Statistical Based Link Insertion for Robust Clock Network Design

Abstract

We present a statistical based non-tree clock distribution construction algorithm that starts with a tree and incrementally insert cross links, such that the skew variation of the final clock network is within a certain confidence interval under variations in wire width. Monte Carlo simulations show that the robustness of the final clock network can be significantly improved with a small increase in wire length.

1 Introduction

As the minimum feature sizes of VLSI circuits get smaller while the clock frequency increases, the effects of process variations on clock skews become significant. In [10], it is shown that interconnect variations in clock trees can cause as much as 25% variation in clock skew.

To counter the effects of skew variations, two categories of approaches have been proposed. One is the optimization on clock trees [17] [12]. These works perform wire sizing to minimize the skew variations.

The other category is the construction of hybrid non-trees, or in general, clock networks [15, 9, 14, 13]. The main reasoning behind this approach is to provide additional paths for the signal to reach the clock sink nodes such that the variations can be compensated. In [9], a center fat wire is used. The idea behind the center fat wire is that if clock signals from a tree arrive at multiple evenly distributed locations at the center chunk, the clock skew within the fat wire is negligible. In [15], a top (root) level mesh drives multiple clock trees in the lower (leaf) level. In [14], a top level tree drives a mesh in the lower level. In [13], cross links are added to convert a clock tree to a non-tree.

The main objective of the above works is to minimize the skew between clock sinks by performing optimizations that is based on computing the exact skew using delay models. However, process variations are statistically distributed and maybe spatially correlated. Therefore, it is advantageous to perform clock tree optimizations that are based on statistical information of the process variations. It is shown in [13] that inserting links to a clock network can reduce the skew variability without a significant increase in wirelength. In this work we utilize the concept of link insertion to increase the robustness of the clock network. The problem can be stated as follows: *Given a clock tree that satisfies the skew constraints between the clock sinks, successively insert links to the tree/network such that the skew variation is within a certain confidence interval, under variations in wire width.*

We propose an incremental statistical based optimization approach to increase the robustness of a clock distribution network to process variation by the addition of cross links [13]. We use statistical timing analysis similar to [5, 2] to obtain the statistical distribution of the skew of the clock distribution network to determine the optimal insertion point. Once the cross links are inserted, the statistical skew distributions are updated before more links are inserted. In [13], the statistical skew analysis is performed only at the final step, when all links have been added. Monte Carlo simulations show that the robustness of the final clock network can be increased with a small increase in wire length. In this work, we use a fitted Elmore delay model [1] and apply this to the Elmore delay computation of both clock trees and non-trees.

2 Statistical Distribution of Parameters

Consider a wire of two segments. e_i has a uniform width of w_i and of length l_i. e_j has a width of w_j and of length l_j. If we use a RC π-type circuit to model this wire and evaluate the Elmore delay of the wire from one end of e_i (source node) to the end of e_j, we have:

$$d_j = \frac{r_\square l_i}{w_i}\left(\frac{c_a l_i w_i}{2} + \frac{c_f l_i}{2} + c_a l_j w_j + c_f l_j\right) + \frac{r_\square l_j}{w_j}\left(\frac{c_a l_i w_j}{2} + \frac{c_f l_j}{2}\right), \quad (1)$$

where $r_\square$ is the wire resistance per unit square, c_a is the area capacitance per unit length and c_f is the fringing capacitance per unit length. If we rewrite the expression based on the widths from Eqn. (1), we have:

$$d_j = \left[\frac{r_\square c_a l_i^2}{2} + \frac{r_\square c_a l_j^2}{2}\right] + \left[\frac{r_\square c_f l_i}{2} + r_\square c_f l_j\right]\left(\frac{1}{w_i}\right) + \left[\frac{r_\square c_f l_j}{2}\right]\left(\frac{1}{w_j}\right) + [r_\square c_a l_i l_j]\left(\frac{w_j}{w_i}\right). \quad (2)$$

The Elmore delay model has been widely used because it is extremely efficient to compute and has a good fidelity with respect to HSPICE simulation [3]. The primary disadvantage of is that it has limited accuracy and it always overestimates the delay [8]. To increase the accuracy , we use a fitted Elmore delay model [1] in this work, although we refer to it as Elmore delay throughout the paper.

Due to process variations, the width of a fabricated wire may deviate from the designed width and it would follow a statistical distribution. Therefore, we replace all occurrences of w_i and w_j with random variables (RV) W_i and W_j and express the Elmore delay of the wire as another RV denoted as D_j:

$$D_j = \left[\frac{r_\square c_a l_i^2}{2} + \frac{r_\square c_a l_j^2}{2}\right] + \left[\frac{r_\square c_f l_i}{2} + r_\square c_f l_j\right]\left(\frac{1}{W_i}\right)$$

$$+\left[\frac{r_{\square}c_f l_j}{2}\right]\left(\frac{1}{W_j}\right)+[r_{\square}c_a l_i l_j]\left(\frac{W_j}{W_i}\right). \quad (3)$$

We assume that W has a normal distribution with mean ($\overline{W}$) and standard deviation (σ_W). In this work, we consider only wire width variations, but it can be extended to handle other sources of variations such as wire height.

Using a Taylor series expansion of Eqn. (2) up to the second order, we have:

$$D_j = d_0 + \sum_{k=1}^{m} \left.\frac{\partial D_j}{\partial W_k}\right|_{W_k=\overline{W}_k} \Delta W_k$$
$$+\frac{1}{2}\sum_{k_1=1}^{m}\sum_{k_2=1}^{m} \left.\frac{\partial^2 D_j}{\partial W_{k_1}\partial W_{k_2}}\right|_{W_{k_1}=\overline{W}_{k_1},W_{k_2}=\overline{W}_{k_2}} \Delta W_{k_1}\Delta W_{k_2}, \quad (4)$$

where $d_0 = d_j$, evaluated at the expansion point (which is the Elmore delay if we do not consider process variations), m is the number of RVs involved in Eqn. (3), and $\Delta W_k = W_k - \overline{W}_k$. From this we can see that the mean of Elmore delay is not obtained by setting the wire widths to their mean values, due to other higher terms that shift the mean. Using a second order expansion mainly increases the accuracy of the mean computation. The effect of higher order terms on the variance is not as pronounced.

Therefore the Elmore delay of D_j is in the form:

$$D_j = d_j + \sum_{k=1}^{m} a_k \Delta W_k + \sum_{k_1=1}^{m}\sum_{k_2=1}^{m} b_{k_1k_2}\Delta W_{k_1}\Delta W_{k_2}, \quad (5)$$

where a_k and $b_{k_1k_2}$ are scalar constants. Therefore for each D_j, there are $1+m+m(m+1)/2$ terms.

To find the mean, $E(D_j)$, in Eqn. (5), we have:

$$E(D_j) = d_j + \sum_{k=1}^{m} a_k E(\Delta W_k) + \sum_{k_1=1}^{m}\sum_{k_2=1}^{m} b_{k_1k_2} E(\Delta W_{k_1}\Delta W_{k_2}), \quad (6)$$

where $E(\Delta W_{k_1}\Delta W_{k_2})$ is the covariance between ΔW_{k_1} and ΔW_{k_2} (since both ΔW_{k_1} and ΔW_{k_2} have zero mean). To find the variance, $Var(D_j)$, we have:

$$Var(D_j) = \sum_{k_1=1}^{m}\sum_{k_2=1}^{m} a_{k_1}a_{k_2} E(\Delta W_{k_1}\Delta W_{k_2}) + H.O.T., \quad (7)$$

where $H.O.T.$ are higher order terms. In this work, we do not consider $H.O.T.$ for variance computation. The RVs are assumed to be spatially correlated and is given in a covariance matrix. The spatial correlation is modeled by imposing a set of grids on the design space where parameters of devices or wires within the same grid have perfect correlation while the parameters of devices or wires that are in distant grids have a weaker correlation [5, 2]. Therefore if there are m grids, there would be m RVs and covariance matrix would be of size $m \times m$. Each grid represents the distribution of the deviation of wire widths ΔW_k. Therefore, if we know the mean values of each RV, we can easily compute the mean and variance of the delay of the wire. We assume all wires in the same grid have the same nominal width. The model can be easily generalized to handle the case where wires in the same grid may have different widths.

For a clock tree, we denote the skew between any two nodes i and j to be $q_{ij} = d_i - d_j$. Therefore, $Q_{ij} = D_i - D_j$ and can similarly be expressed in the form:

$$Q_{ij} = q_{ij} + \sum_{k=1}^{m} \tilde{a}_k \Delta W_k + \sum_{k_1=1}^{m}\sum_{k_2=1}^{m} \tilde{b}_{k_1k_2} E(\Delta W_{k_1}\Delta W_{k_2}), \quad (8)$$

where $\tilde{a}_k$ and $\tilde{b}_k$ are the scalar constants obtained from $D_i - D_j$. Hence, $E(Q_{ij})$ and $Var(Q_{ij})$ can be computed in a similar fashion as in Eqn. (6) and (7). The mean and the variance can be found in $O(m^2)$ run time (we do not use higher orders for variance computation).

To reduce the complexity of the mean and variance computation, we make use of Principal Component Analysis (PCA) [11] to convert the set of correlated RVs to an uncorrelated set as in [5]. In turn, we can express all the delays and skews with the new uncorrelated set of RVs. Since the cross terms no longer exist, there are only $1+2m$ terms for every D and Q. Once the delay or skew is expressed as a set of independent RVs, the mean and variance can now be found in $O(m)$ run time. For the rest of the paper, D's and Q's are assumed to be expressed with the new uncorrelated set of RVs.

Table 1 shows the accuracy of the statistical skew analysis compared to Monte Carlo HSPICE simulations. Max mean skew is defined as the largest mean skew in the circuit and Max SD skew is the largest standard deviation of skew in the circuit. It also shows the use of fitted Elmore delay on a tree. From the results, we see that using a fitted Elmore delay enhances the accuracy when performing statistical analysis, which is crucial when we are optimizing the variation of skews such that it is within a certain confidence level.

3 Statistical Delay and Skew

A challenge of adding links to trees is that once the link is added and a non-tree is formed, Eqn. (1) and consequently Eqn. (5) can no longer be used directly to compute the delays. Suppose that n is the number of nodes in the clock tree. In general, given the node conductance matrix G of size $n \times n$, and the nodal capacitance matrix C that is of size $n \times 1$, the RC delay can be found by computing $G^{-1}C$ or RC, where $R = G^{-1}$.

In this work, since we assume that the wire widths are RVs, we need to capture this information in the resistance and capacitance of each wire. In this work, a wire that passes through multiple grids is treated as a path p of multiple nodes and edges. The capacitance (c_L) and resistance (r_L) of the wire can be expressed as:

$$c_L = \sum_{i \in path\ p} l_i(c_a W_i + c_f), \quad (9)$$

and

$$r_L = r_{\square} \sum_{i \in path\ p} \frac{l_i}{W_i}. \quad (10)$$

From these two equations, we can express c_L and r_L in the form of a Taylor series expansion similar to Eqn. (5), then represent them with a set of uncorrelated RVs obtained by PCA. To

Table 1: **Comparison of statistical skew analysis with Monte Carlo (MC) HSPICE simulations for trees**

Benchmark Circuit				(MC)		Stat. analysis (SA)		$\Delta = \frac{SA-MC}{MC}$		(*SA)		$\Delta = \frac{*SA-MC}{MC}$	
Circuit	Clock pins #	Nodes #	Grids #	Max mean skew (*ps*)	Max SD skew (*ps*)	Max mean skew (*ps*)	Max SD skew (*ps*)	ΔMax mean skew (%)	ΔMax SD skew (%)	Max mean skew (*ps*)	Max SD skew (*ps*)	ΔMax mean skew (%)	ΔMax SD skew (%)
s1423	74	262	16	1119	22.4	1134	22.9	1.3	2.2	1305	26.6	16.6	18.8
s5378	179	594	64	950	102	946	108	-0.4	7.8	1061	124	11.7	21.6
s15850	597	2162	64	1853	455	1779	464	-4.0	1.9	2259	579	21.9	27.3

*Without using fitted Elmore delay

avoid adding new notation, we again use r_L and c_L to denote resistance and capacitance expressed in terms of the set of uncorrelated RVs. To find the conductance, we take the Taylor series expansion of $\frac{1}{r_L}$ and represent it with a set of uncorrelated RVs.

Since we can express resistance and capacitances with sets of uncorrelated RVs, the remaining challenge is in computing RC and updating it as successive links are inserted.

3.1 Incremental Updates

After each subsequent link insertion, RC must be updated. Suppose the x^{th} link wire that we add to the tree has a resistance of r_x and a capacitance of c_x and the link wire is added between nodes i and j.

Updating C is trivial because we add $\frac{c_x}{2}$ to two entries of C, namely C_i and C_j. This can be viewed as adding a column vector (ΔC_x) to C where ΔC_x has a capacitance of $\frac{c_x}{2}$ at the i^{th} and j^{th} entries with the rest set to zero:

$$C_x = C_{x-1} + \Delta C_x, \tag{11}$$

where:

$$\Delta C_x = \begin{bmatrix} c_1 \\ \vdots \\ c_n \end{bmatrix}, \quad c_k = \begin{cases} \frac{c_x}{2} & k = \{i, j\}, \\ 0 & otherwise. \end{cases} \tag{12}$$

In this work, we do not construct R. However, to demonstrate the main concept of updating RC, assume that it is given. Updating R is not as straightforward. One way of performing the update is by adding $\frac{1}{r_x}$ to the appropriate entries of G, and perform an inverse operation (assuming that it is possible) to obtain the new R. If we observe the update of G, we see that adding a link resistance r_L between nodes i and j is equivalent to subtracting (adding) the link conductance $\frac{1}{r_L}$ from (to) two entries of G, namely G_{ii}, G_{jj} and adding (subtracting) $\frac{1}{r_L}$ to (from) G_{ij}, G_{ji}. If we put these four link conductances in a matrix ΔG, the new G, is given by:

$$G_x = G_{x-1} + \Delta G_x, \tag{13}$$

such that:

$$\Delta G_x = -\frac{1}{r_x} \underbrace{\begin{bmatrix} v_1 \\ \vdots \\ v_n \end{bmatrix}}_{V_x} \underbrace{\begin{bmatrix} v_1 \cdots v_n \end{bmatrix}}_{V_x^T}, \quad v_k = \begin{cases} 1 & k = i, \\ -1 & k = j, \\ 0 & otherwise, \end{cases} \tag{14}$$

and we denote the vector of v's as V_x.

From Eqn. (14), we can see that ΔG is a matrix of rank-one. Therefore, instead of performing an inverse computation of G_x after every link inserted, we perform a rank-one update on R_{x-1} (G_{x-1}^{-1}) to obtain R_x (G_x^{-1}) directly. The rank-one update of R_{x-1} is given by [7]:

$$R_x = [R_{x-1} - \frac{1}{r_x} V_x V_x^T]^{-1} = R_{x-1} + \beta R_{x-1} V_x V_x^T R_{x-1}, \tag{15}$$

where:

$$\beta = \frac{r_x}{r_x - V_x^T R_{x-1} V_x}$$

By using a rank-one update, we can obtain R_x quickly after a link insertion. The new delay of every node can then be obtained by multiplying R_x with the updated C_x. This allows us to use a rank-one update to compute the statistical delay and skew of a clock network which is constructed from a clock tree by link insertion in $O(mn^2)$. However, in this work, we are only interested in the product of R and C. Therefore, R is not constructed and it is not necessary to update R and C separately. By directly updating RC, there is a reduction in computation complexity.

This approach is similar to [4] in a sense that they are also incrementally computing the new RC delay as resistive links are added back to a tree to compute the RC delay of the original network (the resistive links were removed from the original RC network to form a tree). However, in [4], their objective is to compute the final delay of a given RC network without the presence of variations. In this work, we are computing the statistical delays and skews in the presence of variations. We also add new capacitances to the network and therefore we have additional terms that we need to compute. The main concept behind our work is the use of a rank-one update and the use of the unique structure in ΔG_x and ΔC_x, for a direct update of the statistical delay.

4 Direct Incremental Statistical Delay Update

In this section, we introduce a method to compute and update RC directly after a link is inserted.

From Eqns. (11, 13 and 15), we have:

$$R_xC_x = \left[R_{x-1} + \frac{r_xR_{x-1}V_xV_x^TR_{x-1}}{r_x - V_x^TR_{x-1}V_x}\right](C_{x-1} + \Delta C_x)$$

$$= R_{x-1}C_{x-1} + R_{x-1}\Delta C_x + \frac{r_xR_{x-1}V_x(V_x^TR_{x-1}C_{x-1})}{r_x - V_x^TR_{x-1}V_x} + \frac{r_xR_{x-1}V_x(V_x^TR_{x-1}\Delta C_x)}{r_x - V_x^TR_{x-1}V_x} \quad (16)$$

To compute R_xC_x, we must obtain the following terms:

1. r_x: This is the resistance of the link expressed with a set of uncorrelated RVs.

2. V_x or V_x^T: This is a column vector that corresponds to the two nodes (i and j) that are connected by the x^{th} link. The vector has values of 1 and -1 in the i^{th} and j^{th} entries and zeros elsewhere. This term is stored for successive link insertions and requires $O(1)$ space complexity for each link.

3. $R_{x-1}C_{x-1}$: This term corresponds to the RC delay before the x^{th} link is inserted. Before the first link is inserted, this corresponds to the case where $x = 1$ and R_0C_0 corresponds to the original Elmore delay of the clock tree, which can be found by a bottom-up, top-down tree traversal in $O(mn)$. This is the key idea why we do not require the construction of R. Once this term is computed, it is stored for successive link insertions and requires $O(mn)$ space complexity.

4. $V_x^TR_{x-1}C_{x-1}$: From Eqn. (14), we see that V_x^T is a row vector of zeros except with values of 1 and -1 in the i^{th} and j^{th} entries. Therefore the product of V_x^T and $R_{x-1}C_{x-1}$ can be easily found by taking the difference between the i^{th} the j^{th} entry of $R_{x-1}C_{x-1}$ and can be done in $O(m)$ run time.

5. $R_{x-1}V_x$: By replacing the x's with $y-1$ in the brackets and replacing $(C_{x-1} + \Delta C_x)$ with V_x in Eqn. (16), this term can be computed using the recursive function as follows:

$$R_{y-1}V_x = \underbrace{R_{y-2}V_x}_{(a)} + \underbrace{\frac{r_{y-1}R_{y-2}V_{y-1}V_{y-1}^T(R_{y-2}V_x)}{r_{y-1} - V_{y-1}^TR_{y-2}V_{y-1}}}_{(b)}. \quad (17)$$

y is used to indicate the difference between the V's of previous links that have been already added and the V's of the last or current link. The term $R_{y-2}V_{y-1}$ (or $R_{x-2}V_{x-1}$) in (b) has been computed in the $(x-1)^{th}$ link and is known if we have stored this in the previous link insertions. $V_{y-1}^TR_{y-2}V_x$ in (b) can be computed in a similar fashion as $V_x^TR_{x-1}C_{x-1}$. Therefore, (b) can be computed as soon as $R_{y-2}V_x$ (or (a)) is obtained.

6. $V_x^TR_{x-1}V_x$: Similar to $V_x^TR_{x-1}C_{x-1}$, the product of V_x^T and $R_{x-1}V_x$ can be found by taking the difference between the i^{th} the j^{th} entry of $R_{x-1}V_x$. This term can be computed in $O(m)$ as soon as $R_{x-1}V_x$ is obtained. We do not store this for successive link insertions.

   We compute $R_{y-2}V_x$ in (a) by doing recursive operations of Eqn. (17). This will generate terms similar to (b), which we can compute once the term like (a) is known, and a term like (a) which again will require further recursive operations. Eventually the term (a) will reach R_0V_x. Note that this is very similar to R_0C_0, which is the Elmore delay of the original tree, before any links are inserted. Therefore, we can treat R_0V_x as the Elmore delay of a special tree that has the same exact topology and resistance values as the original tree (Fig. 1). However, the tree has zero node capacitances except for node i and node j. These two nodes have a "capacitance" of 1 and -1 respectively when the x^{th} link is inserted. We will show how R_0V_x is obtained in the next section. Since R_{x-1} is symmetric:

$$(V_x^TR_{x-1}) = (R_{x-1}V_x)^T.$$

   Once this term is computed, it is stored for successive link insertions and requires $O(mn)$ space complexity. Note that we store terms that are $R_{x-1}V_x$, i.e., where the index differs by one. The rest are not stored because they are not reused again in future link insertions.

7. $R_{x-1}\Delta C_x$: The method of computing this term is very similar to finding $R_{x-1}V_x$. It will require recursive operations similar to Eqn. (17), except V_x is replaced with ΔC_x. Eventually the term (a) will reach $R_0\Delta C_x$. We can treat $R_0\Delta C_x$ as the Elmore delay of a special tree that has the same exact topology and resistance values as the original tree. However, the tree has zero node capacitances except for node i and node j. These two nodes both have a capacitance of $\frac{c_x}{2}$ when the x^{th} link is inserted. We will show how $R_0\Delta C_x$ is obtained in the next section. This term is not stored for successive link insertions.

   $V_x^TR_{x-1}\Delta C_x$ can be computed in $O(m)$ as soon as $R_{x-1}\Delta C_x$ is obtained.

In [4], no new capacitances are added when the links are added back to form the original RC network. In this work, we have the extra term $R_0\Delta C_x$, which needs to be computed and will be presented in the following section. The other difference is in the space complexity. Using the same notation, [4] requires $O(mnx^2)$. In this work, we require $O(mnx)$ to store the terms $R_{x-2}V_{x-1}$ and R_xC_x.

4.1 Computing R_0V_x and $R_0\Delta C_x$

Note that the special trees corresponding to the terms R_0V_x and $R_0\Delta C_x$ are essentially the same except that the two capacitances at node i and node j are different (Fig. 1). Therefore, we can

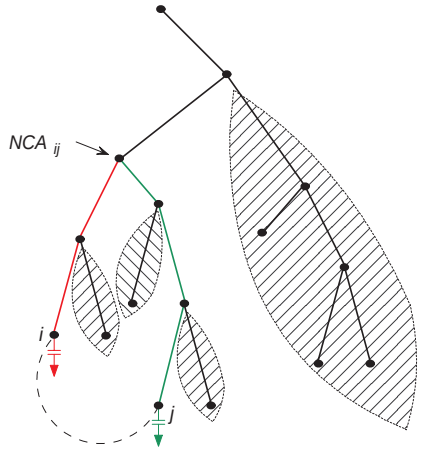

Fig. 1: **Figure showing the special tree and the off-path subtrees. All nodes have zero capacitance except nodes i and j (where the link is inserted)**

compute both terms simultaneously with a single bottom-up, top-down traversal. There are some unique properties that allows a quick computation the Elmore delay of these special trees. We denote the path in the tree from node a to node b as $path_{a\rightsquigarrow b}$ and denote the node NCA_{ij} as the nearest common ancestor of nodes i and j. In this work, we only insert links between nodes i and j such that $NCA_{ij} \neq i$ and $NCA_{ij} \neq j$.

Suppose s is the root of the original clock tree. We denote T_k as an off-path subtree rooted at node k: $k \in path_{i\rightsquigarrow s} \cap path_{j\rightsquigarrow s}$, and no edges in T_k intersects with any edge in $path_{i\rightsquigarrow s} \cap path_{j\rightsquigarrow s}$ (Fig. 1). The nodes in T_k have zero capacitances, which implies that the Elmore delay of these nodes will have the same delay as node k. Therefore, we can simply fill in these values in R_0V_x and $R_0\Delta C_x$ once the delay to node k is found. This means that we do not need to traverse those nodes in T_k during the Elmore delay computation. Although it still requires $O(n)$ time to fill in the entries, we only perform a number of multiplications (product of node resistance with downstream capacitance of the node) that is bounded by the height of the original tree.

Suppose $u = NCA_{ij}$. It is interesting to note that to fill the entries of R_0V_x, we really do not need to traverse above u for the Elmore delay computation. This is because the downstream capacitance of those nodes k: $k \in path_{u\rightsquigarrow s}$, are zero (sum of 1 and -1). Therefore the Elmore delays from the root to nodes k and to nodes in subtrees T_k, where $k \in path_{u\rightsquigarrow s}$, are all zero.

5 Statistical based link selection

The major challenge is to determine where the links should be inserted. This is because as we can expect that, in general, inserting a link between two nodes will change the skew between all node pairs.

In a synchronous circuit, between every pair of sinks, say i and j, there are two inequalities that pose lower and upper bound constraints on the skew q_{ij} [6]. These are the hold time and setup time constraints. For simplicity, we use $a_{ij} \leq q_{ij} \leq b_{ij}$ to represent the lower and upper bound skew constraints between sinks i and j.

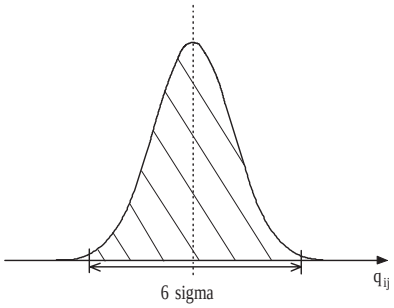

Fig. 2: **Figure showing a PDF of skew (q_{ij}). The shaded area is the 6σ skew variation**

Input: Clock tree T, skew constraints (or bounded skew), grids and Covariance matrix
Output: Clock network $\tilde{T}$

1. Add internal nodes to T based on the grid partition
2. Perform PCA to obtain uncorrelated set of RVs
3. Perform statistical skew analysis on T
4. While skew violation exists: (See Eqn. (18))
   a. For each sink pair considered for link sink insertion, find the worst case skew violation
   b. Insert the link that has the best worst case violation
   c. Update skew

Fig. 3: **Statistical link insertion algorithm**

The objective of this work is to achieve:

$$a_{ij} + \alpha\sigma_{ij} \leq q_{ij} \leq b_{ij} - \alpha\sigma_{ij}, \tag{18}$$

where $\alpha\sigma_{ij}$ is α times the standard deviation of Q_{ij} and σ_{ij} is obtained from the statistical skew analysis. In other words, we would like the mean of the skew of every sink pair to be adjusted such that the $2\alpha\sigma_{ij}$ skew variation ($\mu \pm \alpha\sigma_{ij}$) for that particular pair of sinks lies within their skew constraints. A skew violation occurs when the skew constraints are not satisfied. In this work, we assume $\alpha = 3$ as shown in Fig. 2.

In Section 4, we showed that we can obtain the statistical delay and skew after a link has been inserted. Therefore, for any candidate link, we can temporarily insert the link, perform a statistical delay evaluation and check against the skew constraints for skew violations. We can then determine whether that candidate link should be permanently inserted to resolve the skew violations. Once the link is inserted, the delay and skew is updated and is ready for the next link to be inserted. The algorithm is summarized in Fig. 3.

5.1 Candidate link selection

In a clock tree with n nodes, there are $O(n^2)$ pairs of nodes that could be considered as candidates. Clearly, testing all of them will be expensive to do. Therefore, we reduce the search space by setting a limit on the link distance. The reason for this is that we would like to keep the links as short as possible to minimize the total wirelength. A longer wirelength imposes a larger load

to the clock driver and will consume more power per switching cycle. The other reason is that a shorter link inserted will have a smaller perturbation on the delay and skew of the overall tree because a shorter link would have smaller link capacitances. We can see this from the computation of $R_0\Delta C_x$.

One other constraint that we use is to limit the height of the subtree that is rooted at the NCA of the two nodes where the link is to be inserted. From the computation of $R_0\Delta V_x$, we can see that the taller the subtree, the larger the effect the link has on the tree.

6 Complexity Analysis

In this section, we present and summarize a run time complexity analysis of the major components of the algorithm:

1. Statistical Elmore delay evaluation of the original tree with n nodes: We first start with a bottom-up traversal to gather the all the downstream capacitances of every node, which takes $O(n)$. Then we perform a top-down traversal to compute the Elmore delay of every node by taking the product of the edge resistance and its downstream capacitance and adding the delay of the parent node. Each product takes $O(m)$ and we go through n nodes. Therefore the run time is $O(nm)$. Once the delays D_i's to each node is found, to compute the mean and variance requires $O(m)$ time. Therefore, the overall complexity of this part is $O(nm)$.

2. Statistical skew evaluation: Once we have obtained all the delays D_i's to each node, we can find the skew Q_{ij}'s. Since there are $O(n^2)$ skew pairs, and each pair requires $O(m)$ time to compute the mean and the variance of the skew, the overall complexity of this part is $O(mn^2)$. However, in practice, we only evaluate the skew of the sink pairs because we are interested in the skews of the clock sinks.

3. Direct incremental statistical delay update: This component has several smaller parts:

   (a) Computation of R_0V_x and $R_0\Delta C_x$: This is done by finding the Elmore delay of two special trees as shown in Section 4.1. However, since the two trees are very similar, we can compute these two terms at the same time. Due to the fact that most node capacitances of these special trees are zero, we do not need to traverse all n nodes when computing the Elmore delay. This means that although it is required to fill n entries of R_0V_x and $R_0\Delta C_x$, we only perform a very small amount of computation (less than n). We state that this part takes $O(nm)$.

   (b) Computation of R_xC_x: Assuming we have all the terms shown in Section 4, we can start combining the terms together. Since $V_x^TR_{x-1}C_{x-1}$, $V_x^TR_{x-1}\Delta C_x$ and $\left(\frac{r_x}{r_x-V_x^TR_{x-1}V_x}\right)$ are all of size 1×1, therefore we can view these terms as constant factors. We can therefore rewrite Eqn. (16) as:

$$R_xC_x = (R_{x-1}C_{x-1}) + (R_{x-1}\Delta C_x) + (R_{x-1}V_x)\cdot \left[\left(\frac{r_x}{r_x-V_x^TR_{x-1}V_x}\right)\left(V_x^TR_{x-1}C_{x-1}+V_x^TR_{x-1}\Delta C_x\right)\right] \quad (19)$$

   Computing the term in brackets takes one multiplication and that term is multiplied n times with $R_{x-1}V_x$. We then add this to $R_{x-1}C_{x-1}$ and $R_{x-1}\Delta C_x$ to obtain R_xC_x. Therefore, this operation takes $O(mn)$ time.

   If x is the number of links, to compute $R_{x-1}V_x$, it requires x recursive operations of Eqn. (17) and one tree traversal to obtain R_0V_x. Each recursive operation should be computed in the same manner as the computation of R_xC_x, which takes $O(nm)$. Therefore the overall time to compute the final result of R_xC_x in Eqn. (16) for the x^{th} link is $O(xmn+nm)$.

Therefore, the overall time to insert the x^{th} link, update R_xC_x followed by a statistical skew evaluation takes $O(mn^2+xmn+nm)$ time. Assuming for the link selection process, we consider h candidate pairs of nodes as the x^{th} link inserted. Therefore, if we have to insert a total of x links, the run time complexity of our algorithm is $O(x(h(mn^2+xmn+nm)))$. From the experiments shown, we only insert a very small amount of links, therefore x is very much less than n.

7 Experiments and Results

The proposed statistical link insertion algorithm has been implemented in Matlab and tested on three ISCAS89 benchmark circuits on a Sun UltraSparc-II workstation. The wires are assumed to have a resistance of 0.03Ω per unit square, an area capacitance of $1fF/\mu m$ and a fringing capacitance of $1.2fF/\mu m$. These numbers are typical for a $0.18\mu m$ process technology.

To truly evaluate the quality of our solution we performed Monte Carlo HSPICE simulations on the original clock tree and the final clock tree with inserted links. We assume that the process variations are spatially correlated. We use a grid partitioning to section the die into a number of grids (m). Given the clock tree, we first add additional internal nodes to the tree based on the grid partitioning. Long wires that pass through several grids will be treated as multiple segments and each segment's wire width follows the statistical distribution associated with the grid it passes through.

In these experiments, the clock tree (network) is fed into the simulator and wire widths are assigned to all the edges, while retaining the topology. The assignment of wire widths follows a normal distribution that has a mean of w_{mean} and a standard deviation of σ_w. The $3\sigma_w$ variation of the wire widths in every grid has been set to an arbitrary value of $\pm 20\%$ of the nominal value. In other words, the minimum (-3σ) is set at 80% of the nominal value and the maximum ($+3\sigma$) is set at 120% of the nominal value. The skews between all pairs of sinks are

Table 2: Comparison of statistical skew analysis with Monte Carlo (MC) HSPICE simulations for non-trees

Benchmark Circuit				(MC)		Stat. analysis (SA)		$\Delta = \frac{SA-MC}{MC}$		(*SA)		$\Delta = \frac{*SA-MC}{MC}$	
Circuit	Clock pins #	Nodes #	Grids #	Max mean skew (*ps*)	Max SD skew (*ps*)	Max mean skew (*ps*)	Max SD skew (*ps*)	ΔMax mean skew (%)	ΔMax SD skew (%)	Max mean skew (*ps*)	Max SD skew (*ps*)	ΔMax mean skew (%)	ΔMax SD skew (%)
s1423	74	262	16	1053	21.1	1084	21.9	2.9	3.8	1241	25.4	17.9	20.4
s5378	179	594	64	945	103	942	110	-0.3	6.8	1055	128	11.6	24.3
s15850	597	2162	64	1841	453	1758	463	-4.5	2.2	2247	569	22.1	25.6

*Without using fitted Elmore delay

Table 3: **Monte Carlo HSPICE skew and yield analysis before link insertion**

Circuit	UST/DME Safety margin (*ps*)	Wire-length (*mm*)	Max mean skew (*ps*)	Max sd skew (*ps*)	Yield (%)
s1423	0	107.3	1119	22.4	0
	1500	141.2	1077	21.8	100
s5378	0	176.5	950	102	0
	400	363.5	941	101	100
s15850	0	448.6	1853	455	0
	300	1079.7	1821	451	100

evaluated based on this width assignment using HSPICE. For each clock tree, 2000 simulations are performed with a new set of width assignment for every simulation.

Table 2 shows the comparison of statistical skew analysis (using fitted Elmore and without) with Monte Carlo HSPICE. The results show that there is an increased accuracy when fitted Elmore delay is used for non-trees. Table 3 shows the Monte Carlo HSPICE skew analysis on the clock trees before links are inserted. It shows the number of clock sinks, the wire-length, the maximum of the mean of all skews and the maximum of the standard deviation of all skews. The trees are generated using the UST/DME clock routing algorithm [16] with different safety margins. The idea of safety margins is to push the clock skew away from the bounds during clock routing such that they provide an increased tolerance to process variations. This, however, increases the wirelength. We show the increased wirelength for each circuit with a minimal safety margin (at intervals) such that a 100% yield is achieved. The yield is defined as the percentage of clock trees/networks tested in the simulations that meet all skew constraints under the presence of process variations, based on the 6σ definition in Eqn. (18). Our algorithm starts with clock trees with zero safety margins.

The abrupt changes in the yield stems from the fact that the safety margin increments in the experiments are done in intervals. It would require very large amount of resources to run Monte Carlo HSPICE simulations to determine the yield using fine safety margin increments. For the smallest benchmark circuit s1423, we had to increase the wire width variation to $\pm$30% to introduce a yield loss. This is because smaller clock trees typically has shorter branches and therefore are less affected by variations.

Table 4 shows the Monte Carlo HSPICE skew analysis on the clock trees after the links are inserted. For each benchmark circuit, it shows the change in both the maximum mean skew and the maximum standard deviation of skew from the original clock tree, the wirelength and the number of links inserted. From the results, we see that we can increase the yield with a small number of links that do not significantly increase the wirelength.

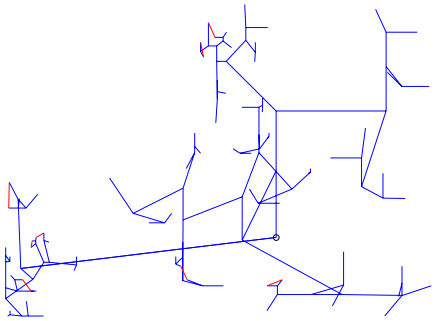

Fig. 4: **Circuit s1423 with links inserted (shown in red)**

8 Summary and Conclusions

In conclusion, we have presented a method to incrementally add cross links to a clock tree such that the skew variation of the final clock network is within a certain confidence interval. The link selection is determined by statistical skew analysis which is incrementally updated after links are inserted. The final result is a non-tree that has skew variation that is within a certain confidence interval, with a small increase in wirelength (12.2% max) compared to the orginal tree with zero saftey margins. The results are verified with Monte Carlo HSPICE simulations.

Table 4: **Monte Carlo HSPICE skew and yield analysis after link insertion**

Circuit	Max mean skew (*ps*)	Δ Max mean skew (%)	Max sd skew (*ps*)	Δ Max sd skew (%)	Wire-length (*mm*)	Δ Wire-length (%)	*Δ Wire-length (UST) (%)	Links inserted	CPU time (*min*)	Yield (%)
s1423	1053	-5.9	21.1	-1.3	119.1	10.9	-15.7	10	2.6	100
s5378	945	-0.5	103	1.0	185.7	5.2	-48.9	8	53.8	100
s15850	1841	-0.6	453	-0.4	503.3	12.2	-53.4	20	655	100

*Comparing with UST/DME with 100% yield

References

[1] Arif Ishaq Abou-Seido, Brian Nowak, and Chris Chu. Fitted elmore delay: a simple and accurate interconnect delay model. *IEEE Trans. Very Large Scale Integr. Syst.*, 12(7):691–696, 2004.

[2] A. Agarwal, D. Blaauw, and V. Zolotov. Statistical timing analysis for intra-die process variations with spatial correlations. In *Proc. Int. Conf. on Computer Aided Design*, pages 900–907, 2003.

[3] K. D. Boese, A. B. Kahng, B. A. McCoy, and G. Robins. Fidelity and near-optimality of Elmore-based routing constructions. In *Proc. IEEE Int. Conf. on Computer Design*, pages 81–84, 1993.

[4] P. K. Chan and K. Karplus. Computing signal delay in general rc networks by tree/link partitioning. In *DAC '89: Proceedings of the 26th ACM/IEEE conference on Design automation*, pages 485–490. ACM Press, 1989.

[5] H. Chang and S. Sapatnekar. Statistical timing analysis considering spatial correlations using a single pert-like traversal. In *Proc. Design Automation Conf*, 2003.

[6] J. P. Fishburn. Clock skew optimization. *IEEE Trans. on Computers*, 39(7):945–951, July 1990.

[7] G. H. Golub and C. F. van Loan. *Matrix Computations*. The Johns Hopkins University Press, Baltimore, Maryland, third edition edition, 1996.

[8] R. Gupta, B. Tutuianu, B. Krauter, and L. T. Pillage. The Elmore delay as a bound for RC trees with generalized input signals. In *Proc. Design Automation Conf*, pages 364–369, June 1995.

[9] S. Lin and C. K. Wong. Process-variation-tolerant clock skew minimization. In *Proc. Int. Conf. on Computer Aided Design*, pages 284–288, 1994.

[10] Y. Liu, S. Nassif, L. Pileggi, and A. Strojwas. Impact of interconnect variations on the clock skew of a gigahertz microprocessor. In *Proc. Design Automation Conf*, 2000.

[11] D. F. Morrison. *Multivariate Statistical Methods*. New York: McGraw-Hill, 1976.

[12] S. Pullela, N. Menezes, and L. T. Pillage. Reliable non-zero skew clock trees using wire width optimization. In *Proceedings of the 30th international conference on Design automation*, pages 165–170. ACM Press, 1993.

[13] A. Rajaram, J. Hu, and R. Mahapatra. Reducing clock skew variability via cross links. In *Proceedings of the 41st annual conference on Design automation*, pages 18–23. ACM Press, 2004.

[14] P.J. Restle, T.G. McNamara, D.A. Webber, P.J. Camporese, K.F. Eng, K.A. Jenkins, D.H. Allen, M.J. Rohnand, 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. In *IEEE Journal of Solid-State Circuits*, volume 36 (5), pages 792 – 799, May 2001.

[15] H. Su and S. Sapatnekar. Hybrid structured clock network construction. In *Proc. Int. Conf. on Computer Aided Design*, pages 333 – 336, 2001.

[16] C.-W. A. Tsao and C.-K. Koh. UST/DME: A clock tree router for general skew constraints. *ACM (TODAES)*, 7:359–379, 2002.

[17] D. Velenis, E. Friedman, and M. Papaefthymiou. A clock tree topology extraction algorithm for improving the tolerance of clock distribution networks to delay uncertainty. In *Proc. IEEE Int. Symp. on Circuits and Systems*, pages 4.422–4.425, May 2001.