Process Variation Aware Clock Tree Routing

[Extended Abstract]

Bing Lu
Cadence Design Sys. Inc.
35 Spring Street
New Providence, NJ 07974
bingl@cadence.com

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

Gary Ellis
IBM Microelectronics
11400 Burnet Road
Austin, TX 78758
ellinote@us.ibm.com

Haihua Su
IBM Austin Research Lab
11400 Burnet Road
Austin, TX 78758
haihua@us.ibm.com

ABSTRACT

Fast progress on VLSI technology makes clock skew more susceptible to process variations. We propose DME/BST based algorithms for clock tree routing to improve skew tolerance to process variations. The worst case skew due to process variations is estimated and employed to guide the decision making during the routing. Our method can be applied to general non-zero skew requirements. Minimizing total wirelength is considered as a secondary objective at the same time. Experimental results on benchmark circuits demonstrate great improvement on process variation tolerance through our algorithms.

Categories and Subject Descriptors

B.7.2 [**Hardware**]: Integrated Circuits—*Design Aids*

General Terms

Algorithms, Performance

Keywords

VLSI, interconnect, physical design, clock tree synthesis

1. INTRODUCTION

In synchronous VLSI designs, the pace of data transfer is generally coordinated by clock signals, thus clock network quality plays a key role in determining VLSI system performance. Conventional clock designs have placed emphasis on seeking zero clock skew, since the clock skew sets a lower bound on clock cycle time. Previous work has focused on zero-skew clock design under either pathlength or Elmore delay model [8, 9, 16]. Further, since the clock network is a major source of power consumption, reducing clock network wirelength to lower power consumption is always desired. The DME(Deferred-Merge Embedding) algorithm is proposed in [1, 2, 7] to achieve zero-skew with a minimal wirelength. In practice, however, circuits are able to operate correctly within some non-zero skew bound [4]. In [4], a bounded skew clock tree(BST) algorithm extends DME to further reduce clock tree wirelength.

ISPD'03, April 6-9, 2003, Monterey, California, USA.

In reality, designed skews may not be guaranteed after chip manufacturing because the fabricated wire width may be different from expected values due to process variations such as etching errors, mask misalignment and spot defects. Process variations are mitigated in clock design through buffer insertion/sizing[3, 18] and non-tree topology[10]. Other types of topologies such as mesh or combined mesh-tree [6, 14, 15] have also been proposed to pursue process variation tolerant zero-skew. Even though non-tree structures are generally effective in overcoming process variations, it does not permit the widely applied clock gating technique and excessive usage of non-tree structures may aggravate the already severe power consumption problem for the clock network.

In modern VLSI circuit designs, clock skew is more and more susceptible to process variations because of the increasingly high clock frequency, large chip area and shrinking feature size. This requires that skew tolerance to process variation needs to be handled in a meticulous manner. Thus, a process variation aware clock *tree* routing is needed in complement with non-tree structure, buffer insertion/sizing and wire sizing to achieve a greater process variation tolerance. Our experimental results show that a careful clock tree design itself will make a great difference on skew tolerance to process variations. In [17], an abstract tree topology for improving process variation tolerance is suggested based on the observation that skew between two clock sinks will not be affected by the process variations along the shared portion of their source-sink paths. Hence, skew tolerance to process variation may be improved through letting a pair of clock sinks with a smaller skew permissible range share a greater portion of their source-sink paths in the abstract topology. However, this work uses a crude delay model neglecting sink location/capacitance, and more importantly it is incomplete in that it does not show how to implement the abstract topology through physical routing.

In this work, we improve upon these deficiencies by focusing on the construction of the clock *tree* to improve skew tolerance with respect to process variations. The process variation tolerant tree can be adopted as either a complete clock network or a local clock system driven by a buffer or a mesh of a global clock network. The primary objective is to minimize the maximum skew violation due to process variations and the routing wirelength is minimized as secondary objective. We propose DME[1, 2, 7] and BST [4] based algorithms to solve these problems. The major distinction from DME/BST is that a worst case process variation limit is considered in addition to skew permissible ranges during the routing. In experiments, we compared our method with a naive extension of DME and our method exhibits great improvement on tolerance to process variations.

The remainder of this paper is organized as follows. Section 2 presents problem formulation. The algorithm of minimizing skew violation is described in Section 3. Then the experimental results are shown in Section 4. Our algorithm is extended to minimize wirelength subject to skew bounds in Section 5. Finally, Section 6 summarizes the primary results of this paper and outlines further research.

2. PROBLEM FORMULATION

Let $S = \{s_1, s_2, \ldots, s_n\} \subset \Re^2$ denote a set of clock sinks in the Manhattan plane and s_0 be a clock source if given. An abstract topology $T_t(S)$ is defined as a rooted binary tree with its n leaves be $s_1, s_2, \ldots, s_n$. A physical embedding of $T_t(S)$ is a rooted rectilinear Steiner tree $T_e(S)$ such that each internal node $v_i \in T_t(S)$ is mapped to a location $l(v_i)$ in the Manhattan plane. In a binary rooted tree, each node v is connected to its unique parent by an edge e_v. Suppose the cost of e_v is its wirelength $\mid e_v \mid$, then the overall cost of $T_e(S)$ is the total wirelength of the edges in $T_e(S)$.

For the wire width variations, both global spatial variations and local random variations are appended to the nominal wire width w_0 through the following expression[15]:

$$w = w_0 + \lambda x + \theta y + \delta \quad (1)$$

where coefficient λ indicates the spatial variation along the x coordinates and coefficient θ models the variation along the y direction. The fourth term δ is a random variable following a normal probability distribution with mean value 0 and standard deviation σ. As an approximation, we assume that the wire width w of is bounded by $0 < W_l \leq w \leq W_u$ where $W_l = w_0 + \lambda x + \theta y - 3\sigma$ and $W_u = w_0 + \lambda x + \theta y + 3\sigma$.

For each sink s_i, let $t(s_0, s_i)$ denote the delay time from root s_0 to s_i. Then for any two sinks s_i and s_j, the clock skew $t_{skew}(s_i, s_j)$ between them is[1] $t(s_0, s_i) - t(s_0, s_j)$. Each skew $t_{skew}(s_i, s_j)$ is allowed to change within a permissible range $[LPR_{ij}, UPR_{ij}]$ without affecting circuit performance[12, 17]. The *skew violation* is defined as $SV_{ij} = \max(LPR_{ij} - t_{skew}(ij), t_{skew}(ij) - UPR_{ij})$. In order to improve tolerance to process variation, we will first solve the minimal skew violation under process variation problem as follows.

Minimizing Skew Violation (MinSV) Problem: *Given a set $S = \{s_1, s_2, \ldots, s_n\} \subset \Re^2$ of clock sinks, skew permissible ranges for all pairs of clock sinks, find a clock routing tree $T_e(S)$ such that the maximum skew violation among all pairs of sinks is minimized when wire width varies between W_l and W_u.*

Note that the source location s_0 is not included in our formulation since our proposed methods can handle any prescribed s_0 transparently. Even though we consider only wire width variation here, our method can be easily extended to account for wire thickness and spacing variations. The interconnect delay is evaluated through Elmore delay model.

3. MINIMAL SKEW VIOLATION CLOCK TREE

Given an abstract topology, the procedure to build the minimal skew violation clock tree is divided into two phases as [2, 4, 7, 9]: bottom-up tree of merging segments construction and top-down routing tree embedding. During the bottom-up phase, a set of candidate locations, which is called merging segment, is found for each internal node of the given abstract topology in order to achieve certain skew objectives. Every candidate location along a single merging segment should yield the same skew behavior and the tree obtained is called *tree of merging segments*. Examples of abstract topology and tree of merging segments are illustrated in Figure 1. In the top-down embedding process, one candidate location is selected for each merging segment such that the total wirelength is minimized while the skew performance obtained in the bottom-up phase is retained.

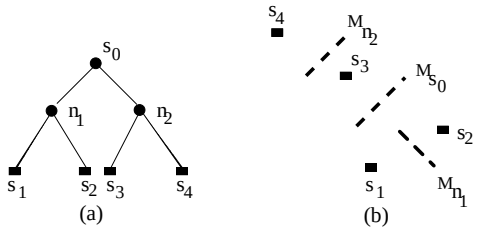


Figure 1: (a) abstract topology tree, (b) tree of merging segments for the abstract topology of (a).

3.1 Notations and Definitions

DEFINITION 1. ***Segment Distance:*** *For any two merging segments $\mathbf{M}(n_i)$ and $\mathbf{M}(n_j)$, the* segment distance *between $\mathbf{M}(n_i)$ and $\mathbf{M}(n_j)$ is defined as $D_{ij} = d(\mathbf{M}(n_i), \mathbf{M}(n_j)) = \{min \ \ d_r(v_a, v_b) \mid v_a \in \mathbf{M}(n_i), v_b \in \mathbf{M}(n_j)\}$, where $d_r(v_a, v_b) = |x_a - x_b| + |y_a - y_b|$ is the rectilinear distance between v_a and v_b.*

DEFINITION 2. ***Nearest Pair:*** *For any two merging segments $\mathbf{M}(n_1)$ and $\mathbf{M}(n_2)$, a pair of points $v_1 \in \mathbf{M}(n_1)$ and $v_2 \in \mathbf{M}(n_2)$ is defined as nearest pair if $d_r(v_1, v_2) = d(\mathbf{M}(n_1), \mathbf{M}(n_2))$.*

DEFINITION 3. ***Shortest Distance Region:*** *For any two merging segments $\mathbf{M}(n_1)$ and $\mathbf{M}(n_2)$, the shortest distance region $\mathbb{R}(\mathbf{M}(n_1), \mathbf{M}(n_2))$ between $\mathbf{M}(n_1)$ and $\mathbf{M}(n_2)$ is the set of points with the minimum sum of Manhattan distances to $\mathbf{M}(n_1)$ and $\mathbf{M}(n_2)$, i.e., $\mathbb{R}(\mathbf{M}(n_1), \mathbf{M}(n_2)) = \{p \mid d_r(p, \mathbf{M}(n_1)) + d_r(p, \mathbf{M}(n_2)) = d(\mathbf{M}(n_1), \mathbf{M}(n_2))\}$.*

3.2 Tree of Merging Segments Construction

Our objective in searching the merging segments is to minimize the skew violation due to process variations. More specifically speaking, for an internal node n with two children nodes n_i and n_j, we look for a merging segment for n such that the skew violation between any sink $s_r \in T_{ni}$ and sink $s_l \in T_{nj}$ is minimized. We use T_{ni} and T_{nj} to denote the subtrees rooted at n_i and n_j, respectively. In order to guide the searching for the merging segment, we need to estimate the range of the skew between s_r and s_l. Because of process variations, the skew between a particular pair of sinks $s_r \in T_{ni}$ and $s_l \in T_{nj}$ is not a unique value, instead it is within a range $[\underline{t}_{skew}(s_r, s_l), \overline{t}_{skew}(s_r, s_l)]$ and different sink pairs may have different skew ranges under process variations.

To minimize the maximum skew violations among all sink pairs, we need to estimate skew ranges for all sink pairs between two subtrees[19]. For example, if there are 8 sinks in T_{nr} and 7 sinks in T_{nl}, then skew ranges for 56 pairs need to be estimated. This is in contrast to the traditional zero-skew routing where only skew between one pair of sinks from two subtrees need to be computed. However, estimating skew ranges for all pairs of sinks between two subtrees greatly increases the computation time. Hence, we trace

[1] Some works treat the absolute value of the delay difference as the skew without differentiating the signs.

the skew range for only the most critical sink pair between two subtrees. Let $P_{rl} = UPR_{rl} - LPR_{rl}$ and P_{min} be the minimum skew permissible range among all pairs of sinks for the given net. The criticality between a sink pair s_r and s_l is defined as:

$$Criticality_{rl} = \gamma \frac{P_{min}}{P_{rl}} + (1-\gamma)\frac{d_{rl}}{d_{max}} \quad (2)$$

where d_{rl} is the rectilinear distance between s_r and s_l, d_{max} is the maximum sink pair distance among all pairs of sinks of the total net, and γ is a constant weighting factor. On the right-hand side of the above equation, the first term represents the criticality due to the skew permissible range and the second term describes the criticality from the spatial distance, since the skew between two sinks far apart from each other is more susceptible to process variations. We can pre-select the most critical sink pair between any two subtrees according to equation (2) and perform the skew estimation for only this critical sink pair when the merging segment for their corresponding subtrees is searched.

The skew range between s_r and s_l can be obtained if the minimum and the maximum delay from node n to s_r and s_l are available. Estimating the minimum interconnect delay due to wire width variation is actually very similar to the delay driven wire sizing problem[5]. The difference is that the wire width variation range due to process variation is much smaller than that in the wire sizing problem. Such small variation range allows us to employ some simple wire sizing scheme to estimate the minimum interconnect delay. It is stated in [5] that single width sizing is a reasonable approximation to the optimal wire sizing. For a wire segment of length l and width w, its resistance is rl/w and its capacitance is clw, where r and c are coefficients for resistance and capacitance. If it has a capacitive load C_L, its interconnect delay in Elmore delay model is $t = \frac{1}{2}rcl^2 + \frac{rl}{w}C_L$. Evidently, the minimum delay is obtained when $w = W_u$ and the maximum delay occurs if $w = W_l$. When we estimate the minimum delay from a node n to a sink s_r in a routing *tree* instead of a 2-pin path, the wire width along the path from n to s_r is W_u while wire width for any branching segment not on this path has to be the minimum W_l. Since those branching segments are pure capacitive load to the $n \rightarrow s_r$ path, the minimum wire width implies the minimum load. The maximum delay due to process variation in a routing tree can be estimated similarly. Another observation is that process variations at any wires not in T_n do not affect the skew between s_r and s_l. Therefore, the merging segments for n's parent/ascendant nodes found in later searchings may maintain the skews between any sink pairs between T_{nr} and T_{nl}.

Once the skew range $[\underline{t}_{skew}(s_r, s_l), \overline{t}_{skew}(s_r, s_l)]$ for the most critical sink pair $s_r \in T_{nr}$ and $s_l \in T_{nl}$ is obtained, we can choose the merging segment for the internal node n, which is the parent node of n_r and n_l, such that the center of this skew range coincides with the center of permissible range $[LPR_{rl}, UPR_{rl}]$ for s_r and s_l. When the skew range is greater than the skew permissible range, such selection minimize the maximum positive skew violation. If the skew range is smaller than the permissible range, this selection method maximize the safety margin.

Suppose we are working on the tree of merging segments for a node n with children n_i and n_j. Furthermore, let $T_{\mathbf{M}(n_i)}$ and $T_{\mathbf{M}(n_j)}$ denote two subtrees of merging segments rooted at n_i's merging segment $\mathbf{M}(n_i)$ and n_j's merging segment $\mathbf{M}(n_j)$, respectively. Let $v_i \in \mathbf{M}(n_i)$ and $v_j \in \mathbf{M}(n_j)$. The searching for the merging point v is started by modeling the minimum(maximum) delay $\underline{t}(v, s_r)(\overline{t}(v, s_r))$ and $\underline{t}(v, s_l)(\overline{t}(v, s_l))$ from v to s_r and s_l, respectively.

$$\begin{cases} \underline{t}(v, s_r) = \underline{t}(v_i, s_r) + \frac{rc}{2}\cdot|e_{v_i}|^2 + \frac{r\cdot|e_{v_i}|}{W_u}\cdot\underline{C}_{T_i} \\ \overline{t}(v, s_r) = \overline{t}(v_i, s_r) + \frac{rc}{2}\cdot|e_{v_i}|^2 + \frac{r\cdot|e_{v_i}|}{W_l}\cdot\overline{C}_{T_i} \\ \underline{t}(v, s_l) = \underline{t}(v_j, s_l) + \frac{rc}{2}\cdot|e_{v_j}|^2 + \frac{r\cdot|e_{v_j}|}{W_u}\cdot\underline{C}_{T_j} \\ \overline{t}(v, s_l) = \overline{t}(v_j, s_l) + \frac{rc}{2}\cdot|e_{v_j}|^2 + \frac{r\cdot|e_{v_j}|}{W_l}\cdot\overline{C}_{T_j} \end{cases}$$

where e_{v_i} is the edge between v and v_i, e_{v_j} is the edge between v and v_j. Notation $\underline{C}_{T_i}$ is the tree capacitance of $T_{\mathbf{M}(n_i)}$ where the wire width along the path from v_i to s_r is W_u and any other wire width in this subtree is w_l. Notations of $\overline{C}_{T_i}$, $\underline{C}_{T_j}$ $\overline{C}_{T_j}$ are defined similarly. Therefore, the lower bound $\underline{t}_{skew}(s_r, s_l)$ and upper bound $\overline{t}_{skew}(s_r, s_l)$ of the skew between s_r and s_l are

$$\begin{cases} \underline{t}_{skew}(s_r, s_l) = \underline{t}(v, s_r) - \overline{t}(v, s_l) \\ \overline{t}_{skew}(s_r, s_l) = \overline{t}(v, s_r) - \underline{t}(v, s_l) \end{cases}$$

If we let $|e_{v_i}| = z$ and $|e_{v_j}| = D_{ij} - |e_{v_i}| = D_{ij} - z$, where $D_{ij} = d(\mathbf{M}(n_i), \mathbf{M}(n_j))$, then

$$\begin{cases} \underline{t}^z_{skew}(s_r, s_l) = rc\cdot z\cdot D_{ij} + rz(\frac{\underline{C}_{T_i}}{W_u} + \frac{\overline{C}_{T_j}}{W_l}) + \underline{K} \\ \overline{t}^z_{skew}(s_r, s_l) = rc\cdot z\cdot D_{ij} + rz(\frac{\overline{C}_{T_i}}{Wl} + \frac{\underline{C}_{T_j}}{W_u}) + \overline{K} \end{cases}$$

where $\underline{K} = \underline{t}(v_i, s_r) - \overline{t}(v_j, s_l) - (\frac{rc\cdot D_{ij}^2}{2} + \frac{r\cdot D_{ij}\cdot\overline{C}_{T_j}}{W_l})$ and $\overline{K} = \overline{t}(v_i, s_r) - \underline{t}(v_j, s_l) - (\frac{rc\cdot D_{ij}^2}{2} + \frac{r\cdot D_{ij}\cdot\underline{C}_{T_j}}{W_u})$.

Thus, for any $0 \le z \le D_{ij}$, the skew is within the bound $\mathbf{B}_z = [\underline{t}^z_{skew}(s_r, s_l), \overline{t}^z_{skew}(s_r, s_l)]$. In particular, if $|e_{v_i}| = 0$, the minimum and the maximum skew are given as $\underline{t}^0_{skew}(s_r, s_l) = \underline{K}$ and $\overline{t}^0_{skew}(s_r, s_l) = \overline{K}$. When $|e_{v_i}| = D_{ij}$, then

$$\begin{cases} \underline{t}^{D_{ij}}_{skew}(s_r, s_l) = \underline{t}(v_i, s_r) - \overline{t}(v_j, s_l) + (\frac{rc\cdot D_{ij}^2}{2} + \frac{r\cdot D_{ij}\cdot\underline{C}_{T_i}}{W_u}) \\ \overline{t}^{D_{ij}}_{skew}(s_r, s_l) = \overline{t}(v_i, s_r) - \underline{t}(v_j, s_l) + (\frac{rc\cdot D_{ij}^2}{2} + \frac{r\cdot D_{ij}\cdot\overline{C}_{T_i}}{W_l}) \end{cases}$$

Suppose that the skew permissible range for s_r and s_l is $[LPR_{rl}, UPR_{rl}]$ and let $M^0_{skew} = [\underline{t}^0_{skew}(s_r, s_l) + \overline{t}^0_{skew}(s_r, s_l)]/2$, $M^{D_{ij}}_{skew} = [\underline{t}^{D_{ij}}_{skew}(s_r, s_l) + \overline{t}^{D_{ij}}_{skew}(s_r, s_l)]/2$, and $M_{PR} = [LPR_{rl} + UPR_{rl}]/2$. In other words, M_{PR} is the center of the permissible range, $M^0_{skew}(M^{D_{ij}}_{skew})$ is the center of the skew range when $z = 0 (z = D_{ij})$. There exist three scenarios (Figure 2): (i) $M^0_{skew} \le M_{PR} \le M^{D_{ij}}_{skew}$; (ii) $M^{D_{ij}}_{skew} < M_{PR}$; (iii) $M^0_{skew} > M_{PR}$

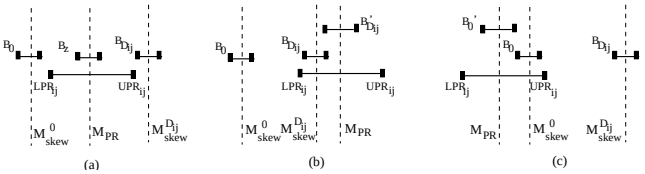


Figure 2: (a) $M^0_{skew} \le M_{PR} \le M^{D_{ij}}_{skew}$**, (b)** $M^{D_{ij}}_{skew} < M_{PR}$**, (c)** $M^0_{skew} > M_{PR}$**.**

Let $Q = \underline{t}(v_i, s_r) + \overline{t}(v_i, s_r) - (UPR_{rl} + LPR_{rl} + \underline{t}(v_j, s_l) + \overline{t}(v_j, s_l))$.

In case (i), no wire snaking[16] is necessary and we can obtain

$$z = \frac{rcD_{ij}^2 + r\cdot D_{ij}(\frac{\overline{C}_{T_j}}{W_l} + \frac{\underline{C}_{T_j}}{W_u}) - Q}{r(2c\cdot D_{ij} + \frac{\underline{C}_{T_i} + \underline{C}_{T_j}}{W_u} + \frac{\overline{C}_{T_i} + \overline{C}_{T_j}}{W_l})} \quad (3)$$

by aligning the center of the skew range to the center of the permissible range. This alignment is equivalent to solving the following equation:

$$\underline{t}^z_{skew}(s_r, s_l) - LPR_{rl} = UPR_{rl} - \overline{t}^z_{skew}(s_r, s_l)$$

For case (ii), we let $|e_{v_j}| = 0$ and need to extend the edge length of e_{v_i} so that $|e_{v_i}| > D_{ij}$. By solving

$$\underline{t}^{|e_{v_i}|}_{skew}(s_r, s_l) - LPR_{rl} = UPR_{rl} - \overline{t}^{|e_{v_i}|}_{skew}(s_r, s_l)$$

we have

$$|e_{v_i}| = \frac{\sqrt{[r(\frac{\overline{C}_{T_i}}{W_l} + \frac{C_{T_i}}{W_u})]^2 - 4rcQ} - r(\frac{\overline{C}_{T_i}}{W_l} + \frac{C_{T_i}}{W_u})}{2rc}. \quad (4)$$

Since $M^{D_{ij}}_{skew} < M_{PR}$, $Q < 0$ and therefore $[r(\frac{\overline{C}_{T_i}}{W_l} + \frac{C_{T_i}}{W_u})]^2 - 4rcQ > 0$. For the same reason, the RHS of equation (4) is guaranteed to be positive and there is always a feasible solution.

Similarly, for case (iii), we let $|e_{v_i}| = 0$ and extend the edge length of e_{v_j} so that $|e_{v_j}| > D_{ij}$ and by solving

$$\underline{t}^{|e_{v_j}|}_{skew}(s_r, s_l) - LPR_{rl} = UPR_{rl} - \overline{t}^{|e_{v_j}|}_{skew}(s_r, s_l)$$

we have

$$|e_{v_j}| = \frac{\sqrt{[r(\frac{\overline{C}_{T_j}}{W_l} + \frac{C_{T_j}}{W_u})]^2 + 4rcQ} - r(\frac{\overline{C}_{T_j}}{W_l} + \frac{C_{T_j}}{W_u})}{2rc}. \quad (5)$$

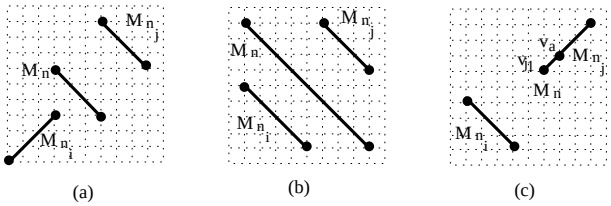
From the above analysis, a merging point minimizing skew violation for s_r and s_l always exists. A feasible merging segment is a collection of all such merging points of s_r and s_l. In addition, $|e_{v_i}| + |e_{v_j}|$ is defined as merging cost. It is obvious that for each pair of v_i and v_j, the feasible merging segment of v is actually the intersection of two Manhattan circles[2] centered at v_i and v_j, respectively. Consequently, it is a Manhattan arc[3]. The merging segment $\mathbf{M}(n)$ is chosen among these feasible merging segments in a way such that the tolerance violation is minimized and merging cost is also minimized. If v_i and v_j is a nearest pair, then the merging cost is minimized. Therefore, it is sufficient to select nearest points of $\mathbf{M}(n_i)$ and $\mathbf{M}(n_j)$ to construct $\mathbf{M}(n)$. Similar to [1], we have the following Lemma:

LEMMA 1. *If two merging segment $\mathbf{M}(n_i)$ and $\mathbf{M}(n_j)$ do not intersect, then there exists a nearest pair $v_i \in \mathbf{M}(n_i)$ and $v_j \in \mathbf{M}(n_j)$ such that either v_i or v_j or both is an end point of $\mathbf{M}(n_i)$ or $\mathbf{M}(n_j)$.*

According to Lemma 1, the merging segment $\mathbf{M}(n)$ is built as follows. Suppose $\mathbf{M}(n_i)$'s end points are v_{i_1} and v_{i_2} and $\mathbf{M}(n_j)$'s end points are v_{j_1} and v_{j_2}. First, the rectilinear distances $d_r(v_{i_1}, v_{j_1})$, $d_r(v_{i_1}, v_{j_2})$, $d_r(v_{i_2}, v_{j_1})$, and $d_r(v_{i_2}, v_{j_2})$ are computed. Then all pairs of points whose rectilinear distances are equal to $d(\mathbf{M}(n_i), \mathbf{M}(n_j))$ are selected. For each such pair of points, the feasible merging segment is found. Any two feasible merging segments are merged if they overlap. The process repeats until no feasible merging segment overlap with each other. Finally, the longest feasible merging segment is picked up as merging segment if there are more than one feasible merging segments left. Basically, if there is no need to extend edge, the resulting merging segment

[2]**Manhattan Circle** is a collection of all points with equal distance from a point (center). Basically, Manhattan circle is a square rotated by 45 degree.

[3]**Manhattan Arc** is a line segment of a Manhattan circle. Its slope is either -1 or $+1$.



(a) (b) (c)

Figure 3: (*a*) An example such that $D_{ij} = d(\mathbf{M}(n_i), \mathbf{M}(n_j)) = 9$ and $\mathbf{M}(n_i)$ is not parallel to $\mathbf{M}(n_j)$, suppose no edge extension is needed, $|e_{v_i}|$ is found to be 3, (*b*) An example such that $D_{ij} = d(\mathbf{M}(n_i), \mathbf{M}(n_j)) = 9$ and $\mathbf{M}(n_i)$ is parallel to $\mathbf{M}(n_j)$, suppose no edge extension is needed, $|e_{v_i}|$ is 4, (*c*) An example such that $D_{ij} = d(\mathbf{M}(n_i), \mathbf{M}(n_j)) = 7$ and $\mathbf{M}(n_i)$ is not parallel to $\mathbf{M}(n_j)$, suppose edge e_{v_i} needs to be extended to $|e_{v_i}| = 9$.

$\mathbf{M}(n)$ is inside the shortest distance region $\mathbb{R}(\mathbf{M}(n_i), \mathbf{M}(n_j))$ (Figure 3(*a*) and (*b*)). Otherwise, some adjustment is applied to $\mathbf{M}(n)$ as follows. For instance, in Figure 3 (*c*), $\mathbf{M}(n)$ contains only v_{j_1} is merging segment if it is restricted within $\mathbb{R}(\mathbf{M}(n_i), \mathbf{M}(n_j))$. In order to obtain more flexibility, $\mathbf{M}(n)$ is allowed to extend along $\mathbf{M}(n_j)$ such that all points v_c of $\mathbf{M}(n_j)$ with $d_r(v_c, \mathbf{M}(n_i)) \leq 9$ may be included in $\mathbf{M}(n)$. Therefore, in Figure 3 (*c*), $\mathbf{M}(n)$ is the line segment $v_{j_1} v_c$.

3.3 Algorithms and Their Analyses

Algorithm 1 Merging_Segment_Construction(lnode, rnode).

Input: Two nodes lnode and rnode of $T_t(S)$
Output: A routing tree with internal nodes embedded.

1. pick a lnode's clock sink s_r ;
2. pick a rnode's right sinks_l ;
3. $D \leftarrow d(\mathbf{M}(lnode), \mathbf{M}(rnode))$;
4. $LPR \leftarrow$ lower bound of permissible range of s_r and s_l;
5. $UPR \leftarrow$ upper bound of permissible range of s_r and s_l;
6. compute $\underline{t}^0_{skew}$, $\overline{t}^0_{skew}$, $\underline{t}^D_{skew}$ and $\overline{t}^D_{skew}$;
7. $M^0_{skew} \leftarrow (\underline{t}^0_{skew} + \overline{t}^0_{skew})/2$;
8. $M^D_{skew} \leftarrow (\underline{t}^D_{skew} + \overline{t}^D_{skew})/2$;
9. $M_{PR} \leftarrow (LPR + UPR)/2$;
10. **if** $(M^0_{skew} \leq M_{PR} \leq M^D_{skew})$ **then**
    compute $|e_{lnode}|$ according to Equation (3);
    $|e_{rnode}| = D - |e_{lnode}|$;
11. **else if** $(M^D_{skew} < M_{PR})$ **then**
    extend $e_{lnode}|$ according to Equation (4);
    $|e_{rnode}| = 0$;
12. **else**
    $e_{lnode}| = 0$;
    extend $e_{rnode}|$ according to Equation (5);
13. **end if**
14. construct the merging segment according to rules described in Section 3.2 and return the resulting merging segment;

To build the tree of merging segments, the initial clock abstract topology tree is processed in bottom-up fashion. Merging segment of an internal node is based on its left and right children's merg-

ing segments. Merging_Segment_Construction (Algorithm 1) is the main routine constructing such merging segments. Inside the route, sink s_r of lnode and sink s_l of rnode are identified, distance between $\mathbf{M}(lnode)$ and $\mathbf{M}(rnode)$ are computed, permissible range of s_r and s_l is fetched and all skews are calculated. Finally, the merging segment is formed according to rules described in Section 3.2.

After tree of merging segments is finalized, the clock routing tree is embedded top-down same as in DME [1]. First the location for clock source is determined, then all descendent's locations are recursively located. In particular, each clock sink's location is itself and each internal node's location is derived from its parent's location and the condition whether there exists extension.

In fact, Algorithm 1 is analogous to a postorder tree traversal. The running time of postorder is $O(N)$ where N is the number of edges of the tree. Furthermore, all computations in the algorithm can be done in constant time. Therefore, both tree of merging segments and clock routing embedding can be made in time $O(N)$. Note that if a tree has n leaves, the number of the tree's edges is usually $O(n)$.

4. EXPERIMENTAL RESULTS

The algorithm of MinSV have been tested on five benchmark circuits widely employed in the literature [4, 7, 8, 9, 15, 16]. As [15], the wire widths are scaled under the 0.18 micron technology while the sink's loading capacitances remain the same. We assume that the wire width follows the variation defined in equation (1) with nominal width $w_0 = 0.54$, standard deviation $\sigma = 0.162$ for the local variation term δ, and spatial variation coefficient $\lambda = \theta = \frac{3\sigma}{d_{max}}$ where d_{max} is the maximum rectilinear distance among all pair of sinks. The weighting factor γ in selecting the critical sink pairs is 0.5. The wire resistance per unit length is $0.0042/w\Omega$ where w is the wire width, and per unit length and width capacitance is $3.18e-6pF$, the driver's resistance is 18.2Ω, and the unit of pin's loading cap is $0.01pF$. The initial clock routing abstract topology is constructed by applying the method introduced in [16], which recursively partition sinks into two equal parts in horizontal and vertical directions alternately. However, our algorithms are not dependent on specific topologies. Any other methods such as [1, 2, 7, 8, 9, 11, 12, 13, 17] can be used to generate the initial abstract topology. The experiments are carried out on a SUN Blade-100 workstation with 512M memory.

Since there is no previous work which addresses the construction of process variation tolerant clock *trees*, we have implemented an extended DME algorithm for comparisons. Instead of seeking zero-skew, as in the original DME algorithm, the objective has been modified to target the skew between each sink pair to be at the center of its permissible range. Since DME algorithm itself is not capable of dealing with random process variations, the local variation term δ is ignored for the skew estimation during the DME algorithm. Even though the corresponding change on the algorithm is trivial, skew tolerance to any design uncertainties is improved due to the increased guarding band. However, the wire width variation is not handled directly as in our algorithms.

Different permissible ranges for each pair of sinks were applied using uniformly distributed random numbers over [0.0, 1.0]. For each pair of sinks s_i and s_j, LPR_{ij} is negative and UPR_{ij} is positive, respectively. We use a constant SCALE to control the LPR_{ij} and UPR_{ij} (= random number * SCALE). For each SCALE, two types of cases are considered: (1) LPR_{ij} and UPR_{ij} are symmetric, that is $|LPR_{ij}| = |UPR_{ij}|$; (2) $|LPR_{ij}|$ and $|UPR_{ij}|$ differ, but not a lot, that is $|LPR_{ij} + UPR_{ij}| \leq \frac{SCALE}{10}$. Results for $SCALE = 1$ and $SCALE = 0.1$ are shown in Tables (1–2), where S means all permissible ranges are symmetric while NS indicates that all tolerance ranges are not symmetric.

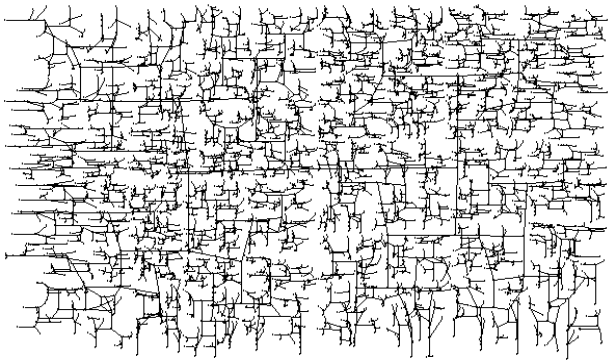


Figure 4: Clock routing tree for $r5$.

To compare the tolerance to the process variation, two clock routing trees are constructed with the same set of permissible ranges by DME and MinSV, respectively. For each routing tree, Elmore delay of each sink and max skew and max skew violation among sinks are computed with width variation. Tables (1–2) exhibit experimental results for benchmarks $r1$–$r5$. The total wirelength are listed in column 2 and 3. The wirelength generated from MinSV is sometimes greater and sometimes less than the wirelength resulted from DME, but all of the differences are on a negligible level. The skew performance is evaluated through the number of skew violations and the maximum skew violations. The number of skew violations are shown in column 4 and 5, and the percentage improvements from MinSV are in column. Our algorithm constantly outperforms the DME by large margin of $32\% - 57\%$ in term of number of skew violations. In column 7 and 8, the experimental results show that MinSV always yields significantly less value on the maximum skew violation and the average improvement from MinSV is 32%. The rightmost two columns show the CPU time which indicate that our algorithm runs at speed about the same as DME.

Comparing the results between symmetric permissible range and non-symmetric permissible range, we can see that non-symmetric permissible ranges usually cause greater wirelength and worse skew violations. This is due to the fact that wire snaking occurs more often when we align the skews to non-symmetric permissible ranges. The smaller SCALE value in Table (2) implies tighter permissible ranges and consequently the skew violations are worse. Figure 4 shows the embedded clock routing tree for the benchmark $r5$.

5. MINIMIZING WIRELENGTH SUBJECT TO SKEW CONSTRAINTS

Besides improving process variation tolerance, it is desirable to minimize total wirelength for the clock tree and thereby reduce power dissipation. Therefore, we further propose an algorithm on minimizing wirelength subject to skew constraints.

Minimizing Wirelength Subject to Skew Constraints (MinWSC) Problem: *Given a set $S = \{s_1, s_2, \ldots, s_n\} \subset \Re^2$ of clock sinks, a set of skew permissible ranges for each pair of clock sinks, find a clock routing tree $T_e(S)$ such that the total wirelength is minimized while the maximum positive skew violation among all pairs of sinks is non-positive when wire width varies in $[W_l, W_u]$.*

	total wirelength (μm)		#skew violations			max skew violation (ns)			CPU(s)	
	DME	MinSV	DME	MinSV	imprv	DME	MinSV	imprv	DME	MinSV
r1-S	168376	168251	89	40	55%	0.141	0.126	11%	0.1	0.1
r1-NS	179702	179630	102	58	43%	0.082	0.057	30%	0.1	0.1
r2-S	351085	352044	190	91	52%	0.365	0.206	44%	0.2	0.2
r2-NS	383169	384651	243	157	35%	0.492	0.392	20%	0.2	0.2
r3-S	440309	440989	264	114	57%	0.302	0.250	17%	0.4	0.4
r3-NS	494090	494161	318	189	41%	0.329	0.274	17%	0.4	0.4
r4-S	885678	885523	496	230	54%	1.543	0.454	71%	2.2	2.2
r4-NS	1009232	1008957	659	383	42%	0.588	0.359	39%	2.1	2.1
r5-S	1309794	1311689	834	393	53%	1.449	0.649	57%	6.9	7.0
r5-NS	1552413	1552719	1139	671	41%	1.215	1.101	9%	6.9	6.9

Table 1: The results when SCALE is 1.

	total wirelength (μm)		#skew violations			max skew violation (ns)			CPU(s)	
	DME	MinSV	DME	MinSV	imprv	DME	MinSV	imprv	DME	MinSV
r1-S	168376	168252	161	100	38%	0.142	0.127	11%	0.1	0.1
r1-NS	169835	170020	166	101	39%	0.098	0.093	5%	0.1	0.1
r2-S	351085	352044	354	225	36%	0.371	0.207	44%	0.2	0.2
r2-NS	349704	351290	351	235	33%	0.171	0.060	65%	0.2	0.2
r3-S	440309	440990	462	285	38%	0.320	0.251	22%	0.4	0.4
r3-NS	443812	443676	491	311	37%	0.302	0.248	18%	0.4	0.4
r4-S	885678	885523	1028	616	40%	1.537	0.454	70%	2.1	2.2
r4-NS	888685	888342	1051	655	38%	0.458	0.363	21%	2.2	2.2
r5-S	1309794	1311689	1449	989	32%	1.449	0.650	55%	7.0	6.9
r5-NS	1310697	1311156	1665	1045	37%	0.993	0.763	23%	7.0	6.9

Table 2: The results when SCALE is 0.1.

The procedure to build the minimal positive skew violation and minimal wirelength clock tree is similar to the algorithm MinSV except that merging regions are exploited other than mere merging segments. In section 3, points whose skew range $[\underline{t}_{skew}, \overline{t}_{skew}]$ closer to the center of permissible range $[LPR, UPR]$ are preferred during tree of merging segments construction. If we only seek non-positive skew violation, such merging scheme may cause unnecessary extra wirelength. In order to avoid such extra wirelength, all merging points whose corresponding skew ranges are within permissible range are considered as candidates. Usually these merging points may form a polygon *merging region* instead of a merging segment and any merging within this region will not cause positive skew violation. Merging regions allow us more room during the top-down routing phase to find a minimum total wirelength. The basic idea is similar to the BST algorithm[4]. However, the scenarios in our work are much more complicated due to the fact that each sink pair may have its distinct permissible range in contrast to the single global skew bound in BST. Moreover, the fact that at each point v_i on a joining segment $\mathbf{J_n}$ (of an internal node n), the skew between any pair of sinks in the subtree rooted at n is a range instead of a single value makes the merging regions more difficult to obtain than BST.

5.1 Notations and Definitions

DEFINITION 4. ***Shortest Distance Region:*** [4] *For any two convex polygonal regions P_1 and P_2 with boundaries $\mathcal{B}(P_1)$ and $\mathcal{B}(P_2)$, respectively, the* shortest distance region $\Re(P_1, P_2)$ *between P_1 and P_2 is the set of points with minimum sum of Manhattan distances to $\mathcal{B}(P_1)$ and $\mathcal{B}(P_2)$, i.e., $\Re(P_1, P_2) = \{p \mid d_r(p, \mathcal{B}(P_1)) + d_r(p, \mathcal{B}(P_2)) = d(\mathcal{B}(P_1), \mathcal{B}(P_2))\}$.*

[4]Note that shortest distance region here is defined based on merging regions and is different from the shortest distance region defined in Section3.

DEFINITION 5. ***Shortest Distance Segments:*** *For any two convex polygonal regions P_1 and P_2 with boundaries $\mathcal{B}(P_1)$ and $\mathcal{B}(P_2)$, respectively, the* shortest distance segments *of $\mathcal{B}(P_1)$ and $\mathcal{B}(P_2)$ are defined as $\mathbb{S}_{P_2}(P_1) = \mathcal{B}(P_1) \cap \Re(P_1, P_2)$ and $\mathbb{S}_{P_1}(P_2) = \mathcal{B}(P_2) \cap \Re(P_1, P_2)$*

DEFINITION 6. ***Joining Segment:*** *Let n be an internal node with children n_i and n_j with merging region $\mathcal{M}(n_i)$ and $\mathcal{M}(n_j)$, respectively. The segments of $\mathcal{M}(n_i)$ and $\mathcal{M}(n_j)$ used to build the merging region $\mathcal{M}(n)$ are called* joining segments, *denoted as $\mathbf{J}_{n_i}$ and $\mathbf{J}_{n_j}$.*

DEFINITION 7. ***Well-behaved Elmore Delay Line Segment:*** *Let a joining segment $\mathbf{J_n}$ with respect to an internal node n be a line segment with ending points v_1 and v_2. For each point v_p on $\mathbf{J_n}$, let u denote $d_r(v_1, v_p)$, then $d_r(v_2, v_p) = |\mathbf{J_n}| - u$. If for any leaf node s_k in the subtree rooted at n*

$$\begin{cases} \overline{t}(v_p, s_k) = K \cdot u^2 + \alpha_1 \cdot u + \beta_1 \\ \underline{t}(v_p, s_k) = K \cdot u^2 + \alpha_2 \cdot u + \beta_2 \end{cases}$$

where K, α_1, α_2, β_1, and β_2 are constants, then $\mathbf{J_n}$ is a well-behaved Elmore delay line segment.

5.2 Tree of Merging Regions Construction

Suppose we are working on the tree of merging regions for a node n with children n_i and n_j and let $T_{\mathcal{M}(n_i)}$ and $T_{\mathcal{M}(n_j)}$ denote two subtrees of merging regions rooted at n_i's merging region $\mathcal{M}(n_i)$ and n_j's merging region $\mathcal{M}(n_j)$, respectively. $\mathcal{M}(n)$ is constructed based on the skew between $T_{\mathcal{M}(n_i)}$'s sink s_r and $\mathcal{M}(n_j)$'s sink s_l. First, shortest distance segments $\mathbb{S}_{\mathcal{M}(n_i)}(\mathcal{M}(n_j))$ and $\mathbb{S}_{\mathcal{M}(n_j)}(\mathcal{M}(n_i))$ are found and are used as joining segments $\mathbf{J}_{n_i}$ and $\mathbf{J}_{n_j}$. We assume that $\mathbf{J}_{n_i}$ and $\mathbf{J}_{n_j}$ are well-behaved: when $\mathbf{J}_{n_i}$ and $\mathbf{J}_{n_j}$ are parallel Manhattan arcs, delay functions

on $\mathbf{J}_{n_i}$ and $\mathbf{J}_{n_j}$ are constants and when $\mathbf{J}_{n_i}$ and $\mathbf{J}_{n_j}$ are parallel non-Manhattan arcs, delay functions on $\mathbf{J}_{n_i}$ and $\mathbf{J}_{n_j}$ have same quadratic terms.

First of all, let us look at certain pair of points v_i and v_j on $\mathbf{J}_{n_i}$ and $\mathbf{J}_{n_j}$ such that $d_r(v_i, v_j) = d(\mathbf{J}_{n_i}, \mathbf{J}_{n_j})$. If the skew range $\mathbf{B}_{skew} = [\underline{t}^0_{skew}(s_r, s_l),\ \overline{t}^{D_{ij}}_{skew}(s_r, s_l)]$ and permissible range $\mathbf{B}_{PR} = [LPR_{rl},\ UPR_{rl}]$, where $D_{ij} = d(\mathbf{J}_{n_i}, \mathbf{J}_{n_j})$, there are four scenarios: (1) $\mathbf{B}_{PR}$ and $\mathbf{B}_{skew}$ do not overlap, (2) $\mathbf{B}_{PR}$ and $\mathbf{B}_{skew}$ partially overlap, (3) $\mathbf{B}_{PR}$ covers $\mathbf{B}_{skew}$ completely, and (4) $\mathbf{B}_{PR}$ is covered by $\mathbf{B}_{skew}$ completely. When $LPR_{rl} = \underline{t}^0_{skew}(s_r, s_l)$ and $UPR_{rl} = \overline{t}^{D_{ij}}_{skew}(s_r, s_l)$, both case (3) and case (4) occur and we treat this as case (3) for simplicity.

Case (1). happens when (a) $UPR_{rl} < \underline{t}^0_{skew}(s_r, s_l)$ or (b) $LPR_{rl} > \overline{t}^{D_{ij}}_{skew}(s_r, s_l)$. In subcase ($a$), we let $|e_{v_i}| = 0$ and e_{v_j} has to be extended according to Equation (5). In subcase (b), we first let $|e_{v_j}| = 0$ and compute $|e_{v_i}| = z$ by solving $\underline{t}^z_{skew}(s_r, s_l) = LPR_{rl}$. If the result satisfies $\overline{t}^{|e_{v_i}|}_{skew}(s_r, s_l) \le UPR_{rl}$, then we are done. Otherwise, $|e_{v_j}| = 0$ and e_{v_i} has to be extended according to Equation (4). In this case, the merging region is reduced to a merging segment.

Case (2). happens when $LPR_{rl} \le \overline{t}^{D_{ij}}_{skew}(s_r, s_l)$ and $\underline{t}^0_{skew}(s_r, s_l) \le UPR_{rl}$. If $\underline{t}^0_{skew}(s_r, s_l) < UPR_{rl} < \overline{t}^0_{skew}(s_r, s_l)$ or $\underline{t}^{D_{ij}}_{skew}(s_r, s_l) < LPR_{rl} < \overline{t}^{D_{ij}}_{skew}(s_r, s_l)$, then e_{v_j} or e_{v_i} is extended as Case (1). Otherwise, either (c) $\overline{t}^0_{skew}(s_r, s_l) < UPR_{rl} < \overline{t}^{D_{ij}}_{skew}(s_r, s_l)$ or (d) $\underline{t}^0_{skew}(s_r, s_l) < LPR_{rl} < \underline{t}^{D_{ij}}_{skew}(s_r, s_l)$ is true. By solving $\overline{t}^{|e_{v_i}|}skew(s_r, s_l) = UPR_{rl}$, we have

$$z_{lo} = \frac{UPR_{rl} - \overline{t}(v_i, s_r) + \underline{t}(v_j, s_l) + (\frac{rc \cdot D_{ij}^2}{2} + \frac{r \cdot D_{ij} \cdot \underline{C}_{T_j}}{W_u})}{rc \cdot D_{ij} + r(\frac{\overline{C}_{T_i}}{Wl} + \frac{\underline{C}_{T_j}}{W_u})}, \tag{6}$$

and the edge length of e_{v_i} can be any value in $[0, z_{lo}]$ for subcase (c). Similarly, we can obtain

$$z_{hi} = \frac{LPR_{rl} - \underline{t}(v_i, s_r) + \overline{t}(v_j, s_l) + (\frac{rc \cdot D_{ij}^2}{2} + \frac{r \cdot D_{ij} \cdot \overline{C}_{T_j}}{W_l})}{rc \cdot D_{ij} + r(\frac{\underline{C}_{T_i}}{Wu} + \frac{\overline{C}_{T_j}}{W_l})} \tag{7}$$

by solving $\underline{t}^{|e_{v_i}|}skew(s_r, s_l) = LPR_{rl}$ and $|e_{v_i}|$ can be any value in $[z_{hi}, D_{ij}]$ for subcase (d).

Case (3). happens when $LPR_{rl} \le \underline{t}^0_{skew}(s_r, s_l) \le \overline{t}^{D_{ij}}_{skew}(s_r, s_l) \le UPR_{rl}$. The value of $|e_{v_i}|$ is in $[0, D_{ij}]$ and $|e_{v_j}| = D_{ij} - |e_{v_i}|$.

Case (4). happens when $\underline{t}^0_{skew}(s_r, s_l) < LPR_{rl} \le UPR_{rl} < \overline{t}^{D_{ij}}_{skew}(s_r, s_l)$. If there exists some $z = |e_{v_i}|$ such that $LPR_{rl} \le \underline{t}^z_{skew}(s_r, s_l) \le \overline{t}^z_{skew}(s_r, s_l) \le UPR_{rl}$, then $|e_{v_i}|$ has to be within $[z_{lo}, z_{hi}]$ and $|e_{v_j}| = D_{ij} - |e_{v_i}|$. Otherwise, $|e_{v_i}|$ is obtained by solving

$$\underline{t}^{|e_{v_i}|}_{skew}(s_r, s_l) - LPR_{rl} = UPR_{rl} - \overline{t}^{|e_{v_i}|}_{skew}(s_r, s_l)$$

and $|e_{v_j}| = D_{ij} - |e_{v_i}|$.

Then let us consider when points v_i and v_j move along $\mathbf{J}_{n_i}$ and $\mathbf{J}_{n_j}$ simultaneously. The delay functions are $\underline{t}(v_i, s_r) = K \cdot u_i^2 + \alpha_1 \cdot u_i + \beta_1$, $\overline{t}(v_i, s_r) = K \cdot u_i^2 + \alpha_2 \cdot u_i + \beta_2$, $\underline{t}(v_j, s_l) = K \cdot u_j^2 + \alpha_3 \cdot u_j + \beta_3$, and $\overline{t}(v_j, s_l) = K \cdot u_j^2 + \alpha_4 \cdot u_j + \beta_4$, where $0 \le u_i \le |\mathbf{J}_{n_i}|$ and $0 \le u_j \le |\mathbf{J}_{n_j}|$ and K, α_1, α_2, α_3, α_4, β_1, β_2, β_3, and β_4 are constants.

If we let $|e_{v_i}| = z$ and $|e_{v_j}| = D_{ij} - z$, where $D_{ij} = d(\mathbf{J}_{n_i}, \mathbf{J}_{n_j})$, then for merging point v,

$$\begin{cases} \underline{t}(v, s_r) = \frac{rc}{2} \cdot z^2 + \frac{r \cdot \underline{C}_{T_i}}{W_u} \cdot z + \underline{t}(v_i, s_r) \\ \overline{t}(v, s_r) = \frac{rc}{2} \cdot z^2 + \frac{r \cdot \overline{C}_{T_i}}{W_l} \cdot z + \overline{t}(v_i, s_r) \\ \underline{t}(v, s_l) = \frac{rc}{2} \cdot (D_{ij} - z)^2 + \frac{r \cdot \underline{C}_{T_j}}{W_u} \cdot (D_{ij} - z) + \underline{t}(v_j, s_l) \\ \overline{t}(v, s_l) = \frac{rc}{2} \cdot (D_{ij} - z)^2 + \frac{r \cdot \overline{C}_{T_j}}{W_l} \cdot (D_{ij} - z) + \overline{t}(v_j, s_l) \end{cases}$$

Let us consider the following two cases:

Case (a). $\mathbf{J}_{n_i}$ and $\mathbf{J}_{n_j}$ are both Manhattan arcs. Since $\underline{t}(v_i, s_r)$, $\overline{t}(v_i, s_r)$, $\underline{t}(v_j, s_l)$, and $\overline{t}(v_j, s_l)$ are constants (see explanation later), without loss of generality, we may assume that $\underline{t}(v_i, s_r) = \beta_1$, $\overline{t}(v_i, s_r) = \beta_2$, $\underline{t}(v_j, s_l) = \beta_3$, and $\overline{t}(v_j, s_l) = \beta_4$. Therefore,

$$\begin{cases} \underline{t}^z_{skew}(s_r, s_l)_a = (rc \cdot D_{ij} + \frac{r \cdot \underline{C}_{T_i}}{W_u} + \frac{r \cdot \overline{C}_{T_j}}{W_l}) \cdot z + \underline{t}^0_{skew,a} \\ \overline{t}^z_{skew}(s_r, s_l)_a = (rc \cdot D_{ij} + \frac{r \cdot \overline{C}_{T_i}}{W_l} + \frac{r \cdot \underline{C}_{T_j}}{W_u}) \cdot z + \overline{t}^0_{skew,a} \end{cases}$$

and

$$\begin{cases} \underline{t}^0_{skew,a} = (\beta_1 - \beta_4) - (\frac{rc \cdot D_{ij}^2}{2} + \frac{r D_{ij} \cdot \overline{C}_{T_j}}{W_l}) \\ \overline{t}^0_{skew,a} = (\beta_2 - \beta_3) - (\frac{rc \cdot D_{ij}^2}{2} + \frac{r D_{ij} \cdot \underline{C}_{T_j}}{W_u}) \end{cases} \tag{8}$$

Case (b). When $\mathbf{J}_{n_i}$ and $\mathbf{J}_{n_j}$ are not Manhattan arcs, we have $u_i = u_j$. Therefore, the skew range is given by

$$\begin{cases} \underline{t}^z_{skew}(s_r, s_l)_b = (\alpha_1 - \alpha_4)u_i + \underline{t}^z_{skew}(s_r, s_l)_a \\ \overline{t}_{skew}(s_r, s_l)_b = (\alpha_2 - \alpha_3)u_i + \overline{t}_{skew}(s_r, s_l)_a \end{cases}$$

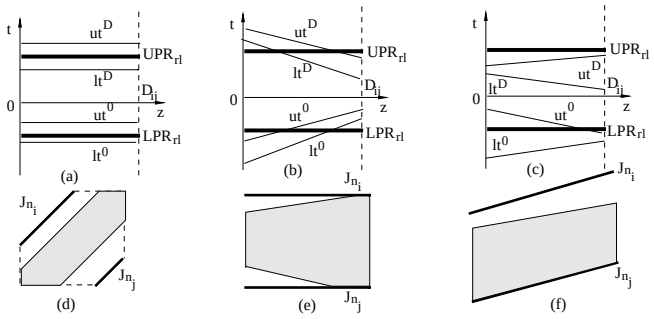


Figure 5: (d) merging region when $\mathbf{J}_{n_i}$ and $\mathbf{J}_{n_j}$ are parallel well-behaved Manhattan arcs and skew condition is in (a), (e) merging region when $\mathbf{J}_{n_i}$ and $\mathbf{J}_{n_j}$ are parallel well-behaved horizontal line segments and skew condition is in (b), and (f) merging region when $\mathbf{J}_{n_i}$ and $\mathbf{J}_{n_j}$ are parallel well-behaved with slope $m > 0$ and skew condition is in (c), where lt^0 stands for $\underline{t}^0_{skew}(s_r, s_l)$, ut^0 for $\overline{t}^0_{skew}(s_r, s_l)$, lt^D for $\underline{t}^{D_{ij}}_{skew}(s_r, s_l)$, and ut^D for $\overline{t}^{D_{ij}}_{skew}(s_r, s_l)$.

Based on the above discussions, when v_i and v_j are moved along $\mathbf{J}_{n_i}$ and $\mathbf{J}_{n_j}$ simultaneously, one case is replaced by another if one of the four line segments representing $\underline{t}^0_{skew}(s_r, s_l)$, $\overline{t}^0_{skew}(s_r, s_l)$, $\underline{t}^{D_{ij}}_{skew}(s_r, s_l)$, and $\overline{t}^{D_{ij}}_{skew}(s_r, s_l)$ intersects either one of horizontal line segments representing LPR_{rl} or UPR_{rl}. Therefore, the merging region of $\mathbf{J}_{n_i}$ and $\mathbf{J}_{n_j}$ can be formed in constant time by considering conditions at all possible intersections since there are at most 8 intersections. Figure 5 illustrates some examples.

Note that during the construction, the following may happen: (i) $\mathbf{J}_{n_i}$ and $\mathbf{J}_{n_j}$ are parallel Manhattan arcs with non-constant delay functions or (ii) the delay functions on $\mathbf{J}_{n_i}$ and $\mathbf{J}_{n_j}$ have different quadratic terms. Under these situations, any pair of points can be picked up to construct the merging region as long as they satisfy $v_i \in \mathbf{J}_{n_i}$, $v_j \in \mathbf{J}_{n_j}$, and $d_r(v_i, v_j) = d(\mathbf{J}_{n_i}, \mathbf{J}_{n_j})$.

5.3 Merging Region Construction Procedure

Suppose we are working on the merging region for a node n with children n_i and n_j.

Step 1. Find joining segments $\mathbf{J}_{n_i} = \mathbb{S}_{\mathcal{M}(n_j)}(\mathcal{M}(n_i))$ and $\mathbf{J}_{n_j} = \mathbb{S}_{\mathcal{M}(n_i)}(\mathcal{M}(n_j))$ and then compute $D_{ij} = d(\mathbf{J}_{n_i}, \mathbf{J}_{n_j})$.

Step 2. Based on skew permissible range between s_r and s_l $[LPR_{rl}, UPR_{rl}]$ and delay functions $\underline{t}(v_i, s_r)$, $\overline{t}(v_i, s_r)$, $\underline{t}(v_j, s_l)$ and $\overline{t}(v_j, s_l)$ defined for points v_i on $\mathbf{J}_{n_i}$ and v_j on $\mathbf{J}_{n_j}$, compute delay functions of $\underline{t}(v, s_r)$, $\overline{t}(v, s_r)$, $\underline{t}(v, s_l)$ and $\overline{t}(v, s_l)$. Next, functions $\underline{t}^z_{skew}(s_r, s_l)$ and $\overline{t}^z_{skew}(s_r, s_l)$ are calculated.

Step 3. If $\mathbf{J}_{n_i}$ and $\mathbf{J}_{n_j}$ are both well-behaved Manhattan arcs parallel to each other, then we may obtain the merging region according to the relation between $\mathbf{B}_{skew}$ and $\mathbf{B}_{PR}$ as described in Section 5.2.

Step 4. If $\mathbf{J}_{n_i}$ and $\mathbf{J}_{n_j}$ are both well-behaved non-Manhattan arcs parallel to each other, then all possible intersections between $\underline{t}^0_{skew}(s_r, s_l)$, $\overline{t}^0_{skew}(s_r, s_l)$, $\underline{t}^{D_{ij}}_{skew}(s_r, s_l)$, or $\overline{t}^{D_{ij}}_{skew}(s_r, s_l)$ and line segments LPR_{rl} or UPR_{rl} are obtained first. For each intersection, the positions of v_i and v_j are located on $\mathbf{J}_{n_i}$ and $\mathbf{J}_{n_j}$, and a feasible range $[L_{e_{v_i}}, U_{e_{v_i}}]$ ($L_{e_{v_i}} \leq U_{e_{v_i}}$) for e_{v_i} is found. Furthermore, feasible range is also computed when v_i and v_j are ending points of $\mathbf{J}_{n_i}$ and $\mathbf{J}_{n_j}$. Then for each feasible range, the possible locations for merging point v when $|e_{v_i}| = L_{e_{v_i}}$ or $|e_{v_i}| = U_{e_{v_i}}$ are located inside shortest distance region $\Re(\mathbf{J}_{n_i}, \mathbf{J}_{n_j})$ of $\mathbf{J}_{n_i}$ and $\mathbf{J}_{n_j}$ in the Manhattan plane. Finally, the minimal polygon enclosing these locations is constructed as the merging region.

The following Lemmas show the correctness of the merging region construction procedure(the proofs are omitted due to page limit). The boundaries of merging region are either Manhattan arcs or rectilinear line segments when $\mathbf{J}_{n_i}$ and $\mathbf{J}_{n_j}$ are parallel well-behaved Manhattan arcs. However, boundaries of merging region could be any kinds of line segments if $\mathbf{J}_{n_i}$ and $\mathbf{J}_{n_j}$ are parallel well-behaved non-Manhattan arcs.

LEMMA 2. *If $\mathbf{J}_{n_i}$ and $\mathbf{J}_{n_j}$ are two well-behaved Manhattan arcs parallel to each other, then any Manhattan arc or rectilinear line segment $l \in \Re(\mathbf{J}_{n_i}, \mathbf{J}_{n_j})$ is also well-behaved.*

LEMMA 3. *If $\mathbf{J}_{n_i}$ and $\mathbf{J}_{n_j}$ are two parallel well-behaved non-Manhattan joining segments, then any line segment $l \in \Re(\mathbf{J}_{n_i}, \mathbf{J}_{n_j})$ is also well-behaved if the delay functions $\overline{t}$ and $\underline{t}$ defined over $\mathbf{J}_{n_i}$ and $\mathbf{J}_{n_j}$ have the same quadratic term.*

LEMMA 4. *$\mathbf{M}(v)$ is a polygon with at most 20 boundary segments.*

6. CONCLUSION AND FUTURE WORK

In order to cope with the increasingly severe impact on clock skew from process variation, we propose DME/BST based process variation aware clock tree routing algorithms to improve skew tolerance to process variations. Experimental results show that our method provides great improvement on skew tolerance to process variations. In this work, we adopt existing techniques on constructing initial abstract topology. However, these techniques are either based on clock sink location distributions [16], or only skew permissible ranges between clock sink pairs [17]. In future, we will investigate new techniques on abstract topology construction that considers both the sink physical proximities and sink temporal specifications to enable a better skew tolerance for clock routing tree. Process variations on clock buffers will be considered in future as well.

7. REFERENCES

[1] T.-H. Chao, Y.-C. Hsu, and J.-M. Ho, "Zero Skew Clock Net Routing", *Proc. DAC*, pp. 518-523, 1992.

[2] T.-H. Chao, Y.-C. Hsu, J.-M. Ho, K. D. Boese, and A. B. Kahng, "Zero Skew Clock Routing with Minimum Wirelength", *IEEE Trans. on Circuits and Systems-II: Analog and Digital Signal Processing*, Vol. CAS-39, No. 11, pp. 799-814, November, 1992.

[3] J. Chung and C.-K. Cheng, "Skew Sensitivity Minimization of Buffered Clock Tree", *Proc. ICCAD*, pp. 280-283, 1994.

[4] J. Cong, A. B. Kahng, C.-K. Koh, and C.-W. Albert Tsao, "Bounded-Skew Clock and Steiner Routing", *ACM Trans. on Design Automation of Electronic Systems*, vol. 3, pp. 341-388, 1998.

[5] J. Cong and Z. Pan, "Interconnect Performance Estimation Models for Design Planning", *IEEE Trans. on CAD*, vol. 20, No. 6, pp. 739-752, June, 2001.

[6] M. P. Desai, R. Cvijetic, and J. Jensen, "Sizing of Clock Distribution Networks for High Performance CPU Chips", *Proc. DAC*, pp. 389-394, June, 1996.

[7] M. Edahiro, "Minimum Path-Length Equi-Distant Routing", *Proc. of Asia-Pacific Conference on Circuits and Systems*, pp. 41-46, December, 1992.

[8] M. Edahiro, "A Clustering-Based Optimization Algorithm in Zero-Skew Routing", *Proc. DAC*, pp. 612-616, June, 1993.

[9] M. Edahiro, "Delay Minimization for Zero-Skew Routing", *Proc. ICCAD*, pp. 563-566, 1993.

[10] S. Lin and C. K. Wong, "Process-Variation-Tolerant Clock Skew Minimization", *Proc. ICCAD*, pp. 284-288, 1994

[11] J. L. Neves and E. G. Friedman, "Topological Design of Clock Distribution Networks Based on Non-Zero Clock Skew Specification", *Proc. of IEEE Midwest Symp. on Circuits and Systems*, pp. 468-471, 1993.

[12] J. L. Neves and E. G. Friedman, "Design Methodology for Synthesizing Clock Distribution Networks Exploiting Nonzero Localized Clock Skew", *IEEE Trans. on VLSI Systems*, Vol 4, No 2., pp. 286-291, June, 1996.

[13] J. L. Neves and E. G. Friedman, "Optimal Clock Skew Scheduling Tolerant to Process Variations", *Proc. DAC*, pp. 623-628, 1996.

[14] P. J. Restle, A. E. Ruehli, and S. G. Walker, "Multi-Ghz Interconnect Effects in Microprocessors", *Proc. ISPD*, pp. 93-97, 2001.

[15] H. Su and S. S. Sapatnekar, "Hybrid Structured Clock Network Construction", *Proc. ICCAD*, pp. 333-336, 2001.

[16] R.-S. Tsay, "An exact zero-skew clock routing algorithm", *IEEE Trans. on CAD*, vol. 12, No. 2, pp. 242-249, February, 1993.

[17] D. Velenis, E. Friedman, and M. C. Papaefthymiou, "A Clock Tree Topology Extraction Algorithm for Improving the Tolerance of Clock Distribution Networks to Delay Uncertainty", *Proc. ISCAS*, pp. 4.422-4.425, May, 2001.

[18] J.G. Xi and W. W.-M. Dai, "Buffer Insertion and Sizing Under Process Variation for Low Power Clock Distribution", *Proc. DAC*, pp. 491-496, 1995.

[19] J.G. Xi and W. W.-M. Dai, "Useful-skew Clock Routing with Gate Sizing for Low Power Design", *J. of VLSI Signal Processing*, vol. 16, pp. 163–179, June/July, 1997.