Journal of VLSI Signal Processing 16, 199–215 (1997)


# Practical Bounded-Skew Clock Routing*

ANDREW B. KAHNG AND C.-W. ALBERT TSAO
*UCLA Computer Science Dept., Los Angeles, CA 90095-1596*

*Received September 24, 1996; Revised October 11, 1996*

**Abstract.** In Clock routing research, such practical considerations as hierarchical buffering, rise-time and overshoot constraints, obstacle- and legal location-checking, varying layer parasitics and congestion, and even the underlying design flow are often ignored. This paper explores directions in which traditional formulations can be extended so that the resulting algorithms are more useful in production design environments. Specifically, the following issues are addressed: (i) clock routing for varying layer parasitics with non-zero via parasitics; (ii) obstacle-avoidance clock routing; and (iii) hierarchical buffered tree synthesis. We develop new theoretical analyses and heuristics, and present experimental results that validate our new approaches.

## 1. Preliminaries

Control of signal delay skew has become a dominant objective in the routing of VLSI clock distribution networks and large timing-constrained global nets. Thus, the "zero-skew" clock tree and performance-driven routing literatures have seen rapid growth over the past several years; see [1, 2] for reviews. "Exact zero skew" is typically obtained at the expense of increased wiring area and higher power dissipation. In practice, circuits still operate correctly within some non-zero skew bound, and so the actual design requirement is for a *bounded-skew routing tree* (BST). This problem is also significant in that it unifies two well-known routing problems—the *Zero Skew Clock Routing Problem* (ZST) for skew bound $B = 0$, and the classic *Rectilinear Steiner Minimum Tree Problem* (RSMT) for $B = \infty$.

In our discussion, the *distance* between two points $p$ and $q$ is the Manhattan (or rectilinear) distance $d(p, q)$, and the distance between two sets of points $P$ and $Q$ is $d(P, Q) = \min\{d(p, q) \mid p \in P \text{ and } q \in Q\}$. The *cost* of the edge $e_v$ is simply its wirelength, denoted $|e_v|$; this is always at least as large as the Manhattan distance between the endpoints of the edge, i.e., $|e_v| \geq d(l(p), l(v))$. *Detour wiring*, or *detouring*, occurs when $|e_v| > d(l(p), l(v))$. The cost of $T$, denoted $cost(T)$, is the total wirelength of the edges in $T$. We denote the set of sink locations in a clock routing instance as $S = \{s_1, s_2, \ldots, s_n\} \subset \Re^2$. A *connection topology* is a binary tree with $n$ leaves corresponding to the sinks in $S$. A *clock tree* $T_G(S)$ is an embedding of the connection topology in the Manhattan plane, i.e., each internal node $v \in G$ is mapped to a location $l(v)$ in the Manhattan plane. (If $G$ and/or $S$ are understood, we may simply use $T(S)$ or $T$ to denote the clock tree.) The root of the clock tree is the *source*, denoted by $s_0$. When the clock tree is rooted at the source, any edge between a parent node $p$ and its child $v$ may be identified with the child node, i.e., we denote this edge as $e_v$. If $t(u, v)$ denotes the signal delay between nodes $u$ and $v$, then the *skew* of clock tree $T$ is given by

$$\begin{aligned} skew(T) &= \max_{s_i, s_j \in S} |t(s_0, s_i) - t(s_0, s_j)| \\ &= \max_{s_i \in S}\{t(s_0, s_i)\} - \min_{s_i \in S}\{t(s_0, s_i)\} \end{aligned}$$

The BST problem is formally stated as follows.

**Minimum-Cost Bounded Skew Routing Tree (BST) Problem:** Given a set $S = \{s_1, \ldots, s_n\} \subset \mathcal{R}^2$ of sink locations and a skew bound $B$, find a routing topology $G$ and a minimum-cost clock tree $T_G(S)$ that satisfies $\text{skew}(T_G(S)) \leq B$.

*Support for this work was provided by Cadence Design Systems, Inc.

### *1.1. The Extended DME Algorithm*

The BST problem has been previously addressed in [3–5]. Their basic method, called the *Extended DME* (Ex-DME) algorithm, extends the DME algorithm of [6–9] via the enabling concept of *merging region*, which is a set of embedding points with feasible skew and minimum merging cost if no detour wiring occurs[1]. For a fixed tree topology, Ex-DME follows the 2-phase approach of the DME algorithm in constructing a bounded-skew tree: (i) a bottom-up phase to construct a binary tree of merging regions which represent the loci of possible embedding points of the internal nodes, and (ii) a top-down phase to determine the exact locations of the internal nodes. The reader is referred to [4, 3, 5, 10] for more details (the latter is available by anonymous ftp). In the remainder of this subsection, we sketch several key concepts from [4, 3, 5].

Let $max\_t(p)$ and $min\_t(p)$ denote the maximum and minimum delay values (max-delay and min-delay, for short) from point $p$ to all leaves in the subtree rooted at $p$. The skew of point $p$, denoted $skew(p)$, is $max\_t(p) - min\_t(p)$. (If all points of a pointset $P$ have identical max-delay and min-delay, and hence identical skew, we similarly use the terms $max\_t(P)$, $min\_t(P)$ and $skew(P)$.) As $p$ moves along any line segment the values of $max\_t(p)$ and $min\_t(p)$, along with $skew(p)$, respectively define the *delay* and *skew functions* over the segment.

For a node $v \in G$ with children $a$ and $b$, its merging region, denoted $mr(v)$, is constructed from the so-called "joining segments" $L_a \in mr(a)$ and $L_b \in \mathrm{mr}(b)$, which are the closest boundary segments of $mr(a)$ and $mr(b)$. In practice, $L_a$ and $L_b$ are either a pair of parallel Manhattan arcs (i.e., segments with possibly zero length having slope $+1$ or $-1$) or a pair of parallel rectilinear segments (i.e., horizontal or vertical line segments). The set of points with minimum sum of distances to $L_a$ and $L_b$ form a *Shortest Distance Region* $SDR(L_a, L_b)$, where the points with skew $\leq B$ (i.e., feasible skew) in turn form the merging region $mr(v)$. [5] prove that under Elmore delay each line segment $l = \overline{p_1 p_2} \in SDR(L_a, L_b)$ is *well-behaved*, in that the max-delay and min-delay functions of point $p \in l$ are of the forms $max\_t(p) = \max_{i=1,\ldots,n_1}\{\alpha_i \cdot x + \beta_i\} + K \cdot x^2$ and $min\_t(p) = \min_{i=1,\ldots,n_2}\{\alpha'_i \cdot x + \beta'_i\} + K \cdot x^2$, where $x = d(p_1, a)$ or $d(p_2, b)$. In other words, the skew values along a well-behaved segment $l$ can be either a constant (when $K = \alpha_i = \alpha'_i = 0$) or piecewise-linear decreasing, then constant, then piecewise-linear increasing along $l$. This important property enables [5] to develop a set of construction rules for computing the merging region $mr(v) \in SDR(L_a, L_b)$ efficiently in $O(n)$ time. The resulting merging region is shown to be a convex polygon bounded by at most 2 Manhattan arcs and 2 horizontal/vertical segments when $L_a$ and $L_b$ are Manhattan arcs, or a convex polygon bounded by at most $4n$ (with arbitrary slopes) segments where $n$ is the number of the sinks. The empirical studies of [5] show that in practice each merging region has at most 9 boundary segments, and thus is computed in constant time.

Since each merging region is constructed from the closest boundary segments of its child regions, the method for constructing the merging region is called *Boundary Merging and Embedding* (BME). [5] also propose a more general method called *Interior Merging and Embedding* (IME), which constructs the merging region from segments which can be interior to the children regions. The routing cost is improved at the expense of longer running time. For arbitrary topology, [3] propose the Extended Greedy-DME algorithm (ExG-DME), which combines merging region computation with topology generation, following the Greedy-DME algorithm approach of [11]. The distinction is that ExG-DME allows merging at non-root nodes whereas Greedy-DME always merges two subtrees at their roots; see [3] for details.

Experimental results show that ExG-DME can produce a set of routing solutions with smooth skew and wirelength trade-off, and that it closely matches the best known heuristics for both zero-skew routing and unbounded-skew routing (i.e., the rectilinear Steiner minimal tree problem).

### *1.2. Contributions of the Paper*

In this paper, we will show that these nice properties of merging regions and merging segments still exist when layer parasitics (i.e., the values of per-unit capacitance and resistance) vary among the routing layers and when there are large routing obstacles. Therefore, the ExG-DME algorithm can be naturally extended to handle these practical issues which are encountered in the real circuit designs. Section 2 extends the BME construction rules for the case of varying layer parasitics. We prove that if we prescribe the routing pattern between any two points, any line segment in $SDR(L_a, L_b)$ is well-behaved where $L_a$ and $L_b$ are two single points. Hence, the BME construction rules are still applicable. Section 3 proposes new merging region construction rules when there are obstacles in the routing plane. The

solution is based on the concept of a *planar merging region*, which contains all the minimum-cost merging points when no detouring occurs. Finally, Section 4 extends our bounded-skew routing method to handle the practical case of buffering hierarchies in large circuits, assuming (as is the case in present design methodologies) that the buffer hierarchy (i.e., the number of buffers at each level and the number of levels) is given. Some conclusions are given in Section 5.

## 2. Clock Routing for Non-Uniform Layer Parasitics

In this section, we consider the clock routing problem for non-uniform layer parasitics, i.e., the values of per-unit resistance and capacitance on the V-layer (vertical routing layer) and H-layer (horizontal routing layer) can be different[2]. We first assume that via has no resistance and capacitance, then extend our method for non-zero via parasitics.

Let node $v$ be a node in the topology with children $a$ and $b$, and let merging region $mr(v)$ be constructed from joining segments $L_a \subseteq mr(a)$ and $L_b \subseteq mr(b)$. When both $L_a$ and $L_b$ are vertical segments or are two single points on a horizontal line, only the H-layer will be used for merging $mr(a)$ and $mr(b)$. Similarly, when $L_a$ and $L_b$ are both horizontal or are two single points on a vertical line, only the V-layer will be used for merging $mr(a)$ and $mr(b)$.[3] The original BME construction rules [5] still apply in these cases.

Corollary 1 below shows that for non-uniform layer parasitics, joining segments will never be Manhattan arcs of non-zero length. Thus we need consider only the possible modification of BME construction rules for the case where the joining segments are two single points which do not sit on a horizontal or vertical line. In this case, both routing layers have to be used for merging $mr(a)$ and $mr(b)$. One problem with routing under non-uniform layer parasitics is that different routing patterns between two points will result in different delays, even if the wirelength on both layers are the same. However, if we can prescribe the routing pattern for each edge of the clock tree, the ambiguity of delay values between two points can be avoided. Figure 1 shows the two simplest routing patterns between two points, which we call the HV and VH routing patterns. Other routing patterns can be considered, but may result in more vias and more complicated computation of merging regions.

**Theorem 1.** *Let $v$ be a node in the topology with children $a$ and $b$, with the subtrees rooted at $a$ and $b$ having capacitive load $C_a$ and $C_b$. Assume that joining segments $L_a \subseteq mr(a)$ and $L_b \subseteq mr(b)$ are two single points. Under the the HV routing pattern*, (i) *any line segment $l \in SDR(L_a, L_b)$ is well-behaved*, (ii) *merging region $mr(v)$ has at most* 6 *sides*, *and* (iii) *$mr(v)$ has no boundary segments which are Manhattan arcs of non-zero length.*

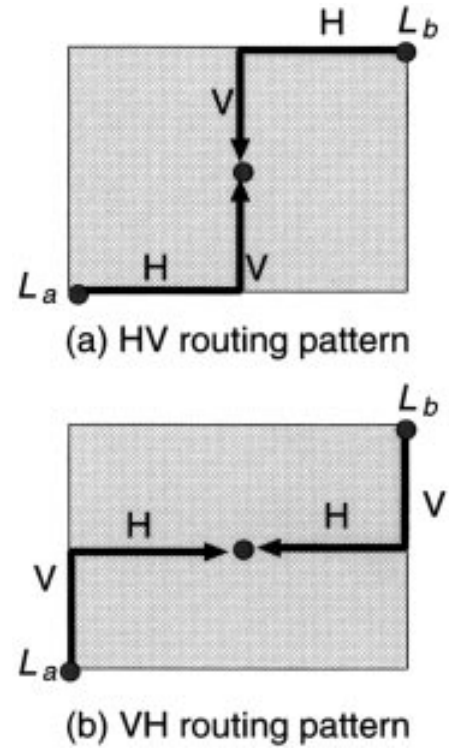



*Figure 1.* Two simple routing patterns between two points: HV and VH.

**Proof:** Without losing generality, we assume that $L_a$ and $L_b$ are located at $(0, 0)$ and $(h, v)$ as shown in Fig. 2. Let $A(x, y)$ and $B(x, y)$ be respectively the average max-delay from $a$ and $b$ to $p$ under the HV routing pattern. Let $r_1$, $c_1$ and $r_2$, $c_2$ be per-unit resistance and capacitance of the H-layer and the V-layer. We refer to the original delays and skew at point $L_a$ as $max\_\bar{t}(L_a)$, $min\_\bar{t}(L_a)$, and $\overline{skew}(L_a)$. Similarly, we refer to the original delays/skew at point $L_b$ as $max\_\bar{t}(L_b)$, $min\_\bar{t}(L_b)$, and $\overline{skew}(L_b)$. For point $p = (x, y) \in SDR(L_a, L_b)$,

$$\begin{aligned} A(x, y) &= max\_\bar{t}(L_a) + r_1 x(c_1 x/2 + C_a) \\ &\quad + r_2 y(c_2 y/2 + C_a + c_1 x) \\ &= K_1 \cdot x^2 + Ex + K_2 \cdot y^2 \\ &\quad + Fy + Gxy + D. \end{aligned} \tag{1}$$

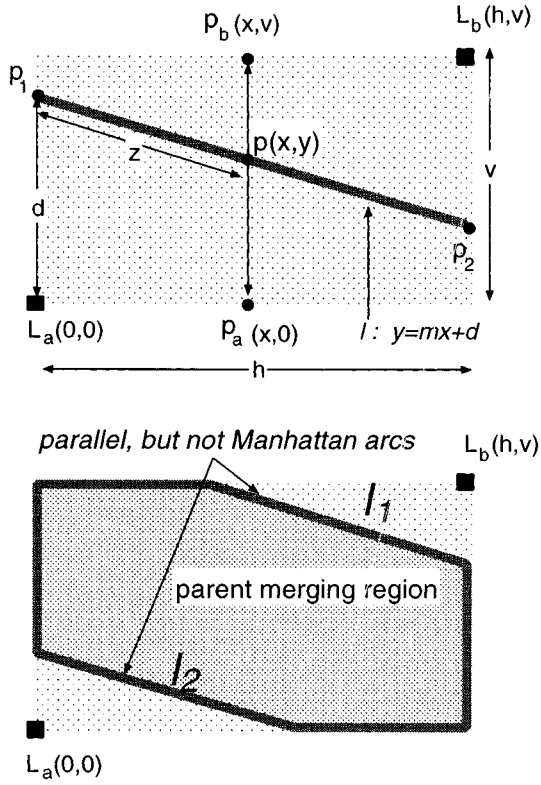



*Figure 2.* The merging region $mr(v)$ constructed from joining segments $L_a$ and $L_b$ which are single points by using the HV routing pattern for non-uniform layer parasitics.

where $K_1 = r_1c_1/2$, $E = r_1C_a$, $K_2 = r_2c_2/2$, $F = r_2C_a$, $G = r_2c_1$, and $D = \overline{max\_t}(L_a)$. Similarly,

$$\begin{aligned} B(x,y) &= \overline{max\_t}(L_b) + r_1(h-x) \\ &\quad \times (c_1(h-x)/2 + C_b) + r_2(v-y) \\ &\quad \times (c_2(v-y)/2 + C_b + c_1(h-x)) \\ &= K_1 \cdot x^2 + Jx + K_2 \cdot y^2 \\ &\quad + Ly + Gxy + M. \end{aligned} \tag{2}$$

where $J$, $L$, and $M$ are also constants. Therefore,

$$\begin{aligned} max\_t(p) &= \max(A(x,y), B(x,y)) \\ &= \max(Ex + Fy + D, Jx + Ly + M) \\ &\quad + K_1 \cdot x^2 + K_2 \cdot y^2 + Gxy \end{aligned} \tag{3}$$

Similarly, we can prove that

$$\begin{aligned} min\_t(p) &= \min(A(x,y), B(x,y)) \\ &= \min(Ex + Fy + D', Jx + Ly + M') \\ &\quad + K_1 \cdot x^2 + K_2 \cdot y^2 + Gxy \end{aligned} \tag{4}$$

where $D' = \overline{min\_t}(L_a)$ and $M' = M - \overline{skew}(L_b)$.

If line segment $l \in SDR(L_a, L_b)$ is vertical, then for point $p(x, y) \in l$ we have

$$max\_t(p) = K_2 \cdot y^2 + \max\{F_v y + O, L_v y + P\} \tag{5}$$

$$min\_t(p) = K_2 \cdot y^2 + \min\{F_v y + O', L_v y + P'\} \tag{6}$$

where $F_v = F + Gx$, $L_v = L + Gx$, $O = D + K_1 \cdot x^2 + Ex$, $O' = D' + K_1 \cdot x^2 + Ex$, $P = M + K_1 \cdot x^2 + Jx$, and $P' = M' + K_1 \cdot x^2 + Jx$ are all constants. So, $l$ is well-behaved.

If $l$ is not vertical and described by the equation $y = mx + d$ where $m \neq \infty$ (see Fig. 2), then from Eqs. (1) and (2)

$$\begin{aligned} A(x,y) &= K_1 \cdot x^2 + Ex + K_2 \cdot (mx+b)^2 \\ &\quad + F(mx+b) + Gx(mx+b) + D \\ &= K \cdot x^2 + Hx + I \\ B(x,y) &= K_1 x^2 + Jx + K_2(mx+b)^2 \\ &\quad + L(mx+b) + Gx(mx+b) + M \\ &= K \cdot x^2 + H'x + I', \end{aligned}$$

where $K$, $H$, $I$, $H'$, and $I'$ are all constants. Hence,

$$max\_t(p) = K \cdot x^2 + \max(Hx + I, H'x + I') \tag{7}$$

$$min\_t(p) = K \cdot x^2 + \min(Hx + Q, H'x + Q') \tag{8}$$

When $max\_t(p)$ and $min\_t(p)$ are written as functions of $z = d(p, p_1) = (1+m)x$, they will still have the same coefficient in the quadratic term; this implies that any line segment $l \in SDR(L_a, L_b)$ is well-behaved.

Let $l_1$ and $l_2$ be the non-rectilinear boundary segments of $SDR(L_a, L_b)$ which have non-zero length. By the fact that $skew(l_1) = skew(l_2) = B$ and Eqs. (3) and (4), $l_1$ and $l_2$ will be two parallel line segments described by equations $(E-J)x + (F-L)y + D - M' = \pm B$. In practice, $|E - J| \neq |F - L|$ unless both layers have the same parasitics, i.e., $r_1 = r_2$ and $c_1 = c_2$. Thus, $l_1$ and $l_2$ will not be Manhattan arcs. □

We similarly can prove that Theorem 1 holds when the routing pattern is VH, or even when the routing pattern is a linear combination of both routing patterns such that each tree edge is routed by HV with probability $0 \leq \alpha \leq 1$ and VH with probability $1 - \alpha$. Notice that at the beginning of the construction, each node $v$ is a sink with $mr(v)$ being a single point. Thus, no merging region can have boundary segments which are Manhattan arcs with constant delays, and we have

**Corollary 1.** *For non-uniform layer parasitics, each pair of joining segments will be either* (i) *parallel rectilinear line segments or* (ii) *two single points.*

Since any line segment in $SDR(L_a, L_b)$ is well-behaved for non-uniform layer parasitics, the BME construction rules are still applicable, except that (i) we have to prescribe the routing pattern for each tree edge, and (ii) the delays are calculated based on Eqs. (5), (6) for points on a vertical line $l \in SDR(L_a, L_b)$, and (7), (8) for points on a non-vertical line $l \in SDR(L_a, L_b)$, whenever $L_a$ and $L_b$ are two single points.

**Theorem 2.** *With non-zero via parasitics (per-unit resistance $r_v \geq 0$, per-unit capacitance $c_v \geq 0$), Theorem 1 still holds except that there will be different delay/skew equations for points on boundary segments and interior segments of $SDR(L_a, L_b)$.*

**Proof:** Again, without losing generality we assume the HV routing pattern. In Fig. 3(a), we assume that points $L_a$ and $L_b$ are both located in the H-layer. Under the HV routing pattern, most merging points $p$ are on the V-layer except the top and bottom boundaries of $SDR(L_a, L_b)$ (e.g., point $q$ in the figure). For point $p$ on the V-layer, there is exactly one via in the path from $p$ to $L_a$ and $L_b$ according to the HV routing pattern. Then, delay equations for merging points $p = (x, y) \in SDR(L_a, L_b)$ on the V-layer become

$$\begin{aligned} A(x,y) &= max\_\bar{t}(L_a) + r_1 x(c_1 x/2 + C_a) \\ &\quad + r_v(C_a + c_1 x + c_v/2) \\ &\quad + r_2 y(c_2 y/2 + C_a + c_1 x + c_v) \\ &= K_1 \cdot x^2 + J_1 x \\ &\quad + K_2 \cdot y^2 + L_1 y + r_2 c_1 xy + M_1, \\ B(x,y) &= max\_\bar{t}(L_b) + r_1(h-x)(c_1(h-x)/2 + C_b) \\ &\quad + r_v(C_b + c_1(h-x) + c_v/2) + r_2(v-y) \\ &\quad \times (c_2(v-y)/2 + C_b + c_1(h-x) + c_v) \\ &= K_1 \cdot x^2 + J_2 x \\ &\quad + K_2 \cdot y^2 + L_2 y + r_2 c_1 xy + M_2 \end{aligned}$$

where $J_1$, $L_1$, $M_1$, $J_2$, $L_2$, and $M_2$ are all constants. Since the quadratic terms $K_1 \cdot x^2$ and $K_2 \cdot y^2$ are the same as before, Theorem 1 holds for the merging points in $SDR(L_a, L_b)$ on the V-layer.

For merging points $q \in SDR(L_a, L_b)$ on the H-layer, the number of vias from $q$ to $L_a$ and $L_b$ can be either 0 or 2. The delay calculations for merging points $p$ and $q$ will not be the same because of the unequal number of vias from the merging points to $L_a$ and $L_b$.

Figure 3(b) shows one of the three cases where without loss of generality either point $L_a$ or $L_b$ is located on the V-layer. As shown in the Figure, we use point $q$ to represent the merging point on the left or right boundary of $SDR(L_a, L_b)$ on the V-layer, point $q'$ to represent the merging point on the top or bottom boundary of $SDR(L_a, L_b)$ on the H-layer, and point $p \in SDR(L_a, L_b)$ to represent the other merging points which are on the V-layer (but not on the right or left boundaries). In this case, the number of vias from point $q$, $q'$ and $p$ to $L_a$ or $L_b$ are not equal; their delay equations will not be identical, but will still have the same quadratic terms $K_1 \cdot x^2$ and $K_2 \cdot y^2$. Therefore, Theorem 1 still holds except that there will be different delay/skew equations for points on boundary segments and interior segments of $SDR(L_a, L_b)$. □

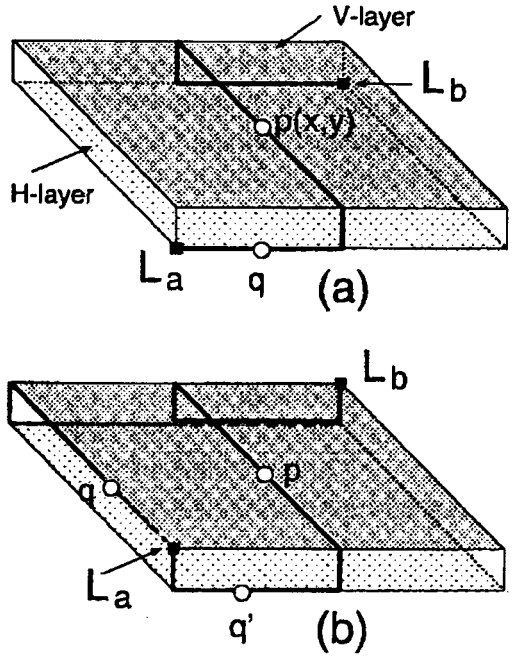


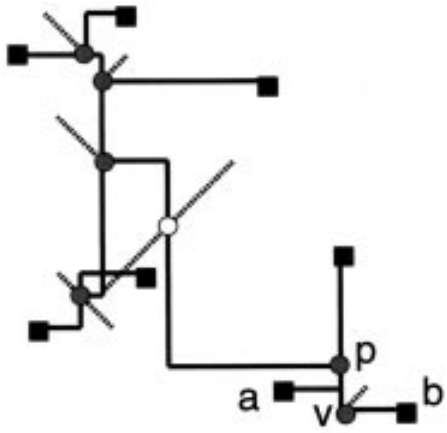

*Figure 3.* Delay/skew equations for points on boundary segments and interior segments of $SDR(L_a, L_b)$ are different when via resistance and/or capacitance are non-zero.

*Table 1.* Comparison of total wirelength of routing solutions under non-uniform and uniform layer parasitics, with ratios shown in parentheses. We mark by $*$ the cases where the routing solution under non-uniform layer parasitics has smaller total wirelength than the solution under uniform layer parasitics.

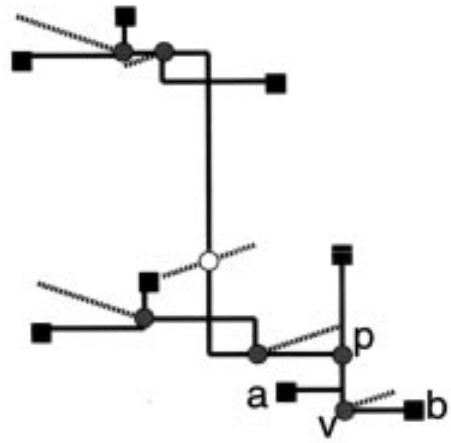
| | r1 | | r2 | | r3 | | r4 | | r5 | |
|---|---|---|---|---|---|---|---|---|---|---|
| Skew bound | Wirelengths under non-uniform layer parasitics (normalized) wirelengths under uniform layer parasitics | | | | | | | | | |
| 0 [11] | 1253.2 | | 2483.8 | | 3193.8 | | 6499.7 | | 9723.7 | |
| 0 | 1332.5<br>1320.7 | (1.01) | 2623.8<br>2603.6 | (1.01) | *3359.1<br>3382.4 | (0.99) | *6810.7<br>6877.5 | (0.99) | *10108.7<br>10138.5 | (1.00) |
| 1 ps | 1283.5<br>1232.2 | (1.04) | 2531.8<br>2401.7 | (1.05) | 3207.0<br>3118.1 | (1.03) | 6461.5<br>6241.1 | (1.04) | 9610.8<br>9190.7 | (1.05) |
| 5 ps | 1182.1<br>1130.6 | (1.05) | 2333.3<br>2256.2 | (1.03) | 2988.6<br>2875.1 | (1.04) | 5979.8<br>5715.1 | (1.05) | 8753.9<br>8371.2 | (1.05) |
| 10 ps | 1158.6<br>1069.2 | (1.08) | 2248.3<br>2183.5 | (1.03) | 2810.7<br>2747.6 | (1.02) | 5719.0<br>5453.8 | (1.05) | 8482.4<br>8063.7 | (1.05) |
| 20 ps | 1071.5<br>1039.6 | (1.03) | 2183.4<br>2069.1 | (1.06) | 2709.8<br>2569.0 | (1.05) | 5474.6<br>5290.1 | (1.03) | 8018.2<br>7695.9 | (1.04) |
| 50 ps | 1058.6<br>1009.3 | (1.05) | 2028.9<br>1917.8 | (1.06) | 2557.0<br>2459.7 | (1.04) | 5195.8<br>5008.0 | (1.04) | 7562.9<br>7248.2 | (1.04) |
| 100 ps | 989.0<br>964.3 | (1.03) | 1929.0<br>1880.7 | (1.03) | 2463.9<br>2350.1 | (1.05) | 4940.1<br>4786.1 | (1.03) | 7193.1<br>6869.6 | (1.05) |
| 200 ps | 936.7<br>895.8 | (1.05) | 1886.7<br>1741.6 | (1.08) | *2356.0<br>2359.5 | (0.99) | 4734.4<br>4540.1 | (1.04) | 6905.9<br>6650.0 | (1.04) |
| 500 ps | 919.4<br>820.4 | (1.12) | 1770.9<br>1754.6 | (1.01) | 2205.2<br>2187.4 | (1.01) | 4635.1<br>4564.2 | (1.02) | 6564.1<br>6449.3 | (1.02) |
| 1 ns | 830.0<br>819.1 | (1.01) | *1664.2<br>1709.4 | (0.93) | *2156.4<br>2175.8 | (0.99) | *4500.5<br>4531.4 | (0.99) | *6395.4<br>6453.4 | (0.99) |
| 10 ns | 775.9<br>775.9 | (1.00) | *1569.4<br>1613.5 | (0.97) | *2160.6<br>2212.4 | (0.98) | *4072.1<br>4184.2 | (0.97) | 6168.5<br>5979.3 | (1.03) |
| $\infty$ | 775.9<br>775.9 | (1.00) | 1522.0<br>1522.0 | (1.00) | 1925.2<br>1925.2 | (1.00) | 3838.2<br>3838.2 | (1.00) | 5625.2<br>5625.2 | (1.00) |
| $\infty$ [12] | 769.3 | | 1498.8 | | 1902.6 | | 3781.4 | | 5571.1 | |

*Experiments and Discussion*

Table 1 compares the total wirelength of routing solutions under non-uniform and uniform layer parasitics for standard test cases in the literature. The per-unit capacitance and per-unit resistance for the H-layer are $c_1 = 0.027\,\mathrm{fF}$ and $r_1 = 16.6\,\mathrm{m}\Omega$, respectively. For the uniform layer parasitics, the per-unit capacitance and per-unit resistance of the V-layer are equal to those of the H-layer, i.e., $c_2 = c_1$ and $r_2 = r_1$. For the non-uniform layer parasitics, we set $c_2 = 2.0 \cdot c_1$ and $r_2 = 3.0 \cdot r_1$, respectively. For simplicity, we use only the HV routing pattern and ignore the via resistance and capacitance. As shown in the Table, the solutions under non-uniform layer parasitics have larger total wirelength than those under uniform layer parasitics in most cases, especially when the skew bound is small. This may be due to the fact that merging regions under non-uniform layer parasitics tend to be smaller (and hence have higher merging cost at the next higher level) because the joining segments cannot be Manhattan arcs of non-zero length. When the skew bound is small, most of the merging regions are constructed from Manhattan arcs, and hence the solutions under non-uniform layer parasitics are more likely to have larger total wirelength. When the skew bound is infinite, no joining segments can be Manhattan arcs of non-zero length, and thus the routing solutions under non-uniform and uniform layer parasitics have identical total wirelength. In all the test cases, the wirelengths are evenly distributed among both routing layers—differences between the wirelengths on both layers are all less than 10% of the total wirelength, and less than 5% in most cases.




(a) Uniform layer parasitics (WL=2978 um)




(b) Non-uniform layer parasitics (WL=2808 um)

*Figure 4.* Examples of 8-sink zero-skew trees for the same uniform and non-uniform layer parasitics used in Table 1. Note that the merging segments (the dashed lines) in (a) are Manhattan arcs while those in (b) are not.

We also perform more detailed experiments on benchmark r1 to compare the total wirelength of zero-skew routing for different ratios of $r_2/r_1$ and $c_2/c_1$. When $(r_2c_2)/(r_1c_1)$ changes from 1 to 10, the total wirelength of solutions only varies between $+4\%$ and $-1\%$ from that obtained for uniform layer parasitics (i.e., $(r_2c_2)/(r_1c_1) = 1$). Hence, the routing solution obtained by our new BME method is insensitive to changes in the ratio of H-layer/V-layer RC values.

Figure 4 shows examples of 8-sink zero-skew clock routing trees using the same HV routing pattern and layer parasitics that are used in the Table 1 experiments. We observe that no merging segments under non-uniform layer parasitics are Manhattan arcs and joining segments are all single points. Notice that under any given routing pattern like HV or VH, some adjacent edges are inevitably overlapped. For example, edges $\overline{av}$ and $\overline{vp}$ in Fig. 4 are overlapped because both edges are routed using the same HV patterns. If edges $\overline{av}$ and $\overline{bv}$ are routed according to the VH routing pattern, the overlapping wire can be eliminated.

Finally, we note that under uniform layer parasitics the IME method [5] is identical to the BME method for zero-skew routing since all merging segments are Manhattan arcs. However, the IME method might be better than the BME method for non-uniform layer parasitics, since merging segments are no longer equal to Manhattan arcs.

## 3. Clock Routing in the Presence of Obstacles

This section proposes new merging region construction rules when there are obstacles in the routing plane. Without loss of generality, we assume that all obstacles are rectangular. We also assume that an obstacle occupies both the V-layer and H-layer (this is of course a strong assumption, and current work is directed to the case of per-layer obstacles). We first present the analysis for uniform layer parasitics, then extend our method to non-uniform layer parasitics; we also give experimental results and describe an application to planar clock routing.

### *3.1. Analysis for Uniform Layer Parasitics*

Given two merging regions $mr(a)$ and $mr(b)$, the merging region $mr(v)$ of parent node $v$ is constructed from joining segments $L_a \subseteq mr(a)$ and $L_b \subseteq mr(b)$. Observe that a point $p \in mr(v)$ inside an obstacle cannot be the feasible merging point. Furthermore, points $p, p' \in SDR(L_a, L_b)$ may have different minimum sums of pathlengths to $L_a$ and $L_b$ because obstacles that intersect $SDR(L_a, L_b)$ may cause different amounts of detour wiring from $p$ and $p'$ to $L_a$ and $L_b$. We define the *planar merging region* $pmr(v)$ to be the set of feasible merging points $p$ such that the pathlength of the shortest planar path (without going through obstacles) from $L_a$ through $p$ to $L_b$ is minimum (when the minimum pathlength from $L_a$ to $L_b$ is equal to $d(L_a, L_b)$, $pmr(v) \subseteq mr(v)$). Just as the merging region $mr(v)$ becomes a merging segment $ms(v)$ under zero-skew routing, the planar merging region $pmr(v)$ becomes the *planar merging segment* $pms(v)$ under zero-skew routing.

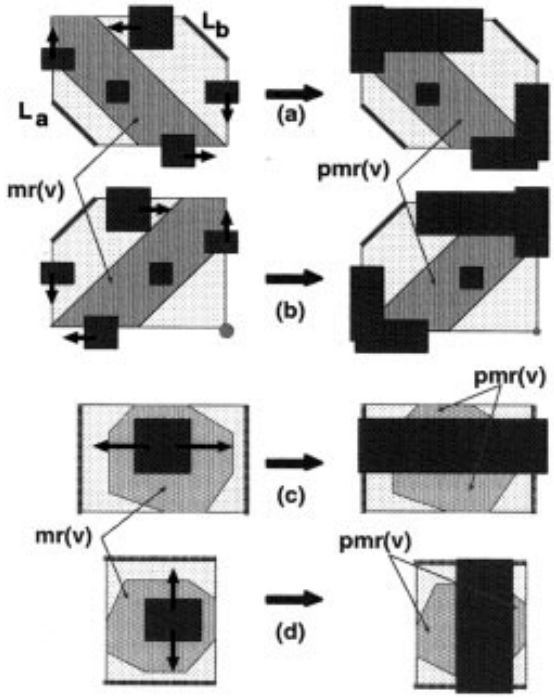



*Figure 5.* Illustration of obstacle expansion rules.

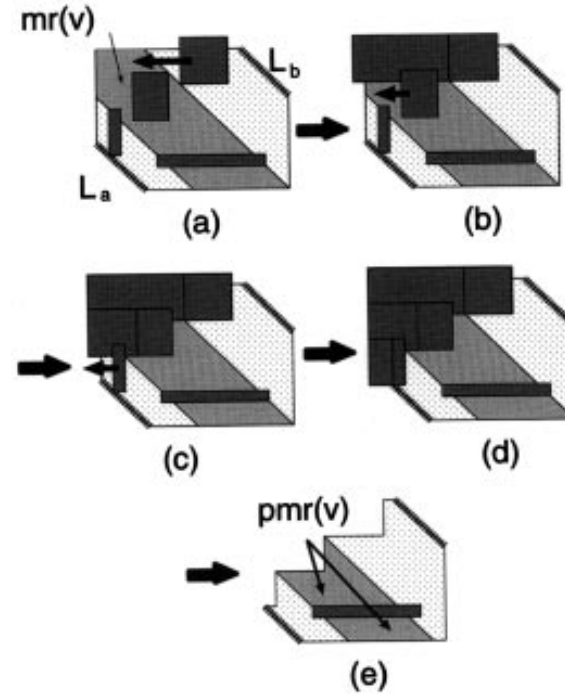



*Figure 6.* A "chain reaction" in the obstacle expansion.

The construction of $pmr(v)$ is as follows. If joining segments $L_a$ and $L_b$ overlap, $pmr(v) = mr(v) = L_a \cap L_b$. Otherwise, we expand any obstacles that intersect with rectilinear boundaries of $SDR(L_a, L_b)$ as illustrated in Fig. 5 for four possible cases; these define the *Obstacle Expansion Rules*.

*Case I.* (expand as in Fig. 5(a)).

1. $L_a = \{p_1\}$, $L_b = \{p_2\}$, and $\overline{p_1 p_2}$ has finite non-zero positive slope $m$, i.e., $0 < m < \infty$.
2. $L_a$ or $L_b$ is a Manhattan arc of non-zero length with slope $-1$.

*Case II.* (expand as in Fig. 5(b)).

1. $L_a = \{p_1\}$, $L_b = \{p_2\}$, and $\overline{p_1 p_2}$ has finite non-zero negative slope $m$, i.e., $-\infty < m < 0$.
2. $L_a$ or $L_b$ is a Manhattan arc of non-zero length with slope $+1$.

*Case III.* (expand as in Fig. 5(c)). Both joining segments are vertical segments, possibly of zero length.

*Case IV.* (expand as in Fig. 5(d)). Both joining segments are horizontal segments, possibly of zero length.

In Case I, an obstacle $O$ which intersects with the top (bottom) boundary of $SDR(L_a, L_b)$ is expanded horizontally toward the left (right) side until $O$ reaches the left (right) boundary of $SDR(L_a, L_b)$. If $O$ intersects with the left (right) boundary of $SDR(L_a, L_b)$, then $O$ is expanded upward (downward) until $O$ reaches the top (bottom) boundary of $SDR(L_a, L_b)$. Case II is symmetric. In Case III, an obstacle $O$ intersecting with $SDR(L_a, L_b)$ is expanded along the horizontal direction until $O$ reaches both joining segments. Case IV is symmetric, with expansion in the vertical direction[4]. Finally, note that in Cases I and II an expanded obstacle $O$ can intersect with another obstacle, which is then expanded in the same way; this sort of "chain reaction" is illustrated in Fig. 6.

With these obstacle expansion rules, we may complete the description of the planar merging region construction. For child regions $mr(a)$ and $mr(b)$ of node $v$, $pmr(v)$ is constructed as follows.

1. Apply the obstacle expansion rules to expand obstacles.
2. Calculate $pmr(v) = \{p \mid p \in mr(v) -$ expanded obstacles$\}$.
3. Restore the sizes of all the expanded obstacles.

4. If $pmr(v) \neq \emptyset$ then stop; continue with next step otherwise.
5. Compute the shortest planar path $P$ between $mr(a)$ and $mr(b)$.
6. Divide path $P$ into a minimum number of subpaths $P_i$ such that the pathlength of $P_i$, $cost(P_i)$, is equal to the (Manhattan) distance between the endpoints of $P_i$, i.e., if subpath $P_i = s \rightsquigarrow t$, then $cost(P_i) = d(s, t)$.
7. Calculate delay and skew functions for each line segment in $P$.
8. For each subpath $P_i$ which has a point $p$ with feasible or minimum skew, use the endpoints of $P_i$ as the new joining segments. Then, calculate the planar merging region $pmr_i(v)$ with respect to the new joining segments, using Steps 1, 2 and 3. (Note that $pmr_i(v) \neq \emptyset$ since $p \in pmr_i(v)$).
9. $pmr(v) = \cup pmr_i(v)$, where subpath $P_i \subseteq P$ contains a point $p$ with feasible or minimum skew.

Notice that the purpose of Step 6 is to maximize the area of $pmr(v)$. As shown in Fig. 7, if we divide subpath $P_2 = y - z - t$ into two smaller subpaths $y - z$ and $z - t$, region $pmr_2(v)$ in the Figure will shrink to be within the shortest distance region $SDR(y, z)$. Thus, like the merging regions constructed by the BME method, the planar merging regions will contain all the minimum-cost merging points when no detouring occurs. For the same reason stated in the Elmore-Planar-DME algorithm [13] the planar merging regions along the shortest planar path will not guarantee minimum tree cost at the next higher level. Thus, it is possible to construct and maintain planar merging regions along several shortest planar paths. At the same time, if an internal node $v$ can have multiple planar merging regions, the number of merging regions may grow exponentially during the bottom-up construction of merging regions (this is the difficulty encountered by the IME method of [5]). Our current implementation simply keeps at most $k$ regions with lowest tree cost for each internal node.

Finally, in the top-down phase of Ex-DME each node $v$ is embedded at a point $q \in L_v$ closest to $l(p)$ (where $p$ is the parent node of $v$), and that $L_v \in mr(v)$ is one of the joining segments used to construct $mr(p)$. When $L_v$ is a Manhattan arc of non-zero length, there can be more than one embedding point for $v$. However, when obstacles intersect $SDR(l(p), L_v)$, some of the embedding points $q \in L_v$ closest to $l(p)$ may become infeasible because the shortest planar path from $q$ to $l(p)$ has pathlength $> d(l(p), L_v)$. To remove infeasible embedding points from $L_v$, we treat $l(p)$ and $L_v$ as two joining segments, then apply the obstacle expansion rules as in Fig. 8(b). If $L'_v$ denotes the portion of $L_v$ left uncovered by the expanded obstacles, the feasible embedding locations for $v$ consist of the points on $L'_v$ that are closest to $l(p)$.

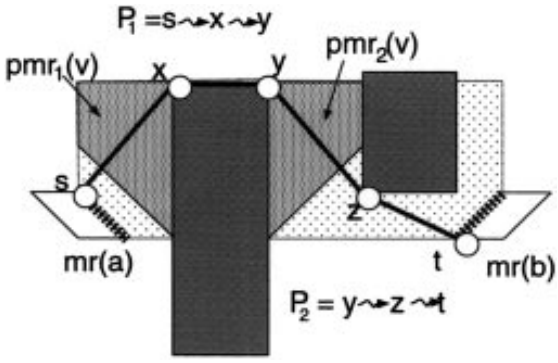



*Figure 7.* Construction of planar merging regions along a shortest planar path between child merging regions.

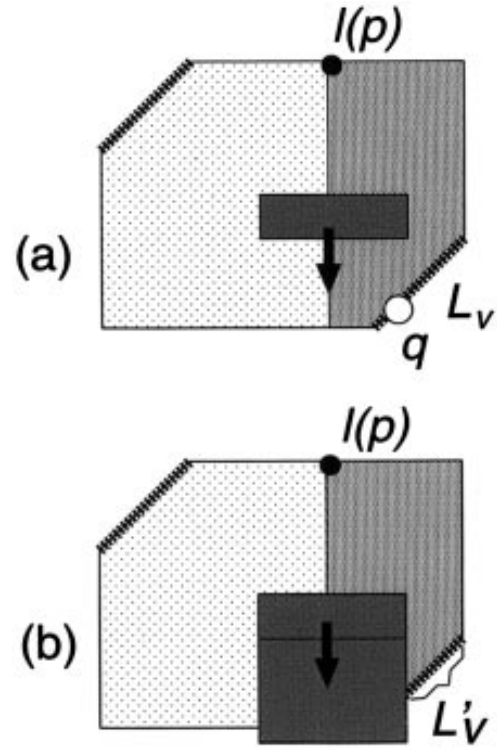



*Figure 8.* Modification of the embedding rule in the top-down phase of the Ex-DME algorithm when there are obstacles in the routing plane.

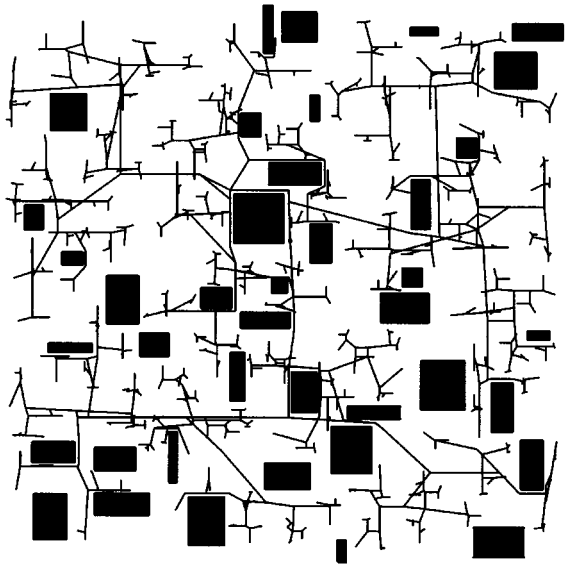


*Figure 9.* A zero-skew solution for the 555-sink test case with 40 obstacles.

### *3.2. Experimental Results*

Our obstacle-avoiding BST routing algorithm was tested on four examples respectively having 50, 100, 150 and 555 sinks with uniformly random locations in a 100 by 100 layout region; all four examples have the same 40 randomly generated obstacles shown in Fig. 9. For comparison, we run the same algorithm on the same test cases without any obstacles. Details of the experiment are as follows. Parasitics are taken from MCNC benchmarks Primary1 and Primary2, i.e., all sinks have identical 0.5 pF loading capacitance and the per-unit wire resistance and wire capacitance are 16.6 mΩ and 0.027 fF. For each internal node, we maintain at most $k = 5$ merging regions with lowest tree cost. We use the procedure Find-Shortest-Planar-Path of the Elmore-Planar-DME algorithm [13] to find shortest planar $s$-$t$ paths. The current implementation uses Dijkstra's algorithm in the visibility graph $G(V, E)$ (e.g., [14, 15]) where $V$ consists of the source and destination points $s$, $t$ along with detour points around the corners of obstacles. The weight $|e|$ of edge $e = (p, q) \in E$ is computed on the fly; if $e$ intersects any obstacle, then $|e| = \infty$, else $|e| = d(p, q)$. The running time of obstacle-avoidance routing can be substantially improved with more sophisticated data structures for detecting the intersection of line segments and obstacles, and faster path-finding heuristic in the geometric plane. Table 2 shows that the wirelengths of routing solutions with obstacles are very close to those of routing solutions without obstacles (typically within a few percent). Runtimes (reported for a Sun 85 MHz Sparc-5) are significantly higher (by factors of up to 18 for the 50-sink instance) when the 40 obstacles are present; we believe that this is due to our current naive implementation of obstacle-detecting and path-finding. Figure 9 shows the zero-skew clock routing solution for the 555-sink test case.

*Table 2.* Total wirelength and runtime for obstacle-avoiding BST algorithm, for various instances and skew bounds. Sizes and locations of obstacles are shown in Fig. 9. Numbers in parentheses are ratios to corresponding (total wirelength, runtime) values when no obstacles are present in the layout.

| #Sinks | 50 | 100 | 150 | 555 |
|---|---|---|---|---|
| Skew bound | Wirelength: $\mu$m (normalized)<br>CPU time: hr:min:sec (normalized) | | | |
| 0 | 8791.1(1.06)<br>00:00:04(4) | 11925.1(1.04)<br>00:00:10(2) | 14747.5(1.03)<br>00:00:15(2) | 28854.8(1.01)<br>00:00:34(1) |
| 1 ps | 8048.7(1.09)<br>00:01:09(6) | 10761.4(1.04)<br>00:05:20(7) | 13388.5(1.03)<br>00:11:36(3) | 26240.0(1.04)<br>00:44:14(10) |
| 2 ps | 7831.9(1.07)<br>00:01:47(8) | 10796.8(1.01)<br>00:08:17(9) | 12643.0(1.02)<br>00:20:55(10) | 25205.2(1.04)<br>01:30:08(13) |
| 5 ps | 7140.9(1.04)<br>00:04:01(13) | 10493.6(1.08)<br>00:15:16(11) | 11598.8(1.01)<br>00:30:34(13) | 23648.0(1.04)<br>01:30:08(13) |
| 10 ps | 7126.2(1.06)<br>00:06:13(14) | 9701.2(1.03)<br>00:19:36(12) | 11426.1(1.07)<br>00:36:30(12) | 22737.3(1.05)<br>01:48:06(13) |
| 20 ps | 6831.6(1.13)<br>00:07:40(15) | 9296.4(1.03)<br>00:21:56(10) | 11606.0(1.10)<br>00:40:39(3) | 21641.7(1.05)<br>03:42:52(24) |
| 50 ps | 6468.4(1.12)<br>00:10:36(15) | 8739.6(1.09)<br>00:26:47(11) | 10194.4(1.10)<br>01:00:50(13) | 22167.1(1.15)<br>02:18:20(14) |
| 100 ps | 6484.7(1.20)<br>00:13:51(18) | 8588.2(1.11)<br>00:30:16(9) | 9295.6(1.02)<br>01:03:00(15) | 19086.6(1.01)<br>03:06:23(17) |
| 1 ns | 6484.7(1.24)<br>00:16:20(18) | 8115.1(1.13)<br>00:36:52(11) | 9265.8(1.10)<br>01:18:36(15) | 17166.8(.99)<br>07:24:38(12) |
| 10 ns | 6484.7(1.24)<br>00:16:19(18) | 8115.1(1.13)<br>00:36:43(11) | 9265.8(1.10)<br>01:20:07(15) | 16698.3(.99)<br>03:18:20(7) |
| $\infty$ | 6484.7(1.24)<br>00:16:43(18) | 8115.1(1.13)<br>00:36:52(11) | 9265.8(1.10)<br>01:20:25(13) | 16698.3(1.02)<br>03:21:11(7) |

### *3.3. Extension to Non-Uniform Layer Parasitics*

When the layer parasitics are non-uniform, no joining segment can be a Manhattan arc, so Cases I.2 and II.2 of the obstacle expansion rules are inapplicable. In Cases III and IV, only one routing layer will be used to merge the child regions, so the construction of planar merging regions will be the same as with uniform layer parasitics. Hence, the construction of planar merging

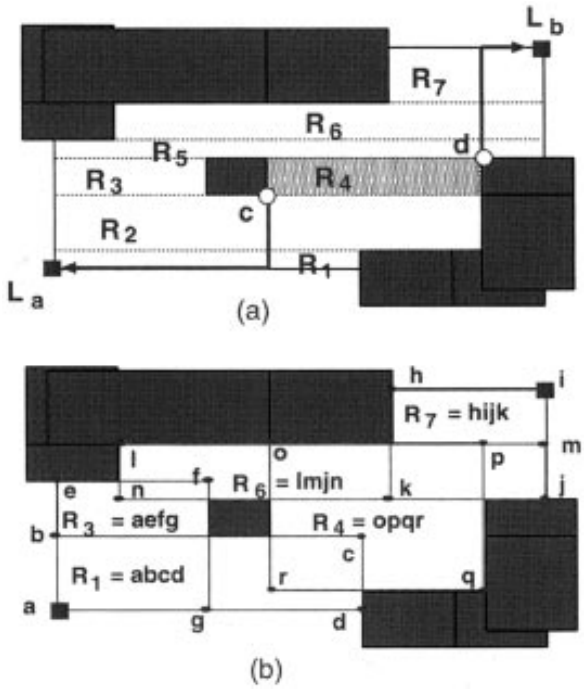



*Figure 10.* Obstacle-avoidance routing for non-uniform layer parasitics when joining segments $L_a$ and $L_b$ are single points not on the same vertical or horizontal line.

regions changes only for Cases I.1 and II.1, i.e., when the joining segments $L_a$ and $L_b$ are two single points which are not on the same vertical or horizontal line.

Since larger merging regions will result in smaller merging costs at the next higher level, a reasonable approach[5] is to maximize the size of the merging region constructed within each rectangle $R_i \subseteq SDR(L_a, L_b)$, by expanding $R_i$ as shown in Fig. 10(b). After expansion, "redundant" rectangles contained in the expansions of other rectangles (e.g., rectangles $R_2$ and $R_5$ in Fig. 10 are contained in the union of expansions of $R_1$, $R_3$, $R_4$, $R_6$ and $R_7$) can be removed to simplify the computation. The merging region construction for Cases I.1 and II.1 with non-uniform layer parasitics is summarized as follows.

1. Divide $SDR(L_a, L_b)$ into a set of disjoint rectangles $R_i$ by extending horizontal boundary segments of the (expanded) obstacles in $SDR(L_a, L_b)$.
2. Expand each rectangle $R_i$ until blocked by obstacles.
3. Remove rectangles $R_i$ that are completely contained by other rectangles.
4. For each rectangle $R_i$ do:
   - Let $c \in R_i$ and $d \in R_i$ be the corner points which are closest to joining segment $L_a$ and $L_b$. Apply prescribed routing patterns from $c$ to $L_a$ and from $d$ to $L_b$.
   - Calculate delays at $c$ and $d$.
   - Construct the merging region from points $c$ and $d$ as as described in Section 2.

Finally, we notice that in planar clock routing, all wires routed at a lower level become obstacles to subsequent routing at a higher level. Also, in the obstacle-avoidance routing, if some obstacle blocks only one routing layer, then the routing over the obstacle must be planar. In such cases, we may apply the concept of the planar merging region to improve the planar clock routing. In particular, we improve the Elmore-Planar-DME algorithm [13, 16] by (i) constructing the planar merging segment $pms(v)$ for each internal node $v$ of the input topology $G$, and (ii) replacing the Find-Merging-Path and Improve-Path heuristics of Elmore-Planar-DME by construction of a shortest planar path $P$ connecting $v$'s children $s$ and $t$ via $v$'s embedding point $l(v) \in pms(v)$. Total wirelength can be reduced because $l(v)$ is now selected by the DME method optimally from $pms(v)$ instead of being selected heuristically by Find-Merging-Path and Improve-Path in Elmore-Planar-DME. Our experiments [17] show that Elmore-Planar-DME is consistently improved by this technique.

## 4. Buffered Clock Tree Synthesis

Finally, we extend our bounded-skew routing method to handle the practical case of buffering hierarchies in large circuits. There have been many works on buffered clock tree designs. [18–20] determine the buffer tree hierarchy for the given clock tree layout or topology. [21, 22] design the buffer tree hierarchy and the routing of the clock net simultaneously. However, the prevailing design methodology for clock tree synthesis is that the buffer tree hierarchy is pre-designed before the physical layout of the clock tree (e.g., see recent vendor tools for automatic buffer hierarchy generation, such as Cadence's CT-Gen tool). In practice, a buffer hierarchy must satisfy various requirements governing, e.g., phase delay ("insertion delay"), clock edge rate, power dissipation, and estimated buffer/wire area. Also, the placement and routing estimation during chip planning must have reasonably accurate notions of buffer and decoupling capacitor areas, location of wide edges in

the clock distribution network, etc. For these reasons, buffer hierarchies are typically "pre-designed" well in advance of the post-placement buffered clock tree synthesis. So our work starts with a given buffer hierarchy as an input; this defines the number of buffer levels and the number of buffers at each level. We use the notation $k_M - k_{M-1} - \cdots - k_0$ to represent a buffer hierarchy with $k_i$ buffers at level $i$, $0 \leq i \leq M$. For example, a 170-16-4-1 hierarchy has 170 buffers at level 3, 16 buffers at level 2, etc. Note that we always have $k_0 = 1$ since there is only one buffer at the root of the clock tree. As in [19, 20, 22], to minimize the skew induced by the changes of buffer sizes due to the process variation, we assume that identical buffers are used at the same buffer level. (From the discussion of our method below, we can see that our method can work without this assumption by minor modification.)

We propose an approach to bounded-skew clock tree construction for a given buffer hierarchy. Our approach performs the following steps at each level of the hierarchy, in bottom-up order.

1. Cluster the nodes in the current level (i.e., roots of subtrees in the buffer hierarchy, which may be sinks or buffers) in the current level into the appropriate number of clusters (see Section 4.1).
2. Build a bounded-skew tree for each cluster by applying the ExG-DME algorithm under Elmore delay [5].
3. Reduce the total wirelength by applying a buffer sliding heuristic (see Section 4.2).

### *4.1. Clustering*

The first step is to assign each node (e.g., sink or buffer) in the current level $i$ of the buffer hierarchy to some buffer in level $i-1$. The set of nodes assigned to a given level $i-1$ buffer constitute a *cluster*. If there are $k$ buffers in the next higher level of the buffer hierarchy, then this is a $k$-way clustering problem. Numerous algorithms have been developed for geometric clustering (see, e.g., the survey in [23]); our empirical studies show that the K-Center technique of Gonzalez [24] tends to produce more balanced clusters than other techniques. Furthermore, the K-Center heuristic has only $O(nk)$ time complexity (assuming $n$ nodes at the current level). The basic idea of K-Center is to iteratively select $k$ cluster *centers*, with each successive center as far as possible from all previously selected centers. After all $k$ cluster centers have been selected, each node at the current level is assigned to the nearest center. Pseudo-code for K-Center is given in Fig. 11 (reproduced from [23]), with Steps 0 and 3*a* added to heuristically maximize the minimum distance among the $k$ cluster centers.

We propose to further balance the clustering solution from K-Center using the iterative procedure PostBalance in Fig. 12, which greedily minimizes the objective function $\sum_{i=1,k} Cap(X_i)^w$. Here,

- $Cap(X_i)$ is the estimated total capacitance of the BST (to be constructed in the second major step of our approach) over sinks in cluster $X_i$. In other words, $Cap(X_i) = \sum_{v \in X_i} c_v + d(l(v), center(X_i)) \cdot c$,

**Algorithm** $k$-center($\mathcal{S}, X_1 \ldots, X_k, k$)
**Input:** Set of subtree roots (e.g., sinks or buffers) $\mathcal{S}$;
number of clusters $k$
**Output:** Sets of clusters $\{X_1, X_2, \ldots, X_k\}$
0. Calculate $V = \mathcal{S} \bigcup \mathcal{U}$, where $\mathcal{U} = \{ u \mid u =$ a grid point of $\sqrt{|\mathcal{S}|}$ uniformly spaced horizontal and vertical lines inside $bbox(\mathcal{S})$ $\}$
1. Initialize $W$, a set of cluster *centers*, to empty.
2. Choose some random $v$ from $V$ and add it to $W$.
3. **while** $|W| \leq k$, find $v \in V$ s.t. $d_W = \min_{w \in W} d(v, w)$ is maximized, and add it to $W$.
3a. **while** $\exists v_1 \in W, v_2 \in V - W$ s. t. $d_W$ can be increased by swapping $v_1$ and $v_2$, then swap $v_1$ and $v_2$ (i.e., $W = W + \{v_2\} - \{v_1\}$).
4. Form clusters $X_1, X_2, \ldots, X_k$ each containing a single point of $W$;
place each $v \in \mathcal{S}$ into the cluster of the closest $w_i \in W$.

*Figure 11.* Pseudocode for a modified K-center heuristic.

**Procedure** PostBalance($X_1 \ldots, X_k$)
**Input:** Sets of clusters $\{X_1 \ldots, X_k\}$ s.t. $X_i \bigcap X_j = \emptyset \quad \forall 1 \leq i \neq j \leq k$
**Output:** Sets of clusters $\{X_1, \ldots, X_k\}$ s.t. $X_i \bigcap X_j = \emptyset \quad \forall 1 \leq i \neq j \leq k$
1. Calculate $S = \bigcup_{i=1,k} X_i$
2. **do**
3. Sort clusters in increasing order of estimated load capacitance
4. **for** each cluster $X_i$ in the sorted order
5. $n\_move = 0$
6. Let $V = \{ v \mid v \in S - X_i \}$
7. Sort nodes $v \in V$ in increasing order of $d(v, center(X_i))$
8. **for** each node $v \in V$ in the sorted order
Suppose $v \in X_j$, $1 \leq j \neq i \leq k$
9. **if** $\sum_{i=1,k} (Cap(X_i))^5$ decreases by moving $v$ to cluster $X_i$
10. Move $v$ to cluster $X_i$ (i.e., $X_i = X_i + \{v\}$, $X_j = X_j - \{v\}$)
11. $n\_move = n\_move + 1$
12. **if** $n\_move > 3$ **Go To** 4
13. **while** there is any sink moved in current iteration.

*Figure 12.* Procedure PostBalance.

where $c_v$ is the input capacitance of node $v$ and $center(X_i)$ is the Manhattan center of the nodes in cluster $X_i$ as defined in [25, 16].[6]

- The number $w$ is used to trade off between balance among clusters and the total capacitive load of all clusters. A higher value of $w$ favors balanced clustering, which usually leads to lower-cost routing at the next higher level but can cause large total capacitive load at the current level. On the other hand, $w = 1$ favors minimizing the total capacitive load at the current level without balancing the capacitive load among the clusters. Based on our experiments, we use $w = 5$ to obtain all the results reported below; this value seems to reasonably balance the goals of low routing cost at both the current and next higher levels[7].

### 4.2. *Buffer Sliding*

Chung and Cheng [20] shift the location of a buffer along the edge to its parent node to reduce or eliminate excessive detouring. The motivation for their technique is straightforward. In Fig. 13, subtree $T_1$ rooted at $v_1$ is driven by buffer $b_1$, and subtree $T_2$ rooted at $v_2$ is driven by buffer $b_2$. Let $t_2$ be the delay from parent node $p$ to child node $v_2$, and let $t_2'$ be the delay from parent node $p$ to child node $v_2$ after buffer $b_2$ slides toward node $p$ over a distance of $x$ units. Let $l = d(l(p), l(v_2))$. We now have

$$\begin{aligned} t_2 &= rl(cl/2 + c_b) + t_b + r_b \cdot Cap(T_2) \\ t_2' &= r(l-x)(c(l-x)/2 + c_b) + t_b \\ &\quad + r_b(cx + Cap(T_1)) + rx(cx/2 + Cap(T_2)) \\ t_2' - t_2 &= rcx^2 + r_b cx + r(Cap(T_2) - cl - c_b)x \end{aligned} \quad (9)$$

Notice that the coefficient of the last term in Eq. (9), $Cap(T_2) - cl - c_b$, is always positive in practice because (i) the total wirelength of $T_2$ is larger than that of the parent edge of $T_2$, and (ii) the sum of sink capacitances in $T_2$ is larger than the input capacitance of a buffer, so that $t_2' > t_2$. Also, as buffer $b_2$ is moved closer to its parent node $p$, delay $t_2'$ will increasingly exceed $t_2$. In the case where $t_1$ is so much larger than $t_2$ that detour wiring is necessary, we can slide buffer $b_2$ so that delay balance is achieved at point $p$ using less detour wiring (see Fig. 13(a)). Even when no detour wiring is necessary, the buffer sliding technique can still be used to reduce routing wirelength at the next higher level of the hierarchy. In Fig. 13(b), we reduce the wirelength by constructing a minimal Steiner tree over $b_1$ and $b_2$. Suppose the delay from $p'$ to buffer $b_1$ is larger than that from $p'$ to buffer $b_2$, we can slide buffer $b_2$ toward the left, thus increasing the delay from $p'$ to $b_2$ such that $p'$ can become the delay balance point.

There is a similar idea in [21], which reduces wirelength by inserting an extra buffer. However, adding a buffer will cause large extra delay and power dissipation. Indeed, when $T_a$ and $T_b$ have similar delays, excessive detour wirelength is inevitable when a buffer is added at the parent edge of just one subtree. Hence, the technique of [21] will be effective in reducing power dissipation and wirelength only when the delays of $T_a$ and $T_b$ are very different. ([21] also consider buffer insertion only for the zero-skew case.)

We now give a buffer sliding heuristic, called **H3** (see Fig. 14) that does not add any extra buffers and that can handle any skew bound (we find, however, that it is less effective for large skew bounds; see Section 4.3). H3 builds a low-cost tree $T_{\text{opt}}$ over a set of of buffers $\mathcal{S} = \{b_1, \ldots, b_k\}$ as follows. First, we construct a BST $\bar{T}$ under a new skew bound $\bar{B} \geq B$ without buffer sliding. Next, we calculate the delay $d^i_{\max}$ ($d^i_{\min}$) which is the maximum (minimum) delay along any root-sink path in $\bar{T}$ that passes through buffer $b_i$ (Line 7). We then calculate $d_{\max} = \max_{i=1,k}\{d^i_{\max}\}$ at Line 8. At Line 10, we slide each buffer $b_i$ such that the min-delay at its input is increased by $\max\{0, d_{\max} - d^i_{\min} - B\}$ and $skew(\bar{T})$ is reduced toward $B$. Finally, we build a new tree $T$ by re-embedding the topology of $\bar{T}$ according to the original skew bound $B$ (Line 9); this will minimize

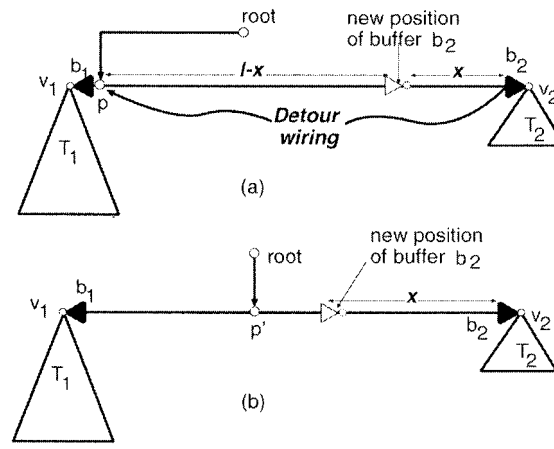



*Figure 13.* Two examples showing how the buffer sliding technique can eliminate (a) detour wiring or (b) routing wirelength at higher levels of the buffer hierarchy.

**Procedure** H3($\mathcal{S}$)
**Input:** Set of buffers $\mathcal{S} = \{b_1, \cdots, b_k\}$; Skew Bound $B$; Set of subtrees $T_i$ driven by buffer $b_i$ with $skew(T_i) < B$;
**Output:** Tree $T_{opt}$ with $skew(T_{opt}) \leq B$; Set of wirelength $L_i \geq 0$ inserted between buffer $b_i$ and its subtree root $root(T_i)$.

```
1.  min_cost = ∞
2.  Set new skew bound B̄ = B
3.  do
4.      Build tree T̄ over buffers in S with new
        skew bound B̄ (no buffer sliding)
5.      for i = 1 to k do
        /* Let max_t̄(b_i) (max_t(b_i)) be max-delay
           from input of buffer b_i to sinks which are
           descendants of b_i before (after) buffer
           sliding */
6.          Calculate x = delay from root(T̄)
            along the unique path in T̄ to b_i
7.          Calculate d^i_max = max_t̄(b_i) + x, and
            d^i_min = min_t̄(b_i) + x
8.      Calculate d_max = max{d^i_max}
9.      for i = 1 to k do
10.         Calculate the length of wire L_i between
            b_i and root(T_i) s.t. min_t(b_i) = min_t̄(b_i)
            + max{0, d_max − d^i_min − B}
11.     Build tree T by re-embedding topology of T̄
            under original skew bound B with wire of
            length L_i inserted between b_i and root(T_i),
            ∀i = 1, ··· , k
12.     if cost(T) < min_cost
13.         T_opt = T
14.         min_cost = cost(T)
15.     B̄ = B̄ + 3ps
16. while min_cost ever decreased in last 10
        iterations
```

*Figure 14.* Procedure H3 (buffer sliding).

any potential increase in tree cost $cost(T) - cost(\bar{T})$. The above steps are iterated for different skew bounds $\bar{B} > B$, and the tree $T$ with smallest total wirelength is chosen as $T_{\text{opt}}$. In general, when the new skew bound $\bar{B}$ is increasing, $cost(\bar{T})$ will be decreasing. However, the length of the wire inserted between each buffer and its subtree root will increase when the $\bar{B}$ becomes too large, and $cost(T)$ will stop decreasing after a certain number of iterations. In all of our experiments, the procedure stops within 50 iterations.

### *4.3. Experimental Results*

For the sake of comparison, we have also implemented the following buffer sliding heuristics.

**H0** No buffer sliding.

**H1** Slide buffers to equalize $max\_t(b_i)$ for all $1 \leq i \leq k$, i.e., the max-delay from the input of each buffer $b_i$ to sinks which are the descendants of $b_i$. This is the buffer sliding technique used in [19, 22].

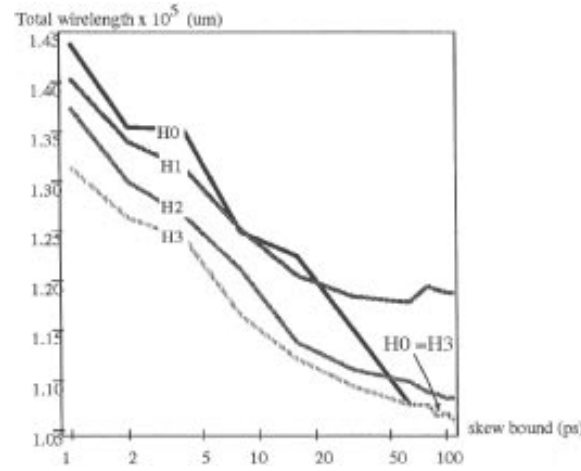



*Figure 15.* Total wirelength achieved by different buffer sliding heuristics on benchmark circuit r1 with a 32-1 buffer hierarchy. The wirelength unit is 100 $\mu$m. Buffer parameters are output resistance $r_b = 100\ \Omega$, input capacitance $c_b = 50$ fF, and internal delay $t_b = 100$ ps. Note that the X axis is on a logarithmic scale.

**H2** Slide buffers to equalize $max\_t(b_i)$ and $max\_t(b_j)$ where $b_i$ and $b_j$ are the sibling buffers.

Figure 15 shows the total wirelength reduction achieved by the various buffer sliding heuristics on benchmark circuit r1 with a 32-1 buffer hierarchy. H3 is consistently better than other heuristics for the skew bound from 0 to 50 ps. When the skew bound $B$ is larger than 50 ps, the tree cost reduction $cost(T) - cost(\bar{T})$ is very slight for any $\bar{B} > B$, and hence when we push $skew(\bar{T})$ back to $B$ by buffer sliding, there is almost no gain in the total wirelength. Therefore, heuristic H3 will be the same as H0 when the skew bound is sufficiently large. A more detailed comparison of total wirelength reduction achieved by different buffer sliding heuristics is given in Table 3, which shows that H3 is consistently better than other heuristics for different skew bounds and buffer hierarchies. In the table, we also report ratios of tree costs, averaged over the five test cases, for each heuristic versus H3 (i.e., we normalize the tree costs against the H3 tree cost). For the zero skew regime, the heuristics H0, H1 and H2 respectively require 6.9%, 10.6% and 3.0% more wirelength on average than our heuristic H3. And for the 50 ps skew regime, the heuristics H0, H1 and H2 respectively require 3.1%, 17.0% and 1.1% more wirelength on average than our heuristic H3. Notice that heuristic H1, the method used in [19, 22], actually has the largest total wirelength in most cases.

*Table 3.* Detailed comparison of total wirelength achieved by different buffer sliding heuristics on benchmark circuits r1–r5 with two types of 2-level buffer hierarchy and one type of 3-level buffer hierarchy. The wirelength unit and buffer parameters are the same as those in Fig. 15.

| | r1 | r2 | r3 | r4 | r5 | r1 | r2 | r3 | r4 | r5 |
|---|---|---|---|---|---|---|---|---|---|---|
| | Skew bound = 0 | | | | | Skew bound = 10 ps | | | | |
| | Buffer hierarchy: $2\sqrt{n}-1$ | | | | | | | | | |
| $H0$ | 1,486 | 2,984 | 3,728 | 7,718 | 11,193 (1.059) | 1,242 | 2,446 | 3,095 | 6,279 | 9,102 (1.061) |
| $H1$ | 1,483 | 3,207 | 3,855 | 8,829 | 11,567 (1.119) | 1,232 | 2,850 | 3,175 | 7,710 | 9,785 (1.164) |
| $H2$ | 1,458 | 2,941 | 3,651 | 7,408 | 10,852 (1.032) | 1,185 | 2,400 | 3,012 | 6,018 | 8,773 (1.025) |
| $H3$ | 1,404 | 2,802 | 3,553 | 7,261 | 10,589 (1.000) | 1,172 | 2,314 | 2,921 | 5,907 | 8,549 (1.000) |
| | Buffer hierarchy: $\sqrt{n}-1$ | | | | | | | | | |
| $H0$ | 1,497 | 2,923 | 3,733 | 7,476 | 11,185 (1.044) | 1,231 | 2,516 | 3,053 | 6,128 | 9,128 (1.060) |
| $H1$ | 1,447 | 2,896 | 3,848 | 7,661 | 11,418 (1.051) | 1,219 | 2,408 | 3,207 | 6,297 | 9,470 (1.073) |
| $H2$ | 1,450 | 2,825 | 3,646 | 7,319 | 10,878 (1.015) | 1,170 | 2,419 | 2,982 | 5,972 | 8,820 (1.023) |
| $H3$ | 1,432 | 2,797 | 3,584 | 7,175 | 10,713 (1.000) | 1,159 | 2,340 | 2,893 | 5,872 | 8,597 (1.000) |
| | Buffer hierarchy: $n^{2/3}-n^{1/3}-1$ | | | | | | | | | |
| $H0$ | 1,626 | 3,259 | 4,017 | 7,971 | 11,989 (1.104) | 1,306 | 2,693 | 3,375 | 6,713 | 9,816 (1.112) |
| $H1$ | 1,558 | 3,297 | 4,168 | 8,912 | 12,982 (1.149) | 1,258 | 2,926 | 3,547 | 7,870 | 10,954 (1.198) |
| $H2$ | 1,556 | 2,989 | 3,808 | 7,594 | 11,368 (1.042) | 1,234 | 2,470 | 3,136 | 6,379 | 9,204 (1.040) |
| $H3$ | 1,476 | 2,877 | 3,636 | 7,374 | 10,921 (1.000) | 1,193 | 2,361 | 3,020 | 6,152 | 8,813 (1.000) |
| | Skew bound = 20 ps | | | | | Skew bound = 50 ps | | | | |
| | Buffer hierarchy: $2\sqrt{n}-1$ | | | | | | | | | |
| $H0$ | 1,185 | 2,375 | 2,950 | 6,021 | 8,736 (1.059) | 1,074 | 2,168 | 2,780 | 5,630 | 8,245 (1.018) |
| $H1$ | 1,182 | 2,626 | 3,127 | 7,323 | 9,401 (1.155) | 1,200 | 2,565 | 3,110 | 7,061 | 9,021 (1.174) |
| $H2$ | 1,147 | 2,245 | 2,893 | 5,816 | 8,397 (1.021) | 1,109 | 2,170 | 2,745 | 5,530 | 7,918 (1.010) |
| $H3$ | 1,112 | 2,216 | 2,845 | 5,695 | 8,231 (1.000) | 1,073 | 2,158 | 2,736 | 5,477 | 7,822 (1.000) |
| | Buffer hierarchy: $\sqrt{n}-1$ | | | | | | | | | |
| $H0$ | 1,196 | 2,404 | 2,971 | 5,937 | 8,708 (1.058) | 1,127 | 2,224 | 2,772 | 5,504 | 8,416 (1.064) |
| $H1$ | 1,153 | 2,370 | 3,116 | 6,116 | 9,248 (1.077) | 1,127 | 2,169 | 2,920 | 5,706 | 8,855 (1.089) |
| $H2$ | 1,146 | 2,280 | 2,944 | 5,743 | 8,397 (1.022) | 1,061 | 2,115 | 2,685 | 5,404 | 8,005 (1.020) |
| $H3$ | 1,135 | 2,228 | 2,839 | 5,617 | 8,271 (1.000) | 1,053 | 2,080 | 2,607 | 5,261 | 7,854 (1.000) |
| | Buffer hierarchy: $n^{2/3}-n^{1/3}-1$ | | | | | | | | | |
| $H0$ | 1,267 | 2,538 | 3,125 | 6,396 | 9,350 (1.089) | 1,132 | 2,344 | 2,913 | 5,823 | 8,551 (1.011) |
| $H1$ | 1,256 | 2,780 | 3,494 | 7,532 | 10,806 (1.206) | 1,262 | 2,891 | 3,439 | 7,735 | 11,182 (1.248) |
| $H2$ | 1,191 | 2,401 | 2,969 | 6,077 | 8,811 (1.030) | 1,145 | 2,301 | 2,856 | 5,786 | 8,475 (1.003) |
| $H3$ | 1,158 | 2,339 | 2,893 | 5,864 | 8,536 (1.000) | 1,112 | 2,312 | 2,937 | 5,684 | 8,327 (1.000) |

## 5. Conclusions

In this work, we have extended the bounded-skew routing methodology to encompass several very practical clock routing issues: non-uniform layer parasitics, non-zero via resistance and/or capacitance, existing obstacles in the metal routing layers, and hierarchical buffered tree synthesis. For the case of varying layer parasitics, we prove that if we prescribe the routing pattern between any two points, merging regions are still bounded by well-behaved segments except that no boundary segments can be Manhattan arcs of non-zero length. Our experimental results show that taking into account non-uniform layer parasitics can be accomplished without significant penalty in the clock tree cost. Our solution to obstacle-avoidance routing

is based on the concept of a *planar merging region* which contains all the feasible merging points $p$ such that the shortest planar path between child merging regions via $p$ is equal to the shortest planar path between child merging regions taking into consideration the given obstacles. Again, our experimental results are quite promising: even for the relatively dense obstacle layout studied, obstacle-avoidance clock routing seems achievable without undue penalty in clock tree cost. Finally, we extend the bounded-skew routing approach to address buffered clock trees, assuming (as is the case in present design methodologies) that the buffer hierarchy (i.e., the number of buffers at each level and the number of levels) is given. A bounded-skew buffered clock tree is constructed by performing three steps for each level of the buffer hierarchy, in bottom-up order: (i) cluster sinks or roots of subtrees for each buffer; (ii) build a bounded-skew tree using the ExG-DME algorithm under Elmore delay [5] for each cluster; and (iii) reduce the total wirelength by the H3 buffer sliding heuristic. Our experimental results show that H3 achieves very substantial wirelength improvements over the method used by [19, 22], for a range of buffer hierarchy types and skew bounds.

## Notes

1. One minor caveat is that the "merging region" of [3–5] is not a complete generalization of the DME merging segment: when detour wiring occurs or when sibling merging regions overlap, the merging region may not contain all the minimum-cost merging points.
2. We assume that there are only two routing layers. Our approach can easily extends to multiple routing layers.
3. However, when detouring occurs, both the H-layer and V-layer will be used for the detour wiring. It is easy to calculate the extra wirelength on both layers if we prescribe the routing pattern for detour wiring.
4. Strictly speaking, there can be joining segments with slopes other than $\pm 1$, 0, and $\infty$ although they are not encountered in practice. For the case of joining segments with slopes $m$ with $|m| > 1$ ($|m| < 1$), we expand obstacles as in Case III (IV).
5. The simplest approach is to divide $SDR(L_a, L_b)$ into a set of disjoint rectangles $R_i$ that contains no obstacles, as shown in Fig. 10(a). Let $c \in R_i$ and $d \in R_i$ be the corner points closest to joining segments $L_a$ and $L_b$. If prescribed routing patterns are assumed for the shortest paths from $c$ to $L_a$ and from $d$ to $L_b$, delays at $c$ and $d$ are well-defined. Since there are no obstacles inside $R_i$, the planar merging region can be constructed from points $c$ and $d$ for non-uniform layer parasitics using the methods of Section 2.
6. More accurate models for estimating the load capacitance of a cluster are of course possible, but have surprisingly little effect. Indeed, we implemented the best possible model (which is to actually execute the BST construction whenever a BST estimate is required) but this did not result in noticeable performance improvement.
7. We also investigated less greedy iterative methods that have the same general structure as the classic KL-FM partitioning heuristics. For example, an analog of the KL-FM pass might always expand the cluster with smallest estimated load capacitance by shifting the closest "unlocked" node in another cluster; as in KL-FM, a node that is moved becomes locked for the remainder of the pass to prevent cycling. In our experience, such more complicated heuristics do not achieve noticeably different results from the simple method we describe.

## References


1. A.B. Kahng and G. Robins, *On Optimal Interconnections for VLSI*, Kluwer Academic Publishers, 1995.
2. E.G. Friedman (Ed.), *Clock Distribution Networks in VLSI Circuits and Systems: A Selected Reprint Volume*, IEEE Press, 1995.
3. J.H. Huang, A.B. Kahng, and C.-W.A. Tsao, "On the bounded-skew clock and steiner routing problems," in *Proc. ACM/IEEE Design Automation Conf.*, pp. 508–513, 1995. Also available as Technical Report CSD-940026x, Computer Science Dept., UCLA.
4. J. Cong and C.-K. Koh, "Minimum-cost bounded-skew clock routing," in *Proc. IEEE Intl. Symp. Circuits and Systems*, Vol. 1, pp. 215–218, April 1995.
5. J. Cong, A.B. Kahng, C.-K. Koh, and C.-W.A. Tsao, "Bounded-skew clock and Steiner routing under Elmore delay," in *Proc. IEEE Intl. Conf. Computer-Aided Design*, pp. 66–71, Nov. 1995.
6. K.D. Boese and A.B. Kahng, "Zero-skew clock routing trees with minimum wirelength," in *Proc. IEEE Intl. Conf. on ASIC*, pp. 1.1.1–1.1.5, 1992.
7. 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. Circuits and Systems*, Vol. 39, No. 11, pp. 799–814, Nov. 1992.
8. T.-H. Chao, Y.-C. Hsu, and J.-M. Ho, "Zero skew clock net routing," in *Proc. ACM/IEEE Design Automation Conf.*, pp. 518–523, 1992.
9. M. Edahiro, "Minimum skew and minimum path length routing in VLSI layout design," *NEC Research and Development*, Vol. 32, No. 4, pp. 569–575, 1991.
10. J. Cong, A.B. Kahng, C.-K. Koh, and C.-W.A. Tsao, "Bounded-skew clock and Steiner routing under Elmore delay," Technical Report CSD950030, Computer Science Dept., University of California, Los Angeles, Aug. 1995. Available by anonymous ftp to ftp.cs.ucla.edu, also available at http://vlsicad.cs.ucla.edu/~tsao.
11. M. Edahiro, "A clustering-based optimization algorithm in zero-skew routings," in *Proc. ACM/IEEE Design Automation Conf.*, pp. 612–616, June 1993.
12. M. Borah, R.M. Owens, and M.J. Irwin, "An edge-based heuristic for rectilinear Steiner trees," *IEEE Trans. Computer-Aided Design*, Vol. 13, No. 12, pp. 1563–1568, Dec. 1994.
13. A.B. Kahng and C.-W.A. Tsao, "Low-cost single-layer clock trees with exact zero Elmore delay skew," in *Proc. IEEE Intl. Conf. Computer-Aided Design*, 1994.
14. T. Asano, L. Guibas, J. Hershberger, and H. Imai, "Visibility-polygon search and euclidean shortest paths," in *Proc.*

*IEEE Symp. Foundations of Computer Science*, pp. 155–164, 1995.
15. E. Welzl, "Constructing the visibility graph for $n$ line segments in $o(n^2)$ time," *Information Processing Letters*, Vol. 20, pp. 167–171, 1985.
16. A.B. Kahng and C.-W.A. Tsao, "Planar-dme: A single-layer zero-skew clock tree router," *IEEE Trans. Computer-Aided Design*, Vol. 15, No. 1, Jan. 1996.
17. C.-W.A. Tsao, "VLSI Clock Net Routing," Ph.D. thesis, University of California, Los Angeles, Oct. 1996.
18. J.G. Xi and W.W.-M. Dai, "Buffer insertion and sizing under process variations for low power clock distribution," in *Proc. ACM/IEEE Design Automation Conf.*, pp. 491–496, 1995.
19. S. Pullela, N. Menezes, J. Omar, and L.T. Pillage, "Skew and delay optimization for reliable buffered clock trees," in *Proc. IEEE Intl. Conf. Computer-Aided Design*, pp. 556–562, 1993.
20. J. Chung and C.-K. Cheng, "Skew sensitivity minimization of buffered clock tree," in *Proc. IEEE Intl. Conf. Computer-Aided Design*, pp. 280–283, 1994.
21. A. Vittal and M. Marek-Sadowska, "Power optimal buffered clock tree design," in *Proc. ACM/IEEE Design Automation Conf.*, San Francisco, June 1995.
22. Y.P. Chen and D.F. Wong, "An algorithm for zero-skew clock tree routing with buffer insertion," in *Proc. European Design and Test Conf.*, pp. 652–657, 1996.
23. C.J. Alpert and A.B. Kahng, "Geometric embeddings for faster (and better) multi-way partitioning," in *Proc. ACM/IEEE Design Automation Conf.*, pp. 743–748, 1993.
24. T.F. Gonzalez, "Clustering to minimize the maximum intercluster distance," *Theoretical Computer Science*, Vol. 38, Nos. 2–3, pp. 293–306, June 1985.
25. A.B. Kahng and C.-W. Albert Tsao, "Planar-dme: Improved planar zero-skew clock routing with minimum pathlength delay," in *Proc. European Design Automation Conf. with with EURO-VHDL*, Grenoble, France, pp. 440–445, Sept. 1994. Also available as Technical Report CSD-940006, Computer Science Dept., UCLA.

**Andrew B. Kahng** received the A.B. degree in applied mathematics and physics from Harvard College, and the M.S. and Ph.D. degrees in computer science from the University of California at San Diego. He joined the computer science department at UCLA in 1989, and has been an associate professor there since 1994. His honors include NSF Research Initiation and Young Investigator awards. He is General Chair of the 1997 ACM International Symposium on Physical Design, and a member of the working group that is defining the Design Tools and Test portion of the 1997 SIA National Technology Roadmap for Semiconductors. Dr. Kahng's research interests include VLSI physical layout design and performance verification, combinatorial and graph algorithms, and the theory of iterative global optimization. Currently, he is Visiting Scientist (on sabbatical leave from UCLA) at Cadence Design Systems, Inc.
abk@cs.ucla.edu

**Chung-Wen Albert Tsao** received the B.S. degree from National Taiwan University in 1984, and the M.S. degree from National Sun Yat-Sen University in 1988, both in electrical engineering. With assistance from a Fellowship from the Ministry of Education, Taiwan, he received the M.S. degree and Ph.D. in Computer Science from UCLA in 1993 and 1996, majoring in Theory with minors in Architecture/VLSI CAD and Network Modeling/Analysis. He is currently working at Cadence Design Systems, Inc., San Jose, California. His Ph.D. research focused on VLSI clock net routing. His current research interests include VLSI routing, partitioning and placement, computational geometry, and delay modeling.
tsao@cadence.com