

Matching-Based Methods for High-Performance Clock Routing

Jason Cong, *Member, IEEE*, Andrew B. Kahng, *Associate Member, IEEE*, and Gabriel Robins, *Member, IEEE*

Abstract—Minimizing clock skew is important in the design of high performance VLSI systems. We present a general clock routing scheme that achieves very small clock skews while still using a reasonable amount of wirelength. Our routing solution is based on the construction of a binary tree using geometric matching. For cell-based designs, the total wirelength of our clock routing tree is on average within a constant factor of the wirelength in an optimal Steiner tree, and in the worst case is bounded by $O(\sqrt{l_1 l_2} \cdot \sqrt{n})$ for n terminals arbitrarily distributed in the $l_1 \times l_2$ grid. The bottom-up construction readily extends to general cell layouts, where it also achieves essentially zero clock skew within reasonably bounded total wirelength. We have tested our algorithms on numerous random examples and also on layouts of industrial benchmark circuits. The results are promising: our clock routing yields near-zero average clock skew while using total wirelength competitive with previously known methods.

I. INTRODUCTION

CIRCUIT speed is a major consideration in the design of high-performance VLSI systems. In a synchronous VLSI design, limitations on circuit speed are determined by two factors: the delay on the longest path through combinational logic, and the maximum clock skew among the synchronizing components. With advances in VLSI fabrication technology, the switching speed of combinational logic increases dramatically. Thus, the clock skew induced by non-symmetric clock distribution has become a more significant limitation on circuit performance.

Minimization of clock skew has been studied by a number of researchers in recent years. H-tree constructions have been used extensively for clock routing in regular systolic arrays [2], [11], [15], [34]. Although the H-tree structure can significantly reduce clock skew [11], [34], it is applicable primarily when all of the synchronizing

components are identical in size and are placed in a symmetric array. Ramananathan and Shin [23] proposed a clock distribution scheme for building block design where all blocks are organized in a hierarchical structure. They assume that all clock entry points are known at each level of the hierarchy and, moreover, that the number of blocks at each level is small since an exhaustive search algorithm is used to enumerate all possible routes. Fishburn [14] gave methods to maximize the margin of error in clocking constraints, and to minimize the clock period while avoiding clock hazards, or race conditions. This is accomplished via a linear programming formulation. However, the approach assumes that the entire clock tree topology is already known.

Jackson, Srinivasan, and Kuh [18] presented a clock routing scheme for circuits with many small cells. Their algorithm recursively partitions a circuit into two equal parts, and then connects the center of mass of the whole circuit to the centers of mass of the two sub-circuits. Although it was shown that the maximum difference in path length from the root to different synchronizing components is bounded by $O(\sqrt{l_1 l_2}/n)$ in the average case, small examples exist for which the wirelengths between clock source and clock pins can vary by as much as half the chip diameter.

In this paper, we first study the problem of high-performance clock routing for cell-based designs, i.e., circuits with many small cells, such as with standard-cell or sea-of-gates design styles. We then extend our method to general cell (also known as building-block) layouts, where the wiring is restricted to specific channels. In either of these scenarios, the H-tree approach cannot be used since synchronizing components may be of different sizes and may be in arbitrary locations in the layout. The method of [23] cannot be applied either, since there is no natural hierarchical structure associated with the design and the number of synchronizing components is typically too large to allow exhaustive examination of all possible routes. The algorithm of [18] is not completely satisfactory since large skews may result even for small examples, while the approach of [14] does not construct an actual clock routing topology. With this in mind, the goal of our present work is to develop a clock routing methodology which minimizes skew while incurring little added wiring expense.

We present a basic algorithm and several variants,

Manuscript received October 25, 1991; revised August 24, 1992. This work was supported by the National Science Foundation under Grant MIP-9110696 and Grant MIP-9110511, by a California MICRO grant from Cadence Design Systems, by the Army Research Office under Contract DAAK-70-92-K-0001, and Grant DAAL-03-92-G-0050, and by an IBM Graduate Fellowship. This paper was recommended by Associate Editor R. H. J. M. Otten. Preliminary versions of this work appeared in the *Proc. ACM/IEEE Design Automation Conference*, June 1991 and also in the *Proc. Int. ASIC Conference*, September 1991.

J. Cong and A. B. Kahng are with the Department of Computer Science, UCLA, Los Angeles, CA 90024-1596.

G. Robins was with the Department of Computer Science, UCLA, Los Angeles, CA. He is now with the Department of Computer Science, University of Virginia, Charlottesville, VA 22903.

IEEE Log Number 9206840.

which minimize skew by constructing a clock tree that is balanced with respect to root-leaf pathlengths in the tree (these notions will be formalized below). The approach is based on geometric matching: we start with a set of trees, each containing a single terminal of the clock signal net. At each level, we combine the trees into bigger trees using the edges of geometric matching. The end result is a binary tree whose leaves are the terminals in the clock signal net and whose root is the clock entry point. Our method is particularly suitable for designs which employ a single large buffer to drive the entire clock tree, rather than a buffer hierarchy. There are a number of reasons for such a design choice, as discussed in [2]. We note that the recently announced DEC Alpha processor uses such a single-buffer design style [12].

In the cell-based design regime, our algorithm guarantees perfect pathlength balanced trees for inputs of four or less pins. Extensive experimental results indicate that even for large clock signal nets, the maximum difference of pathlengths in the clock tree constructed by our algorithm remains essentially zero. This performance is obtained without undue sacrifice of wirelength: we prove that on average the total wirelength in our clock tree construction is within a constant factor of the wirelength in the optimal Steiner tree. Furthermore, our worst-case clock tree cost is bounded by $O(\sqrt{l_1 l_2} \cdot \sqrt{n})$ for n terminals in the $l_1 \times l_2$ grid,¹ which is the same bound as for the worst-case cost of the optimal Steiner tree.

Since the work in [18] addresses minimum-skew clock routing for cell-based designs, we implemented the algorithm of [18] for comparison purposes. For uniformly random sets of up to 1024 pins in the $l_1 \times l_2$ grid, our method produced clock routings with near-zero clock skew both in the average case and worst case, with total wirelength of the clock tree significantly lower than that produced by the method of [18]. In addition, our routing results for layouts of the MCNC Primary1 and Primary2 benchmarks are significantly better than those reported by [18]; we obtain perfectly balanced root-leaf pathlengths in the clock tree using several percent less total wire than the method of [18]. Actual clock skews for our benchmark routings, as determined by SPICE simulation, are reasonable.

We then apply our method to general cell design, by extending the notion of matching to arbitrary weighted graphs. In this scenario our algorithm produces a clock routing tree that is embedded in the channel intersection graph [10] of an arbitrary building-block layout. The clock routing trees produced by our method attain almost zero skew with only modest wirelength penalty. Experimental results show that the pathlength skew of our routing tree is less than 2% of the skew for a heuristic Steiner tree. This is achieved on average with less than 50% increase in wiring cost over the Steiner tree.

¹The $l_1 \times l_2$ grid consists of all lattice points (x, y) with x, y integers and $0 \leq x \leq l_1, 0 \leq y \leq l_2$.

The remainder of this paper is organized as follows. Section II defines a number of basic concepts and gives a precise formulation of our skew minimization problem. In Section III, we present the clock routing algorithm in detail for cell-based designs; Section IV extends the algorithm to general cell layouts. Experimental results of our algorithm and comparisons with previous methods are presented in Section V, and Section VI concludes with possible extensions of the method. Early versions of this paper were presented in [19] and [8].

II. PRELIMINARIES

A synchronous VLSI circuit consists of two types of elements, synchronizing elements (such as registers) and combinational logic gates (such as NAND gates and NOR gates). The synchronizing elements are connected to one or more system-wide clock signals. Every closed path in a synchronous circuit contains at least one synchronizing element (Fig. 1). The speed of a synchronous circuit is mainly determined by the clock periods. It is well known [1], [18] that the clock period C_P of each clock signal net satisfies the inequality:

$$C_P \geq t_d + t_{skew} + t_{su} + t_{ds}$$

where t_d is the delay on the longest path through combinational logic, t_{skew} is the clock skew, t_{su} is the set up time of the synchronizing elements (assuming that the synchronizing elements are edge triggered), and t_{ds} is the propagation delay within the synchronizing elements.

The term t_d itself can be further decomposed into $t_d = t_{d_interconnect} + t_{d_gates}$, where $t_{d_interconnect}$ is the delay associated with the interconnect of the longest path through combinational logic, and t_{d_gates} is the delay through the combinational logic gates on this path. As VLSI feature sizes become smaller, the terms t_{su} , t_{ds} , and t_{d_gates} all decrease significantly. Therefore, as noted above, $t_{d_interconnect}$ and t_{skew} become more dominant factors in determining circuit performance. It was noted in [1] that t_{skew} may account for 10% or more of the system cycle time. The objective of this paper is to minimize t_{skew} , while we have subsequently addressed the problem of minimizing $t_{d_interconnect}$ in a different work [9].

Given a routing solution for a clock signal net, the clock skew is defined to be the maximum difference among the delays from the *clock entry point* (CEP) to synchronizing elements in the net. The delay from the CEP to any synchronizing element depends on the wirelength from the CEP to the synchronizing element, the RC constants of wire segments in the routing, and the overall topology of the routing solution. Usually, the clock routing may be described as an RC tree [24], and we commonly use the first-order moment of the impulse response (also called Elmore delay) to approximate delay in an RC tree. The formulas derived by Rubinstein, Penfield and Horowitz [24] give both upper and lower bounds on delay in an RC tree.

However, although both the formula for Elmore delay

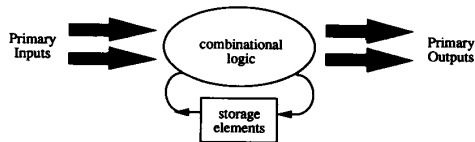


Fig. 1. A typical combinational circuit.

and those in [24] are very useful for simulation or timing verification, they involve sums of quadratic terms and are more difficult to compute and optimize during layout design. Thus, a simpler, linear *RC* model is often used (e.g., [23], [22]) so that wirelength between CEP and the synchronizing elements approximate the circuit delay. In this paper, we also use wirelength as a simple approximation of delay in a routing solution. The clock skew is hence defined to be the maximum difference in wirelength from the CEP to synchronizing elements in the clock signal net. We now give several definitions, along with a formal statement of the skew minimization problem.

A clock routing solution is represented by a rooted (Steiner) tree in the layout whose root is the CEP and whose leaves are synchronizing elements in the clock signal net. The *length*, or *cost*, of an edge in the tree is the Manhattan distance between the two endpoints of the edge, and the tree cost is the sum of all edge costs in the tree.

Definition: The *pathlength skew* of a tree is the maximum difference of the pathlengths in the tree from the root to any two leaves.

A tree is called a *perfect pathlength balanced tree* if its pathlength skew is zero. It is not difficult to construct a perfect pathlength balanced tree if we are allowed to use an arbitrary amount of wire. However, a routing tree with very high cost may distort the clock signal due to longer signal rise and fall times. Thus, we wish to construct a clock routing tree whose pathlength skew is as small as possible, without making the total tree cost too large. With this in mind, we formulate the clock routing problem as follows:

The Pathlength Balanced Tree (PBT) Problem: Given a set of n terminals, N , and real numbers B and S , find a clock routing tree connecting N such that the pathlength skew of the tree is bounded by S and the tree cost is bounded by B .

The following is immediately evident:

Theorem 1: the PBT problem is NP-hard.

Proof: Set $S = \infty$ so that the PBT problem simplifies to the minimum rectilinear Steiner tree problem, which is known to be NP-complete [17]. \square

Our objective is to give a heuristic algorithm for the PBT problem. For cell-based design methodologies, we wish to construct a clock tree with pathlength skew as small as possible, using wirelength as close as possible to that in an optimal Steiner tree. Specifically we would like to obtain a clock routing solution in the $l_1 \times l_2$ grid which

uses $O(\sqrt{l_1 l_2} \cdot \sqrt{n})$ total wirelength because an optimal Steiner tree will also use $O(\sqrt{l_1 l_2} \cdot \sqrt{n})$ wirelength in the average case [28].

III. A CLOCK ROUTING ALGORITHM FOR CELL-BASED DESIGN

For cell-based design, point-to-point interconnection cost is closely approximated by (Manhattan) geometric distance. Thus, in developing our clock routing algorithm for cell-based layouts, we introduce the notion of a geometric matching:

Definition: Given a set of $2k$ terminals, a *geometric matching* on this set consists of k line segments between terminals, with no two of the k segments sharing an endpoint.

Each line segment in the matching defines a *matching edge*. The cost of a geometric matching is the sum of the costs of its matching edges. A geometric matching on a set of terminals is optimal if its cost is minimum among all possible geometric matchings. An example of an optimal geometric matching over four terminals is shown in Fig. 2.

To construct a tree by iterative matching, we begin with a forest of n isolated terminals (for convenience, assume that n is a power of 2), each of which is considered to be a tree with CEP equal to the location of the terminal itself. The minimum-cost geometric matching on these n CEPs yields $n/2$ segments, each of which defines a subtree with two nodes. The optimal CEP into each subtree of two nodes is the midpoint of the corresponding segment, i.e., such that the clock signal will have zero skew between the segment endpoints.

In general, the matching operation will pair up the CEP's (roots) of all trees in the current forest. At each level, we choose the root of each new merged tree to be the *balance point* which minimizes pathlength skew to the leaves of its two subtrees (see Fig. 3). The balance point is the point p along the "straightline" connecting the roots of the two subtrees, such that the maximum difference in pathlengths from p to any two leaves in the combined tree is minimized. Computing the balance point requires constant time if we know the minimum and maximum root-leaf pathlengths in each of the two subtrees, and these values can be maintained incrementally using constant time per each node added to the clock tree.

Notice that at each level of the recursion, we only have to match half as many nodes as in the previous level. Thus, after $\lceil \log n \rceil$ matching iterations, we obtain the complete clock tree topology. In practice, we actually compute min-cost *maximum cardinality* matchings, i.e., if there are $2m + 1$ nodes, we find the optimal m -segment matching and match $m + 1$ CEPs at the next level. Fig. 4 describes our clock routing algorithm ALG1 for cell-based design.

The following two results show that ALG1 indeed uses a reasonable amount of wirelength. We prove that our

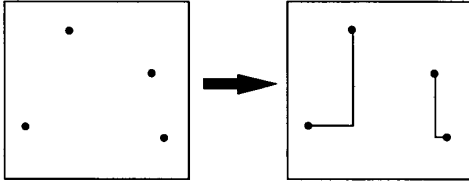


Fig. 2. An optimal geometric matching over four terminals.

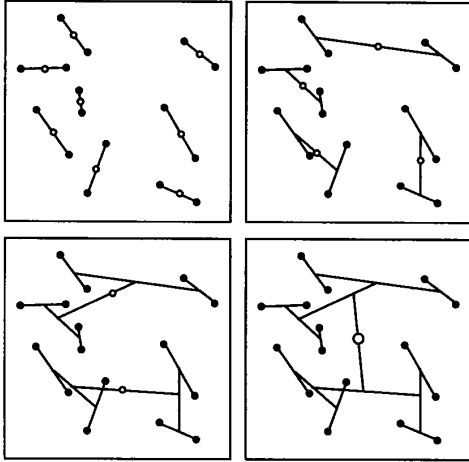


Fig. 3. ALG1 execution on a set of 16 terminals. Solid dots denote terminals, and hollow dots represent the balance points of the corresponding edges. At each level a geometric matching is computed on the balance points of the previous level. Note that although edges are depicted as straight lines, they are actually routed rectilinearly.

clock tree cost grows at the same asymptotic rate as the worst-case optimal Steiner tree cost over n terminals; we also show that our tree cost is on average within a constant factor of the optimal Steiner tree cost.

Theorem 2: For n terminals arbitrarily distributed in the $l_1 \times l_2$ grid, the maximum total wirelength of T_{ALG1} is $O(\sqrt{l_1 l_2} \cdot \sqrt{n})$.

Proof: For n terminals in the $l_1 \times l_2$ grid, the worst-case cost of an optimal matching is $O(\sqrt{l_1 l_2} \cdot \sqrt{n})$ [31]. Since the clock tree is formed by the edges of a matching on n terminals, plus the edges of a matching on $n/2$ terminals, etc., the total edgelenh in the tree is

$$O(\sqrt{l_1 l_2} \cdot \sqrt{n}) + O\left(\sqrt{l_1 l_2} \cdot \sqrt{\frac{n}{2}}\right) + \dots$$

$$= O(\sqrt{l_1 l_2} \cdot \sqrt{n}) \quad \square$$

This is of the same order as the maximum possible total edge length for the optimal Steiner tree on n terminals [28]. Note that Theorem 2 does not directly relate the cost of our clock routing construction to the cost of the optimal Steiner tree; this is partially addressed by the following.

ALG1: A Clock Routing Algorithm for Cell-Based Designs

Input: A set of terminals N

Output: A clock tree topology T_{ALG1} with root CEP

$T = \emptyset$

$P = N$

While $|P| > 1$

$M =$ the edges of the optimal geometric matching over P

$P' = \emptyset$

For $(p_1, p_2) \in M$ **Do**

$T_1 =$ the subtree of T rooted at p_1

$T_2 =$ the subtree of T rooted at p_2

$p =$ a point lying between p_1 and p_2 on the line containing p_1 and p_2 , such that p minimizes skew of the tree $T_1 \cup T_2 \cup \{(p, p_1), (p, p_2)\}$ rooted at p

$P' = P' \cup \{p\}$

$T = T \cup \{(p, p_1), (p, p_2)\}$

$P = P'$ plus a possible unmatched node if $|P|$ is odd

$CEP =$ root of $T =$ single remaining point in P

Output clock routing tree $= T_{ALG1} = T$

Fig. 4. The matching-based clock tree routing algorithm.

Theorem 3: For random sets of terminals chosen from a uniform distribution in the $l_1 \times l_2$ grid, the total edgelenh of the ALG1 clock tree will be on average within a constant factor of the total edgelenh of the optimal Steiner tree.

Proof: The minimum Steiner tree cost for n terminals randomly chosen from a uniform distribution in the $l_1 \times l_2$ Manhattan grid grows as $\beta \cdot \sqrt{l_1 l_2} \cdot \sqrt{n}$ for some constant β [28]. The claim follows from the $O(\sqrt{l_1 l_2} \cdot \sqrt{n})$ worst-case bound on the minimum-cost matching at any level of the construction [31]. \square

The balancing operation to determine the CEP of a merged tree is necessary because the root-leaf pathlength might vary between subtrees at a given stage of the construction. In general, when we merge subtrees T_1 and T_2 into a higher level subtree T , the optimal entry point of T will not be equidistant from the entry points of T_1 and T_2 (this can be seen in the example of Fig. 3). Intuitively, balancing entails "sliding" the CEP along the "bar of the H." However, it might not always be possible to obtain perfectly balanced pathlengths in this manner (see Fig. 5).

We therefore use a further optimization, which we call *H-flipping*: for each edge e added to the layout which matches CEPs on edges e_1 and e_2 , replace the "H" formed by the three edges e , e_1 , and e_2 by the "H" over the same four terminals which (i) minimizes pathlength skew, and (ii) to break ties, minimizes tree cost. We now prove that for four terminals it is always possible find an "H" orientation which achieves zero clock skew, and we also bound the increase in wirelength caused by H-flipping for nets of size four. As discussed below, extensive empirical tests confirm that even for very large inputs, the H-flipping refinement almost always yields perfectly path-balanced trees with essentially no increase in wirelength.

If a net is of size two, ALG1 selects the midpoint of the segment connecting the two terminals as the balance point, and this clearly yields a perfect pathlength balanced tree. Now we show that for nets of size four, ALG1 with

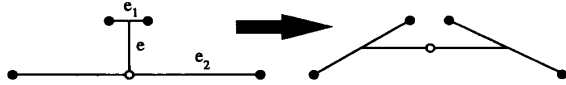


Fig. 5. Example of flipping an H to minimize clock skew: the tree on the left has no zero-skew balance point along the middle segment of the ‘H’, while the tree on the right does.

the H-flipping refinement also yields perfect pathlength balanced trees (a net of size three can be treated as a net of size four in which two terminals coincide).

Let $a, b, c,$ and d be the terminals in a net of size four. Without loss of generality, assume that ab and cd are the edges in an optimal matching and $ab \geq cd$. (for convenience, we use ab to denote both the segment ab and also its length. Let m_1 and m_2 be the midpoints of ab and cd , respectively. According to ALG1, m_1 is chosen to be the root of the subtree for a and b , and m_2 is chosen to be the root of the subtree for c and d . Then, the algorithm tries to choose the balance point p on segment m_1m_2 such that

$$\frac{ab}{2} + pm_1 = \frac{cd}{2} + pm_2. \quad (1)$$

It is easy to see that if $m_1m_2 \geq (ab - cd)/2$, we can always choose p satisfying (1). In this case, the pathlengths from p to all four terminals are the same, so that we have a perfect pathlength balanced tree. However, if $m_1m_2 < (ab - cd)/2$, we perform H-flipping and replace ab and cd by ad and bc . Then the midpoint n_1 on bc is the root of the subtree for b and c , and the midpoint n_2 on ad is the root of the subtree for a and d . We then seek p' on n_1n_2 such that

$$\frac{ad}{2} + p'n_1 = \frac{bc}{2} + p'n_2. \quad (2)$$

According to the following lemma, we are guaranteed to find p' on n_1n_2 satisfying (2).

Lemma 1: If $m_1m_2 < (ab - cd)/2$ then $n_1n_2 \geq (bc - ad)/2$.

Proof: If we have both $m_1m_2 < (ab - cd)/2$ and $n_1n_2 < (bc - ad)/2$ then (see Fig. 6):

$$m_1m_2 + n_1n_2 < \frac{ab - cd}{2} + \frac{bc - ad}{2},$$

therefore,

$$\frac{ab + bc}{2} > m_1m_2 + n_1n_2 + \frac{cd + ad}{2}. \quad (3)$$

Let x be the midpoint of bd . Using similar triangles and the triangle inequality, we obtain

$$\frac{ab}{2} = xn_2 \leq n_1n_2 + xn_1 = n_1n_2 + \frac{cd}{2}$$

and

$$\frac{bc}{2} = xm_2 \leq m_1m_2 + xm_1 = m_1m_2 + \frac{ad}{2}$$

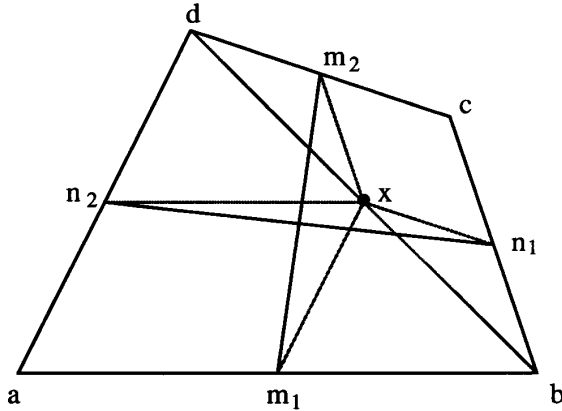


Fig. 6. Illustration for the proof of Lemma 1.

so that

$$\frac{ab + bc}{2} \leq m_1m_2 + n_1n_2 + \frac{cd + ad}{2},$$

contradicting (3). Therefore if $m_1m_2 < (ab - cd)/2$ we must have $n_1n_2 \geq (bc - ad)/2$. \square

Lemma 1 implies that we can always choose the balance point p' on n_1n_2 after H-flipping. Therefore, ALG1 always constructs a perfect pathlength balanced tree for a net of size four. The following lemma shows that when we replace ab and cd by ad and bc in the H-flip, the wirelength increase is bounded by a constant factor.

Lemma 2: If $m_1m_2 < (ab - cd)/2$ then $bc + ad \leq 3(ab + cd)$.

Proof: Let x be the midpoint of bd . Again applying similar triangles and the triangle inequality, we obtain (see Fig. 7):

$$\frac{bc}{2} = xm_2 \leq xd + dm_2 = xd + \frac{cd}{2}$$

and

$$\frac{ad}{2} = xm_1 \leq xb + bm_1 = xb + \frac{ab}{2}$$

so that

$$\frac{bc + ad}{2} \leq bd + \frac{ab + cd}{2}. \quad (4)$$

Let y be the intersection of bd and m_1m_2 . We then have

$$by \leq m_1y + m_1b = m_1y + \frac{ab}{2}$$

$$dy \leq m_2y + m_2d = m_2y + \frac{cd}{2}$$

$$bd \leq m_1m_2 + \frac{ab + cd}{2} < \frac{ab - cd}{2} + \frac{ab + cd}{2} = ab. \quad (5)$$

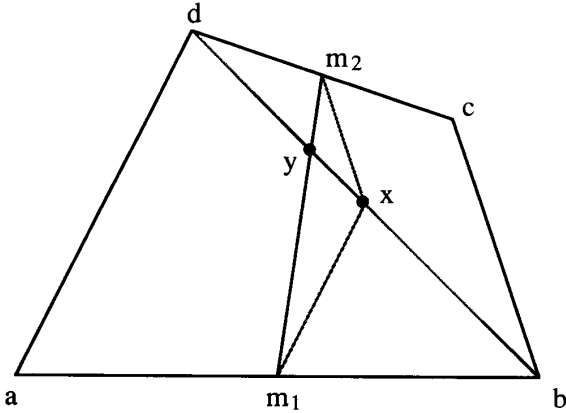


Fig. 7. Illustration for the proof of Lemma 2.

Thus, from (4) and (5) we have

$$\frac{bc + ad}{2} \leq ab + \frac{ab + cd}{2} \leq \frac{3(ab + cd)}{2}$$

or $bc + ad < 3(ab + cd)$. \square

Together, these lemmas imply:

Theorem 4: It is always possible to find an ‘‘H’’ orientation over four terminals which achieves zero clock skew, using at most a constant factor extra wirelength.

We now briefly discuss complexity issues and the requirement of an efficient implementation. Since our method is based on geometric matching, its time complexity depends on that of the matching subroutine. A well-known algorithm for general weighted matching requires time $O(n^3)$ [16], [21]. By taking advantage of the planar geometry, the algorithmic complexity can be reduced to $O(n^{2.5} \log n)$ [33]. However, even this may be excessive for large problem instances.

In order to solve problems of practical interest, and since there is no clear relationship between the optimality of the matching and the magnitude of the skew of the resulting clock tree, we may choose to speed up the implementation by using efficient geometric matching heuristics [3], [29], [30]. Although most of these methods were designed for the Euclidean plane, they also perform well in the Manhattan metric, especially if their output is further improved by uncrossing pairs of intersecting edges in the heuristic matching (in any metric, this reduces the matching cost due to the triangle inequality; to this end, note that k intersections of n line segments may be found efficiently in time $O(n \log n + k)$ [7]).

We shall later discuss empirical results from implementation of ALG1 based on three matching methods which require time $O(n)$, $O(n \log n)$ and $O(n \log^2 n)$, respectively. Each of these three matching heuristics yields very good clock routing solutions.

The basic approach of ALG1 thus consists of $\lceil \log n \rceil$ applications of the matching algorithm. H-flipping requires constant time per node, and therefore does not add to the asymptotic time complexity. If the underlying

matching algorithm runs within monotonically non-decreasing time $S(n) = \Omega(n)$, we may write $S(n) = n \cdot T(n)$ where $T(n) = S(n)/n$ is monotonically non-decreasing, and hence the total time required by ALG1 is

$$\begin{aligned} & S(n) + S\left(\frac{n}{2}\right) + S\left(\frac{n}{4}\right) + \dots \\ &= n \cdot T(n) + \frac{n}{2} \cdot T\left(\frac{n}{2}\right) + \frac{n}{4} \cdot T\left(\frac{n}{4}\right) + \dots \\ &\leq n \cdot T(n) + \frac{n}{2} \cdot T(n) + \frac{n}{4} \cdot T(n) + \dots \\ &= T(n) \cdot \left(n + \frac{n}{2} + \frac{n}{4} + \dots\right) \\ &\leq 2n \cdot T(n) = 2S(n) = O(S(n)) \end{aligned}$$

i.e., the time complexity of ALG1 is asymptotically equal to the time complexity of the underlying matching algorithm.

IV. A CLOCK ROUTING ALGORITHM FOR GENERAL CELL DESIGN

The same idea of bottom-up iterative matching which we developed in the preceding section may be easily generalized to clock routing in block layouts. In this section, we extend our method to such general cell designs, where a circuit is partitioned into a set of arbitrarily-sized rectangular cells (also referred to as blocks). Blocks may be of widely varying sizes, and are not necessarily placed in any regular arrangement. The routing is carried out in the channels between blocks, with routing over blocks prohibited. For this design style, the approximation of routing cost by geometric distance, which we used for cell-based design in the previous section, does not apply. The feasible routing regions are represented by the channel intersection graph (CIG) [10], which represents the available routing channels induced by a module layout. To capture the locations of clock pins within channels, we use the *augmented channel intersection graph* (ACIG), which is constructed as follows: for each pin incident to a routing channel, introduce a new node into the channel intersection graph which breaks the channel edge into two new edges (see the top left of Fig. 9).

Our goal is still to construct a clock signal tree with both skew and total wirelength as small as possible, except that routing of tree edges is now restricted to lie within prescribed routing channels. Given a graph G with positive edge costs, we let $\text{minpath}_G(x, y)$ denote the minimum cost path between nodes x and y , and use $\text{dist}_G(x, y)$ to denote the cost of $\text{minpath}_G(x, y)$. The notion of a matching may be extended to arbitrary weighted graphs as follows:

Definition: Given a graph $G = (V, E)$ with a positive cost function on the edges, a *generalized matching* M in G is a set of shortest paths connecting m mutually disjoint

node pairs, i.e., $M = \{\minpath_G(x_1, y_1), \minpath_G(x_2, y_2), \dots, \minpath_G(x_m, y_m)\}$, where the x_i 's and y_i 's are all distinct.

A generalized matching on a set of nodes $N \subseteq V$ in G is *complete* if $m = \lfloor |N|/2 \rfloor$. The *cost* of a generalized matching M is the sum of the costs of the shortest paths in the matching, i.e., $cost(M) = \sum_{i=1}^m dist_G(x_i, y_i)$. An *optimal complete generalized matching* on $N \subseteq V$ is one with least cost. We can show the following properties of optimal complete generalized matchings:

Lemma 3: Each edge of G belongs to at most one shortest path in an optimal complete generalized matching on $N \subseteq V$ in G .

Proof: Let M be an optimal complete generalized matching on N . Suppose that edge e appears in both $\minpath_G(x_i, y_i)$ and $\minpath_G(x_j, y_j)$, where (x_i, y_i) and (x_j, y_j) are in M and $i \neq j$ (see Fig. 8). Because (x_i, y_i) and $(x_j, y_j) \in M$ are shortest paths in G , we have

$$\begin{aligned} dist_G(x_i, x_j) + dist_G(y_i, y_j) \\ \leq dist_G(x_i, y_i) \\ + dist_G(x_j, y_j) - 2 \cdot cost(e). \end{aligned}$$

Therefore, replacing $\minpath_G(x_i, y_i)$ and $\minpath_G(x_j, y_j)$ by $\minpath_G(x_i, x_j)$ and $\minpath_G(y_i, y_j)$ would yield a complete generalized matching on N with smaller cost, a contradiction. \square

Henceforth, we will assume that there are b blocks in the design. G is the underlying augmented channel intersection graph and we assume that the n clock terminals are embedded on edges of G .

Lemma 4: The routing cost between any two clock terminals in G is bounded by $l_1 + l_2$.

Proof: Let x and y be two clock terminals in G . Let P_1 be any monotone (staircase) path passing through x and connecting two opposite corners w and w' of the layout grid. Clearly, $cost(P_1) = l_1 + l_2$. Similarly, let P_2 be a monotone path passing through y and connecting w and w' . Then, $cost(P_1) + cost(P_2) = 2 \cdot (l_1 + l_2)$. Since at least one of w or w' can be reached from *both* x and y with cost at most $l_1 + l_2$, the shortest path between x and y has cost no more than $l_1 + l_2$. \square

Proof: Let x and y be two clock terminals in G . Let P_1 be any monotone (staircase) path passing through x and connecting two opposite corners w and w' of the layout grid. Clearly, $cost(P_1) = l_1 + l_2$. Similarly, let P_2 be a monotone path passing through y and connecting w and w' . Then, $cost(P_1) + cost(P_2) = 2(l_1 + l_2)$. Since at least one of w or w' can be reached from *both* x and y with cost at most $l_1 + l_2$, the shortest path between x and y has cost no more than $l_1 + l_2$. \square

It is clear from Lemma 4 that an optimal complete generalized matching on the clock terminals in G has cost no more than $(l_1 + l_2) \cdot \lfloor n/2 \rfloor$.

As in the previous section, our basic strategy is to con-

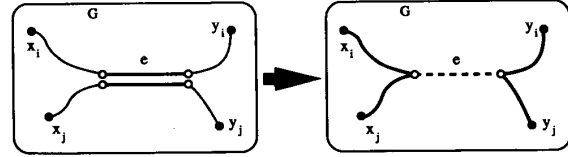


Fig. 8. Each edge belongs to at most one shortest path in an optimal complete generalized matching.

struct a clock tree by computing a sequence of generalized matchings on the clock terminals. We begin with a forest of n isolated clock terminals in G (again for convenience, we assume that n is a power of 2), each of which is a degenerate tree with CEP being the terminal itself. The optimal complete generalized matching on these n terminals yields $n/2$ paths, each of which defines a subtree. The optimal CEP into each subtree is the midpoint of the corresponding path, so that the clock signal will have zero skew between the two terminals. At each level, we compute an optimal generalized matching on the set of CEPs (roots) of all subtrees in the current forest and merge each pair of subtrees into a larger subtree. As before, the root (CEP) of the resulting tree is chosen to be the *balance point* on the path connecting the two subtrees such that the pathlength skew in the resulting tree is minimized (see Fig. 9).

Notice that at each level of the recursion, we only have to match half as many nodes as at the previous level. Thus, in $\lceil \log n \rceil$ matching iterations, we obtain a complete clock tree topology. If n is not a power of 2, then as noted in the discussion of ALG1, there will be an odd number $2m + 1$ of nodes to match at some level. For such cases, we compute an optimal maximum-cardinality generalized matching on $2m$ nodes, and then match $m + 1$ nodes at the next level. Fig. 10 gives a formal description of our clock routing algorithm ALG2 for general cell design.

The worst-case clock tree cost produced by the algorithm can be bounded as follows:

Theorem 5: Given b blocks in the $l_1 \times l_2$ grid and n terminals of a clock signal net, the cost of the clock tree created by ALG2 is at most $(l_1 + l_2) \cdot n$.

Proof: By Lemma 4, the cost of a generalized matching on n terminals is bounded by $(l_1 + l_2) \cdot \lfloor n/2 \rfloor$. After each iteration, the number of nodes to be matched is reduced by half. Therefore, the total clock tree cost is bounded by

$$\begin{aligned} (l_1 + l_2) \cdot \frac{n}{2} + (l_1 + l_2) \cdot \frac{n}{4} + \dots \\ \leq (l_1 + l_2) \cdot n. \end{aligned}$$

\square

In order to compute an optimal generalized matching on a set of nodes N in G , we construct a weighted complete graph G' on N such that $weight(x, y) = dist_G(x, y)$ for each pair of nodes x and y in N . This can be accomplished by applying an $O(|E| \cdot |V| + |V|^2)$ implemen-

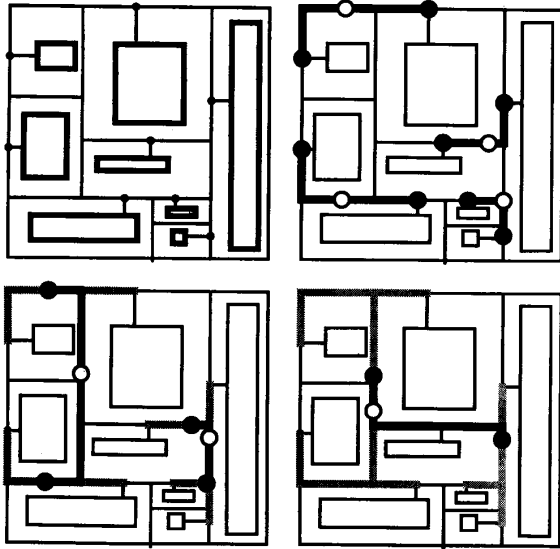


Fig. 9. ALG2 execution on a random module placement with an 8-terminal net. Solid dots are roots of subtrees in the previous level; hollow dots are roots (CEPs) of new subtrees computed at the current level. At each level an optimal generalized matching is computed on the solid points. For clarity, only the newly added wires are highlighted at each level.

ALG2: A Clock Routing Algorithm for General Cell Designs
Input: A set of terminals N embedded in a CIG G
Output: A clock tree topology T_{ALG2} with root CEP
$T = \emptyset$
$P = N$
While $ P > 1$
$M = \text{opt complete generalized matching on } P$
$P' = \emptyset$
For $\{p_1, p_2\} \in M$ Do
$T_1 = \text{subtree of } T \text{ rooted at } p_1$
$T_2 = \text{subtree of } T \text{ rooted at } p_2$
$p = \text{balance point on } \min\text{path}_G(p_1, p_2) \text{ minimizing the skew of the tree } T_1 \cup T_2 \cup \min\text{path}_G(p_1, p_2)$
$P' = P' \cup \{p\}$
$T = T \cup \{\{p, p_1\}, \{p, p_2\}\}$
$P = P'$ plus an unmatched node if $ P $ is odd
$CEP = \text{Root of } T = \text{single remaining point in } P$
Output clock routing tree $= T_{ALG2} = T$

Fig. 10. The matching-based clock tree algorithm for general cell design.

tation of Floyd's all-pairs shortest path algorithm [25] to the graph $G = (V, E)$. Note that G is a planar graph and therefore $|E| = O(|V|)$. Since for the augmented channel intersection graph we have $|V| = O(b + n)$, and typically $b > n$, the overall time complexity for this step is $O(b^2)$. We may then apply an $O(n^3)$ algorithm for computing an optimal complete matching in general graphs [21]. However, this complexity will result in long runtimes for large problem instances. Therefore, in order to achieve an efficient implementation, we use the greedy matching heuristic [26]. Such a heuristic matching may be improved by removing overlapping edges of shortest paths, as described in the proof of Lemma 3, so that no edge is used in more than one shortest path. The time complexity of each iteration of ALG2 is dominated by the $O(b^2)$ all-pairs shortest paths computation, which we invoke $\lceil \log n \rceil$

times, so that the overall time complexity of ALG2 is $O(b^2 \cdot \log n)$. This complexity is reasonable since the number of blocks is typically not large.

V. EXPERIMENTAL RESULTS

Both ALG1 and ALG2 were implemented in ANSI C for the Sun-4, Macintosh, and IBM 3090 environments. This section summarizes the simulation results.

5.1. Empirical Data for Cell-Based Designs

We have implemented three basic heuristic variants of ALG1, corresponding to different matching subroutines. The first heuristic variant (SP) uses the linear-time space partitioning heuristic of [30] to compute an approximate matching; the second variant (GR) uses an $O(n \log^2 n)$ greedy matching heuristic [29]; and the third variant (SFC) uses an $O(n \log n)$ spacefilling curve-based method [3]. We have further tested these three variants by running each both with and without two refinements: (1) removing all edge crossings in the heuristic matching, and (2) performing "H-flipping" as necessary. Either of these optimizations can be independently added to any of the three variants, yielding a total of twelve distinct versions of the basic algorithm. The variants of the algorithm are denoted and summarized as follows:

- **SP:** Use the space-partitioning matching heuristic of [30], which induces the matching through recursive bisection of the region (rather than bisection of the set of terminal locations).
- **GR:** Use a greedy matching heuristic, which always adds the shortest edge between unmatched terminals [29].
- **SFC:** Use a space-filling curve to map the plane to a circle, then choose the better of the two embedded matchings (i.e., either all odd edges or all even edges in the induced Hamiltonian cycle through the terminal locations) [3].
- **SP+E, GR+E, SFC+E:** Same as SP, GR, and SFC, respectively, except that the heuristic matching cost is further improved by edge-uncrossing.
- **SP+H, GR+H, SFC+H:** Same as SP, GR, and SFC, respectively, except that pathlength skew is further reduced by H-flipping.
- **SP+E+H, GR+E+H, SFC+E+H:** Same as SP, GR, and SFC, respectively, except that both edge-uncrossing and H-flipping are performed.

For comparison, we also implemented

- **MMM:** The method of means and medians, similar to that of Jackson, Srinivasan and Kuh [18].

The algorithms were tested on random sets of up to 1024 terminals generated from a uniform distribution in the 1000×1000 grid (i.e., $l_1 = l_2 = 1000$). Results for a sample run with 50 random terminal sets at each cardinality are summarized here: Table I compares the average tree costs and Table II compares the average clock skews

TABLE I
AVERAGE TREE COSTS, IN GRID UNITS, FOR THE VARIOUS HEURISTICS

Pts	MMM	SP	GR	SFC	SP+E	GR+E	SFC+E
4	1197	1155	1136	1140	1129	1129	1130
8	2136	2075	2032	2031	1990	1990	1992
16	3506	3582	3409	3527	3343	3326	3343
32	5598	5922	5481	5788	5342	5277	5326
64	8377	9184	8526	9048	8100	8032	8068
128	12276	13793	12632	13656	11912	11725	11976
256	17874	20765	18625	20354	17573	17024	17768
512	25093	30443	27055	29618	25341	24548	25720
1024	36765	44304	38688	42750	36444	35086	37056

Pts	SP+H	GR+H	SFC+H	SP+E+H	GR+E+H	SFC+E+H	Meta
4	1125	1125	1125	1125	1125	1125	1125
8	2027	2028	1994	1971	1979	1980	1960
16	3502	3416	3428	3333	3322	3329	3268
32	5860	5628	5577	5329	5273	5304	5151
64	9226	8794	8748	8076	7982	8047	7844
128	13997	3315	13159	11871	11697	11914	11566
256	21307	19611	19713	17457	16955	17629	16919
512	31646	29175	28688	25188	24465	25483	24480
1024	46417	42110	41540	36276	34965	36814	34992

TABLE II
AVERAGE SKEW VALUES, IN GRID UNITS, FOR THE VARIOUS HEURISTICS

Pts	MMM	SP	GR	SFC	SP+E	GR+E	SFC+E
4	112.31	3.98	15.52	0.00	0.00	0.00	0.00
8	186.10	45.79	76.71	4.26	0.66	0.66	0.66
16	234.72	70.93	141.22	19.47	4.01	3.54	3.66
32	262.61	143.85	200.33	28.29	8.14	7.85	6.14
64	229.15	179.83	273.04	51.36	6.93	8.65	5.29
128	201.55	226.61	314.05	64.86	11.52	14.18	11.26
256	183.28	286.90	324.57	85.10	17.25	13.85	15.04
512	153.90	321.23	399.29	85.46	14.79	15.26	15.73
1024	125.34	339.34	402.59	89.75	17.14	16.71	15.35

Pts	SP+H	GR+H	SFC+H	SP+E+H	GR+E+H	SFC+E+H	Meta
4	0.00	0.00	0.00	0.00	0.00	0.00	0.00
8	3.38	0.12	0.00	0.00	0.00	0.00	0.00
16	1.80	3.80	0.12	0.00	0.00	0.00	0.00
32	3.53	8.64	0.00	0.00	0.00	0.00	0.00
64	13.17	27.69	1.26	0.00	0.00	0.00	0.00
128	20.79	40.34	3.18	0.00	1.02	0.24	0.00
256	41.79	51.87	7.49	0.00	0.92	0.00	0.00
512	76.35	90.66	13.51	0.39	0.62	0.39	0.00
1024	75.92	94.99	16.62	0.44	0.08	0.38	0.00

for all heuristics. The data in the tables is given in grid units.

The computational results indicate that both optimizations (edge-uncrossing and H-flipping) significantly improve both skew and total wirelength. When the refinements are combined, average pathlength skew essentially vanishes, and the wirelength of several variants is superior to the output of MMM. The best variant appears to be GR+E+H, which is based on the greedy matching heuristic together with edge-uncrossing and H-flipping. Note that the cost of the greedy matching is asymptotically as good as that of the optimal matching [26]. Tables III and IV highlight the contrast between GR+E+H and MMM, showing minimum, maximum and average values

TABLE III
MINIMUM, AVERAGE, AND MAXIMUM SKEW VALUES, IN GRID UNITS, FOR GR+E+H AND MMM

Pts	MMM			GR+E+H		
	min	ave	max	min	ave	max
4	2	112.31	379	0	0.00	0
8	46	186.10	407	0	0.00	0
16	86	234.72	416	0	0.00	0
32	118	262.61	540	0	0.00	0
64	141	229.15	337	0	0.00	0
128	120	201.55	282	0	1.02	30
256	127	183.28	250	0	0.92	46
512	103	153.90	203	0	0.62	31
1024	94	125.34	167	0	0.08	4

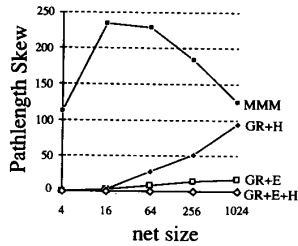


Fig. 11. Overall pathlength skew comparisons between ALG1 (GR + E + H) and MMM.

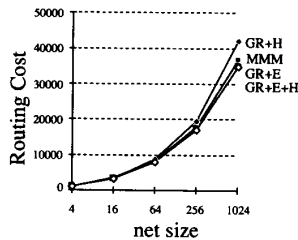


Fig. 12. Overall tree cost comparisons between ALG1 (GR + E + H) and MMM.

TABLE IV
MINIMUM, AVERAGE, AND MAXIMUM TOTAL WIRELENGTH VALUES, IN GRID UNITS, FOR GR + E + H AND MMM

Pts	MMM			GR + E + H		
	min	ave	max	min	ave	max
4	656	1197	1823	555	1125	1668
8	1089	2136	2943	1123	1979	2810
16	2841	3506	4221	2793	3322	3993
32	4813	5598	6216	4695	5273	5866
64	7624	8377	9266	7372	7982	8556
128	11439	12276	13136	11052	11697	12243
256	17220	17874	18549	16379	16955	17543
512	25093	25666	26291	23866	24465	25325
1024	36126	36765	37561	34231	34965	36179

for both total wirelength and skew. Figs. 11 and 12 depict these same comparisons graphically.

As noted in [20], any set of approximation heuristics induces a *meta-heuristic* which returns the best solution found by any heuristic in the set; we also implemented this (denoted as "Meta"), which returns the minimum-skew result from all of the other variants. Interestingly, in our experience Meta *always* returns a perfect pathlength balanced tree, i.e., for each problem instance, at least one of the other heuristic variants will yield a zero clock skew solution. This is very useful, especially when the heuristics are of similar complexity. For example, we can solve the Primary1 benchmark using all twelve methods in under two minutes on a Sun-4/60 workstation.

Fig. 13 and 14 illustrate the output of variant GR + E + H on the Primary1 and Primary2 benchmarks, using the same placement solutions as in [18]; note that although edges are depicted as straight lines in these diagrams, they are actually routed rectilinearly. Table V compares the results of GR + E + H and the results of [18] which were provided by the authors [27]: GR + E + H

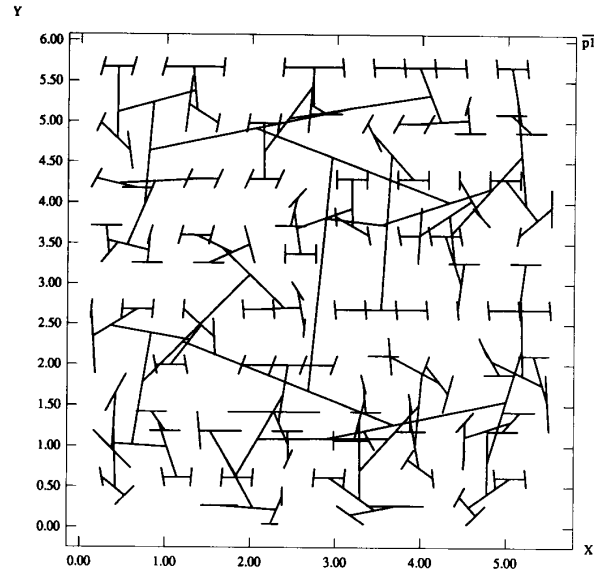


Fig. 13. Output of variant GR + E + H on the Primary1 benchmark layout.

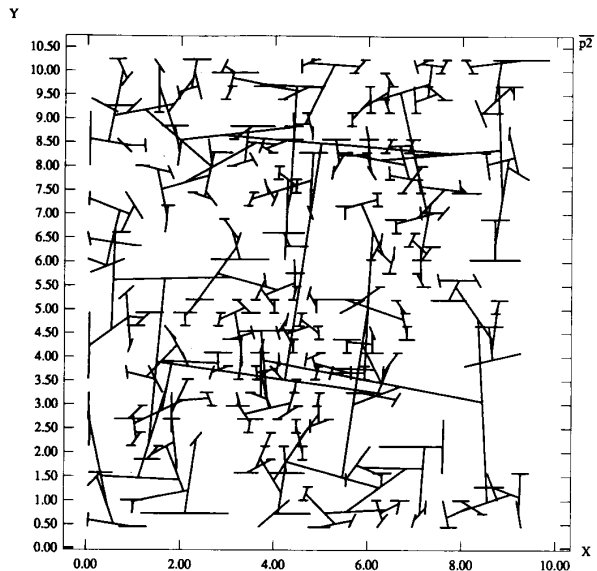


Fig. 14. Output of GR + E + H on the Primary2 benchmark layout.

completely eliminates pathlength skew while using 5%–7% less wirelength. To confirm the correlation between the linear delay model and actual delay, we ran SPICE simulations on the Primary1 and Primary2 clock trees using MOSIS 2.0- μm CMOS technology parameters and 0.3-pF sink loading capacitance; the simulated skews of our clock trees for Primary1 and Primary2 were 181 ps and 741 ps, respectively². Notice that this clock skew was

²Vias and parasitic difference between metal layers were not considered in our simulation because detailed layer assignment has not been determined at this stage of clock routing.

TABLE V
COMPARISONS OF GR + E + H AND MMM ON PRIMARY1 AND PRIMARY2. "SKEW (STD)" DENOTES THE STANDARD DEVIATION OF THE PATH LENGTH, AND "COST" DENOTES THE TOTAL WIRELENGTH

	Skew (STD) MMM	Cost MMM	Skew (STD) GR + E + H	Cost GR + E + H	Reduction Skew (STD)	Reduction Cost (%)
Primary1	0.29	161.7	0.00	153.9	0.29	4.8
Primary2	0.74	406.3	0.00	376.7	0.74	7.3

TABLE VI
AVERAGE CLOCK TREE COSTS AND PATHLENGTH SKEWS, IN GRID UNITS, OF ALG2 AND THE STEINER TREE HEURISTIC, RESPECTIVELY.

# of Modules	Net Size	Skew		Cost	
		Steiner	ALG2	Steiner	ALG2
16	4	511.0	0.8	1537	1921
16	8	794.9	12.9	2328	3478
16	16	1101.5	22.1	3332	5873
32	4	445.0	0.4	1401	1729
32	8	804.4	4.4	2261	3407
32	16	1136.9	12.0	3357	5847

TABLE VII
AVERAGE CLOCK TREE COSTS AND PATHLENGTH SKEWS, OF ALG2 OUTPUT, NORMALIZED (PER INSTANCE) TO CORRESPONDING HEURISTIC STEINER TREE VALUES.

# of Modules	Net Size	Pathlength Skew	Tree Cost	Edge Density in Channels
16	4	0.00	1.26	1.24
16	8	0.02	1.49	1.40
16	16	0.02	1.77	1.63
32	4	0.01	1.24	1.21
32	8	0.01	1.52	1.36
32	16	0.01	1.74	1.48

obtained simply by balancing CEP-leaf pathlengths; as discussed in Section VI, more sophisticated delay models can yield a better choice of balance points in the matching-based construction.

5.2. Empirical Data for General Cell Designs

We have tested ALG2 on two sets of test cases. One set of examples contains clock nets of sizes 4, 8, and 16 on 16 blocks, and the other set contains clock nets of sizes 4, 8, and 16 on 32 blocks. Block sizes and layouts were assigned randomly in the grid by creating a fixed number of non-overlapping blocks, with length, width, and lower-left coordinates all chosen from uniform distributions on the interval [0, 1000] (i.e., $l_1 = l_2 = 1000$).

For each net size (and block number), 100 instances were generated randomly, and we compared the skew and cost of the ALG2 routing trees with those produced by the 1-Steiner heuristic [20]. Results are shown in Tables VI and VII. The skew of our clock tree is very close to zero. In no case is it more than 2% of the skew of the Steiner tree routing. The increase in total wirelength of our routing tree varies from 24% to 77% when compared with the Steiner tree. The data in the tables is given in grid units.

As with the cell-based layout benchmarks, we ran SPICE simulations on a number of examples (again using MOSIS 2.0- μ m CMOS technology and 0.3-pF gate loading capacitance). The actual skew of our clock tree is consistently much smaller than that of a Steiner tree. For a typical 16-pin clock net in a 16-block design, the skews of our clock tree and the Steiner tree are 18 and 69 ps, respectively.

For the routing tree produced by ALG2, we may have overlapping edges in a channel because matching paths at different levels may use the same channel. However, by Lemma 3, no channel segment will appear in more than

a single path in a matching. Therefore, there are at most $\lceil \log n \rceil$ overlapping edges in each channel. The last column in Table VII shows the average edge density in channels, computed as the average of non-zero local column densities over all columns in all channels.

VI. REMARKS AND EXTENSIONS

We recommend that the global clock routing of ALG1 or ALG2 be performed before other wiring, following standard practice. In this way, there are no wire-crossing conflicts since two layers of metal are used, one for horizontal wires, and the other for vertical wires. The exact routing of the clock tree topology may be determined in the detailed routing step.

For cell-based design, we can realize additional wire-length savings in our clock tree routing by varying the geometric embedding of individual wires in the layout. In the Manhattan metric, the "balance point" of a wire connecting two terminals is not unique but is rather a locus of many possible terminals (Fig. 15), with the extremes corresponding to the two L-shaped wire orientations. Our current implementation sets the balance point of a segment to be its "Euclidean" midpoint, but this is not necessarily an optimal choice. Using a graph-theoretic formulation, we can easily derive a polynomial-time method, based on general graph matching, for finding the optimal set of balance points within these loci.

The wire embedding at each level of our algorithm may also benefit from *lookahead* of one or more levels, i.e., when we reach a situation where pathlength skew cannot be eliminated even via the utilization of an H-flip, we can go back one or two levels in the subtrees involved and try different H-flips during previous iterations on those subtrees. In our experience, this strategy easily allows complete elimination of clock skew at the current level, and

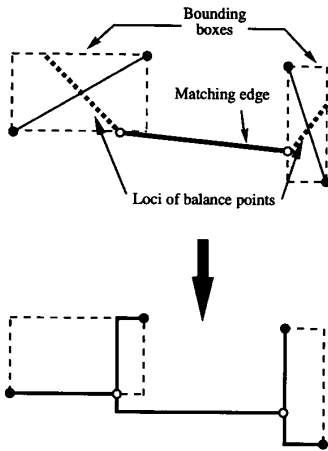


Fig. 15. Further optimizations are possible by matching over the loci of balance point candidates.

requires only a constant amount of computation provided the lookahead depth (i.e., number of levels) is bounded by a constant. With respect to Fig. 15, note that because the routing layers have different electrical characteristics, the choice of balance points must be optimized both with respect to locations and the actual embeddings of the wires incident to the balance point. If the layer assignment is prescribed, the balance point computation is straightforward. Alternatively, deciding between various optional embeddings may be accomplished using one level of lookahead as in [32].

Another important extension lies in the selection of the CEP at each level. Instead of using the linear delay model to select a CEP, we may use a more accurate distributed RC model, to select the CEP so that clock skew is reduced by as much as possible. This is a strictly local modification of our method and does not affect the execution of the rest of the algorithm (or any variant). Such an extension applies to both ALG1 and ALG2, and is particularly useful when varying capacitive loadings exist at the terminals of the clock net. Since our algorithm operates in a bottom-up fashion, and since we treat each level independently, our method is able to accommodate variable gate loading very naturally.³

³We note that Tsay [32] recently gave a clock routing algorithm which uses a bottom-up construction approach similar to the one described in this paper. Tsay's algorithm incorporates one level of look-ahead and the introduction of "extra" wire to achieve an exact zero-skew tree with respect to the Elmore delay model [13]. At each step, Tsay's method combines a pair of zero-skew trees to yield a new zero-skew tree of larger size. The linear-time "Deferred-Merge Embedding" (DME) algorithm of [4]-[6] generalizes look-ahead in maintaining all loci of CEP's that are compatible with a zero-skew tree construction. DME thus reduces the cost of an initial clock tree topology computed by any previous method, while maintaining exact zero clock skew. In regimes where the linear delay model applies, the DME method produces the *optimal* (i.e., minimum-cost) zero-skew clock tree with respect to the prescribed topology, and this tree will also enjoy optimal source-terminal delay [4], [5]. It is noteworthy that with respect to DME, our present matching-based approach yields topologies which lead to lower cost trees than such other initial topologies as those of [6], [18], [32].

Finally, we mention that the PBT problem is interesting from a theoretical standpoint: the tradeoff between path-length balance and total edgelenh appears important not only for clock skew minimization, but also for a number of applications in areas ranging from computational geometry to network design.

In summary, we have presented a bottom-up approach for constructing clock routing trees, for both cell-based and general cell designs. Skew minimization is achieved by constructing the clock tree iteratively through geometric or graph matchings, while carefully balancing the pathlengths from the root to all leaves at each level of the construction. We verified our algorithm on numerous random examples, on industry benchmark circuits, and by SPICE timing simulations; the results show near-zero average clock skew while using total wirelength that compares favorably with previous work.

REFERENCES

- [1] H. Bakoglu, *Circuits, Interconnections and Packaging for VLSI*. Reading, MA: Addison-Wesley, 1990.
- [2] H. Bakoglu, J. T. Walker, and J. D. Meindl, "A symmetric clock-distribution tree and optimized high-speed interconnections for reduced clock skew in ulsi and wsi circuits," in *Proc. IEEE Intl. Conf. on Computer Design*, Port Chester, NY, Oct. 1986, pp. 118-122.
- [3] J. J. Bartholdi and L. K. Platzman, "A fast heuristic based on spacefilling curves for minimum-weight matching in the plane," *Inf. Proc. Lett.*, vol. 17, pp. 177-180, 1983.
- [4] K. D. Boese and A. B. Kahng, "Zero skew clock routing with minimum wirelength," Tech. Rep. CSD-TR-920012, Computer Sci. Dep., UCLA, Mar. 1992.
- [5] —, "Zero skew clock routing with minimum wirelength," in *Proc. IEEE Intl. ASIC Conf.*, Rochester, NY, September 1992, pp. 1-1.1-1-1.5.
- [6] T. H. Chao, Y. C. Hsu, and J. M. Ho, "Zero skew clock net routing," in *Proc. ACM/IEEE Design Automation Conf.*, Anaheim, CA, June 1992, pp. 518-523.
- [7] B. Chazelle and H. Edelsbrunner, "An optimal algorithm for intersecting line segments," *J. Ass. Comput. Mach.*, vol. 39, pp. 177-180, 1992.
- [8] J. Cong, A. B. Kahng, and G. Robins, "On clock routing for general cell layouts," in *Proc. IEEE Intl. ASIC Conf.*, Rochester, NY, Sept. 1991, pp. 14:5.1-14:5.4.
- [9] J. Cong, A. B. Kahng, G. Robins, M. Sarrafzadeh, and C. K. Wong, "Provably good performance-driven global routing," *IEEE Trans. Computer-Aided Design*, vol. 11, pp. 739-752, 1992.
- [10] W. M. Dai, T. Asano, and E. S. Kuh, "Routing region definition and ordering scheme for building-block layout," *IEEE Trans. Computer-Aided Design*, vol. CAD-4, pp. 189-197, 1985.
- [11] S. Dhar, M. A. Franklin, and D. F. Wann, "Reduction of clock delays in vlsi structures," in *Proc. IEEE Intl. Conf. on Computer Design*, Port Chester, NY, Oct. 1984, pp. 778-783.
- [12] D. Dobberpuhl, et al., "A 200 MHz 64b dual-issue cmos microprocessor," in *Proc. IEEE Intl. Solid State Circuits Conf.*, San Francisco, Feb. 1992, pp. 106-107.
- [13] W. C. Elmore, "The transient response of damped linear networks with particular regard to wide-band amplifiers," *J. Appl. Phys.*, vol. 19, pp. 55-63, 1948.
- [14] J. Fishburn, "Clock skew optimization," *IEEE Trans. on Computers*, vol. 39, pp. 945-951, 1990.
- [15] A. L. Fisher and H. T. Kung, "Synchronizing large systolic arrays," in *Proc. SPIE*, May 1982, pp. 44-52.
- [16] H. Gabow, "An efficient implementation of edmonds' algorithm for maximum matching on graphs," *J. Ass. Comput. Mach.*, vol. 23, pp. 221-234, 1976.
- [17] M. Garey and D. S. Johnson, "The rectilinear steiner problem is np-complete," *SIAM J. Appl. Math.*, vol. 32, pp. 826-834, 1977.
- [18] M. A. B. Jackson, A. Srinivasan, and E. S. Kuh, "Clock routing for high-performance IC's," in *Proc. ACM/IEEE Design Automation Conf.*, June 1990, pp. 573-579.

[19] A. B. Kahng, J. Cong, and G. Robins, "High-performance clock routing based on recursive geometric matching," in *Proc. ACM/IEEE Design Automation Conf.*, June 1991, pp. 322-327.

[20] A. B. Kahng and G. Robins, "A new class of iterative steiner tree heuristics with good performance," *IEEE Trans. Computer-Aided Design*, vol. 11, pp. 893-902, 1992.

[21] E. Lawler, *Combinatorial Optimization: Networks and Matroids*. New York: Holt Rinehart and Winston, 1976.

[22] I. Lin and D. H. C. Du, "Performance-driven constructive placement," in *Proc. ACM/IEEE Design Automation Conf.*, June 1990, pp. 103-106.

[23] P. Ramanathan and K. G. Shin, "A clock distribution scheme for non-symmetric VLSI circuits," in *Proc. IEEE Intl. Conf. on Computer-Aided Design*, Santa Clara, CA, Nov. 1989, pp. 398-401.

[24] J. Rubinstein, P. Penfield, and M. A. Horowitz, "Signal delay in RC tree networks," *IEEE Trans. Computer-Aided Design*, vol. CAD-2, pp. 202-211, 1983.

[25] R. Sedgewick, *Algorithms*. 2nd ed. Reading, MA: Addison-Wesley, 1988.

[26] T. L. Snyder and J. M. Steele, "Worst-case greedy matchings in the unit d-cube," *Networks*, vol. 20, pp. 779-800, 1990.

[27] A. Srinivasan, private communication, Oct. 1991.

[28] J. M. Steele, "Growth rates of euclidean minimal spanning trees with power weighted edges," *Ann. Probability*, vol. 16, pp. 1767-1787, 1988.

[29] K. J. Supowit, "New techniques for some dynamic closest-point and farthest-point problems," in *Proc. ACM/SIAM Symp. on Discrete Algorithms*, Jan. 1990, pp. 84-90.

[30] K. J. Supowit and E. M. Reingold, "Divide and conquer heuristics for minimum weighted euclidean matching," *SIAM J. Comput.*, vol. 12, pp. 118-143, 1983.

[31] K. J. Supowit, E. M. Reingold, and D. A. Plaisted, "The travelling salesman problem and minimum matching in the unit square," *SIAM J. Comput.*, vol. 12, pp. 144-156, 1983.

[32] R. S. Tsay, "Exact zero skew," in *Proc. IEEE Intl. Conf. on Computer-Aided Design*, Santa Clara, CA, Nov. 1991, pp. 336-339.

[33] P. Vaidya, "Geometry helps in matching," in *Proc. ACM Symp. on Theory of Computing*, 1988, pp. 422-425.

[34] D. F. Wann and M. A. Franklin, "Asynchronous and clocked control structure for VLSI based interconnection networks," *IEEE Trans. on Computers*, vol. 21, pp. 284-293, 1983.

Jason Cong (S'88-M'90), for a photograph and biography please see page 78 of the January 1993 issue of this TRANSACTIONS.



He is a member of ACM and SIAM.

Andrew B. Kahng (A'89) 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.

Since July 1989, he has been an assistant professor in the computer science department at UCLA, where he has received NSF Research Initiation and Young Investigator Awards. His research interests include computer-aided design of VLSI circuits, combinatorial and parallel algorithms, computational geometry, and the theory of global optimization.



Gabriel Robins (S'91, M'91) received the Ph.D. degree from the UCLA Computer Science Department.

Currently, he is Assistant Professor of Computer Science in the University of Virginia at Charlottesville. His primary areas of research are VLSI CAD and geometric algorithms, with recent work focusing on performance-driven routing, Steiner tree heuristics, motion planning, pattern recognition algorithms, and computational biology.

Dr. Robins is the author of a Distinguished Paper at the 1990 IEEE International Conference on Computer-Aided Design. He is a member of ACM, MAA, and SIAM. He has won the Distinguished Teaching Award and held an IBM Graduate Fellowship at UCLA.