More Practical Bounded-Skew Clock Routing [1]

Andrew B. Kahng and C.-W. Albert Tsao
UCLA Computer Science Dept., Los Angeles, CA 90095-1596
Cadence Design Systems, Inc., San Jose, CA 95134

Abstract: Academic clock routing research results has often had limited impact on industry practice, since 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 nonzero via parasitics; (ii) obstacle-avoidance clock routing; (iii) a new topology design rule for prescribed-delay clock routing; and (iv) predictive modeling of the clock routing itself. 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; see [13, 9] 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 nonzero skew bound, hence the actual design requirement is for a *bounded-skew routing tree* (BST).

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. 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 d_i denotes the signal delay from clock source s_0 to sink s_i, then the *skew* of clock tree T is given by $skew(T) = \max_{s_i,s_j \in S} |d_i - d_j|$. The BST problem is formally stated as follows.

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

[1] This work was supported by a grant from Cadence Design Systems. A. B. Kahng is currently Visiting Scientist (on sabbatical leave from UCLA), and C.-W. A. Tsao is Senior Member of Technical Staff, at Cadence.

The BST problem has been previously addressed in [12, 4, 3]. The basic *Extended DME* (Ex-DME) approach extends the DME algorithm [2, 5] via the concept of a *merging region*, which is a set of embedding points with feasible skew and minimum merging cost if no detour wiring occurs. 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. We now review necessary concepts from [4, 12, 3].

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 $L_b \in mr(b)$, 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 $mr(v)$. It is observed in [3] 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 skew values along l can be either a constant (when L_a and L_b are Manhattan arcs) or piecewise-linear decreasing, then constant, then piecewise-linear increasing along l. This important property enables the merging region $mr(v) \in SDR(L_a, L_b)$ to be constructed in $O(n)$ time [3]. The resulting merging region is 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.

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). When the topology is not prescribed, [12] propose the Extended Greedy-DME algorithm (ExG-DME), which combines merging region computation with topology generation, following the Greedy-DME approach of [6]. ExG-DME allows merging at non-root nodes, whereas Greedy-DME always merges two subtrees at their roots.

2 Non-Uniform Layer Parasitics

Consider the practical scenario where per-unit resistance and capacitance values differ between the V-layer (vertical routing layer) and H-layer (horizontal routing layer).[1] We first assume that vias have no resistance and capacitance, then extend our analysis to nonzero 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

[1] We assume that there are only two routing layers. Our approach easily extends to multiple routing layers.

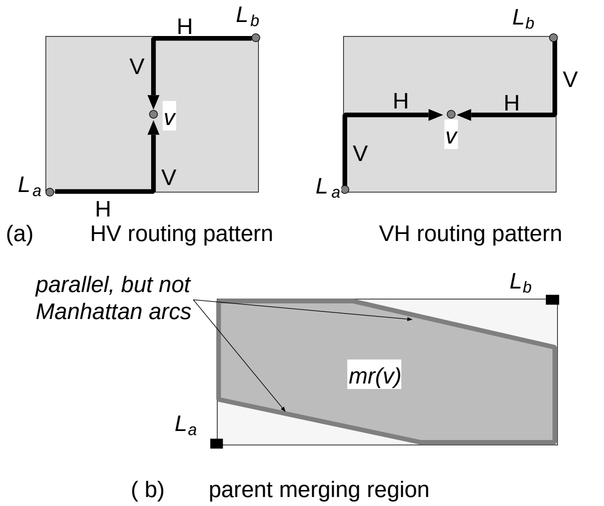

Figure 1: (a) Two simple routing patterns between two points: HV and VH for non-uniform layer parasitics. (b) Resulting merging regions according to the the HV routing pattern.

joining segments $L_a \subseteq mr(a)$ and $L_b \subseteq mr(b)$. When both L_a and L_b are rectilinear segments or are two single points sitting on a vertical or horizontal line, only one routing layer is needed for merging $mr(a)$ and $mr(b)$. Thus, the original BME construction rules [3] still apply in these cases.

Corollary 1 below shows that for non-uniform layer parasitics, joining segments will never be Manhattan arcs of nonzero length. Thus we need consider only the possible modification of BME construction rules for the case where the joining segments are two single points not on the same 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. Fig. 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. In [16], we prove the following theorems.

Theorem 1 *Let v be a node in the topology with children a and b. Assume that joining segments $L_a \subseteq mr(a)$ and $L_b \subseteq mr(b)$ are two single points. By using the HV (or VH) routing pattern for non-uniform layer parasitics, (i) any line segment $l \in SDR(L_a, L_b)$ is well-behaved, (ii) merging region $mr(v)$ has at most 6 sides with no boundary segments which are Manhattan arcs (of nonzero length).*

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.*

From the observation in Fig. 1 that vias are only located at the boundary of $SDR(L_a, L_b)$, we have

Theorem 2 *With nonzero via resistance and/or capacitance, Theorem 1 still holds except that there will be different delay/skew equations for points on boundary segments and interior segments of $SDR(L_a, L_b)$.*

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. Let c_1, r_1 and c_2, r_2 be the per-unit capacitance and per-unit resistance for the H-layer and V-layer, respectively. For the uniform layer parasitics, we set $c_1 = c_2 = 0.027fF$ and $r_1 = r_2 = 16.6m\Omega$. For the non-uniform layer parasitics, we set $c_2 = 2.0 \cdot c_1$ and $r_2 = 3.0 \cdot r_1$. For simplicity, we use only the HV routing pattern and ignore via resistance and capacitance.

We see that solutions under non-uniform layer parasitics average 2% more total wirelength than those under uniform layer parasitics. This may be due to merging regions under non-uniform layer parasitics being smaller (and thus having higher merging cost at the next higher level) since the joining segments cannot be Manhattan arcs of nonzero length. Note that when the skew bound is infinite, all joining segments are rectilinear, and thus the routing solutions under non-uniform and uniform layer parasitics have identical total wirelength. Separately, detailed experiments on benchmark r1 have compared the total wirelength of zero-skew routing for different ratios of r_2/r_1 and c_2/c_1. Even as $(r_2c_2)/(r_1c_1)$ 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, our new BME method has routing costs that are insensitive to changes in the ratio of H-layer/V-layer RC values.

	r1	r2	r3	r4	r5
Skew Bound	Wirelengths under Non-uniform layer parasitics Wirelengths under Uniform layer parasitics				
0 [6]	1253.2	2483.8	3193.8	6499.7	9723.7
0	1332.5 1320.7	2623.8 2603.6	*3359.1 3382.4	*6810.7 6877.5	*10108.7 10138.5
1ps	1283.5 1232.2	2531.8 2401.7	3207.0 3118.1	6461.5 6241.1	9610.8 9190.7
5ps	1182.1 1130.6	2333.3 2256.2	2988.6 2875.1	5979.8 5715.1	8753.9 8371.2
10ps	1158.6 1069.2	2248.3 2183.5	2810.7 2747.6	5719.0 5453.8	8482.4 8063.7
20ps	1071.5 1039.6	2183.4 2069.1	2709.8 2569.0	5474.6 5290.1	8018.2 7695.9
50ps	1058.6 1009.3	2028.9 1917.8	2557.0 2459.7	5195.8 5008.0	7562.9 7248.2
100ps	989.0 964.3	1929.0 1880.7	2463.9 2350.1	4940.1 4786.1	7193.1 6869.6
200ps	936.7 895.8	1886.7 1741.6	*2356.0 2359.5	4734.4 4540.1	6905.9 6650.0
500ps	919.4 820.4	1770.9 1754.6	2205.2 2187.4	4635.1 4564.2	6564.1 6449.3
1ns	830.0 819.1	*1664.2 1709.4	*2156.4 2175.8	*4500.5 4531.4	*6395.4 6453.4
10ns	775.9 775.9	*1569.4 1613.5	*2160.6 2212.4	*4072.1 4184.2	6168.5 5979.3
∞	775.9 775.9	1522.0 152.20	1925.2 1925.2	3838.2 3838.2	5625.2 5625.2
∞[1]	769.3	1498.8	1902.6	3781.4	5571.1

Table 1: Comparison of total wirelength of routing solutions under non-uniform and uniform layer parasitics. 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.

3 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.[2] We first present the analysis for uniform layer parasitics along with experimental results, then extend our method to non-uniform layer parasitics.

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 $L_b \subseteq mr(b)$. Obviously, points $p \in mr(v)$ covered by an obstacle are not feasible merging points. Also, 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 detouring from p and p' to L_a and L_b. Thus, we seek points $p \in mr(v)$ which have minimum sum of pathlengths to L_a and L_b. Define a path $s \rightsquigarrow t$ to be a sequence of line segments from point s to t, with pathlength denoted by $cost(P)$; a *planar path* is a path that does not cross any obstacles. For each node v, the *planar merging region* $pmr(v)$ is the set of feasible merging point p such that the cost of the shortest planar path $P = s \rightsquigarrow p \rightsquigarrow t$ is minimum, where $s \in L_a$ and $t \in L_b$. Note that $cost(P) \geq d(L_a, L_b)$, and $pmr(v) \subseteq mr(v)$ when $cost(P) = d(L_a, L_b)$. 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.

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

Case I. (expand as in Fig. 2(a))

1. $L_a = \{p_1\}, L_b = \{p_2\}$, and $\overline{p_1 p_2}$ has finite nonzero positive slope m, i.e., $0 < m < \infty$.

2. L_a or L_b is a nonzero-length Manhattan arc with slope -1.

In other words, an obstacle O intersecting the top (bottom) boundary of $SDR(L_a, L_b)$ is expanded horizontally toward the left (right) until it reaches the left (right) boundary of $SDR(L_a, L_b)$. If O intersects the left (right) boundary of $SDR(L_a, L_b)$, then O is expanded upward (downward) until it reaches the top (bottom) boundary of $SDR(L_a, L_b)$.

Case II. (symmetric to Case I)

1. $L_a = \{p_1\}, L_b = \{p_2\}$, and $\overline{p_1 p_2}$ has finite nonzero negative slope m, i.e., $-\infty < m < 0$.

2. L_a or L_b is a nonzero-length Manhattan arc with slope $+1$.

Case III. (expand as in Fig. 2(b)) Both joining segments are vertical segments, possibly of zero length.
In other words, an obstacle O intersecting with $SDR(L_a, L_b)$ is expanded along the horizontal direction until O reaches both joining segments.

Case IV. (symmetric to Case III) Both joining segments are horizontal segments, possibly of zero length.

In Cases I and II an expanded obstacle O can intersect with another obstacle, which must then also expand in the same direction via a sort of "chain reaction". With these obstacle expansion rules, we construct $pmr(v)$ from child regions $mr(a)$ and $mr(b)$ as follows.[3]

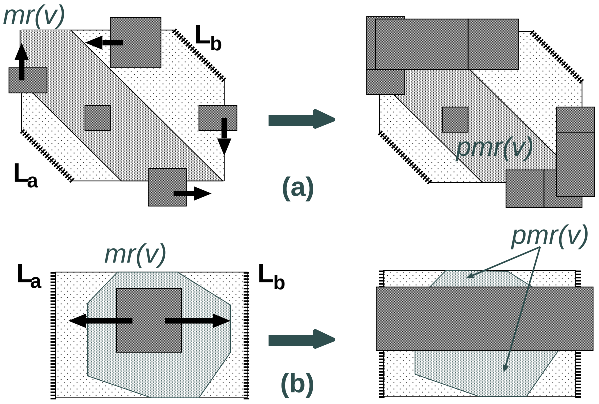

Figure 2: Illustration of Obstacle Expansion Rules.

1. Apply the obstacle expansion rules to expand obstacles, and calculate $pmr(v) = \{p | p \in mr(v) - \text{expanded obstacles}\}$.

2. If $pmr(v) \neq \emptyset$ then stop; otherwise, restore the sizes of all the expanded obstacles and continue with next step.

3. Compute the shortest planar path $P = s \rightsquigarrow t$, where $s \in mr(a)$ and $t \in mr(b)$.

4. Divide path P into a minimum number of subpaths $P_i = s_i \rightsquigarrow t_i$ such that $cost(P_i) = d(s_i, t_i)$.

5. Calculate delay and skew functions for each line segment in P.

6. For each subpath P_i which has a point p with feasible or minimum skew, use s_i and t_i (the endpoints of P_i) as the new joining segments. Then, calculate the planar merging region $pmr_i(v)$ from the new joining segments (actually, the points s_i and t_i) using Steps 1, 2 and 3. (Note that $pmr_i(v) \neq \emptyset$ since $p \in pmr_i(v)$.)

7. $pmr(v) = \bigcup pmr_i(v)$, where subpath $P_i \subseteq P$ contains a point p with feasible or minimum skew.

Notice that the purpose of Step 4 is to maximize the total area of $pmr(v)$. As shown in Fig. 3, if we divide subpath $P_2 = y \rightsquigarrow z \rightsquigarrow t$ into two smaller subpaths $y \rightsquigarrow z$ and $z \rightsquigarrow t$, region $pmr_2(v)$ in the Figure will shrink to be within the shortest distance region $SDR(y, z)$. As we can see from Fig. 3, $pmr(v)$ actually consists of several convex polygonal regions. So the number of regions per node may grow exponentially during the bottom-up construction of merging regions (this is the difficulty encountered by the IME method of [3]).[4] Our current implementation simply keeps at most k regions with lowest tree cost for each internal node.

Recall that 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 $L_v \in mr(v)$ is one of the joining segments used

[2] If some obstacle occupies only one routing layer, then the pre-routed wires over the obstacle become the obstacles for the later routing. In other words, the routing over the obstacle has to be planar. Indeed, our obstacle-avoidance routing was originally applied to improve planar clock routing [16].

[3] Strictly speaking, there can be joining segments with slopes other than ± 1, 0, and ∞ although they are not encountered in practice. For joining segments having slopes m with $|m| > 1$ ($|m| < 1$), we expand obstacles as in Case III (IV).

[4] Moreover, it may be better to construct and maintain planar merging regions along several shortest planar paths since the planar merging regions along the shortest planar path will not guarantee minimum tree cost at the next higher level, as stated in the Elmore-Planar-DME algorithm [14]

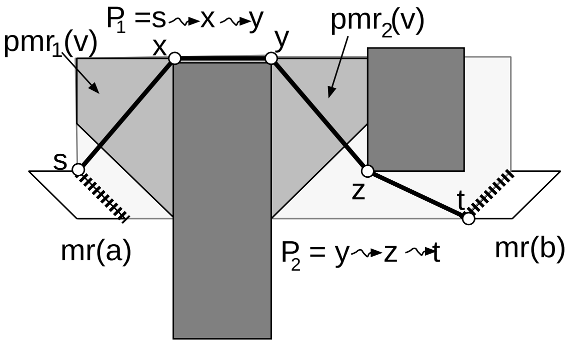

Figure 3: Construction of planar merging regions along a shortest planar path between child merging regions.

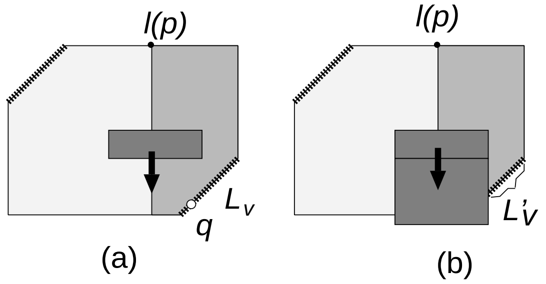

Figure 4: Modification of the embedding rule in the top-down phase of the Ex-DME algorithm when there are obstacles in the routing plane.

to construct $mr(p)$. However, when L_v is a Manhattan arc and there are obstacles intersecting $SDR(l(p), L_v)$, some of the embedding points $q \in L_v$ closest to $l(p)$ may become infeasible because the shortest path from q to $l(p)$ is blocked by some obstacle (Fig. 4(a)). These infeasible embedding points can be removed from L_v by applying the obstacle expansion rules with $l(p)$ and L_v being the joining segments (Fig. 4(b)). The remaining points of L_v left uncovered by the expanded obstacles are the feasible embedding locations for v.

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. 5. 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 $0.5pF$ loading capacitance and the per-unit wire resistance and wire capacitance are $16.6m\Omega$ and $0.027fF$. 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 [14] to find shortest planar s-t paths. The current implementation uses Dijkstra's algorithm in the visibility graph $G(V, E)$ (e.g., [10]) 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 $\Omega(n^2)$ worst-case running time, where n is the total number of vertices in the obstacle polygons, can be reduced to $O(n \log^2 n)$ using techniques in [11]. Table 2 shows that the wirelengths of routing solutions with obstacles are very close

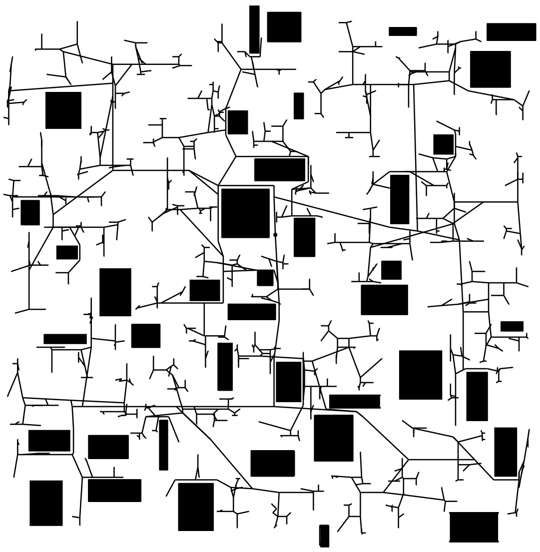

Figure 5: A zero-skew solution for the 555-sink test case with 40 obstacles.

to those of routing solutions without obstacles (typically within a few percent). The higher runtimes (reported for a Sun 85 MHz Sparc-5) in the presence of obstacles are due to the current naive implementation of obstacle detection and path-finding.

#sinks	50	100	150	555
Skew Bound	Wirelength: μm (normalized) CPU time: hr:min:sec (normalized)			
0	8791(1.06) 0:00:04(4)	11925 (1.04) 0:00:10(2)	14748 (1.03) 0:00:15(2)	28855 (1.01) 0:00:34(1)
1ps	8049 (1.09) 0:01:09(6)	10761 (1.04) 0:05:20(7)	13389 (1.03) 0:11:36(3)	26240 (1.04) 0:44:14(10)
2ps	7832 (1.07) 0:01:47(8)	10797 (1.01) 0:08:17(9)	12643 (1.02) 0:20:55(10)	25205 (1.04) 1:30:08(13)
5ps	7141 (1.04) 0:04:01(13)	10494 (1.08) 0:15:16(11)	11599 (1.01) 0:30:34(13)	23648 (1.04) 1:30:08(13)
10ps	7126 (1.06) 0:06:13(14)	9701 (1.03) 0:19:36(12)	11426 (1.07) 0:36:30(12)	22737 (1.05) 1:48:06(13)
20ps	6832 (1.13) 0:07:40(15)	9296 (1.03) 0:21:56(10)	11606 (1.10) 0:40:39(3)	21642 (1.05) 3:42:52(24)
50ps	6468 (1.12) 0:10:36(15)	8740 (1.09) 0:26:47(11)	10194 (1.10) 0:01:50(13)	22167 (1.15) 2:18:20(14)
100ps	6485 (1.20) 0:13:51(18)	8588 (1.11) 0:30:16(9)	9296 (1.02) 01:03:00(15)	19087 (1.01) 3:06:23(17)
1ns	6485 (1.24) 0:16:20(18)	8115 (1.13) 0:36:52(11)	9266 (1.10) 1:18:36(15)	17167 (.99) 7:24:38(12)
10ns	6485 (1.24) 0:16:19(18)	8115 (1.13) 0:36:43(11)	9266 (1.10) 1:20:07(15)	16698 (.99) 3:18:20(7)
∞	6485 (1.24) 0:16:43(18)	8115 (1.13) 0:36:52(11)	9266 (1.10) 1:20:25(13)	16698 (1.02) 3:21:11(7)

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. 5. Numbers in parentheses are ratios to corresponding (total wirelength, runtime) values when no obstacles are present in the layout.

3.2 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 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 not on the same vertical or horizontal line.

We construct planar merging regions for Cases I.1 and II.1 as follows. First, we divide $SDR(L_a, L_b)$ into a set of disjoint rect-

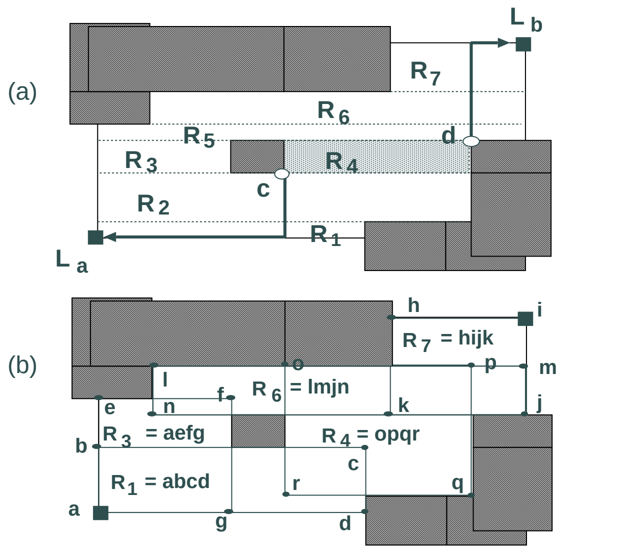

Figure 6: Obstacle-avoidance routing for non-uniform layer parasitics.

angles R_i that contains no obstacles, as shown in Fig. 6(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 planar 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. Since larger merging regions will result in smaller merging costs at the next higher level, we further 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. 6(b). After expansion, "redundant" rectangles contained in the expansions of other rectangles (e.g., rectangles R_2 and R_5 in Fig. 6 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 obstacles in $SDR(L_a, L_b)$, and then expand each rectangle R_i until blocked by obstacles.

2. Remove rectangles R_i that are completely contained by other rectangles.

3. For each remaining 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, and construct the merging region from points c and d as described in Section 2.

4 A New Prescribed-Delay Topology Rule

Prescribed-delay routing is motivated by hierarchical clock tree constructions used with clock gating, building-block design, and the general trend to lower fanouts in deep-submicron technologies. The prescribed-delay formulation is also useful in that it allows existing zero-skew routing algorithms to address the prescribed (local) skew problem, as follows. Let $skew(i, j) = d_i - d_j$ denote the prescribed local skew for sinks s_i and s_j. By rearranging skew constraint equations, we can express each sink delay relative to the delay of sink s_1, i.e., $d_i = d_1 + D_i$, with $D_1 = 0$. Let $D = \max_{i=1}^{n} D_i$. By adding a pseudo-delay element with delay $D - D_i$ to each sink s_i and performing zero-skew routing, the resulting clock tree will satisfy the prescribed skew constraints after we remove the pseudo-delay elements.[5]

Note that the useful skew problem addressed in [8, 17] is a more general problem of prescribed skew, in that the useful skew specifies a "range" of allowed skew values (rather than an "exact" skew value) for each pair of sinks. (For example, the range can be $[-\infty, \infty]$ if there is no skew constraint between a certain pair of sinks.) However, the prescribed skew formulation is still very useful for cases where the "negative skew" (or "signed skew") is desired [17].

	Orig		*Meta(Orig,New)*	
n	*Rank*	*Cost Subopt*	*Rank*	*Cost Subopt*
4	1/15/1.78	0/.64/.02	1/9/1.49	0/.31/.01
5	1/79/5.22	0/.65/.04	1/28/2.68	0/.21/.01
6	1/752/19.41	0/.65/.05	1/413/7.23	0/0.34/0.03

Table 3: Solutions with Original BST topology rule and meta-heuristic of (Original,New) topology rule, compared against optimal topology BST costs. Triples indicate (min,max,avg). There are 15 possible topologies for 4 sinks, 105 for 5 sinks, and 945 for 6 sinks.

With prescribed delays, the BST topology construction must take greater care with temporal (as opposed to spatial) compatibilities. While our goal to to minimize the total wirelength, ignoring the balance of subtree delays during the construction can lead to a great deal of detour wiring. Our studies of small instances show that our original BST merging rule, while generally quite effective, can yield topologies that have rank 752 (out of 945), with 65% cost suboptimality, even for instances as small as 6 sinks (see Table 3). We have studied a new topology construction rule that merges two subtrees so as to minimize $\alpha \times MC + (1 - \alpha) \times MD$, where MC and MD are respectively the total wirelength and maximum source-sink delay of the newly merged subtree. We study this new rule in the context of a meta-heuristic that takes the better of the tree costs according to the original and new merging rules, i.e., we attempt to address the difficult cases for the original rule, rather than find a completely new and general-purpose rule.[6] The parameter α depends on technology and units, since it captures a tradeoff between wirelength and delay. For our technology parameters, we found $\alpha = 0.67$ to be most effective; all our experimental data reflect this value.[7] Table 3 shows that our metaheuristic substantially improves both average-case and worst-case suboptimality for small instances; Table 4 shows that even for larger instances the metaheuristic can offer large improvements over the original BST-DME topology construction.

[5] For example, suppose we have $skew(s_1,s_2) = -10$ and $skew(s_2,s_3) = +50$ for a 3-sink clock net. Then we have $d_2 = d_1 + 10$ and $d_3 = d_2 - 50 = d_1 - 40$, with $D = 10$. The pseudo-delay elements with delays 10, 0, and 50 are added to sink s_1, s_2, and s_3, respectively.

[6] Edahiro [7] proposed similar greedy min-wirelength and min-delay based topology generation heuristics in the context of wire sizing. Here, we in some sense extend the application of his two heuristics for uniform wire width. We find that neither heuristic is overly strong by itself, and that the combination of both is superior.

[7] Our technology parameters are again taken from MCNC benchmarks; see Section 3.1. All experiments were for 1000 random instances with random sink locations in bounding box area = 500000 square units, skew bound B uniformly random in $[1, 20]$ *ps*, and prescribed delays uniformly random in the range $[0, max\_delay]$ where max_delay is itself random in $[1, 15]$ *ps*. Our ongoing work studies the subtle relationships that unify sink placement, skew assignment, clock tree topology construction, and bounded-skew embedding.

5 Predictive Modeling

Finally, our work has explored the issue of predictive modeling. The combined effects of deep-submicron physics and constraint-dominated designs have led to a recent trend of "constructive estimation" in place of analytic or empirical estimation. Nevertheless, efficient design optimization will always require estimators that are less expensive than actual constructions. A case in point is the clustering objective for hierarchical buffered clock tree synthesis: how can such an objective capture the actual performance of the bounded-skew clock routing algorithm that will be invoked after the buffer hierarchy has been determined?

We have recently implemented a generic model-building capability in our group, and have applied it to model prescribed-delay BST routing cost. Our package uses a slight enhancement of the Levenberg-Marquardt global optimization iteration from *Numerical Recipes in C* [15], and for the present experiment we use a simple "sum of powers" model, i.e., $cost(BST) = c_1 p_1^{e_1} + c_2 p_2^{e_2} + \ldots + c_k p_k^{e_k}$ for k parameters, along with a simple bottom-up approach for variable identification. Table 5 shows that very reasonable model accuracy can be easily obtained. Furthermore, using even three inexpensive parameters (center-of-gravity star cost, skew bound B, and maximum prescribed delay max_delay) to supplement the traditional MST cost can significantly improve model accuracy over using the MST cost alone. While the BST construction is actually quite efficient, Table 5 also shows that accurate models can be found for *optimal* BST costs (which can be obtained only with exponential runtimes).

Improvement of Meta(Orig,New) over Orig (%)			
# sinks	*max*	*avg*	*std*
8	26.327	1.378	3.496
12	25.790	1.234	3.021
16	27.928	0.812	2.659
20	34.390	0.559	2.473
24	36.814	0.413	2.462

Table 4: Improvement of Meta(Orig,New) over Orig topology construction, expressed as percentage tree cost reduction.

#	*f(MSTcost) Model*		*f(MSTcost++) Model*	
sinks	*AvgErr*	*MaxErr*	*AvgErr*	*MaxErr*
4	0.067	0.474	0.056	0.416
4opt	0.042	0.504	0.038	0.474
5	0.083	0.480	0.063	0.392
5opt	0.058	0.482	0.047	0.420
6	0.099	0.491	0.075	0.457
6opt	0.067	0.426	0.050	0.371
8	0.107	0.507	0.077	0.365
10	0.121	0.525	0.086	0.403
12	0.121	0.574	0.083	0.434
16	0.124	0.525	0.084	0.361

Table 5: Average and worst-case relative errors for fitted sum-of-powers prescribed-delay BST cost models, taken over 1000 trials. Default model is for original BST topology construction; *opt models are for optimal-cost topology construction. *f(MSTcost)* = model based on MST cost only. *f(MSTcost++)* = model based on MST cost, center-of-gravity star cost, skew bound B, and maximum sink delay.

6 Conclusions

We have extended the bounded-skew routing methodology to address a number of practical clock routing issues. Specifically, we have extended the BST-DME construction to handle non-uniform layer parasitics, nonzero via parasitics, and large obstacles on the clock distribution layers. We have also addressed hierarchical clock routing applications via a new prescribed-delay topology construction rule; our experiments indicate that this rule nicely complements the original ExG-DME topology rule for BST construction. The complementary nature of the two rules is particularly useful for small instances, since most clock subtrees are small in a buffered clock tree. Finally, we have proposed a predictive modeling methodology that can allow close integration of a given BST routing algorithm with a higher-level clock topology generation (sink and buffer clustering) tool. We are continuing to develop further practical clock routing extensions, while also pursuing integration within a commercial cell-based layout tool.

7 Acknowledgments

We are greatly indebted to Kenneth Yan for performing the experimental analyses of Sections 4 and 5.

REFERENCES

[1] M. Borah, R. M. Owens, and M. J. Irwin, "An edge-based heuristic for rectilinear steiner trees", *IEEE Trans. Computer-Aided Design*, 13(12):1563–1568, December 1994.

[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. Circuits and Systems*, 39(11):799–814, November 1992.

[3] J. Cong, A. B. Kahng, C.-K. Koh, and C.-W. A. Tsao, "Bounded-skew clock and steiner routing under elmore delay", *Proc. IEEE Intl. Conf. Computer-Aided Design*, pp. 66–71, November 1995. Also available as *UCLA Computer Science Department Technical Report 950030*, Aug. 1995 by anonymous ftp to ftp.cs.ucla.edu or to http://vlsicad.cs.ucla.edu/~tsao/htdocs/950030.ps.

[4] J. Cong and C.-K. Koh, "Minimum-cost bounded-skew clock routing", *Proc. IEEE Intl. Symp. Circuits and Systems*, volume 1, pp. 215–218, April 1995.

[5] M. Edahiro, "Minimum skew and minimum path length routing in vlsi layout design. *NEC Research and Development*, 32(4):569–575, 1991.

[6] M. Edahiro, "A clustering-based optimization algorithm in zero-skew routings", *Proc. ACM/IEEE Design Automation Conf.*, pp. 612–616, June 1993.

[7] M. Edahiro, "Delay Minimization for Zero-Skew Routing" *Proc. IEEE Intl. Conf. Computer-Aided Design*, pp, 563–566, November, 1993.

[8] J. P. Fishburn, "Clock Skew Optimization" *IEEE Trans. Computers*, 39 (7): 945-951, july, 1990.

[9] E. G. Friedman, editor. *Clock Distribution networks in VLSI Circuits and Systems: A Selected Reprint Volume. IEEE Press*, 1995.

[10] S. K. Ghosh and D. M. Mount, "An output-sensitive algorithm for computing visibility graphs", *SIAM J. on Comput.*, 20(5):888–910, 1991.

[11] J. Hershberger, S. Suri, "Efficient computation of Euclidean shortest paths in the plane", *Pro. 34th IEEE Symp. on Foundations of Computer Science.*, pp. 508–17, 1993.

[12] J. H. Huang, A. B. Kahng, and C.-W. A. Tsao, "On the bounded-skew clock and steiner routing problems", *Proc. ACM/IEEE Design Automation Conf.*, pp. 508–513, 1995.

[13] A. B. Kahng and G. Robins. *On Optimal Interconnections for VLSI*, Kluwer Academic Publishers, 1995.

[14] A. B. Kahng and C.-W. Albert Tsao, "Planar-DME: A single-layer zero-skew clock tree router. *IEEE Trans. Computer-Aided Design*, 15(1), January 1996.

[15] W. H. Press, S. A. Teukolsky, W. T. Vetterling, B. P. Flannery, *Numerical Recipes in C: the Art of Scientific Computing*, 2nd Ed., Cambridge: Cambridge University Press, 1992.

[16] C.-W. A. Tsao. "VLSI Clock Net Routing", *PhD thesis, University of California, Los Angeles*, October 1996.

[17] J. G. Xi and W. W.-M. Dai, "Useful-Skew Clock Routing with Gate Sizing for Low Power Design", *Proc. ACM/IEEE Design Automation Conf.*, pp. 383–388, 1996.