High-Performance Clock Routing Based on Recursive Geometric Matching

Andrew Kahng, Jason Cong, and Gabriel Robins

UCLA Department of Computer Science
Los Angeles, California 90024

Abstract

Minimizing clock skew is a very important problem in the design of high performance VLSI systems. We present a general clock routing scheme that achieves extremely small clock skews, while still using a reasonable amount of wire length. This routing solution is based on the construction of a binary tree using recursive geometric matching. We show that in the average case the total wire length of the perfect path-balanced tree is within a constant factor of the wire length in an optimal Steiner tree, and that in the worst case, is bounded by $O(\sqrt{n})$ when the n leaves are arbitrarily distributed in the unit square. We tested our algorithm on numerous random examples and also on industrial benchmark circuits and obtained very promising results: our clock routing yields near-zero average clock skew while using similar or even shorter total wire length in comparison with the methods of [7].

1 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 becomes a more significant limitation on circuit performance.

Permission to copy without fee all or part of this material is granted provided that the copies are not made or distributed for direct commercial advantage, the ACM copyright notice and the title of the publication and its date appear, and notice is given that copying is by permission of the Association for Computing Machinery. To copy otherwise, or to republish, requires a fee and/or specific permission.

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] [5] [6] [22]. Although the H-tree structure can significantly reduce clock skew [5] [22], it is applicable only when the synchronizing components are identical in size and are placed in a symmetric array.

Ramanathan and Shin [14] proposed a clock distribution scheme for building block design where all blocks are organized in a hierarchical structure. They assume that a clock entry point is given 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.

Jackson, Srinivasan and Kuh [7] 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 $O(\frac{1}{\sqrt{n}})$ on average, one may easily construct examples for which the wirelengths between clock source and clock pins in their solution may vary by as much as the entire chip diameter [8].

In this paper, we study the problem of high-performance clock routing for the design of circuits with many small cells, as in standard-cell or sea-of-gates design styles. Here, 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 [14] cannot be applied either, since there is no natural hierarchy in the design and the number of components is usually too large to allow exhaustive examination of all possible routes.

This paper presents a basic algorithm and several variants, which minimize skew by constructing a clock tree that is balanced with respect to root-leaf pathlengths in the tree. Our algorithm always yields perfect pathlength balanced trees for inputs of two, three or four pins. Extensive experimental results indicate that as the size of the clock signal net becomes large, the maximum difference of pathlengths in the clock tree constructed by our algorithm remains essentially zero.

© 1991 ACM 0-89791-395-7/91/0006/0322 $1.50

This performance is obtained without undue sacrifice of wirelength: we prove that on average the total wire length in our clock tree construction is within a constant factor of the wire length in the optimal Steiner tree. Furthermore, the worst-case heuristic clock tree length is bounded by $O(\sqrt{n})$ for n points in the unit square, which is the same bound as for the worst-case length of the minimal Steiner tree.

Since both our work and the work in [7] are intended to solve the same problem (i.e. clock routing for circuits with a large number of small cells), we implemented the method of [7] and compared the results of the two algorithms. For uniformly distributed sets of up to 1024 pins in the unit square, 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 [7]. In addition, routing results for layouts of the MCNC Primary1 and Primary2 benchmarks are significantly better than those reported in [7]; we obtain perfectly balanced root-leaf pathlengths in the clock tree using several percent less total wire length.

The remainder of this paper is organized as follows. Section 2 defines a number of basic concepts and gives a precise formulation of our skew minimization problem. In Section 3, we present the clock routing algorithm in detail. Experimental results of our algorithm and comparisons with the method of [7] are presented in Section 4, and Section 5 gives concluding remarks.

2 Preliminaries

A synchronous VLSI circuit consists of two types of elements, synchronizing elements and combinational logic gates. The synchronizing elements are connected to one or several system-wide clock signals. Every closed path in a synchronous circuit contains at least one synchronizing element. The speed of a synchronizing circuit is mainly determined by the clock periods. It is well known [3] [7] 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. As VLSI feature sizes become smaller, the terms t_d, t_{su}, and t_{ds} all decrease significantly. Therefore clock skew becomes a more dominant factor in determining circuit performance.

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 wire length from the CEP to the synchronizing element, RC constants of wire segments in the routing, and the topology of the solution. Usually, the clock routing may be described as a RC tree [15], and we commonly use the first-order moment of the impulse response (also called Elmore's delay) to approximate delay in an RC tree. The formulas derived in [15] give both upper and lower bounds on delay in an RC tree, thus yielding a more accurate approximation.

However, although both the formula for Elmore's delay and those in [15] are very useful for simulation or timing verification, they involve sums of quadratic terms and are difficult to compute and optimize during the layout design process. Thus, a linear RC model and the wire length between CEP and the synchronizing element are often used to derive a simpler approximation for circuit delay (e.g., [14] [11]). In this paper, we also use wire length as a simple approximation of the delay in a routing solution. The clock skew is hence defined to be the maximum difference in wire length from the CEP to synchronizing elements in the clock signal net. We now give several important definitions, along with a formal statement of the skew minimization problem.

Recall that 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. A rooted tree is a binary tree if each non-leaf node has exactly two children. The length, or cost, of an edge in the tree is the *Manhattan* or L_1 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 defined to be 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. The objective of our algorithm is to construct a binary tree whose pathlength skew is as small as possible. In the VLSI regime, we may formulate this as follows.

The Path Balanced Tree (PBT) Problem: Given a set of points P in the L_1 unit square and a real number S, find a minimum-cost tree T connecting P such that for some distinguished node r of T, the costs of paths in T from r to any two leaf nodes differ by at most S.

It is not difficult to show that the PBT problem is NP-hard [8]. The objective of this paper is to present a heuristic algorithm for the PBT problem. In particular, we wish to construct a clock tree using $O(\sqrt{n})$ wirelength with pathlength skew as small as possible.[1] Before developing the algorithm, we introduce the notion of a geometric matching:

[1] Note that a zero skew tree can be trivially achieved by routing $n = |P|$ separate wires of constant length from the clock source to all of the clock pins, but this will entail a total clock tree cost of $O(n)$. We would like to obtain a solution which uses $O(\sqrt{n})$ total wirelength because the optimal Steiner tree will also use $O(\sqrt{n})$ wirelength in the average case [18].

Definition: Given a point set S of $2n$ points on the plane, a *geometric matching* on S is a set of n line segments whose endpoints are in S, with no two line segments sharing an endpoint.

Each line segment in the matching defines an *edge*. The cost of a geometric matching is the sum of the lengths of its edges. A geometric matching on S is optimal if its cost is minimum among all possible geometric matchings on S.

3 A Clock Routing Algorithm

To construct a tree by recursive matching, we begin with a forest of n isolated nodes (for convenience, assume $n = 2^k$), each of which is a tree with clock entry point being the node itself. The minimum-cost matching on these n points yields $\frac{n}{2}$ 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., so that the clock signal will have zero skew between the segment endpoints. In general, the matching operation will pair up the clock entry points (i.e., roots) of all trees in the current forest. At each level, we choose the root of the new merged tree to be the *balance point* which minimizes the pathlength skew to the leaves of the two subtrees (Figure 1).

The balance point is computed by finding the point p along the straight line connecting the roots of the two subtrees, such that the difference in pathlengths from p to any two leaves in the combined tree is minimum. 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 node added to the clock tree.

Notice that at each level of the recursion, we only have to match half as many nodes as before. Thus, in $k = \lfloor \log n \rfloor$ matching iterations, we obtain the complete clock tree topology. (In practice, we actually compute a min-cost *maximum cardinality* matching, i.e., if there are $2m+1$ nodes, we find the optimal m segment matching and match $m+1$ points at the next level.) Figure 2 gives a formal description of the algorithm.

The following two results show that our recursive matching approach indeed uses a reasonable amount of wirelength.

Theorem: For n points arbitrarily distributed in the unit square, the total edge length of any clock tree derived in this manner will be $O(\sqrt{n})$, which is of the same order as the maximum possible edgelength for the optimal Steiner tree on n points [18]. □

Theorem: For pointsets taken from a uniform distribution in the unit square, the total edgelength of our heuristic clock tree will be on average within a constant factor of the total edgelength in the minimum Steiner tree. □

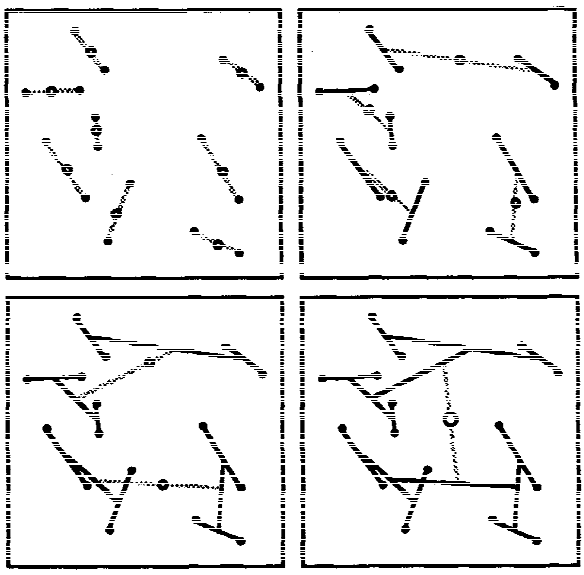

Figure 1: Example of the algorithm running on a random 8-point set. Solid dots denote the original points, and hollow dots represent the balance points of edges. At each level a geometric matching is performed on the balance points of the previous level.

```
T = ∅
while |P| > 1
    M = optimal geometric matching over P
    P' = ∅
    for (p1, p2) ∈ M do
        T1 = subtree of T rooted at p1
        T2 = subtree of T rooted at p2
        p = point on segment between p1 and p2
              such that p minimizes skew of tree
              T1 ∪ T2 ∪ {(p, p1), (p, p2)} w/root p
        P' = P' ∪ {p}
        T = T ∪ {(p, p1), (p, p2)}
    P = P' plus an unmatched node if |P| odd
CEP = Root of T = single remaining point in P
```

Figure 2: The recursive matching-based clock tree algorithm.

For proofs of these results the reader is referred to [8]. 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 by examining the solution of the example in Figure 1). Intuitively, balancing entails "sliding" the CEP along the "bar of the

Figure 3: Example of flipping an H as to minimize clock skew: the clock tree on the left has *no* zero-skew balance point along the middle segment of the H, while the clock tree on the right does.

H". However, it might not always be possible to obtain perfectly balanced pathlengths in this manner (see Figure 3). We therefore use a further optimization, which we call *H-flipping*: for each edge e added to the layout which matches CEP's 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 points which (i) minimizes pathlength skew, and (ii) further minimizes tree cost in the case of ties in pathlength skew. Two formal results are given in [8], proving that for four points it is always possible find an "H" orientation which achieves zero clock skew, and also limiting the increase in wirelength caused by H-flipping for nets of size four. Extensive empirical tests confirm that even for very large inputs, the H-flipping refinement almost always yields perfectly path-balanced trees with essentially no added wirelength expense.

We now briefly discuss complexity issues and the requirement of an efficient implementation. Notice that since our algorithm is based on geometric matching, its time complexity depends on that of the matching subroutine. The fastest known algorithms for general matching are $O(N^3)$ [10]. By taking advantage of planar geometry, the algorithmic complexity can be reduced to $O(N^{2.5} \log n)$ [21]. However, even this lower-complexity method will require long runtimes for large problem instances. To solve problems of practical interest, we chose to speed up the implementation by using efficient geometric matching heuristics [1] [19] [20]. 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 matching; to this end, note that k intersections of n line segments may be found efficiently in time $O(k + N(\log n)^2/\log\log n)$ [13].

In the following section, we discuss empirical results based on three matching methods which are $O(n)$, $O(n \log n)$ and $O(n^{3/2})$ respectively; all three yield very good clock routing solutions. When performance is critical, an optimal geometric matching algorithm might give an improvement over our current implementations, but will also require greater computational resources.

4 Experimental Results

Three main variants of the algorithm were implemented in ANSI C for the Sun-4, Macintosh and IBM environments; code is available from the authors. These variants correspond to the different matching subroutines.

The first heuristic variant (H1) uses the linear-time space partitioning heuristic of [19] to compute an approximate matching; the second variant (H2) uses an $O(n^{3/2})$ greedy matching [1]; and the third variant (H3) uses an $O(n \log n)$ spacefilling curve-based method [4]. We have further tested these three variants by running each with and without two refinements: (1) removing all edge crossings in the heuristic matching, and (2) performing the "H-flipping" described above. Either of these optimizations can be independently added to any of the three variants, giving rise to a total of twelve distinct versions of the basic algorithm. The variants of the algorithm are summarized as follows:

- **H1** - Use the space-partitioning matching heuristic of [19], which induces the matching by recursive bisection of the region (not the pointset).
- **H2** - Use a greedy matching heuristic (i.e., match closest pair of unmatched points) [1].
- **H3** - Use a space-filling curve to induce a Hamiltonian cycle through the points, and then choose the better embedded matching (i.e., either all odd edges or all even edges in the cycle) [4].
- **H4 through H6**- Same as H1 through H3, respectively, except the matching is improved by removing all edge crossings.
- **H7 through H12** - Same as H1 through H6, respectively, except that pathlength skew (or tree cost) is reduced by "H-flipping".

For comparison, we also implemented

- **H0** - The method of Jackson et al. [7].

The algorithms were tested on a large number of random pointsets of up to 1024 points, generated from a uniform distribution in the 1000 x 1000 grid. Results for a sample run with 50 random pointsets at each cardinality are summarized in Tables 1 through 4. Table 1 compares the average tree costs and Table 2 compares the average clock skews for all heuristics.

The computational results indicate that both optimizations (edge-uncrossing and H-flipping) will significantly improve both skew and total wirelength. When the refinements are combined, average clock skew essentially vanishes *completely*, and the wirelength of several variants is noticeably superior to the output of H0 (the method of [7]). The best variant appears to be H11, which is based on the greedy matching heuristic together with edge-uncrossing and H-flipping. This is

noteworthy because the greedy method is asymptotically as good as the optimal matching [16].

Tables 3 and 4 highlight the contrast between H11 and the method of [7], showing minimum, maximum and average values for both total wirelength and skew.

As noted in [9], 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 as H13, which returns the minimum-skew result from H1 through H12. Interestingly, in our experience H13 *always* returns a perfect pathlength balanced tree, i.e., for each problem instance, at least one of the heuristics H1 through H12 will yield a zero clock skew solution (while H0 never does). 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 less than two minutes on a Sun-4/60 workstation.

Finally, we tested our algorithm's performance on the Primary1 and Primary2 benchmarks, using the same layouts as in [7]. Figures 4 and 5 illustrate the output of variant H11. Table 5 compares the results of H11 and the results of [7] which were provided by the authors [17]. H11 completely eliminates clock skew while using 5% - 7% less wirelength.

5 Conclusion

We presented a heuristic method based on recursive matching which constructs clock tree routings with extremely small skew. The method uses total wirelength that is on average within a constant factor of the wirelength in a minimum Steiner tree, and in the worst case bounded by $O(\sqrt{n})$ for n terminals in the unit square. We verified our algorithm on numerous random examples as well as on industry benchmark circuits; results show near-zero average clock skew while using total wirelength that compares very favorably with previous results.

Pts	H0	H1	H2	H3	H4	H5	H6
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	H7	H8	H9	H10	H11	H12	H13
4	1125	1125	1125	1125	1125	1125	1125
8	2027	2028	1994	1971	1979	19800	1960
16	3502	3416	3428	33333	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 1: Average tree costs for the various heuristics.

Pts	H0	H1	H2	H3	H4	H5	H6
4	112.31	3.96	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	8.14
64	229.18	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	18.04
512	153.90	321.23	399.29	85.46	14.79	18.26	15.73
1024	125.34	339.34	402.59	89.75	17.14	16.71	15.35

Pts	H7	H8	H9	H10	H11	H12	H13
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	1.60	3.80	0.12	0.00	0.00	0.00	0.00
32	3.53	6.64	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	51.67	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

Table 2: Average skew values for the various heuristics.

Pts	H0 min	H0 ave	H0 max	H11 min	H11 ave	H11 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

Table 3: Minimum, average, and maximum total wirelength values for H11 and the method of [7].

References

[1] D. Avis, "Worst Case Bounds for the Euclidean Matching Problem", *International J. Comput. Math. Appl.* 7 (1981), pp. 251-257.

[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", *Proc. IEEE ICCD*, Port Chester, Oct. 1986, pp. 118-122.

[3] H. Bakoglu, *Circuits, Interconnections and Packaging for VLSI* , Addison-Wesley, 1990.

[4] J. J. Bartholdi and L. K. Platzman, "A Fast Heuristic Based on Spacefilling Curves for Minimum-Weight Matching in the Plane", *Inf. Proc. Letters* 17 (1983), pp. 177-180.

[5] S. Dhar, M. A. Franklin and D. F. Wann, "Reduction of Clock Delays in VLSI Structures", *Proc. IEEE ICCD*, Port Chester, Oct. 1984, pp. 778-783.

[6] A. L. Fisher and H. T. Kung, "Synchronizing Large Systolic Arrays", *Proc. SPIE* 341, May 1982, pp. 44-52.

[7] M. A. B. Jackson, A. Srinivasan and E. S. Kuh, "Clock Routing for High-Performance ICs", *Proc. ACM/IEEE DAC*, June 1990, pp. 573-579.

[8] A. Kahng, J. Cong and G. Robins, "High-Performance Clock Routing Based on Recursive Geometric Matching", UCLA CSD TR-900045, Nov. 1990.

[9] A. Kahng and G. Robins, "A New Family of Steiner Tree Heuristics With Good Performance: The Iterated 1-Steiner Approach", *Proc. IEEE ICCAD*, Nov. 1990, pp. 428-431.

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

[11] I. Lin and H. C. Du, "Performance-Driven Constructive Placement", *Proc. DAC* (1990), pp. 103-105.

[12] T. M. Lin and C. A. Mead, "Signal Delay in General RC Networks", *IEEE Trans. on CAD* CAD-3(4) (1984), pp. 331-349.

[13] F. P. Preparata and M. I. Shamos, *Computational Geometry: An Introduction*, New York, Springer-Verlag, 1985.

[14] P. Ramanathan and K. G. Shin, "A Clock Distribution Scheme for Non-Symmetric VLSI Circuits", *Proc. IEEE ICCAD*, November 1989, pp. 398-401.

[15] J. Rubinstein, P. Penfield and M. A. Horowitz, "Signal Delay in RC Tree Networks", *IEEE Trans. on CAD* CAD-2(3) (1983), pp. 202-211.

[16] T. L. Snyder and J. M. Steele, "Worst-Case Greedy Matchings in the Unit d-Cube", *Networks* 20 (1990), pp. 779-800.

[17] A. Srinivasan, ***private communication***, Oct. 1990.

[18] J. M. Steele, "Growth Rates of Euclidean Minimal Spanning Trees With Power Weighted Edges", *The Annals of Probability* 16(4) (1988), pp. 1767-1787.

[19] K. J. Supowit and E. M. Reingold, "Divide and Conquer Heuristics for Minimum Weighted Euclidean Matching", *SIAM J. Computing* 12(1) (1983), pp. 118-143.

[20] K. J. Supowit, E. M. Reingold, and D. A. Plaisted, "The Travelling Salesman Problem and Minimum Matching in the Unit Square", *SIAM J. Computing* 12(1) (1983), pp. 144-156.

[21] P. Vaidya, "Geometry Helps in Matching", *ACM Symposium on the Theory of Computing*, pp. 422-425.

[22] D. F. Wann and M. A. Franklin, "Asynchronous and Clocked Control Structure for VLSI Based Interconnection Networks", *IEEE Trans. on Computers* 21(3) (1983), pp. 284-293.

Pts	H0 min	H0 ave	H0 max	H11 min	H11 ave	H11 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	185.28	250	0	0.92	46
512	103	153.90	203	0	0.62	31
1024	94	125.34	157	0	0.08	4

Table 4: Minimum, average, and maximum skew values for H11 and the method of [7].

	H0 skew	H0 cost	H11 skew	H11 cost	skew impr	cost % impr
Prim1	0.29	161.7	0.00	153.9	0.29	4.8
Prim2	0.74	406.3	0.00	376.7	0.74	7.3

Table 5: Comparison of H11 and the method of [7] on Primary1 and Primary2 benchmarks: "skew" denotes standard deviation of path length; "cost" denotes total wirelength.

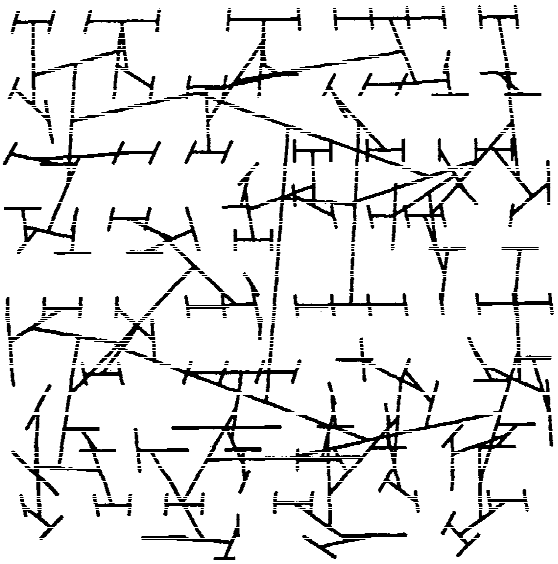

Figure 4: Output of variant H11 on Primary1 benchmark layout.

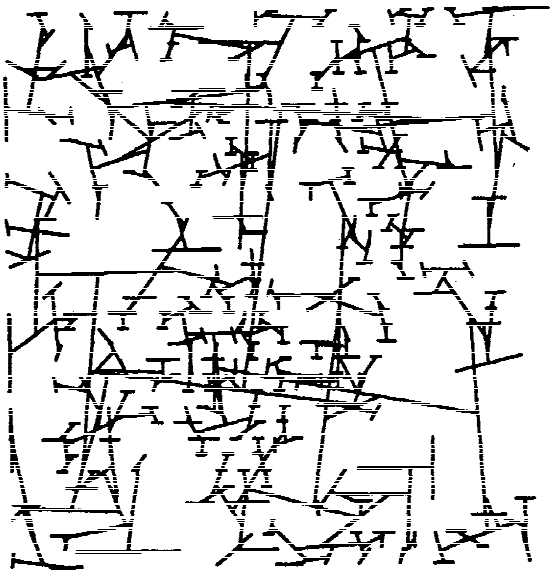

Figure 5: Output of variant H11 on Primary2 benchmark layout.