proposed [14–26]. Among these works, non-tree topology [17, 19, 23–25] is a promising approach, since clock signals propagated through multiple paths can compensate each other on variations. Generally speaking, existing non-tree clock networks can be categorized as 1-dimensional structure [17,24] and 2-dimensional structure [19,23,25]. An early non-tree clock routing work [17] is a 1-dimensional approach (see Figure 1(a)). In the method of [17], a fat metal piece, which is called center chunk, is placed in each subregion and the sinks in each subregion is connected to it directly. A center chunk is driven by a binary tree from the clock source. Since a center chunk is fat and driven at multiple

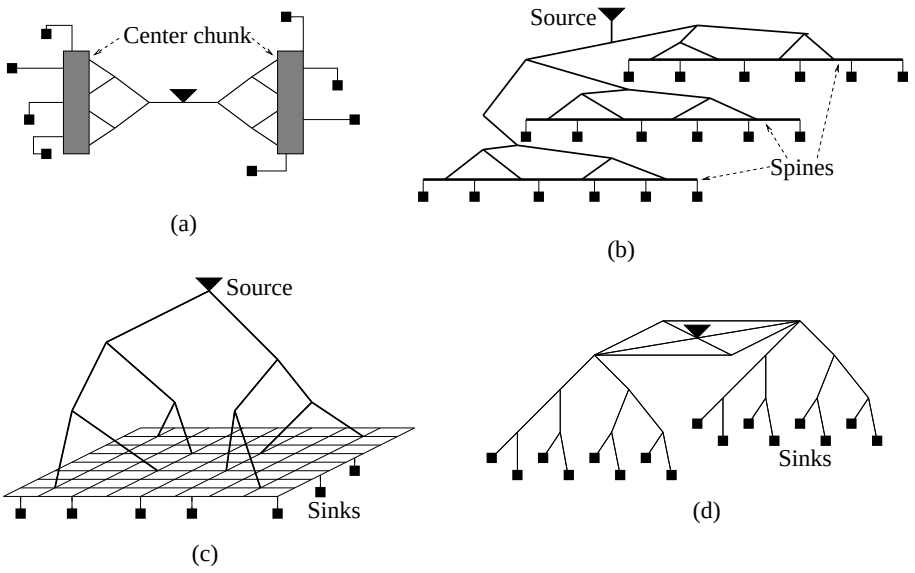

Figure 1: Non-tree clock networks: (a) center chunk; (b) spine; (c) leaf level mesh; (d) top level mesh. Each sink corresponds to a register or a clock subnetwork.

points, the skew between any two points on it is negligible. Another similar approach is the spine method [24] which is employed in $Pentium^{TM}4$ microprocessor and illustrated in Figure 1(b). In the global clock network, there are three spines which span the entire chip and are roughly parallel to each other. Each spine is driven through a tree at multiple points and drives a set of local clock networks. Obviously, a spine in [24] plays a role similar to the center chunk in [17]. One limitation of the 1-dimensional structures is that the variation of skew between different regions are not handled.

The 2-dimensional non-tree structure is also called mesh which can be at either leaf level close to clock sinks or top level close to the clock source. In the leaf level mesh approach [19,23](Figure 1(c)), a metal wire mesh is overlaid on the entire chip area and driven at multiple points directly from clock source [19] or through a routing tree [23]. Each clock sink is connected to the nearest point on the mesh. This technique has been proved to be very effective on suppressing skew variations in microprocessor designs. A leaf level mesh normally consumes enormous wire resources and power. Moreover, it is hard to be integrated with clock gating which is a major low power technique. Therefore, its application is mainly restricted to high-end products such as microprocessors. In contrast, a top level mesh(Figure 1(d)) may consume less wire and power. In the top level mesh approach [25], the clock source drives a coarse mesh directly and clock subtrees are attached to the mesh. The skew variations on the mesh are negligible, but skew variations within each subtree still exist.

Even though several non-tree schemes have been suggested and applied in realistic designs, there are still some important questions without clear and thorough answers.

- Why does a non-tree network have lower variability compared with a tree? Existing answers are based on either common sense or empirical simulation results, however, no theoretical or rigorous explanation has been provided yet.

- Does a non-tree clock network have to be a regular structure? If this restriction is relaxed, a huge solution space of non-tree topology can be explored. The poor tractability of timing

performance in an arbitrary non-tree needs to be solved before it is utilized.

- What is the most efficient usage of wire resource for reducing skew variability? Existing non-tree methods often consumes excessive amount of wires and power. For example, the top level meshes in [25] result in $59\% - 168\%$ more wire area than trees. High expense on wire and power is rarely acceptable in ordinary chip designs except in the case of a few high-end products. Therefore, low cost non-tree networks are strongly needed, particularly for ASIC designs.

- Can non-tree topology be applied to achieve useful non-zero skew efficiently? Useful non-zero skew routing [8,9,14,27–29] attracts many attentions for the sake of timing optimization [5,30] and power/ground noise reduction [2,3]. These methods are restricted to tree topology which could be much less balanced than zero-skew routing tree, and therefore more sensitive to variations. On the other hand, existing non-tree clock networks are mostly designed for zero skew only. Hence, an efficient low variability useful skew routing method needs to be developed.

In this work, efforts are made toward solving the above problems. We propose a new framework of non-tree network which is constructed by inserting cross links in a clock tree. An analysis on impact of link insertion on skew variability is performed. The result of this analysis partially explains the reason why a non-tree may work better than a tree on skew variation reduction. In this structure, cross links can be inserted where they are most effective, so that a great wire efficiency can be obtained. Moreover, cross links can be applied to achieve low variation non-zero skew easily. We suggest link insertion schemes that can quickly convert a clock tree to a non-tree with significantly lower skew variability and very small increase of wirelength. This approach provides a low cost alternative in complement with existing non-tree methods. Monte Carlo simulations on benchmark circuits show that our method can achieve remarkable skew variability reduction with less than 2% increase of wirelength.

2 Skew in RC Network

In this section, delay and skew variation of a non-tree RC network will be analyzed. Elmore delay model is employed due to its high fidelity [31] and ease of computation. A SPICE simulation is performed on a simple case to verify a conclusion from Elmore delay model.

2.1 Delay In RC Networks

The Elmore delay at node i in an RC network is given by [32]:

$$t_i = \sum_j R_{i,j} C_j \tag{1}$$

where C_j is the ground capacitance at node j. The transfer resistance $R_{i,j}$ is equal to the voltage at node i when $1A$ current is injected into node j and all other node capacitors are zero [33]. Even though Elmore delay can be obtained directly from Equation(1), the tree/link partitioning method [34] is employed to reveal the effect of link insertions.

A non-tree RC network can be represented by a graph $G = (V, E)$ with the node set V composed by the source, sinks and Steiner nodes. The graph can be decomposed to a spanning tree $T = (V, E_T)$ and a set of link edges E_l such that $E = E_T \cup E_l$ and $E_T \cap E_l = \emptyset$. The delay from the

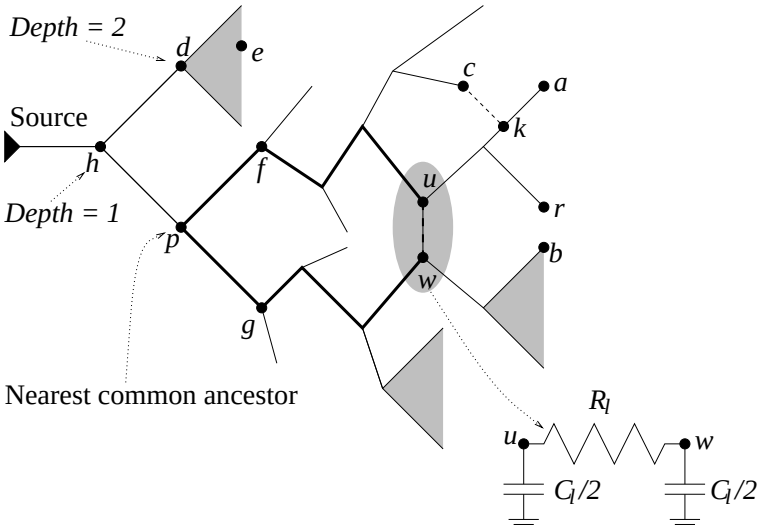

Figure 2: Cross link insertion.

source to each node can be evaluated by starting with delays in the tree T and then incrementally inserting every link edge in E_l[1].

In Figure 2, a network is indicated by the solid lines and a cross link is inserted between node $u \in V$ and node $w \in V$. Without loss of generality, we assume that the degree of each node is no greater than three. If the link has a wire resistance of R_l and wire capacitance of C_l, the link insertion can be decomposed to inserting a resistor of R_l between u and w and adding a capacitor of $\frac{C_l}{2}$ at node u and w. Adding link capacitors does not change the network topology, thus its effect can be estimated easily. If the Elmore delay from the source to any sink i is t_i before the link insertion, the delay $\tilde{t}_i$ after only adding the link capacitors is given by:

$$\tilde{t}_i = t_i + \frac{C_l}{2}(R_{i,u} + R_{i,w}) \tag{2}$$

The impact of the link resistance R_l can be analyzed through the technique developed by Chan and Karplus [34] or equivalently the large change sensitivity analysis in [35]. According to [34], the delay at node i is changed from $\tilde{t}_i$ to $\hat{t}_i$ given by:

$$\hat{t}_i = \tilde{t}_i - \frac{\tilde{t}_u - \tilde{t}_w}{R_l + r_u - r_w} r_i \tag{3}$$

where a node resistance r_i is equal to the Elmore delay at i when $C_u = 1, C_w = -1$ and the other node capacitance are zero.

2.2 Skew Variability Between Link Endpoints

If a link is inserted between node u and node w, let us first look at its impact on skew between u and w. If the delay from the source to u and w are t_u and t_w, respectively, the skew between them is $q_{u,w} = t_u - t_w$. According to Equation(2) and Equation(3), the skew $\hat{q}_{u,w}$ after the link insertion is:

$$\hat{q}_{u,w} = \frac{R_l}{R_l + r_u - r_w}(q_{u,w} + \frac{C_l}{2}(R_{u,u} - R_{w,w})) \tag{4}$$

[1]Another link insertion method was reported in [33] primarily for delay minimization.

The effect of the link capacitance C_l and the link resistance R_l can be separated. The link capacitance often changes the skew value as the value of $R_{u,u}$ is often different from $R_{w,w}$.

The effect of the link resistance R_l can be found by neglecting C_l and the following equation.

$$\hat{q}_{u,w} = \frac{R_l}{R_l + r_u - r_w} q_{u,w} \tag{5}$$

Thus, the effect of R_l depends on the value of $q_{u,w}$. A case of our special interest is when the nominal skew between u and w is zero. In this case, R_l does not affect the nominal skew and $q_{u,w}$ may represent the unwanted skew due to variations. Then, we can reach the following useful conclusions.

Lemma 1: *If two distinctive nodes in an RC network have zero nominal skew between them, inserting a resistor between them can reduce their skew variability.*

Proof: When a resistor R_l is inserted between two distinctive nodes u and w which have zero nominal skew between them, the skew variation $q_{u,w}$ is scaled by $\frac{R_l}{R_l+r_u-r_w}$ according to Equation(5). Since r_u is obtained by computing the Elmore delay when $C_u = 1, C_w = -1$ and the other node capacitors are zero, $r_u = R_{u,u} - R_{u,w}$. According to the computation of the transfer resistance, $R_{u,u} \geq R_{u,w}$ and $r_u \geq 0$. Similarly, $r_w = R_{u,w} - R_{w,w} \leq 0$. Since u and w are two distinctive nodes with zero nominal skew, neither $r_u = 0$ nor $r_w = 0$ may occur. Thus, $r_u - r_w > 0$ is true and $|\hat{q}_{u,w}| < |q_{u,w}|$ is true. □

Lemma 2: *In a clock tree, considering that a resistor R_l is inserted between two disjoint paths $p \rightsquigarrow b$ and $p \rightsquigarrow r$, where b and r are two leaf nodes and p is their nearest common ancestor node, and link endpoints $u \in p \rightsquigarrow r$ and $w \in p \rightsquigarrow b$ always have zero nominal skew, the variability of skew $q_{r,b}$ becomes smaller when the resistor is moved from p toward leaf nodes b and r.*

Proof: The resistor insertion is illustrated in Figure 2. The variation of skew between r and b after inserting resistor R_l between u and w can be expressed as:

$$\hat{q}_{r,b} = \frac{R_l}{R_l + r_u - r_w} q_{u,w} + q_{ur,wb} \tag{6}$$

where $q_{ur,wb} = t_{u,r} - t_{w,b}$ is the difference between delays of path $u \rightsquigarrow r$ and path $w \rightsquigarrow b$. Since the RC network is a tree before inserting the resistance, r_u and $-r_w$ are the total resistance along path $p \rightsquigarrow u$ and $p \rightsquigarrow w$, respectively. Thus $R_{loop} = R_l + r_u - r_w$ is the total resistance along the loop of $p \rightsquigarrow u \rightsquigarrow w \rightsquigarrow p$. Original variation of skew between r and b is $q_{r,b} = q_{u,w} + q_{ur,wb}$. The effect of inserting the resistance is to scale the $q_{u,w}$ by $\frac{R_l}{R_{loop}} < 1$. When the resistor is moved from the nearest common ancestor p toward r and b, $\frac{R_l}{R_{loop}}$ becomes smaller and greater portion of $q_{r,b}$ is scaled down. □

Lemma 1 and Lemma 2 are based on Elmore delay model which is often criticized for its inaccuracy and particularly neglection on the resistive shielding effect [36]. However, the resistive shielding effect is prominent only at nodes close to the source while clock sinks are generally far away from the source node, hence, the inaccuracy of Elmore delay at clock sinks is usually insignificant. A SPICE simulation is performed on a simple case to verify Lemma 2 and demonstrate the fidelity of Elmore delay model. In this simple case, two sinks are driven by a source directly. The two sinks have the same load capacitance and the same distance of $100\mu m$ from the source. We let the driver resistance, wire width and the sink capacitance have $\pm 15\%$ variation following a normal distribution. We test skew variation between two sinks when a link is inserted between two source-sink paths at $20\mu m$, $40\mu m$, $60\mu m$, $80\mu m$ and $100\mu m$ away from the source. The size of the link is constant in each test. The worst case skew and the standard deviation of skew variation from both SPICE model and the Elmore delay model are plotted in Figure 3. This plot shows that there

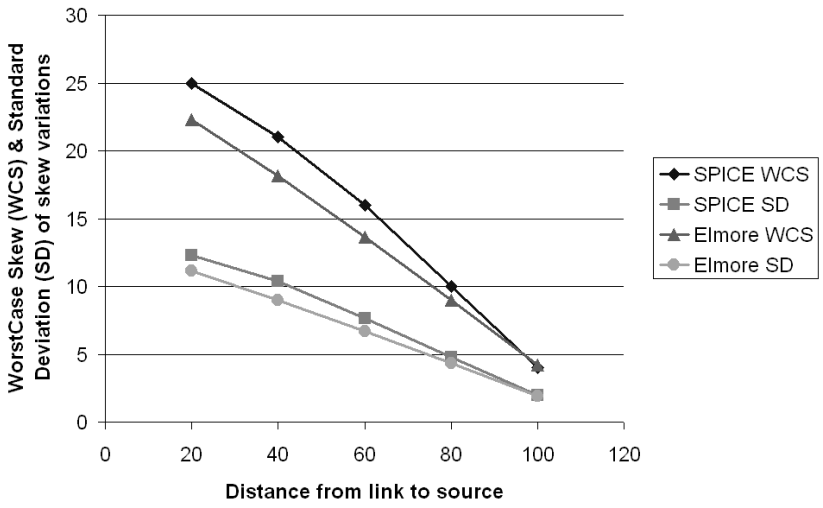

Figure 3: Skew variation vs. link position from both SPICE and Elmore delay model.

is strong correlation between SPICE results and Elmore delay results. In addition, the SPICE simulation results support the conclusion of Lemma 2.

2.3 Skew Variability Between Any Equal Delay Nodes

From Equation (2) and (3), the skew between two arbitrary node i and j after inserting link between u and w is:

$$\hat{q}_{i,j} = q_{i,j} + \frac{C_l}{2}(R_{i,u} + R_{i,w} - R_{j,u} - R_{j,w}) - \frac{\hat{q}_{u,w}}{R_l}(r_i - r_j) \tag{7}$$

Evidently, the link capacitance C_l usually changes the nominal skew. If only the link resistance R_l is considered, the skew becomes:

$$\hat{q}_{i,j} = q_{i,j} - \frac{r_i - r_j}{R_l + r_u - r_w} q_{u,w} \tag{8}$$

We consider the case where the nominal skew is zero between u and w as well as i and j. In this case, R_l does not affect the nominal skew between i and j. Further, both $q_{u,w}$ and $q_{i,j}$ can be treated as skew variations. Then, Equation(8) can be interpreted as that $q_{u,w}$ is scaled by $\frac{r_i - r_j}{R_l + r_u - r_w}$ and added to $q_{i,j}$. Whether the magnitude of $q_{i,j}$ is reduced depends on the signs of $q_{u,w}$, $q_{i,j}$ and $r_i - r_j$.

If we consider in the context of inserting links in a clock tree $T = (V, T_E)$, the node u is in a subtree $T_f \subset T$ and the node w is in another subtree $T_g \subset T$ as shown in Figure 2. The root node of T_f and T_g are the two child nodes of the nearest common ancestor node p between u and w. The effect of R_l on $q_{i,j}$ depends on the locations of i and j in T and can be analyzed in three scenarios as follows.

Scenario 1: One of i and j is in subtree T_f and the other is in subtree T_g. Without loss of generality, we can assume $i \in T_f$ and $j \in T_g$. Since i and u are in the same subtree, their delay t_i and t_u have certain correlation with respect to t_j or t_w. Similarly, delay t_j is more correlated with t_w than with t_i or t_u. Therefore, there is certain correlation between $q_{i,j}$ and $q_{u,w}$. If the correlation is sufficiently strong, $q_{i,j}$ and $q_{u,w}$ usually have the same sign. As $r_u - r_w \geq r_i - r_j \geq 0$ (proof is

similar to Lemma 1), the link resistance may reduce the variability of skew between i and j. In a special case, when i is u and j is w, i.e., $q_{i,j}$ and $q_{u,w}$ have perfect correlation, the link resistance always reduces skew variability as stated in Lemma 1.

Scenario 2: Both i and j are in the same subtree T_f or T_g. In this scenario, r_i and r_j have same sign and $\frac{r_i - r_j}{R_l + r_u - r_w}$ is generally smaller than 0.5. Thus, the skew variation between i and j is increased or decreased by a small fraction of $|q_{u,w}|$. Since i and j are in the same subtree, their skew variation in original tree is usually not large either.

Scenario 3: One of i and j is in the subnetwork G_p rooted at the nearest common ancestor node p and the other node is disjoint with G_p. For example, i is in G_p like b and j is not in G_p like d in Figure 2. If node j is disjoint with G_p, there is not any overlap between any part of source-to-j path and G_p. Hence, $r_j = 0$ and there is no predictable correlation between $q_{i,j}$ and $q_{u,w}$. In the worst case of signs, $q_{i,j}$ has sign opposite to $q_{u,w} r_i$. Since the nominal skew between u and w is zero, $r_u \approx -r_w \approx |r_i|$ in general. Therefore, approximately speaking, R_l may increase or decrease the skew variation between i and j by a half of the skew variation between u and w.

In summary, the link resistance R_l usually reduces skew variability in Scenario 1, may increase skew variability a little in Scenario 2, and may increase skew variability more in Scenario 3.

3 Link Insertion Based Non-tree Clock Routing

3.1 Algorithm Overview

Our objective is to design a clock routing algorithm that can achieve low skew variability and low wire consumptions. Based on the analysis in Section 2, we propose to construct the clock network by inserting cross links in a clock tree. In this incremental approach, many existing clock tree routing algorithms [6–8] can be utilized and the non-tree clock routing problem becomes more manageable.

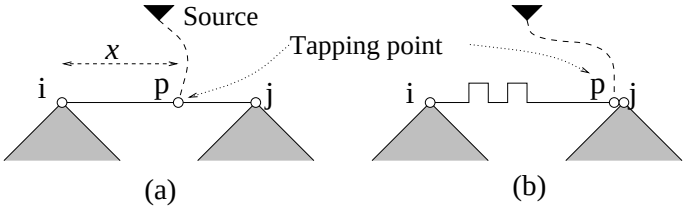

Figure 4: Tune location x of tapping point p such that nominal skew between sinks in subtree T_i and T_j is same as specifications. If there is great imbalance between T_i and T_j, wire snaking may be necessary as in (b).

Each link insertion can be decomposed to adding link capacitance to its endpoints and inserting link resistance. The analysis in Section 2 tells that the nominal skews may be affected by a link capacitance. In other words, after adding link capacitances, the nominal skew may be different from the desired value. The nominal skew change can be removed by tuning tapping points as in [4]. An example of the tuning is shown in Figure 4. Even though the location of the tapping point is determined based on Elmore delay model in [4], the basic idea in [4] can be applied on any delay model. As long as a delay model is monotone, the location of the tapping point can be found through a binary search. After tuning tapping points, the link resistance can be inserted. According to the analysis in Section 2, link resistance does not affect nominal skew when its two endpoints have zero nominal skew before link insertion.

If the cross links are inserted one after another, a tapping point may need to be tuned multiple times as a link inserted later may ruin the nominal skew result of previous tunings. Moreover, links inserted earlier change the initial tree to a non-tree and make subsequent delay computations more difficult. In order to avoid the inefficiency of one-by-one link insertion, we add link capacitance to all selected node pairs simultaneously and perform tuning only once at each tapping point before inserting link resistances. The tunings proceed in a bottom-up traversal of the clock tree. During the traversal, when a tapping point is encountered, the tuning is performed in the same way as in [4]. After the tunings are completed, the clock tree should provide skews same as those in the initial tree.

Our algorithm flow can be summarized as:

1. Input initial clock tree.
2. Select node pairs where cross links will be inserted.
3. Add link capacitance to the selected nodes.
4. Tune tapping points to restore original skew.
5. Insert link resistance to the selected node pairs.

3.2 Selecting Node Pairs for Link Insertion

In the algorithm flow, the major problem is how to choose the node pairs for link insertions. We always choose node pairs with zero nominal skew so that no nominal skew is affected by the link resistance. We also prefer node pairs close to each other so that the link capacitance C_l is small and less wire snakings may happen in tuning tapping points. Based on the analysis in Section 2, we investigate a rule based selection scheme and a minimum weight matching based selection scheme.

3.2.1 Rule Based Selection Scheme

The rule based approach is derived directly from Equation(refequUW) and Lemma 2 in Section 2.2. The conclusions in Section 2.3 are less rigorous than those in Section 2.2 and hard to be translated to clear rules. Lemma 2 indicates that the skew variability reduction for a pair of sinks is more effective when the link is closer to the sinks. Therefore, we restrain the node pairs to be sink pairs for zero skew routing.

α **rule**: In the initial tree, $R_{loop} = R_l + r_u - r_w$ is the total resistance along the loop of $p \rightsquigarrow u \rightsquigarrow w \rightsquigarrow p$. According to Equation(4), the link resistance is more effective when the ratio $\alpha = \frac{R_l}{R_{loop}}$ is smaller. Thus, the first rule is the ratio α is no greater than certain upper bound α_{max}.

β **rule**: Based on Equation(4), the impact of the link capacitance can be reduced when $\beta = |\frac{C_l}{2}(R_{u,u} - R_{w,w})|$ is small or no greater than certain bound β_{max}. In addition to restraining the link size, this rule is in favor of node pairs have similar path length from the source.

γ **rule**: The nearest common ancestor(NCA) node for a sink pair has certain depth in the original tree. For the example in Figure 2, the NCA p of r and b has depth 2. We call this depth as the level of the node pair and denote it as γ. Lemma 2 implies that the value of γ needs to be small or no greater than a bound γ_{max}.

It can be observed that there is redundancy in the three rules, but according to our experimental experiences, all three rules are necessary in general. The rule based node pair selection scheme is simply choosing all *sink* pairs satisfying all three rules. The advantage of this scheme is its simplicity. Its weakness is the neglection on the effects discussed in Section 2.3 which are considered in the minimum weight matching based algorithm.

3.2.2 Minimum Weight Matching Based Selection Algorithm

According to Lemma 1, when a link resistance is inserted between a pair of nodes, it always reduces skew variation between them. However, the effect of this link on skew of other node pairs is very subtle.

According to the analysis in Section 2.3, *scenario 3* needs to be avoided, since a link between u and w in a subtree T_p may hurt the variability of skew between a sink in T_p and a sink outside T_p. Scenario 3 can be avoided by choosing node pairs between left child subtree and right child subtree of a node of depth 1. For example, links between subtree T_p and subtree T_d of depth 1 node h in Figure 2 can avoid scenario 3, since there is no sinks outside of T_h. Node pairs for these links can be characterized by the depth of their nearest common ancestor node h, which can be called γ level by using the term in the γ rule of previous section. Therefore, we need to have node pairs with $\gamma = 1$.

Links inserted between subtree T_d and subtree T_p often improve skew variability between sinks between T_d and T_p according to analysis of *scenario 1*. However, these links may increase skew variability between sinks within T_d or T_p as discussed in *scenario 2*. These degradation can be compensated if links are inserted between sub-subtrees within subtree T_d or T_p. In other words, node pairs of $\gamma = 2$ need to be considered. This procedure can be repeated recursively till γ is sufficiently large. The subtrees corresponding to large γ are mostly small and the skew variability inside is usually insignificant. The main algorithm description on this recursive procedure is given in Figure 5.

Procedure: $SelectNodePairs(T_v)$
Input: Subtree T_v rooted at node v **Output:** Node pair set P
1. $l \leftarrow$ left child node of v 2. $r \leftarrow$ right child node of v 3. $P \leftarrow PairBetweenTrees(T_l, T_r)$ 4. If $Depth(v) == DepthLimit$, return P 5. $P \leftarrow P \cup SelectNodePairs(T_l)$ 6. $P \leftarrow P \cup SelectNodePairs(T_r)$ 7. Return P

Figure 5: Main algorithm of selecting node pairs for link insertion.

A subproblem yet to be solved is how to select node pairs between two subtrees T_l and T_r. This subproblem is the subroutine $PairBetweenTrees(T_l, T_r)$ in line 3 of Figure 5. We decompose each subtree into k sub-subtrees and select k node pairs between sub-subtrees in T_l and sub-subtrees in T_r. This problem can be modeled in a bipartite graph and solved by the minimum weight matching algorithm [37]. If the subtree T_l is decomposed to sub-subtree set $S_l = \{T_{l1}, T_{l2}, ...T_{lk}\}$ and T_r is decomposed to $S_r = \{T_{r1}, T_{r2}, ...T_{rk}\}$, each node in the bipartite graph corresponds to a sub-subtree. There is an edge between every node in S_l and every node in S_r. The edge weight between a sub-subtree pair T_{li} and T_{rj} is the Manhattan distance between the nearest sink pair $u \in T_{li}$ and $w \in T_{rj}$. An example of the bipartite graph with $k = 4$ is shown in Figure 6. The minimum weight matching result may select a set of node pairs between S_l and S_r with the minimum total link length. The rationale behind this scheme is to distribute the links evenly in a subtree such that the *scenario 2* effect becomes less. The algorithm description on selecting node pairs between two subtrees is give in Figure 7.

A demonstration of the node pair selection algorithm is shown in Figure 8 with $DepthLimit = 2$

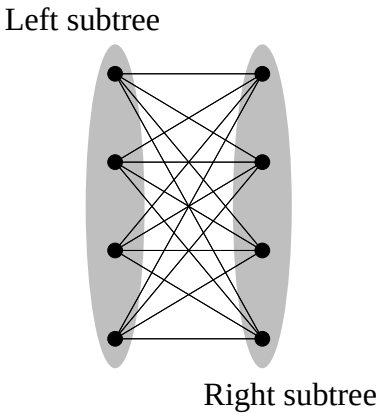

Figure 6: A bipartite graph model for selecting 4 node pairs between two subtrees. Each node corresponds to a sub-subtree in a subtree. An edge weight is the shortest Manhattan distance between leaf(sink) nodes of two sub-subtrees.

Subroutine: $PairBetweenTrees(T_l, T_r)$
Input: Two subtree T_l and T_r **Output:** Node pair set P
S1. Decompose T_l to sub-subtrees $S_l = \{T_{l1}, T_{l2}...T_{lk}\}$ S2. Decompose T_r to sub-subtrees $S_r = \{T_{r1}, T_{r2}...T_{rk}\}$ S3. For every pair (T_{li}, T_{rj}) between S_l and S_r S4. $W(T_{li}, T_{rj}) \leftarrow$ min dist between leaf in T_{li} and T_{rj} S5. Find min weighted matching between S_l and S_r S6. $P \leftarrow k$ leaf pairs in min weighted matching S7. Return P

Figure 7: Algorithm of selecting node pairs between two subtrees. $W(T_{li}, T_{rj})$ represents the edge weight between sub-subtree pair (T_{li}, T_{rj}).

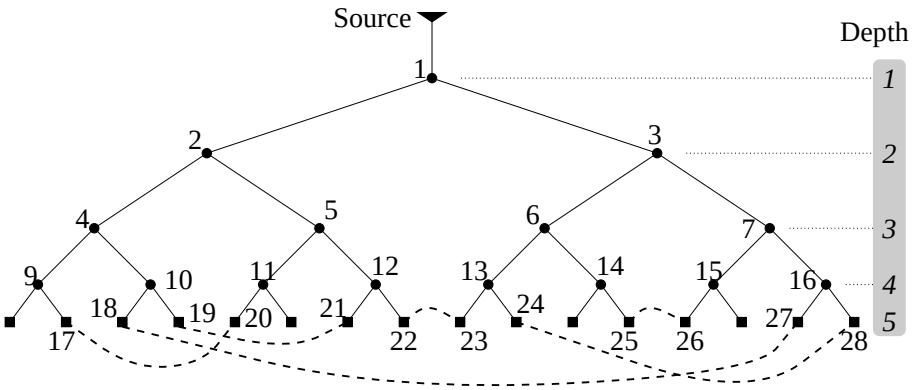

Figure 8: Algorithm example. The dashed curves indicate cross links.

and $k = 2$. The algorithm starts with subtree T_1 rooted at node 1. Two node pairs are selected between its child subtree T_2 and T_3. Since $k = 2$, T_2 is decomposed to T_4 and T_5, and T_3 is decomposed to T_6 and T_7. By running the minimum matching algorithm, node pair $(22, 23)$ and $(18, 27)$ are selected. The same procedure is applied to T_2 and results in selection of $(17, 20)$ and $(19, 21)$. Similarly, $(25, 26)$ and $(24, 28)$ are selected for subtree T_3.

3.3 Non-zero Skew Routing

The link insertion based non-tree clock routing method can be easily extended to achieve non-zero skews. Node pairs can be selected same as in Figure 5. If a sink pair (a, c) have non-zero nominal skew $q_{a,c} = t_a - t_c > 0$, a link can be inserted between sink c and a point k in the parent edge of a such that nominal delay $t_c = t_k$. This is illustrated in Figure 2.

4 Experimental Results

The experiments are performed on a Linux machine with dual $1GHz$ AMD microprocessor and 512M memory. The benchmark circuits are $r1 - r5$ downloaded from GSRC Bookshelf ($http://vlsicad.ucsd.edu/GSRC/bookshelf/Slots/BST/$). The variation factors considered in the experiments include the clock driver resistance, wire width and each sink load capacitance. We assume that all these factors have variations following a normal distribution with standard deviation σ equal to 5% of the nominal values. In the experiment, the skew variations and wirelength are compared among clock trees, tree+links and meshes. For each clock network, a Monte Carlo simulation of 1000 trials is performed to obtain the maximum skew variation (MSV) and the standard deviation (SD) of skew variations. A skew variation is the maximum sink delay minus the minimum sink delay in a trial.

The clock trees are obtained by running the BST [38] code which is also downloaded from the same website of GSRC Bookshelf. When running the BST node, the global skew bound is set to 0 so that zero skew clock trees are obtained. The size of benchmark circuits, skew variations and wirelength of clock trees are given in Table 1.

Table 1: Maximum skew variation (MSV), standard deviation (SD) and total wirelength of trees. The CPU time is from running BST code.

Testcase	# sinks	MSV	SD	wirelen	CPU(s)
r1	267	0.265	0.042	1320665	1
r2	598	0.759	0.112	2602908	3
r3	862	0.934	0.166	3388951	4
r4	1903	2.321	0.317	6828510	12
r5	3101	5.792	1.149	10242660	18

We implemented the proposed link insertion based methods and leaf level mesh methods to construct non-tree networks including four variants:

- Link-R: The proposed link insertion based non-tree, with *rule based* node pair selection.
- Link-M: The proposed link insertion based non-tree, with the *minimum weight matching based* node pair selection.
- Mesh-S: A *sparse* leaf level mesh driven by an H-tree.
- Mesh-D: A *dense* leaf level mesh driven by an H-tree.

The experimental results on these four variants are shown in Table 2. The value of the maximum skew variation(MSV), standard deviation(SD) and wirelength are expressed as the ratio with respect to the results of clock trees. In Table 2, the size for a link method refers to the number of links inserted and the size for a mesh indicates the number of rows and columns.

In *Link-R*, the rules are $\alpha_{max} = 0.1$, $\beta_{max} = 10$ for $r1 - r3$, $\beta_{max} = 50$ for r4 and r5, and $\gamma_{max} = 1$. These rules are chosen empirically as they yield relatively low varition results. The α rule is a ratio regardless net size so that a consistent α rule can be applied to all testcases. In contrast, the β rule depends on net size and need to be increase when the net size is greater. The rule of $\gamma_{max} = 1$ results in *scenario 2* that may aggravate the variations as analyzed in Section 2.3. In practice, it turns out that the *scenario 2* effect is minor, especially when the total number of links is not small. According to our experience, all three rules are necessary to generate a low variation result.

In *Link-M*, $k = 2$ for $\gamma = 1$ for $r1$ and $r2$, i.e., 2 links are inserted at the level of $\gamma = 1$. Since r3 is larger, $k = 4$ at $gamma = 1$ is applied. Testcase of r4 and r5 are the largest, hence, we insert links with $k = 2$ at level $\gamma = 2$ in addition to 4 links at $\gamma = 1$. When the size of a net becomes larger, the levels of the clock tree increases and deeper level (or greater γ) should be allowed. Smaller γ implies greater subtrees, therefore more links needs to be inserted at a smaller γ level in a tree.

Table 2: Skew variations and wirelength in term of tree results. Size of a tree+link network is the number of links. Size of a mesh is $\#rows \times \#columns$.

Case	Method	size	MSV	SD	wirelen	CPU(s)
r1	Link-R	6	0.65	0.97	1.053	0.039
	Link-M	2	0.60	0.80	1.009	0.068
	Mesh-S	11×11	0.92	0.82	1.85	0.045
	Mesh-D	21×21	0.72	0.62	2.51	0.045
r2	Link-R	20	0.72	0.90	1.027	0.087
	Link-M	2	0.68	0.84	1.009	0.098
	Mesh-S	15×15	0.80	0.80	1.87	0.046
	Mesh-D	29×29	0.59	0.47	2.43	0.046
r3	Link-R	29	0.76	0.88	1.074	0.13
	Link-M	4	0.64	0.83	1.017	0.18
	Mesh-S	19×21	0.24	0.35	1.76	0.046
	Mesh-D	35×37	0.14	0.19	2.50	0.046
r4	Link-R	19	0.53	0.35	1.013	0.38
	Link-M	6	0.46	0.35	1.008	0.43
	Mesh-S	27×29	0.23	0.34	1.71	0.048
	Mesh-D	55×57	0.09	0.18	2.33	0.048
r5	Link-R	57	0.31	0.15	1.030	0.53
	Link-M	6	0.26	0.14	1.008	0.52
	Mesh-S	37×39	0.08	0.10	1.64	0.051
	Mesh-D	75×77	0.03	0.06	2.33	0.051

The observations from Table 2 include:

- The minimum weight matching based link method is always superior to the rule based method on both variability and wirelength. For this reason, we skip results of rule based method with other rule parameters.
- A dense mesh always yields less variations than a sparse mesh, but consumes more wirelength as expected.

- All four variants work better on larger nets than on smaller nets. For a large net, a tree is dense in term of the number of wire segments, therefore redundant signal propagation paths can be established with less efforts. In other words, a link of same size has more chance to short two tree segments in a denser tree (or larger net) than in a sparser tree (or smaller net).

- Except for $r1$, a mesh usually can provide lower skew variability than a link based non-tree. However, the wire increase of a mesh is much greater than a link based non-tree. For all solutions from *Link-M*, the wirelength increase over a tree is never greater than 2%.

- The method of *Link-M* results in $32\% - 74\%$ reduction on the maximum skew variation, and $10\% - 86\%$ reduction on the standard deviation except for $r1$. Considering less than 2% wire increase, such significant skew variability reductions indicate great wire usage efficiency.

The CPU time in seconds are displayed in the rightmost column in Table 2. The CPU time for link insertion includes the time for node pair selection and tuning tapping points. Even though the CPU time of link insertion is usually greater than constructing a mesh, it is still trivial, especially compared with the time of clock tree construction.

An experiment on non-zero skew routing is performed on $r1$. The result shows that *Link-M* reduces standard deviation by 11% over tree with 2% increase on wirelength. A dense mesh reduces standard deviation by 15% with 179% increase on wirelength.

5 Conclusion and Future Work

In this paper, a low cost non-tree clock routing method is proposed to reduce skew variability. The non-tree network is obtained by inserting cross links in a given clock tree. The effect of link insertion on skew variation is analyzed. Based on the analysis, link insertion algorithms are developed. Experimental results show that this method can reduce skew variations remarkably with little extra wire resource. This method can be applied to achieve low variation non-zero skew as well.

This work is one step further toward harnessing the challenging skew variation problem. Its accuracy can be improved by adopting a higher order delay model. The efficiency of link insertion can be further improved by considering skew permissible ranges [39] so that links are inserted only between nodes with tight permissible ranges. These potential improvements will be explored in our future research.

References

[1] E. G. Friedman. Clock distribution networks in synchronous digital integrated circuits. *Proceedings of the IEEE*, 89(5):665–692, May 2001.

[2] L. Benini, P. Vuillod, A. Bogliolo, and G. De Micheli. Clock skew optimization for peak current reduction. *Journal of VLSI Signal Processing*, 16(2/3):117–130, Jun./Jul. 1997.

[3] W.-C. D. Lam, C.-K. Koh, and C.-W. A. Tsao. Power supply noise suppression via clock skew scheduling. In *Proceedings of the IEEE International Symposium on Quality Electronic Design*, pages 355–360, 2002.

[4] R.-S. Tsay. Exact zero skew. In *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*, pages 336–339, 1991.

[5] J. P. Fishburn. Clock skew optimization. *IEEE Transactions on Computers*, C-39:945–951, July 1990.

[6] T.-H. Chao, Y.-C. Hsu, J.-M. Ho, K. D. Boese, and A. B. Kahng. Zero skew clock routing with minimum wirelength. *IEEE Transactions on Circuits and Systems - Analog and Digital Signal Processing*, 39(11):799–814, November 1992.

[7] M. Edahiro. A clustering-based optimization algorithm in zero-skew routings. In *Proceedings of the ACM/IEEE Design Automation Conference*, pages 612–616, 1993.

[8] C.-W. A. Tsao and C.-K. Koh. UST/DME: a clock tree router for general skew constraints. In *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*, pages 400–405, 2000.

[9] R. Chaturvedi and J. Hu. A simple yet effective merging scheme for prescribed-skew clock routing. In *Proceedings of the IEEE International Conference on Computer Design*, pages 282–287, 2003.

[10] S. Zanella, A. Nardi, A. Neviani, M. Quarantelli, S. Saxena, and C. Guardiani. Analysis of the impact of process variations on clock skew. *IEEE Transactions on Semiconductor Manufacturing*, 13(4):401–407, November 2000.

[11] Y. Liu, S. R. Nassif, L. T. Pileggi, and A. J. Strojwas. Impact of interconnect variations on the clock skew of a gigahertz microprocessor. In *Proceedings of the ACM/IEEE Design Automation Conference*, pages 168–171, 2000.

[12] R. Saleh, S. Z. Hussain, S. Rochel, and D. Overhauser. Clock skew verification in the presence of IR-drop in the power distribution network. *IEEE Transactions on Computer-Aided Design*, 19(6):635–644, June 2000.

[13] A. H. Ajami, K. Banerjee, M. Pedram, and L. P.P.P. van Ginneken. Analysis of non-uniform temperature-dependent interconnect performance in high performance ICs. In *Proceedings of the ACM/IEEE Design Automation Conference*, pages 567–572, 2001.

[14] S. Pullela, N. Menezes, and L. T. Pillage. Reliable non-zero skew clock trees using wire width optimization. In *Proceedings of the ACM/IEEE Design Automation Conference*, pages 165–170, 1993.

[15] S. Pullela, N. Menezes, J. Omar, and L. T. Pillage. Skew and delay optimization for reliable buffered clock trees. In *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*, pages 556–562, 1993.

[16] J. Chung and C.-K. Cheng. Skew sensitivity minimization of buffered clock tree. In *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*, pages 280–283, 1994.

[17] S. Lin and C. K. Wong. Process-variation-tolerant clock skew minimization. In *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*, pages 284–288, 1994.

[18] J. G. Xi and W. W.-M. Dai. Buffer insertion and sizing under process variations for low power clock distribution. In *Proceedings of the ACM/IEEE Design Automation Conference*, pages 491–496, 1995.

[19] M. P. Desai, R. Cvijetic, and J. Jensen. Sizing of clock distribution networks for high performance CPU chips. In *Proceedings of the ACM/IEEE Design Automation Conference*, pages 389–394, 1996.

[20] J. G. Xi and W. W.-M. Dai. Jitter-tolerant clock routing in two-phase synchronous systems. In *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*, pages 316–320, 1996.

[21] H.-Y. Hsieh, W. Liu, P. Franzon, and R. Cavin III. Clocking optimization and distribution in digital systems with scheduled skews. *Journal of VLSI Signal Processing*, 16(2/3):131–147, Jun./Jul. 1997.

[22] M. Nekili, G. Bois, and Y. Savaria. Pipelined H-tree for high-speed clocking of large integrated systems in presence of process variations. *IEEE Transactions on VLSI Systems*, 5(2):161–174, Jun 1997.

[23] P. J. Restle, T. G. McNamara, D. A. Webber, P. J. Camporese, K. F. Eng, K. A. Jenkins, D. H. Allen, M. J. Rohn, M. P. Quaranta, D. W. Boerstler, C. J. Alpert, C. A. Carter, R. N. Bailey, J. G. Petrovick, B. L. Krauter, and B. D. McCredie. A clock distribution network for microprocessors. *IEEE Journal of Solid-State Circuits*, 36(5):792–799, May 2001.

[24] N. A. Kurd, J. S. Barkatullah, R. O. Dizon, T. D. Fletcher, and P. D. Madland. A multigigahertz clocking scheme for the Pentium 4 microprocessor. *IEEE Journal of Solid-State Circuits*, 36(11):1647–1653, November 2001.

[25] H. Su and S. S. Sapatnekar. Hybrid structured clock network construction. In *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*, pages 333–336, 2001.

[26] B. Lu, J. Hu, G. Ellis, and H. Su. Process variation aware clock tree routing. In *Proceedings of the ACM International Symposium on Physical Design*, pages 174–181, 2003.

[27] J. L. Neves and E. G. Friedman. Topological design of clock distribution networks based on non-zero clock skew specifications. In *Proceedings of the Midwest Symposium on Circuits and Systems*, pages 468–471, 1993.

[28] A. Takahashi, K. Inoue, and Y. Kajitani. Clock-tree routing realizing a clock-schedule for semi-synchronous circuits. In *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*, pages 260–265, 1997.

[29] J. G. Xi and W. W.-M. Dai. Useful-skew clock routing with gate sizing for low power design. *Journal of VLSI Signal Processing*, 16(2/3):163–179, Jun./Jul. 1997.

[30] C. Albrecht, B. Korte, J. Schietke, and J. Vygen. Cycle time and slack optimization for VLSI-chips. In *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*, pages 232–238, 1999.

[31] K. D. Boese, A. B. Kahng, B. A. McCoy, and G. Robins. Near-optimal critical sink routing tree constructions. *IEEE Transactions on Computer-Aided Design*, 14(12):1417–36, December 1995.

[32] S. S. Sapatnekar and S. M. Kang. *Design automation for timing-driven layout synthesis*. Kluwer Academic Publishers, Boston, MA, 1993.

[33] T. Xue and E. S. Kuh. Post routing performance optimization via multi-link insertion and non-uniform wiresizing. In *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*, pages 575–580, 1995.

[34] P. K. Chan and K. Karplus. Computing signal delay in general RC networks by tree/link partitioning. *IEEE Transactions on Computer-Aided Design*, 9(8):898–902, August 1990.

[35] L. T. Pillage, R. A. Rohrer, and C. Visweswariah. *Electronic circuit and system simulation methods*. McGraw-Hill, Inc., 1995.

[36] J. Qian, S. Pullela, and L. T. Pillage. Modeling the effective capacitance for the RC interconnect of CMOS gates. *IEEE Transactions on Computer-Aided Design*, 13(12):1526–1535, December 1994.

[37] C. H. Papadimitriou and K. Steiglitz. *Combinatorial optimization: algorithms and complexity*. Dover Publications, Inc., Mineola, NY, 1998.

[38] J. Cong, A. B. Kahng, C.-K. Koh, and C.-W. A. Tsao. Bounded-skew clock and Steiner routing. *ACM Transactions on Design Automation of Electronic Systems*, 3(3):341–388, July 1998.

[39] J. L. Neves and E. G. Friedman. Optimal clock skew scheduling tolerant to process variations. In *Proceedings of the ACM/IEEE Design Automation Conference*, pages 623–628, 1996.