# Improved Algorithms for Link-Based Non-Tree Clock Networks for Skew Variability Reduction *

Anand Rajaram ‡ †, David Z. Pan ‡, Jiang Hu ¶
‡ Dept. of ECE, University of Texas, Austin, TX 78712
† Dallas DSP Design, Texas Instruments Inc., Dallas, TX 75243
¶ Dept. of EE, Texas A&M University, College Station, TX 77843
anandr@cerc.utexas.edu, dpan@ece.utexas.edu, jianghu@ee.tamu.edu

## ABSTRACT

In the nanometer VLSI technology, the variation effects like manufacturing variation, power supply noise, temperature etc. become very significant. As one of the most vital nets in any synchronous VLSI chip, the Clock Distribution Network (CDN) is especially sensitive to these variations. Recently proposed link-based non-tree [1] addresses this problem by constructing a non-tree that is significantly more tolerant to variations when compared to a clock tree. Although the two algorithms proposed in [1] are effective in reducing the skew variability, they have a few drawbacks including high complexity, lengthy links and uneven link distribution across the clock network. In this paper, we propose two new algorithms that can overcome these disadvantages. The effectiveness of the proposed algorithms has been validated using HSPICE based Monte Carlo simulations. Experimental results show that the new algorithms are able to achieve the same or better skew reduction with an average of 5% wire length increase when compared to the 15% wire length increase of the existing algorithms in [1]. Moreover, the new algorithms scale extremely well to big clock networks, i.e., the bigger the clock network, the less overall link cost (less than 2% for the biggest benchmark we have).

## Keywords

VLSI CAD, Physical Design, Clock Network, Non-tree Clocks

## General Terms

Algorithms, Performance

## Categories and Subject Descriptors

B.7.2 [**INTEGRATED CIRCUITS**]: Design Aids

*This work is partially supported by IBM Faculty Award and SRC under contract 2004-TJ-1205.



## 1. INTRODUCTION

As the VLSI feature size shrinks into the sub-100 nm dimensions, previously negligible variation effects start to drastically affect circuit performance and yield. Effects such as process [2], temperature and power supply voltage [3] variations are becoming more significant as we move from one technology node to the next. Since the CDN is among the largest nets in a chip and at the same time it is the most frequently switching net, any unwanted variation in the clock skew might have catastrophic effect on both energy efficiency and power/ground noise [4]. To address this problem, several solutions like variation aware clock tree routing [5], buffer/wire sizing [6, 7], non-tree clock distribution have been proposed. The non-tree routing is usually the most effective among these methods because of the existence of multiple paths from clock source to the sinks, which makes the delays in the sinks correlated, resulting in reduced skew variation.

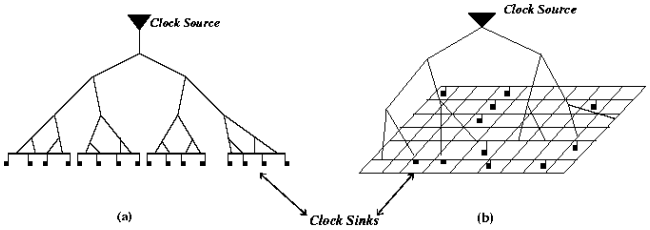


**Figure 1: Non-tree clock networks: (a) one dimensional; (b) leaf level mesh.**

Several types of non-tree CND methodologies have been proposed in literature [1, 8–12]. These non-tree CDNs can be broadly classified as 1-dimensional [8, 9] or 2-dimensional structures [10–12]. In a 1-dimensional approach, several clock sinks are usually attached to a single (usually thick) piece of interconnect. This interconnect will be driven at multiple points from a binary tree and so the skew between any two points on it will be small. As a result, the skew between any two sinks attached to this interconnect will also be small. An example of 1-dimensional non-tree clock network is given in Figure 1(a). Variations of this approach have been used in [8] and in [9] employed in $Pentium^{TM} - 4$ microprocessor. A key limitation of the 1-dimensional structure is that it does not handle the skew variation between different 1-dimensional regions.

The 2-dimensional non-tree structure is also called mesh.

Depending of the location of the mesh w.r.t. the clock source and sinks, it can be further classified as a leaf level (mesh close to clock sinks) or top level (mesh close to the clock source) or a multi-level mesh (several meshes at different levels). In the leaf level mesh approach [10, 11], a metal wire mesh is overlaid on the entire chip area and driven at multiple points directly from clock source [10] or through a routing tree [11]. All the clock sinks are connected to the nearest point on the mesh. This is illustrated in Figure 1(b). The other types of clock meshes are the top level mesh approach [12], and the multi-level mesh approach [13]. The main difference between these methods and the leaf level mesh is the number and location(s) of the mesh(es). As a result, the wire length consumption and skew reduction also varies accordingly. Even though clock mesh has been shown to be highly effective for skew suppression in microprocessors, it consumes enormous wire resources and power. As a result, the use of clock mesh is restricted to high-end products only.

Recently, a link-based non-tree algorithm has been proposed in [1] in which cross links are added in an existing clock tree. By suitably choosing the correct locations of the links, it was shown that significant reduction in skew variability can be achieved with very small increase in wire length when compared to the original clock tree. An example of link-based non-tree is shown in Figure 2. The vital aspect of the link-based methodology is to determine the proper location of the cross links, for which two algorithms were proposed in [1]. While both algorithms are highly effective, they suffer from several drawbacks. Some of these are: high complexity, lengthy links that might cause routing problems and uneven distribution of links across all regions of the clock tree that might limit skew variability reduction.

In this paper, we introduce better algorithms in which the above drawbacks have been addressed. We validate the effectiveness of the new algorithms using HSPICE based Monte Carlo simulations. Experimental results show that the new algorithms are able to achieve the same or better skew reduction with an average of 5% wire length increase when compared to the 15% wire length increase of the existing algorithms.

## 2. REVIEW OF LINK INSERTION BASED NON-TREE CLOCK NETWORKS

In this section, we will briefly review the link-based non-trees proposed in [1]. First, we will review the effect of adding a link on delay and skew in a general RC network. Then we will overview the different scenarios that might result because of link insertion in a RC network. Throughout this paper, Elmore delay model is employed due to its high fidelity [14] and ease of computation. However, HSPICE based Monte Carlo simulations are used for final result validation unlike [1].

### 2.1 Effect of Link Insertion on Delay In RC Network

An iterative method for calculating Elmore delay for any non-tree RC network has been proposed in [15]. In this approach, the non-tree RC network is represented by a graph $G = (V, E)$ with the node set $V$ composed of the source, sinks and Steiner nodes and the edges $E$ composed of all the interconnects. The graph is divided into two parts: a spanning tree $T = (V, E_T)$ and a set of link edges $E_L$ such that $E = E_T \cup E_L$. The final delay of node $i$ is computed by first calculating the Elmore delay of the spanning tree $T = (V, E_T)$ and then incrementally adding the links and updating the delays. For example, for the RC network of Figure 2, the solid lines are the edges of $E_T$ and the dotted lines are the edges of $E_L$.

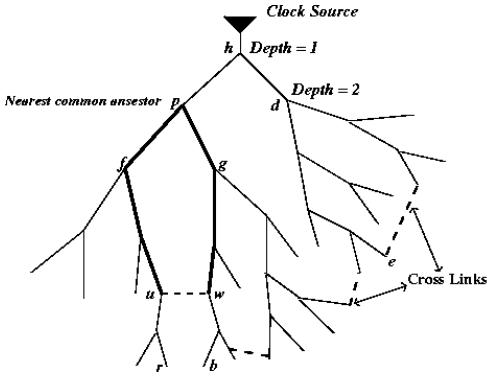


**Figure 2: An example of cross link based non-tree.**

From [1], we can get the effect of inserting a single cross link between node $u \in V$ and node $w \in V$. If the link has a wire resistance of $R_l$ and wire capacitance of $C_l$, the effect of inserting this link into the RC network 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 $w$. Since the addition of link capacitance does not change the network topology, its effect on delay can be easily estimated. If the Elmore delay from the source to any sink $i$ is $t_i$ before the link insertion, the delay $\tilde{t}_i$ after adding the link capacitors *only* is given by:

$$\tilde{t}_i = t_i + \frac{C_l}{2}(R_{i,u} + R_{i,w}) \quad (1)$$

where $R_{i,j}$ for nodes $i$ and $j$ is the transfer resistance between the nodes, which is equal to the voltage at node $i$ when $1A$ current is injected into node $j$ and all other node capacitors are zero [16]. The impact of the link resistance $R_l$ on delay can be obtained by using the technique of Chan and Karplus in [15]. According to [15], the final delay at node $i$ after considering the resistor $R_l$ of the link is given as:

$$\hat{t}_i = \tilde{t}_i - \frac{\tilde{t}_u - \tilde{t}_w}{R_l + r_u - r_w} r_i \quad (2)$$

where $r_i$,$r_u$ and $r_w$ are equal to the Elmore delay at $i$, $u$ and $w$ when $C_u = 1, C_w = -1$ and the other node capacitance are zero. Thus, using the equations (1) and (2), we can easily estimate the effect of inserting a link on delay at any node in a general RC network.

### 2.2 Effect of Link Insertion on Skew Variability

The effect of inserting a link on skew between any two arbitrary points has been analyzed in detail in [1]. In this section, we will review some of their important conclusions for the sake of clarity.

#### 2.2.1 *Skew Variability Between Link Endpoints*

If a link is inserted between nodes $u$ and $w$, then according to [1], the skew between $u$ and $w$ after link insertion is given by:

$$\hat{q}_{u,w} = \frac{R_l}{R_l + r_u - r_w}(q_{u,w} + \frac{C_l}{2}(R_{u,u} - R_{w,w})) \quad (3)$$

where, $q_{u,w} = t_u - t_w$ is the original skew between nodes $u$ and $w$, $\hat{q}_{u,w}$ the final skew after the link insertion, $C_l$ the link capacitance, $R_l$ the link resistance, $R_{u,u}$ and $R_{w,w}$ the transfer resistances of nodes $u$ and $w$. Since the effect of capacitance can be easily estimated and removed [1], equation (3) gets reduced to:

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

Thus, from equation (4), we can see that the final skew is a scaled value of the original skew with the scaling factor of $\frac{R_l}{R_l + r_u - r_w}$. It has been proved in [1] that the scaling factor is always less than 1, thereby proving that the skew between nodes $u$ and $w$ is always reduced as a result of link insertion. It has also been proved in [1] that inserting a link as close to sink nodes as possible is better in terms of skew variability reduction. For example, in Figure 2, when we want to reduce the skew between nodes $r$ and $b$, then it is better for the link to be as close to the nodes $r$ and $b$ as possible.

#### 2.2.2 Skew Variability Between Arbitrary Nodes

The effect of inserting a link between nodes $w$ and $u$ on skew between any two arbitrary nodes $i$ and $j$ is given in [1] as:

$$\hat{q}_{i,j} = q_{i,j} - \frac{r_i - r_j}{R_l + r_u - r_w} q_{u,w} \quad (5)$$

Consider a clock tree $T = (V, E_T)$ as shown in Figure 2 with $E_T$ being the solid lines. Let $T_i$ denote the subtree rooted at node $i$ . The node $u$ is in a subtree $T_f \subset T$ and the node $w$ is in another subtree $T_g \subset T$. The root node of $T_f$ and $T_g$ are the two child nodes of the node $p$. Node $p$ is called the Nearest Common Ancestor (NCA) for the nodes $u$ and $w$. According to [1], when a link is inserted between the nodes $u$ and $w$ the following three scenarios can arise:

**Scenario 1**: One of $i$ and $j$ is in subtree $T_f$ and the other is in subtree $T_g$, for example, $i \in T_f$ and $j \in T_g$. In this case, the link addition introduces a correlation between the delays of nodes $i$ and $j$, which results in reduced skew variability.

**Scenario 2**: Both $i$ and $j$ are in the same subtree $T_f$ or $T_g$. In this scenario, the skew variability might increase. Since $i$ and $j$ are in the same subtree, their skew variation in original tree is usually not that considerable.

**Scenario 3**: One of $i$ and $j$ is in the subnetwork $T_p$ rooted at the NCA node $p$ for nodes $u$ and $w$ and the other node is disjoint with $T_p$. For example, $i$ is in $T_p$ like $b$ and $j$ is not in $T_p$ like $d$ in Figure 2. In this case, there is no predictable correlation between the delays of nodes $i$ and $j$ and so the skew might or might not get reduced.

Therefore, any link insertion scheme must be such that it increases the number of occurrences of scenario 1 while the number of occurrences of scenarios 2 and 3 must be reduced.

### 2.3 Link Insertion Algorithm Overview

The approach taken in [1] towards building a non-tree clock distribution network is an incremental approach. The different steps in the approach are explained below.

1. Obtain initial clock tree from any of the available clock tree routing algorithms in the literature like [17,18,20].
2. Given the clock tree, select node pairs where cross links are to be inserted. The node-pairs must be selected in such a way that the scenario 1 discussed in Section 2.2.2 is maximized and scenarios 2 and 3 minimized. This will make sure that the cross-link addition reduces the skew variability.
3. Since the addition of link capacitance might change the original skew of the clock tree, we need to tune the nodes of the clock tree such that the original skew is not altered. This can be done by considering only the effect of link capacitance on the link end points and tuning the nodes in a bottom-up fashion similar to the method in [19] so as to obtain the original skew after the addition of link capacitance.
4. Finally, the links are added to the selected node pairs. Since the effect of link capacitance has been already removed by bottom-up tuning, only the effect of link resistances will be present, which will reduce the skew variability.

In the above procedure, step 2 is the key step and has a tremendous influence on the quality of the final non tree since selecting the wrong node pairs might result in worse skew variability. The requirements of a good node pair selection algorithm are:

- The algorithm must efficiently distribute the links across all the sections of the clock network. If an algorithm fails to achieve this, it might lead to high skew for the sinks in the region where links are not added.
- It must have a very low complexity in terms of the number of links added. The increase in runtime with increase in the number of links must be less.
- It must make sure that very lengthy links are avoided and the total wire length is reduced.
- The algorithm should be capable of handling asymmetric clock trees as well.

Two algorithms have been proposed in [1] for node pair selection. They are "Rule based" link pair selection algorithm and "min-matching based" node pair selection algorithm. Though both algorithms are able to reduce the skew variability significantly when compared to the original clock tree with little wire consumption, they do not satisfy some of the above mentioned requirements. In this paper, we propose two new algorithms, "Rule Delta" algorithm and "MST with Rule Based Deletion" algorithm, in which the drawbacks of the existing algorithms are addressed. We will discuss these algorithms in the next two sections.

## 3. RULE-DELTA ALGORITHM

The rule-delta algorithm is an improvement of the rule based node pair selection algorithm proposed in [1]. First, we will discuss the merits and demerits of the rule based algorithm of [1] and then introduce the rule-delta algorithm.

### 3.1 Merits and Demerits of the Rule-based Node-pair Selection Algorithm of [1]

The rule based approach of [1] is derived directly from the equation (3). In this method, three rules are defined for node pairs selection, which are given below:

$\alpha$ **rule**: The $\alpha$ value of any link is defined as $\alpha = \frac{R_l}{R_{loop}} \leq \alpha_{max}$, where $R_{loop} = R_l + r_u - r_w$ is the total resistance along the loop of $p \rightsquigarrow u \rightsquigarrow w \rightsquigarrow p$. The lower the value of $\alpha$, the lower the scaling of the original skew and the better the link for skew reduction.

$\beta$ **rule**: The $\beta$ value of any line is defined as $\beta = |\frac{C_l}{2}(R_{u,u} - R_{w,w})| \leq \beta_{max}$. The $\beta$ value determines the extra skew

introduced into the original clock tree due to the link capacitance. The lower the value of $\beta$, the lesser the tuning required in the original tree.

$\gamma$ **rule**: The $\gamma$ rule is intended to make sure that the links added does not worsen the skew variation between any two pairs in the clock network significantly. The 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 using $\gamma$ of the link pair.

In this method, any link that has $\alpha$, $\beta$, $\gamma$ values less than the maximum value set by the user will be added to the tree. The lesser the value of $\alpha$, $\beta$ and $\gamma$, the better the effectiveness of the link in skew variation reduction [1]. But choosing too low values for these variables will result in very less number of links, thereby reducing the effectiveness of link insertion.

### 3.1.1 Merits

The rule based algorithm has the main advantage that the physical characteristics of the links to be inserted are considered before inserting the link in the clock network. Moreover, the runtime does not increase drastically as the number of links to be inserted and the clock tree size increases. Since a lengthy link will usually have high values of $\alpha$ and $\beta$, shorter links will be preferred. Shorter links will be much better in terms of the routability of the final non-tree. Also, when dealing with non-symmetric clock networks, the method will still work because the rules used does not assume any symmetry in the clock network.

### 3.1.2 Demerits

The rule based technique does not have any explicit control over the distribution of the links across the clock network. Generally speaking, we would like the links to be distributed across all the regions of the clock network so that the skew variation of all subtrees in the clock network is reduced. In the case of rule-based method, there is a possibility that the links are added only between a few subtrees thereby not controlling the skew variation in other subtrees. For example, in Figure 3, there are four different subtrees represented by A,B, C and D. It may happen that all the links might get added between the subtrees B and C only, leaving the skew between subtrees A and D completely uncontrolled. This situation can be avoided by making sure that links get added uniformly across all regions of the clock network.

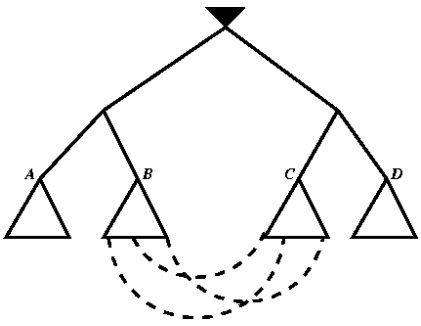


**Figure 3: An example in which the rule based algorithm in [1] might fail.**

## 3.2 Details of Rule Delta Algorithm

The key drawback of the rule-based algorithm is that it cannot guarantee a good distribution of links across all subtrees of the clock network. To overcome this drawback, we propose a new rule called $\delta$ rule in addition to the original $\alpha$, $\beta$ and $\gamma$ rules. The $\delta$ rule is explained below.

$\delta$ **rule**: Let $\delta$ be a valid node level (level = depth of the node) from the clock source. According to the $\delta$ rule, no two links should have the same pair of ancestors at the $\delta$ level from the clock source. When this condition is added to the above rule-based link addition, we can control the distribution of the links among the different sub-trees of the clock tree. For example, in Figure 3, the problem that we would like to avoid is the crowding of the links between the subtrees B and C which leaves the subtrees A and D unconnected. When we apply the $\delta$ rule here with the value of $\delta = 2$, then no more than one link will be inserted between the subtrees B and C.

An important fact to be noted regarding the $\delta$ rule is that the value of $\delta$ must be higher when compared to the $\gamma$ value used. Otherwise, the number of links added will be significantly reduced, thereby reducing the effectiveness of the link insertion. When the value of $\delta$ is increased, more links will be added because of the increase in the number of permitted ancestor node-pairs for link insertion. This will allow the addition of sufficient number of links in a way that they get distributed across all sections of the clock network.

The advantages of Rule-Delta method are listed below:

- The 'rule-delta' algorithm can make sure that the links do not get crowded in the same regions of the clock network. By choosing a proper selection of the $\alpha$, $\beta$, $\gamma$, and $\delta$ values, we can make sure that sufficient number of links get added across all sections of the clock network.
- Since the algorithm just performs a sweep for the different possible links based on the rule values, it is very efficient in terms of the run time.
- The effectiveness of each link is assessed by the rules before adding it. As a result, the situation of addition of lengthy links cannot happen in practice. This is because the lengthy links will have a very high value of both $\alpha$ and $\beta$, which can be removed by using the appropriate bounds for the values of $\alpha$ and $\beta$.
- Since the rules are independent of the structure of the initial clock tree, it can handle even highly unbalanced clock trees.

Thus the rule-delta algorithm has all the required features of a good algorithm that we have listed in Section 2.3.

# 4. MST BASED LINK INSERTION ALGORITHM WITH RULE BASED DELETION

The MST based node pair selection algorithm is an alternative graph theoretical approach to the min-matching based node pair selection algorithm proposed in [1]. First, we will discuss the merits and demerits of the min-matching algorithm of [1] and then introduce the MST based algorithm.

## 4.1 Merits and Demerits of the Min-matching Based Node-pair Selection Algorithm

The min-matching based link insertion algorithm of [1] uses the bipartite min-matching algorithm for selecting the node pairs for link insertion. The main idea of this method is to divide the clock network into several regions and making sure that links are added to every region. This is done by dividing the clock network into two subtrees - the left subtree and the right subtree. Each subtree is further divided into $k$ sub-subtrees. The clock network is represented by a bipartite graph with the two main subtrees as the two sides and the sub-subtrees on either side as the nodes of the bipartite graph with total of $2k$ nodes. The edge weight of the bipartite graph between two nodes is fixed as the minimum rectilinear distance between any two pairs of sinks of those sub-subtrees. For example, in Figure 4, the edge weight between nodes 2 and $b$ is the shortest rectilinear distance between any two pairs of sinks of the sub-subtrees 2 and $b$. Solving the min-matching of this bipartite graph will give us $k$ links with minimum total wire length. This will also make sure that every sub-subtree in the clock network is linked to another by a cross link. This idea is illustrated in Figure 4.

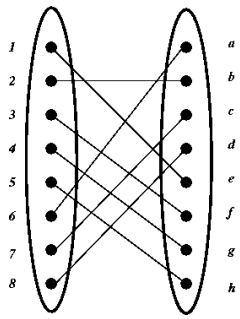


**Figure 4: A bipartite graph model for selecting node pairs between two subtrees. Each subtree is further divided into a number of sub-subtrees for link insertion. An edge weight between two nodes is the shortest rectilinear distance between leaf(sink) nodes of two sub-subtrees.**

### 4.1.1 Merits

The key advantage of the min-matching algorithm is that it guarantees even distribution of links across the clock network. As a result, this method out-performs the rule-based method in skew variation reduction. Also, because of the min-matching nature of the algorithm, it generally gives lesser total wire length when compared to the rule-based method.

### 4.1.2 Demerits

The complexity of the min-matching algorithm in terms of the number of nodes in bipartite graph (number of links added) is $O(n^3)$. This is a major disadvantage because the runtime will drastically increase as the size of the clock network increases. Also, since min-matching does not control the length of individual links, there is high possibility of adding lengthy links. This will make the chip difficult to route. For example, in Figure 4, the links between the nodes 6 and $a$ and nodes 1 and $e$ may be very long. The main reason for lengthy links is that the min-matching algorithm allows a given subtree to be connected to exactly one subtree only. This problem will get worse as the number of links that we want to add increases. Thus, this method will hit a brick-wall in terms of the number of links that can be inserted at a particular level.

Another disadvantage in having lengthy links is that any problem in even one of the links will drastically affect the effectiveness of the non-tree for skew variability reduction. Possible situations in which this can happen are manufacturing defects in the links and unroutability of a link because of the initial clock tree routing. In such cases, having several small links is better than a few lengthy links. Another demerit of the min-matching algorithm is that it inherently assumes a reasonable amount of symmetry in the structure of the clock tree. Otherwise, it might not be very effective in reducing the skew variation. For example, when there are two subtrees on one side of the bipartite graph and five on the other side, then the min-matching algorithm will result in only two links, thereby leaving some of the subtrees without any cross links.

## 4.2 Details of the MST Based Link Insertion Algorithm

As pointed out in section 4.1, one of the key demerits of the min-matching algorithm is the addition of lengthy links. One way to avoid adding lengthy links is to allow a given node to be connected to more than one node on the opposite side of the bipartite graph. Also, not more than one link should be allowed between any given pair of nodes to avoid crowding of links. When these two conditions are satisfied, it is likely that the selected links will not be too lengthy and at the same time, all the sub-subtrees are linked. This is illustrated in Figure 5 in which there are no extremely lengthy links, unlike in Figure 4.

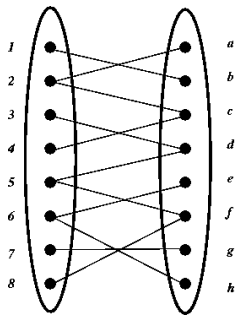


**Figure 5: An example of a new approach that might be better than the min-matching based link insertion.**

To satisfy both these conditions, we propose to construct a MST in the complete bipartite graph. As in [1], the edge weight between any two nodes in the bipartite graph is given as the minimum rectilinear distance between any two pairs of sinks of those two sub-subtrees in the clock network. Constructing a MST from a complete bipartite graph will invariably allow multiple links for a few nodes in the graph. It will also avoid crowding pf links between any two nodes. One possible problem that we might encounter in constructing a complete MST is that the total wire length might become very high. To solve this problem, we selectively remove links

from the selected MST edges based on values of the rules of $\alpha$ and $\beta$ used in the rule based node pair selection algorithms.

Now, we will consider the complete flow of the MST-based link insertion algorithm. From the conclusions in Section 2.2.2, we know that we need to avoid scenario 3 so as to reduce the skew variability. Scenario 3 can be avoided by choosing node pairs between left child subtree and right child subtree of a node of depth 1 from the clock source. For example, in Figure 2, the links between subtree $T_p$ and subtree $T_d$ of depth 1 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 NCA node $h$, which can be called $\gamma$ level as in [1]. Therefore, node pairs with $\gamma = 1$ must be present to effectively reduce the skew variability.

The links inserted between subtrees $T_d$ and $T_p$ will improve skew variability between most sink pairs of those subtrees according to analysis of *scenario 1*. However, these links might worsen the skew variability between sinks pairs within $T_d$ or $T_p$ as discussed in *scenario 2*. This might harm the over all skew variability. This situation can be avoided by inserting links between sub-subtrees within subtree $T_d$ or $T_p$. In other words, node pairs of $\gamma = 2$ need to be considered for link insertion. This procedure can be repeated recursively till $\gamma$ is sufficiently large. The subtrees that correspond to large $\gamma$ are mostly small and usually the skew variability inside is negligible. The main algorithm description on this recursive procedure is given in Figure 6.

| **Procedure:** $Select\_Node\_Pairs(T_v)$ |
|---|
| **Input:** Subtree $T_v$ rooted at node $v$<br>**Output:** Node pair set $P$ |
| 1. $l \leftarrow$ left child node of $v$<br>2. $r \leftarrow$ right child node of $v$<br>3. $P \leftarrow Pair\_Between\_Trees(T_l, T_r)$<br>4. If $Depth(v) == Max\_Depth$, return $P$<br>5. $P \leftarrow P \cup Select_{N}ode_{P}airs(T_l)$<br>6. $P \leftarrow P \cup Select\_Node\_Pairs(T_r)$<br>7. Return $P$ |

**Figure 6: The top-level algorithm of selecting node pairs for link insertion.**

The most important of all the steps in the algorithm in Figure 6 is step 3 in which the node pairs are selected between the two given subtrees. This step is illustrated in Figure 7.

| **Subroutine:** $Pair\_Between\_Trees(T_l, T_r)$ |
|---|
| **Input:** Two subtrees $T_l$ and $T_r$<br>**Output:** Node pair set $P$ |
| A. Decompose $T_l$ into sub-subtrees $S_l = \{T_{l1}, T_{l2}....T_{lk}\}$<br>B. Decompose $T_r$ into sub-subtrees $S_r = \{T_{r1}, T_{r2}....T_{rk}\}$<br>C. For every pair ($T_{li}$, $T_{rj}$) between $S_l$ and $S_r$<br>D. Weight($T_{li}$, $T_{rj}$) = Min distance between leaf pairs in $T_{li}$ and $T_{rj}$<br>E. Find the MST of the complete bipartite graph between $S_l$ and $S_r$.<br>F. Selectively remove the edges that have $\alpha$ and $\beta$ values higher than the set limits of $\alpha_{max}$ and $\beta_{max}$<br>G. $P \leftarrow$ Selected links of MST.<br>H. Return $P$ |

**Figure 7: MST based algorithm of selecting node pairs for link insertion.**

The advantages of MST based link pair selection method are:

- Since the MST based method divides the clock network into subtrees, it can guarantee a good distribution of links across all the regions of the clock network.
- The overall complexity of the algorithm is $O(nlogn)$ in terms of number of nodes and in terms of the number of links to be inserted. This is considerably better than the $O(n^3)$ complexity of the min-matching based algorithm.
- Since multiple links are allowed to be connected to a given node in the bipartite graph, the chances of adding a lengthy link is greatly reduced when compared to the min-matching algorithm.
- The MST based algorithm will work better than the min-matching based algorithm in the case of an unbalanced clock tree. This is because, when the two sides of the bipartite graph have unequal number of nodes, then some nodes will not be linked while using the min-matching algorithm. But in the case of the MST algorithm, all the nodes are guaranteed to be connected to at least one node on the opposite side of the bipartite graph.
- The use of rule based deletion makes sure that the physical characteristics of the links are taken into account before the addition of the link. This will make sure that bad links do not get added to the clock network. We can also effectively control the total wire length consumption by carefully selecting the values of $\alpha_{max}$ and $\beta_{max}$ values for rule based deletion.

Thus we see that the MST algorithm with rule based deletion has all the required features of a good algorithm that we have listed in Section 2.3.

# 5. EXPERIMENTAL RESULTS

To facilitate the comparison between the proposed algorithms and the algorithms of [1], we made sure that our experimental setup is identical to that of [1], i.e., r1-r5 benchmarks obtained from GSRC Bookshelf [21]. The variation factors considered in our experiments are also identical to [1] namely, the clock driver resistance, wire width and the load capacitance of all sinks. The driver resistance, wire width and the sink capacitance have $\pm 15\%$ variation following a normal distribution. Unlike [1] in which only Elmore delay model was used for validation, we also use HSPICE based Monte Carlo simulations for validating our results. For all the clock networks, a Monte Carlo simulation of 1000 trials is performed using both HSPICE and Elmore delay model. We obtain the maximum skew variation in both Elmore delay model (MSV-E) and in HSPICE (MSV-S). Similarly, we obtain the standard deviation values of skew variations for both Elmore delay model (SD-E) and HSPICE(SD-S). These values, along with values of wire length and runtime are compared among clock trees, clock meshes, tree+links with algorithms of [1] and trees+links with our proposed algorithms. The size of benchmark circuits, skew variations and wire length of clock trees are given in Table 1.

Table 2 shows the comparison of five different non-tree structures and algorithms to reduce such clock skew variations. They are Mesh-S (sparse mesh), Mesh-D (dense mesh), Link-M (min-matching based link insertion), all from [1], as well as our two new algorithms Link-RD (rule-delta algorithm) and Link-MST (MST based algorithm). It has

**Table 1: Maximum skew variation (MSV), standard deviation (SD) and total wire length of trees. The CPU time is the time for generating the tree using BST [20] code.**

| Testcase | # sinks | MSV-E | MSV-S | SD-E | SD-S | wirelen | CPU(s) |
|---|---|---|---|---|---|---|---|
| r1 | 267 | 265 | 131 | 42 | 31 | 1320665 | 1 |
| r2 | 598 | 759 | 406 | 112 | 79 | 2602908 | 3 |
| r3 | 862 | 934 | 457 | 166 | 109 | 3388951 | 4 |
| r4 | 1903 | 2321 | 1390 | 317 | 380 | 6828510 | 12 |
| r5 | 3101 | 5792 | 3270 | 1150 | 798 | 10242660 | 18 |

been shown in [1] that the min-matching algorithm always performs better than the rule-based algorithm. So, we compare our results with the min-matching based link insertion from [1] only. The results of the meshes are also provided to make the comparison complete. All the values of MSV-E, MSV-S, SD-E, SD-S and wire length are reported as ratio with respect to the results of the clock trees.

The results for the Link-M method in Table 2 are the best results of the min-matching algorithm in terms of skew variability reduction, which we obtained from the algorithms of [1][1]. The Link-RD rows gives the results for the "rule-delta" link insertion method for each testcase. The values of the parameters $\alpha$, $\beta$ , $\gamma$ and $\delta$ are chosen empirically so as to obtain minimum skew variability. The values of $\alpha$, $\beta$ , $\gamma$ and $\delta$ for the testcase $r1$ are 0.1, 21, 3 and 4 respectively. For all other test cases, namely $r2$ to $r5$, the values of $\alpha$, $\beta$ , $\gamma$ and $\delta$ are 0.06, 100, 3, 5 respectively. The Link-MST rows give the results of the "MST based link insertion" algorithm. As discussed in Section 4.2, we perform selective deletion of links from MST using the values of parameters $\alpha = 0.1$ and $\beta = 100$ for all the clock networks. All the delay values shown are in pico-seconds. The last but one column gives the Average Link Size (ALS) for all the three link insertion schemes.

The important observations from Table 2 are as follows:

- The new algorithms, Link-RD and Link-MST are always able to achieve comparable or better skew variability reduction when compared to the min-matching algorithm of [1] with significantly lower wire length for link insertion. In all test cases, the average wire length increase for the new algorithms was 5%, compared to the 15% cost increase of the min-matching algorithm of [1]. This can be observed from the 'wirelen' column.

- The decrease in the link-cost becomes more significant as the size of the clock network increases. For the test cases of r1 to r5, the wire length costs due to link insertion in our new algorithms (Link-RD and Link-MST) are reduced from around 7-8% to only 1.2% as the number of clock sinks increases from 267 to 3101. However, the previous best algorithm Link-M [1] has a 15% increase. This is mainly because of a drastic reduction in the average link length in the new algorithms. This in turn allows us to insert more links in the clock network, thereby reducing the skew variability more.

- The average length of link for each of the three link insertion methods is given in the right most column. In the case of the min-matching method, for bigger clock networks $(r4, r5)$, the average size of the links becomes significantly higher when compared to the smaller clock networks$(r1 - r3)$. But in the case of the rule-delta algorithm and the MST based algorithm, there is no such significant increase in the average link size. Even in the case of smaller clock networks, the average link length for the proposed algorithms has significantly lower values when compared to the min-matching algorithm.

[1]In [1], the Link-M insertion algorithm was run on a given small number of links, but better MSV and SD may be obtained using the Link-M method if more links are allowed with more wire length penalty.

**Run Time Comparison:** The binary executable used in [1] was used for comparing the run time of our algorithms with that of the min-matching algorithm in [1]. In order to determine how the run time of different algorithms scale with the number of links inserted, we varied the number of links inserted at the highest level (the links with $\gamma = 1$) for each algorithm. Please note that a fair comparison of run time can be done only by comparing link insertion at a given level for all the algorithms. Figure 8 shows the run time values of different algorithms as a function of number of links inserted at the highest level. From Figure 8 we can clearly see that the run time of the min-matching algorithm increases drastically after a point when compared to the other two algorithms. Note that the non-trees of Table 2 are not restricted to a single $\gamma$ level, unlike in this experiment.

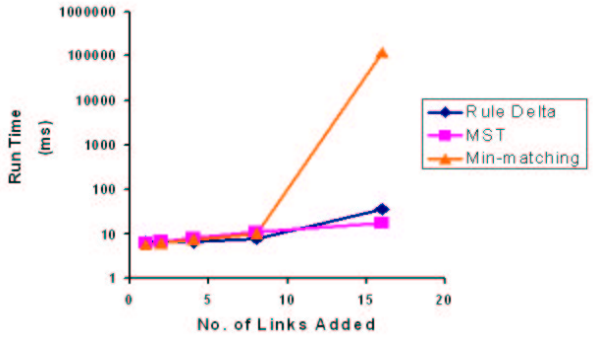


**Figure 8: Runtime comparison between the different link insertion methods as a function of number of links inserted at $\gamma = 1$ level.**

## 6. CONCLUSIONS

We have proposed two new efficient algorithms to overcome some of the drawbacks of the existing algorithms of [1]. The effectiveness of the proposed algorithms has been validated using HSPICE based Monte Carlo simulations. Experimental results show that the new algorithms are able to achieve the same or better skew reduction with an average of 5% wire length increase when compared to the 15% wire length increase of the existing algorithms. The results also show that the links added by the proposed algorithms

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

| Testcase | Method | size | MSV-E | MSV-S | SD-E | SD-S | wirelen | ALS | CPU(s) |
|---|---|---|---|---|---|---|---|---|---|
| r1 | Mesh-S | $11 \times 11$ | 0.92 | 0.36 | 0.82 | 0.06 | 1.850 | N.A | 0.045 |
| | Mesh-D | $21 \times 21$ | 0.72 | 0.25 | 0.62 | 0.04 | 2.510 | N.A | 0.045 |
| | Link-M | 22 | 0.08 | 0.14 | 0.10 | 0.15 | 1.155 | 9004 | 0.069 |
| | Link-RD | 22 | 0.09 | 0.18 | 0.10 | 0.15 | 1.0781 | 4688 | 0.007 |
| | Link-MST | 24 | 0.068 | 0.13 | 0.09 | 0.14 | 1.075 | 4127 | 0.047 |
| r2 | Mesh-S | $15 \times 15$ | 0.80 | 0.58 | 0.80 | 0.05 | 1.76 | N.A | 0.046 |
| | Mesh-D | $29 \times 29$ | 0.59 | 0.15 | 0.47 | 0.03 | 2.430 | N.A | 0.046 |
| | Link-M | 48 | 0.10 | 0.15 | 0.12 | 0.20 | 1.153 | 8296 | 0.095 |
| | Link-RD | 40 | 0.07 | 0.12 | 0.07 | 0.11 | 1.070 | 4555 | 0.032 |
| | Link-MST | 21 | 0.07 | 0.15 | 0.10 | 0.16 | 1.046 | 5701 | 0.075 |
| r3 | Mesh-S | $19 \times 21$ | 0.24 | 0.20 | 0.35 | 0.02 | 1.760 | N.A | 0.046 |
| | Mesh-D | $35 \times 37$ | 0.14 | 0.12 | 0.19 | 0.01 | 2.50 | N.A | 0.046 |
| | Link-M | 64 | 0.09 | 0.16 | 0.11 | 0.19 | 1.144 | 7625 | 0.021 |
| | Link-RD | 51 | 0.08 | 0.15 | 0.10 | 0.12 | 1.06 | 3987 | 0.067 |
| | Link-MST | 41 | 0.08 | 0.18 | 0.08 | 0.13 | 1.053 | 4380 | 0.017 |
| r4 | Mesh-S | $27 \times 29$ | 0.23 | 0.11 | 0.34 | 0.01 | 1.710 | N.A | 0.048 |
| | Mesh-D | $55 \times 57$ | 0.09 | 0.09 | 0.18 | 0.01 | 2.330 | N.A | 0.048 |
| | Link-M | 40 | 0.06 | 0.11 | 0.11 | 0.10 | 1.154 | 26289 | 0.860 |
| | Link-RD | 71 | 0.06 | 0.13 | 0.08 | 0.10 | 1.048 | 4616 | 0.411 |
| | Link-MST | 62 | 0.07 | 0.12 | 0.10 | 0.10 | 1.045 | 4956 | 0.79 |
| r5 | Mesh-S | $37 \times 39$ | 0.08 | 0.07 | 0.10 | 0.01 | 1.640 | N.A | 0.051 |
| | Mesh-D | $75 \times 77$ | 0.03 | 0.06 | 0.06 | 0.04 | 2.330 | N.A | 0.051 |
| | Link-M | 72 | 0.05 | 0.09 | 0.05 | 0.08 | 1.157 | 22334 | 3.050 |
| | Link-RD | 66 | 0.05 | 0.08 | 0.05 | 0.08 | 1.012 | 1862 | 1.945 |
| | Link-MST | 94 | 0.05 | 0.09 | 0.05 | 0.08 | 1.016 | 1743 | 2.91 |

are considerably shorter on the average, thereby making the non-tree more routable than the results of the existing algorithms. The new algorithms are particularly attractive as they scale extremely well to big clock networks, i.e., the bigger the clock network, the less overall link cost (less than 2% for the biggest benchmark we have).

# 7. REFERENCES


[1] A. Rajaram, J. Hu, and R. Mahapatra. "Reducing clock skew variability via cross links," in *Proceedings of the ACM/IEEE DAC*, San Diego, CA, June 2004, pages 18–23.

[2] S. R. Nassif, "Modeling and analysis of manufacturing variations," in *Proceedings of the IEEE CICC*, San Diego, CA, May 2001, pp. 223–228.

[3] R. Saleh, S. Z. Hussain, S. Rochel, and D. Overhauser, "Clock skew verification in the presence of IR-drop in the power distribution network," in *IEEE Transactions on CAD*, vol.19, no.6, pp.635–644, June 2000.

[4] 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 ISQED*, San Jose, CA, March 2002, pp. 355–360.

[5] B. Lu, J. Hu, G. Ellis, H. Su, "Process variation aware clock tree routing," in *Proceedings of the ISPD*, Monterey, CA, April 2003, pp. 174–181.

[6] J. Chung and C.K. Cheng, "Optimal Buffered Clock Tree Synthesis," in *IEEE ASIC conference*, Austin, TX, Sept. 1994, pp. 130–133.

[7] S. Pullela, N. Menezes, and L. T. Pillage, "Reliable non-zero skew clock trees using wire width optimization," in *Proceedings of the ACM/IEEE DAC*, Dallas, TX, June 1993, pp. 165–170.

[8] S. Lin and C. K. Wong, "Process-variation-tolerant clock skew minimization," in *Proceedings of the IEEE/ACM ICCAD*, San Jose, CA, November 1994, pp. 284–288.

[9] 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," in *IEEE Journal of SSC*, vol.36, no.11, pp. 1647–1653, November 2001.

[10] M. P. Desai, R. Cvijetic, and J. Jensen, "Sizing of clock distribution networks for high performance CPU chips," in *Proceedings of the ACM/IEEE DAC*, Las Vegas, NV, June 1996, pp. 389–394.

[11] 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 SSC*, vol.36, no.5, pp. 792–799, May 2001.

[12] H. Su and S. S. Sapatnekar, "Hybrid structured clock network construction," in *Proceedings of the IEEE/ACM ICCAD*, San Jose, CA, November 2001, pp. 333–336.

[13] M. Mori, H. Chen, B. Yao and C.-K. Cheng, "A multilevel network approach for clock skew minimization with process variations," in *Proceeding of the Conference on ASP-DAC*, Yokohama, Japan, January 2004, pp. 263268.

[14] K. D. Boese, A. B. Kahng, B. A. McCoy, and G. Robins, "Near-optimal critical sink routing tree constructions," in *IEEE Transactions on CAD*, vol.14, no.12, pp. 1417–1436, December 1995.

[15] P. K. Chan and K. Karplus, "Computing signal delay in general RC networks by tree/link partitioning," in *IEEE Transactions on CAD*, vol.9, no.8, pp. 898–902, August 1990.

[16] T. Xue and E. S. Kuh. "Post routing performance optimization via multi-link insertion and non-uniform wiresizing," in *Proceedings of the IEEE/ACM ICCAD*, San Jose, CA, November 1995, pp. 575–580.

[17] C.-W. A. Tsao and C.-K. Koh, "UST/DME: a clock tree router for general skew constraints," in *Proceedings of the IEEE/ACM ICCAD*, San Jose, CA, November 2000, pp. 400–405.

[18] T.-H. Chao, Y.-C. Hsu, J.-M. Ho, K. D. Boese, and A. B. Kahng. "Zero skew clock routing with minimum wirelength," in *IEEE Transactions on CS-ADSP*, vol.39, no.11, pp.799–814, November 1992.

[19] R.-S. Tsay, "Exact zero skew," in *Proceedings of the IEEE/ACM ICCAD*, Santa Clara, CA, November 1991, pp. 336–339.

[20] J. Cong, A. B. Kahng, C.-K. Koh, and C.-W. A. Tsao. "Bounded-skew clock and Steiner routing," in *ACM Transactions on DAES*, 3(3):341–388, July 1998.

[21] http://vlsicad.ucsd.edu/GSRC/bookshelf/Slots/BST/