A NOVEL APPROACH TO IC PERFORMANCE OPTIMIZATION BY CLOCK ROUTING

by

Michael A. B. Jackson, Arvind Srinivasan,
and Ernest S. Kuh

Memorandum No. UCB/ERL M90/27

4 April 1990

A NOVEL APPROACH TO IC PERFORMANCE OPTIMIZATION BY CLOCK ROUTING

by

Michael A. B. Jackson, Arvind Srinivasan,
and Ernest S. Kuh

Memorandum No. UCB/ERL M90/27

4 April 1990

ELECTRONICS RESEARCH LABORATORY

College of Engineering
University of California, Berkeley
94720

A Novel Approach to IC Performance Optimization by Clock Routing

Michael A. B. Jackson Arvind Srinivasan Ernest S. Kuh

April 4, 1990

1 Introduction

In today's highly competitive IC marketplace, company survival necessitates product differentiability. Product differentiability may be engendered in many ways, several of which include: increased performance (e.g. lower power, faster timing, etc...), lower cost, more features, or faster time to market. Thus, the motivation for design techniques that enhance chip timing performance are of fundamental importance to the IC community.

The clock is the essence of a synchronous digital system. Physically, the clock is distributed from an external pad to all similarly clocked synchronizing elements through a distribution network that encompasses combinational logic and interconnects. It serves to unify the physical and temporal design abstractions by determining the precise instants in time at which the digital machine changes state. Because the clock is important, optimization of the clock signal can have a significant impact on the chip's cycle time, especially in high-performance designs. Non-optimal clock behavior is caused by two phenomena: the routing to the chip's synchronizing elements, and the asymmetric behavior of the clock distribution logic.

Previous work in clock optimization has been contributed by several authors. H-trees have been recognized for years as a technique to help reduce the skew in synchronous systems [FK82] [KGE82] [DFW84] [BWM86]. For regular structures such as systolic arrays the H-tree works well to reduce skew because the synchronizing elements are distributed in a uniform pattern throughout the chip. However, for general design styles, nonuniform distributions of clock pins are common and the H-tree becomes ineffective as a technique for clock routing. The large size of the clock net has led some researchers [DFW84] [Mij87] to perform buffer optimization within the clock distribution tree. [BWM86] have provided an analysis of the clock tree that considers the transmission line properties of the interconnects. [BBB+89] have presented an approach for ASIC clock distribution that integrates buffer optimization into place and route algorithms. However, in all previous work the routing of the clock net is performed using ordinary global routing tools based on minimum spanning or approximate minimum Steiner tree net models and with detailed routers that have little understanding of clock routing problems. This causes non-optimal clock behavior and as region size or the number of pins in the clock net increases, the detrimental behavior is exacerbated. In this paper, we focus on routing techniques for optimizing clock signals in VLSI circuits. We demonstrate the superiority of our algorithm over standard routing techniques for widely varying region size, clock pin

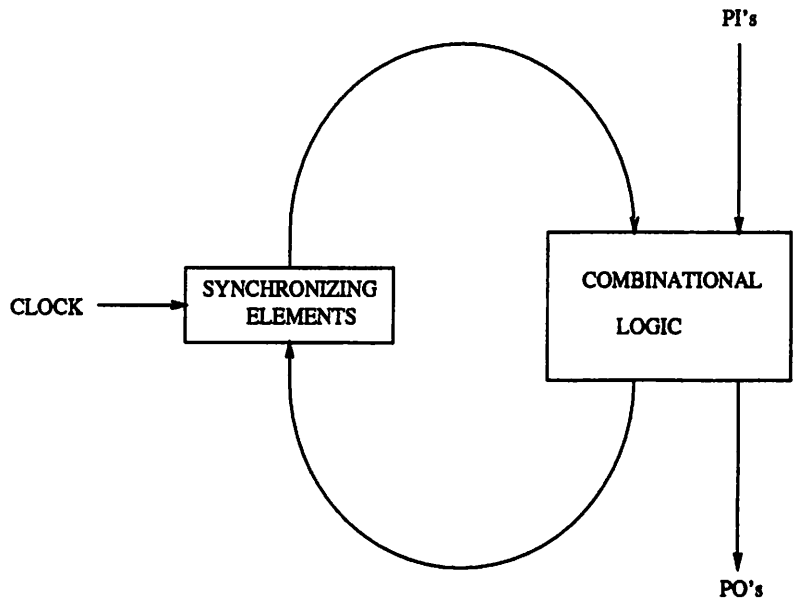

Figure 1: Synchronous digital system model

distributions, numbers of clock pins and technology feature size.

In section two the preliminaries necessary for understanding the paper are presented. Following this, in section three the problem is defined. Section four illustrates the algorithm for clock routing and section five discusses theoretical results. Next, in section six the experimental results are presented, and in section seven possible avenues for future work and conclusions regarding the approach are discussed.

2 Preliminaries

The majority of digital chips are synchronous in nature. Synchronous designs resemble the finite state logic model of Figure 1. The topological requirement imposed on such a system is that all closed signal paths must contain at least one synchronizing element. Satisfaction of this constraint has several valuable ramifications, two of which are: the assurance of deterministic behavior if the physical aspects of the design are correct, and it obviates the requirement that the combinational logic be free of transients as long as next state sampling is performed after the longest path has settled to its final value [MC80]. For simplicity, and without loss of generality, let the synchronizing elements in Figure 1 be edge-triggered. Furthermore, let CP denote the clock period, δ_L the largest path delay through the combinational logic, t_{SKEW} the clock skew, t_{SU} the set-up time of the edge-triggered synchronizing elements, and t_{CQ} the delay from the synchronizing element's clock pins to the Q output pins. In order to guarantee that no long-path timing violations occur in the design, the following equation must be satisfied

$$CP \geq \delta_L + t_{SKEW} + t_{SU} + t_{CQ} \tag{1}$$

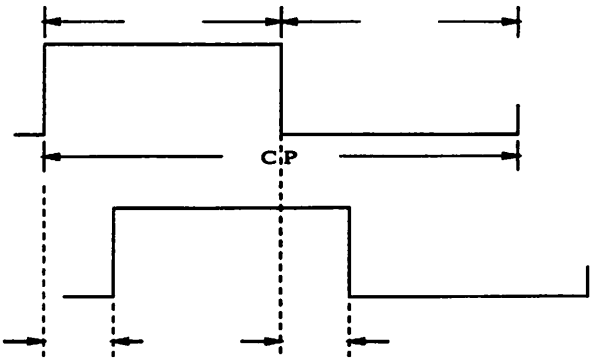

Figure 2: Relationship between τ_H, τ_L, and CP with 50 % duty cycle

The two timing related clock parameters that one must consider for high-performance design are clock skew and phase delay. *Clock skew* may be defined to be the maximum difference in arrival times at any two similarly clocked clock pins. Clock skew is caused by several phenomena: asymmetric routes to the clocked elements, differing interconnect line parameters, different delays through the clock distribution elements, and different device threshold voltages for the clock distribution logic. Equation 1 illustrates the important relationship between skew and the longest combinational logic path delay. As skew increases with the clock period held fixed, the efficiency of the digital system is reduced because valuable computation time is "stolen" from the total cycle time. Frequently in high-performance design environments, skew is constrained to be less than five percent of the clock period. Thus, in a 100 MHz design, skew would be constrained to be less than 500 ps. In this paper, routing techniques to help achieve this goal are presented. Assume for simplicity and without loss of generality that the only logic in the clock distribution tree is the external pad. *Phase delay* may be defined to be the maximum delay to any synchronizing clock pin. The same phenomena causing skew contribute to phase delay. It is convenient to consider phase delay as consisting of two components: an intrinsic rising or falling pad delay t_{IH} or t_{IL} contributed by the externally driven clock pad and the time to charge or discharge the clock net t_{CH} or t_{CL}. Expressions for rising and falling phase delay t_{PH} and t_{PL} may be defined as

$$t_{PH} = t_{IH} + t_{CH} \tag{2}$$

$$t_{PL} = t_{IL} + t_{CL} \tag{3}$$

Phase delay affects chip to chip interfaces by appearing as inter-chip skew and in worst-case scenarios may not provide adequate time to charge and discharge the clock net. For example, in a 100 MHz chip with a duty cycle of 50 %, the high and low portions of the clock period τ_H and τ_L equal 2.5 ns and impose the following constraints

$$t_{CH} < \tau_H \tag{4}$$

$$t_{CL} < \tau_L \tag{5}$$

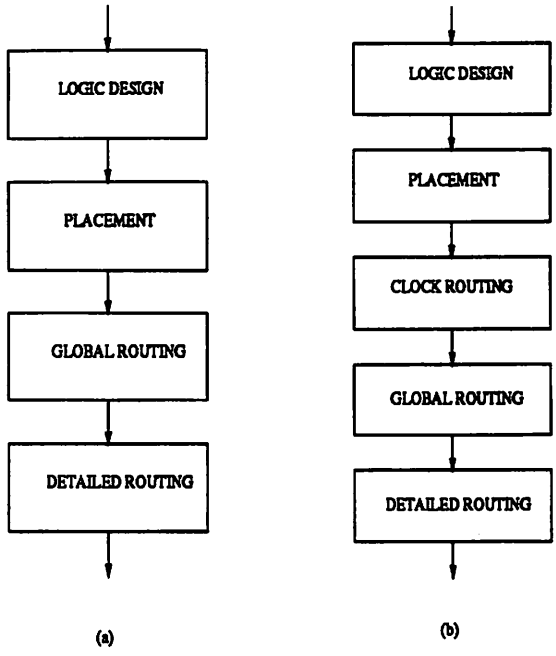

Figure 3: (a) Conventional IC design flow, (b) Proposed design flow

Figure 2 illustrates the relationship between τ_H, τ_L, and CP. These expressions are necessary but not sufficient conditions to guarantee proper clocking. In worst-case situations, t_{CH} and t_{CL} could conceivably constrain the clock period so it may be necessary to minimize the phase delay.

In discussions to this point, we have tacitly assumed that only one clock exists. However, in CMOS design styles it is commonplace to design with more than one clock. The ideas presented in this paper may easily be extended to the case of multiple clocks by treating the clock nets independently. In the presence of multiple clocks, skew and phase delay between (inter) and within (intra) the clocks must be considered. Adopting the policy of routing the clocks independently explicitly solves intra-clock problems. Inter-clock phase delay is implicitly minimized as desired, but the independent treatment of the problem does nothing for inter-clock skew. However, this does not present a problem because circuit techniques that insert delay in the faster clock trees can be used to equalize inter-clock skew should it be a problem. We have observed that the phase delay produced by our clock routing algorithm is smaller in every case when compared to simple routing techniques. Thus, in practice the inter-clock skew is also being reduced. Hereafter, for purposes of simplicity, attention will be restricted to single clock designs.

Prior to delving into the clock routing algorithm, it is necessary to define its role in the context of the overall design flow. A simplified portion of the traditional design flow is seen in Figure 3(a). In this figure the flow proceeds from logic design to physical design. Physical design consists of three classical steps: placement, global routing, and detailed routing. In the proposed design flow (Figure 3(b)), a clock routing step is interposed between placement and global routing. Clock buffer logic is not introduced during the logic design step. Instead, its inclusion is dependent on an accurate SPICE [Nag75] analysis performed after the clock route has been generated. If it is determined that buffers are needed to further reduce skew

or phase delay then they are inserted into the placement after making the necessary local perturbations. In row-based design styles, the placement modifications can be easily anticipated by accurately predicting the expected locations of the buffers, and adjusting the desired row lengths in the pre-placement topography specification. Presently, during the clock routing step, the global and detailed routes of the clock net are determined and passed to the global router as blockages. The clock route is constructed so that the clock signal behavior is optimized. It may be necessary to widen initial portions of the clock net to account for potential electromigration problems [BBB+89].

To understand the consequences of decisions made during physical design, one must model the interconnect parasitics that load the clock driver circuitry. We make the reasonable assumption that in a high-performance design environment that the interconnects are realized with aluminum due to its excellent conductor properties. Interconnect resistance R_{int} is determined using the following expression

$$R_{int} = \frac{\rho L}{WH} \tag{6}$$

where ρ is the resistivity of aluminum (3 $\mu\Omega$ cm), L is the interconnect length, W the interconnect width, and H the interconnect thickness. Interconnect capacitance C_{int} is modeled using a simple parallel-plate model

$$C_{int} = K_c(C_{ox} + C_I) \tag{7}$$

where

$$C_{ox} = \epsilon_{ox}\frac{WL}{t_{ox}} \tag{8}$$

and

$$C_I = \epsilon_{ox}\frac{LH}{L_s} \tag{9}$$

In these expressions K_c is a constant that is inserted to account for fringing effects, and can be calculated using the two-dimensional analysis of [DS80]. It is assumed to be 2.0 in our calculations. L_s represents the line spacing, t_{ox} represents the thickness of the field oxide, and ϵ_{ox} is the permitivity of the oxide. All lengths used to calculate resistance and capacitance are based on Manhattan distances.

Armed with estimates for R_{int} and C_{int}, simple and accurate interconnect delay estimates may be determined based on the first-order moment of the impulse response which has also been called Elmore's delay [RPH83] [Elm48]. The interconnects are treated as distributed RC trees and are modeled using their equivalent T-networks.

3 Problem Definition

Given the IC's placement, the locations of blockages on the routing layers, the positions of all clock pins on the clock net, and the location of the clock pad along the periphery of the chip, the problem may be defined as follows: construct a clock tree consisting only of Manhattan segments that optimizes the clock skew, wirelength and phase delay subject to the blockages on the routing layers. In general, one seeks to minimize clock skew and wirelength, subject to constraints on phase delay and the routing. Formally, clock optimization could be formulated as follows

$$\begin{array}{ll} \min & f(t_{skew}, WL) \\ \text{subject to} & t_{CH} < \tau_H \\ & t_{CL} < \tau_L \\ & NR = 0 \end{array}$$

where WL equals the total wirelength and NR equals the number of no routes.

4 The Algorithm

4.1 The Basic Algorithm

The algorithm which we call the Method of Means and Medians (MMM) is conceptually simple, elegant and yields some theoretical results which are intuitively pleasing. Let $S = \{s_1, s_2, \ldots, s_n\}$ be the set of points in the plane which represent the clock pins. Each s_i is a tuple (x_i, y_i).
Define

$$x_c(S) = \frac{\sum_{i=1}^{n} x_i}{n} \tag{10}$$

$$y_c(S) = \frac{\sum_{i=1}^{n} y_i}{n} \tag{11}$$

$(x_c(S), y_c(S))$ represents the center of mass of the set of points S. We shall use the notation $S_x(S)$ or simply S_x to denote the ordered set of points obtained by ordering the set S by increasing x coordinate, i.e. $x_i \leq x_j$ if $s_i, s_j \in S_x(S)$ and $i < j$. Similarly, $S_y(S)$ or simply S_y represents the ordered set of points obtained by ordering the set S by increasing y coordinate. Define

$$S_L(S) = \{s_i \in S_x \mid i \leq \lceil n/2 \rceil\} \tag{12}$$

$$S_R(S) = \{s_i \in S_x \mid \lceil n/2 \rceil < i \leq n\} \tag{13}$$

$$S_B(S) = \{s_i \in S_y \mid i \leq \lceil n/2 \rceil\} \tag{14}$$

$$S_T(S) = \{s_i \in S_y \mid \lceil n/2 \rceil < i \leq n\} \tag{15}$$

The sets S_L and S_R represent the division of S into two sets about the median x coordinate of the set of points. These sets partition the original region in the x dimension into two regions with approximately equal number of elements in each sub-region. In fact, $| \, |S_L| - |S_R| \, | \leq 1$. Similarly, S_B and S_T represent the division of S into two sets about the median y coordinate of the set of points. Given S, the basic algorithm first splits S into two sets (arbitrarily in the x direction or y direction). Assume that a split of S into S_L and S_R is made. Then, the algorithm routes from the center of mass of S to each of the centers of mass of S_L and S_R respectively. The regions S_L and S_R are then recursively split in the y direction (the direction opposite to the previous one). Thus, splits alternating between x and y are introduced upon the set of points recursively until there is only one point in each sub-region. The pseudo-code for the algorithm is given in Figure. 4.

procedure basic_MMM(S)
begin
    if $|S| \leq 1$ **return;**
    $x_0 = x_c(S)$; $y_0 = y_c(S)$;
    $x_{left} = x_c(S_L(S))$; $y_{left} = y_c(S_L(S))$;
    $x_{right} = x_c(S_R(S))$; $x_{right} = y_c(S_R(S))$;
    route from (x_0, y_0) to (x_{left}, y_{left}) and (x_{right}, y_{right});
    $x_{bot-left} = x_c(S_B(S_L(S)))$; $y_{bot-left} = y_c(S_B(S_L(S)))$;
    $x_{top-left} = x_c(S_T(S_L(S)))$; $y_{top-left} = y_c(S_T(S_L(S)))$;
    $x_{bot-right} = x_c(S_B(S_R(S)))$; $y_{bot-right} = y_c(S_B(S_R(S)))$;
    $x_{top-right} = x_c(S_T(S_R(S)))$; $y_{top-right} = y_c(S_T(S_R(S)))$;
    route from (x_{left}, y_{left}) to $(x_{bot-left}, y_{bot-left})$ and $(x_{top-left}, y_{top-left})$;
    route from (x_{right}, y_{right}) to $(x_{bot-right}, y_{bot-right})$ and $(x_{top-right}, y_{top-right})$;
    basic_MMM($S_B(S_L(S))$);
    basic_MMM($S_T(S_L(S))$);
    basic_MMM($S_B(S_R(S))$);
    basic_MMM($S_T(S_R(S))$);
end;

Figure 4: Pseudo-code for the basic algorithm

4.2 Improvements

The simple algorithm described above yields good results, but there is room for further improvement. In the following discussion we define a cut in the x direction to mean a split resulting in a left region and a right region. A cut in the y direction implies a split resulting in a top and a bottom region.

4.2.1 Delay equalization look-ahead

Consider the example shown in Figure 5, where S is the clock source. If we make a cut in the x direction and then recursively split the left and right regions in the y direction, we get the result shown in Figure 5(a). Clearly, there is skew between points P_{LT} and P_{RT}. However, if we reverse the cut directions, i.e, split in the y direction first followed by a split in the x direction, we get the result shown in Figure 5(b), which has no skew between the endpoints.

This example illustrates the need for making a good choice of cut direction at each level of the recursion tree. We, make the choice by a one level look-ahead technique. Given a region to be split, the algorithm makes an x direction cut followed by a y direction cut on the resulting left and right regions. It also makes a y direction cut followed by an x direction cut. The skews for each of the configurations is compared and the cut direction that minimizes skew between its current endpoints is chosen. The method of estimating the skew between the endpoints is described in the following section.

4.2.2 Delay Calculation

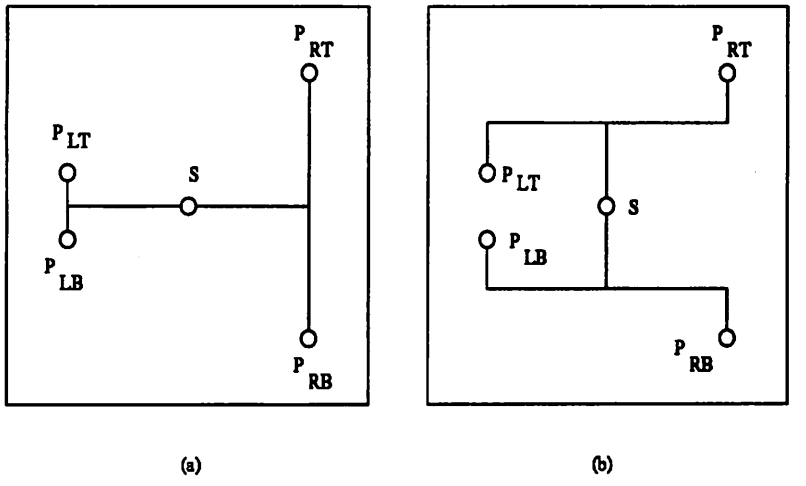

Figure 5: Clock tree (a) without look-ahead, (b) with look-ahead

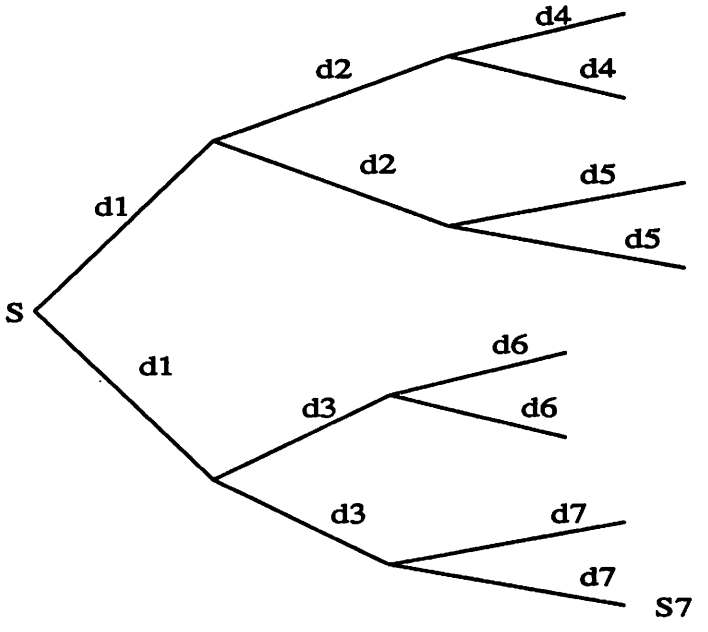

Figure 6: Clock tree representation used for delay calculations

We use the Penfield-Rubinstein [RPH83] algorithm for calculating delays to the endpoints in the grown clock tree. The resistance and capacitance of the tree segments are modeled by a T-network model. Consider the tree shown in Figure 6. Because of a property of the center of mass (see next section), the lengths of tree segments from the center of mass of a region to each of its two sub-regions will always be equal and symmetric. The delay from s to the endpoint $s7$ is calculated as

$$\delta_{s-s7} = R_l(0.5C_l(d_1^2 + d_3^2 + d_7^2) + C_g(4d_1 + 2d_3 + d_7) + 2C_l(d_1d_3 + d_1d_6 + d_1d_7 + d_3d_7)) \qquad (16)$$

where C_g is the gate capacitance at the clock pin (assumed equal for all clock pins), C_l is the capacitance per unit length and R_l is the resistance per unit length. Note that the delay to any endpoint depends on the lengths of other segments connecting different endpoints, so it is important to use delay estimates to drive the look-ahead rather than length calculations. The complexity of the algorithm with look-ahead is $O(n \log n)$ where n is the number of clock pins. We prove this in the theoretical results section. We shall refer to the algorithm as the Method of Means and Medians (MMM) in the following text.

5 Theoretical results

In this section we derive some key results that motivate the Method of Means and Medians. First, we prove that after splitting a region into two sub-regions, the lengths of segments from the center of mass of the region to each of its sub-regions is equal and symmetric. Next, we establish a bound on the total wirelength for a gridded distribution of points and compare it with the wirelength for a minimum rectilinear Steiner tree spanning those points. We also present an interesting result that claims that increasing the number of points within a region reduces the skew. Finally, we prove that the algorithm with one level of look-ahead runs in time $O(n \log n)$ where n is the number of clock pins. All our theoretical results corroborate our experimental results (Section 6).

Theorem 5.1

Given a set of points $S = \{s_1, s_2, \ldots, s_n\}$, where n is a even integer,

$$| x_c(S) - x_c(S_L(S)) | + | y_c(S) - y_c(S_L(S)) | = | x_c(S) - x_c(S_R(S)) | + | y_c(S) - y_c(S_R(S)) |$$

Proof

Assume without loss of generality that the points in S are ordered by increasing x coordinate value.

$$x_c(S) = \frac{\sum_{i=1}^{n} x_i}{n},$$

$$y_c(S) = \frac{\sum_{i=1}^{n} y_i}{n},$$

$$x_c(S_L(S)) = \frac{\sum_{i=1}^{n/2} x_i}{n/2},$$

$$y_c(S_L(S)) = \frac{\sum_{i=1}^{n/2} y_i}{n/2},$$

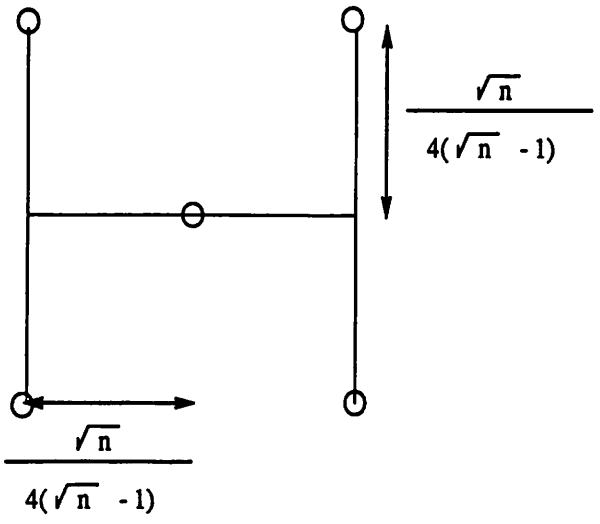

Figure 7: Illustration for total wirelength calculation

$$x_c(S_R(S)) = \frac{\sum_{i=n/2+1}^{n} x_i}{n/2},$$

$$y_c(S_R(S)) = \frac{\sum_{i=n/2+1}^{n} y_i}{n/2},$$

$$x_c(S) - x_c(S_R(S)) = \frac{\sum_{i=1}^{n} x_i}{n} - \frac{\sum_{i=1}^{n/2} x_i}{n/2} = \frac{1}{2}\left(\frac{\sum_{i=n/2+1}^{n} x_i}{n/2} - \frac{\sum_{i=1}^{n/2} x_i}{n/2}\right)$$

Similarly,

$$y_c(S) - y_c(S_L(S)) = \frac{1}{2}\left(\frac{\sum_{i=n/2+1}^{n} y_i}{n/2} - \frac{\sum_{i=1}^{n/2} y_i}{n/2}\right),$$

$$x_c(S) - x_c(S_R(S) = \frac{1}{2}\left(\frac{\sum_{i=1}^{n/2} x_i}{n/2} - \frac{\sum_{i=n/2+1}^{n} x_i}{n/2}\right),$$

$$y_c(S) - y_c(S_R(S)) = \frac{1}{2}\left(\frac{\sum_{i=1}^{n/2} y_i}{n/2} - \frac{\sum_{i=n/2+1}^{n} y_i}{n/2}\right)$$

■

A similar result holds between S, $S_B(S)$ and $S_T(S)$. The significance of the above result is that at every split in the algorithm, the lengths to each of the sub-regions is always equal. Note that as we move deeper into the clock tree, the segments become smaller in length. Thus, at the topmost level of the clock tree when the segments are longest, we ensure exact balance and no skew.

Lemma 5.2

Given a distribution of n points on a uniform grid, where $n = 4^k, k$ an integer ≥ 1, within a region of side 1.0 unit, the total wirelength of the tree produced by the basic algorithm grows as $\frac{3}{2}\sqrt{n}$.

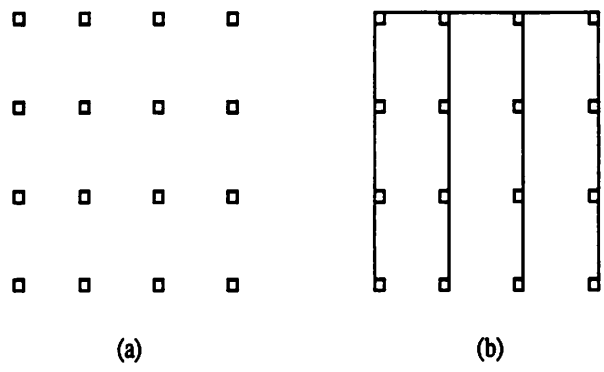

Figure 8: Minimum rectilinear Steiner tree on uniformly spaced points

Proof

The center of mass of the entire set of points lies at $(0.5, 0.5)$. The cut direction does not matter in this case. As shown in Figure 7, at the first level, the lengths of the segments to each of the sub-regions is $\frac{\frac{1}{2}\sqrt{n}}{\sqrt{n}-1}$. At the next level, we have 4 segments of length $\frac{\frac{1}{4}\sqrt{n}}{(\sqrt{n}-1)}$. These six segments make up an "H" shape. The total length of the "H" shape seen at the topmost level is $\frac{\frac{3}{2}\sqrt{n}}{\sqrt{n}-1}$. At the next level, we will have four such "H" patterns of half this length and then sixteen "H" shapes of one-fourth the dimensions of the topmost "H". If we denote the total wirelength to be $L(n)$, we obtain the following expression:

$$L(n) = 3 \times \frac{\frac{1}{2}\sqrt{n}}{(\sqrt{n}-1)} + 4 \times (3 \times \frac{\frac{1}{4}\sqrt{n}}{(\sqrt{n}-1)}) + 16 \times (3 \times \frac{\frac{1}{8}\sqrt{n}}{(\sqrt{n}-1)}) + \ldots + \frac{n}{4} \times (3 \times \frac{1}{(\sqrt{n}-1)}) \quad (17)$$

There are $\log_4(n)$ terms in the series. Thus,

$$L(n) = \frac{3}{2}\frac{\sqrt{n}}{\sqrt{n}-1} \sum_{i=0}^{\log_4(n)} 2^i \quad (18)$$

This expression can be easily shown equal to $\frac{3}{2}\sqrt{n}$. ■

Theorem 5.3

The wirelength for a minimum rectilinear Steiner tree spanning a set of n uniformly spaced points on a grid, where $n = 4^k, k$ an integer ≥ 1, within a region of side 1.0 unit, grows as $\sqrt{n} + 1$. This is also the largest possible wirelength for a rectilinear Steiner tree for a distribution of n points in a unit square. Any other distribution of n points within the unit grid will yield a smaller total wirelength.

Proof

According to a result by Hanan [Han66], the Steiner points for the minimum rectilinear Steiner tree must lie on the intersections of the horizontal and vertical lines drawn through the points in the region. For the case of uniformly spaced points, the Steiner points coincide with the points themselves. Therefore, we can connect each column of points by a single segment. A total of $\sqrt{n}$ segments of length 1 unit each are required to connect up the points in the columns. Exactly one additional segment of length one unit is

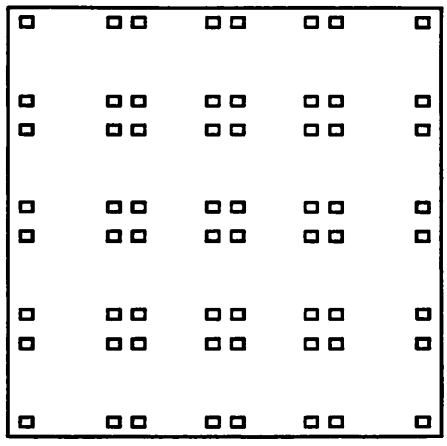

Figure 9: Conjectured worst case configuration

required to connect together the column segments. The total wirelength for this configuration is $\sqrt{n}+1$. This is illustrated in Figure 8. Interestingly, according to a result by Hwang and Chung [CH79] the largest minimal rectilinear Steiner tree on n points in a square of side 1 unit is also $\sqrt{n}+1$. ■

These results indicate that the wirelength of the clock tree is worse only by a constant factor of $\frac{3}{2}$ in comparison to a minimum rectilinear Steiner tree for the particular distribution of points.

We conjecture that the worst case wirelength for the clock tree occurs for the configuration of clock pins shown in Figure 9. The wirelength for this configuration can be shown to be $\frac{9}{4}\sqrt{n}-\frac{3}{2}$. Thus even in this case, the total wirelength is still a constant times the wirelength for the largest minimum rectilinear Steiner tree.

We define the *sparsity* p of a distribution of points in a region to be the total number of points in the region divided by the area of the region. It is a measure of the average number of points per unit area. The next result concerns the variation of skew with sparsity.

Theorem 5.4

For a uniformly randomly distributed set of points inside a box of side n units with sparsity p, the expected maximum difference in length from the center to any endpoint for the basic algorithm is proportional to $\frac{1}{\sqrt{p}}$.

Proof

There are pn^2 points in the region. We assume that since the points are uniformly distributed over the region, each split will divide the points evenly into two sets of equal size. Every split divides the region into two sub-regions each of which has one dimension reduced by a factor of 2 because we always partition about the center point. A total of $\lceil \log_2(pn^2) \rceil$ splits will be made by the algorithm. Half of the splits will occur in the x dimension and the other half in the y dimension. Thus, the size of one side of the smallest region produced by the algorithm will be $\frac{n}{2^{\lfloor 1/2 \log(pn^2) \rfloor}} = \frac{1}{\sqrt{p}}$. Since we assumed a uniform distribution of points, the differences in lengths of segments if any will occur at the last level. The maximum difference

in length that can occur in the last level is one side of the smallest region which is $\frac{1}{\sqrt{p}}$. ∎

This result indicates that as the number of points within the region is increased, the skew between the endpoints reduces! Our experimental results support this claim.

Theorem 5.5

The algorithm with one level look-ahead runs in time $O(n \log n)$, where n is the number of points in the region.

Proof

We maintain two sorted lists of the points, one list by increasing x coordinate value and another by increasing y coordinate value. The sorting requires $2n \log n$ time. There are $O(\log n)$ recursion levels and at each level i we have 2^i regions. For each region in a level, we perform 4 center of mass calculations, two for an x direction split and two for a y direction split. The total number of additions required to compute the center of masses at any given level is thus $4n$. Thus, the total work done in computing the centers of mass is $O(n \log n)$. Within the data structure for every region, we store the delay from the source up to the center of mass of that region. Therefore a constant amount of work is done per region in computing the delays from the source to the region's left and right sub-regions and top and bottom sub-regions corresponding to an x direction split and a y direction split. The total work in computing delays is thus $\sum_{i=0}^{\log n} 2^i = O(n)$. Therefore, the running time of the algorithm is $O(n \log n)$. ∎

The algorithm is extremely fast and the running time for routing a region with 4096 points on a DECStation 3100 computer (14 MIPS) was less than a second of CPU time. Therefore, speed is not an issue of concern in this algorithm.

6 Experimental Results

As a test of the effectiveness of MMM it was run on twenty random examples and the MCNC industrial benchmarks Primary1 and Primary2. The twenty random examples had pin distributions that were uniformly distributed in a square region. For the twenty examples, four equal-sized examples with 16, 32, 64, 256 or 512 pins were generated. For comparative purposes, we routed the same pin distributions using a minimum spanning tree (MST) algorithm because it approximated the routing many global routers would make if a simplistic approach was taken for clock routing. SPICE [Nag75] files were generated for all examples based on Manhattan geometries, and the interconnect was driven by a single I/O buffer pad with equivalent drive of ten times the minimum sized inverter cell in a 2 μm design style. To model gate loading, a capacitance was placed at the leaves of the clock distribution tree of value 0.3 pF.

The first experiment was designed to compare clock skew versus chip size. Chip size was varied by increasing the chip area for each of the randomly distributed examples from 1.56 mm^2 to 100 mm^2, effectively scaling the relative locations of the points. The skew values reported represent the average of the four equal sized pin examples at each chip size. The results are presented in Figure 10 and the superiority of MMM over MST is clearly apparent. The skew growth is practically insignificant for increasing chip size, with the best MST result (16 pins) introducing more than five times the amount contributed by the

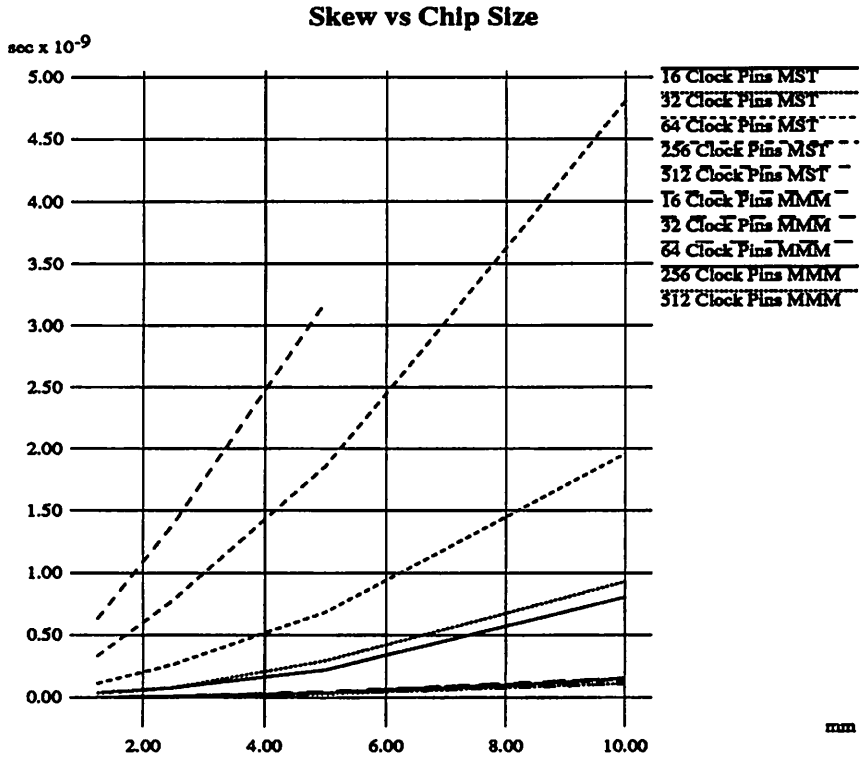

Figure 10: Skew comparison for varying chip size

Chip Size(mm)	MST Skew(ns)	MMM Skew(ns)
1.25	0.63	0.0033
2.50	0.78	0.0087
5.0	1.86	0.035
10.0	4.81	0.14

Figure 11: Skew comparison table for 2 micron feature size, 256 clock pins

worst MMM result at 100 mm^2 and the 512 pin MST example contributing more than 68 times the skew at 25 mm^2 than MMM.

Figures 12 and 13 illustrate the relationship between phase delay and chip size for each of the examples run with MMM and MST. For each example of a fixed pin count, the phase delay is less for MMM on all chip sizes.

To determine the relationship between skew and the number of pins with chip size fixed at 25 mm^2, MMM and MST were compared to one another with the result appearing in Figure 15. Interestingly, the skew decreased with increasing number of pins for MMM and grew linearly for MST. Figure 17 illustrates the inverse relationship between sparsity and the maximum difference in length to the endpoints in the tree.

Similarly, phase delay versus the number of pins for a chip size of 25 mm^2 were compared for MST and MMM. Again, MMM displayed a clear advantage with its growth in phase delay appearing to be sub-linear and MST approximating linear growth. These results can be seen in Figure 18.

The dramatic improvements in clock skew and phase delay are paid for in terms of total wirelength. To illustrate this, the average wirelength for all examples was plotted against the number of pins in Figure 20. The experimental results corroborate the theoretical $\sqrt{n}$ relationship between wirelength and number of points n with the difference appearing as a constant factor. Thus, the improvements in clock behavior are gained with an increase in the clock net's wirelength.

To estimate the effects of future device and interconnect scaling on the problem of clock optimization through routing, a series of experiments were conducted where interconnect and device feature sizes were scaled. Interconnect scaling was carried down to an interconnect width of 0.5 μm at which point further interconnect scaling is unlikely due to the expected presence of four or more layers for interconnect routing [Ko89]. In the experiments the driving buffer was not scaled for simplicity of implementation. To model future trends, a scaling factor was applied to both the devices and the interconnect. Device dimensions were scaled horizontally and vertically by α (gate capacitance reduced by α) while interconnect dimensions were scaled horizontally by α (interconnect resistance per unit length increased by α while capacitance per unit length decreased by α). Interconnect dimensions W and L_s were kept equal while H and t_{ox} were fixed at 1 μm. Even though industrial processes may differ from the estimated trends, the qualitative relationship should remain.

Figure 22 plots clock skew versus minimum feature size. As device and interconnect geometries are reduced, skew as determined by SPICE increases. However, it remains relatively constant for MMM and grows substantially worse for MST. This trend exists for all examples.

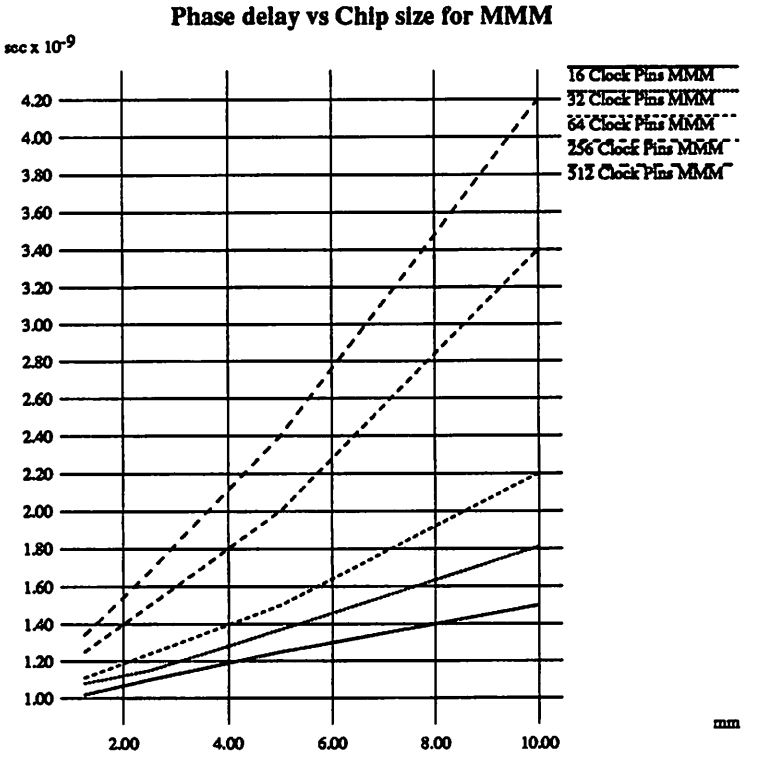

Figure 12: Phase delay for MMM, varying chip size

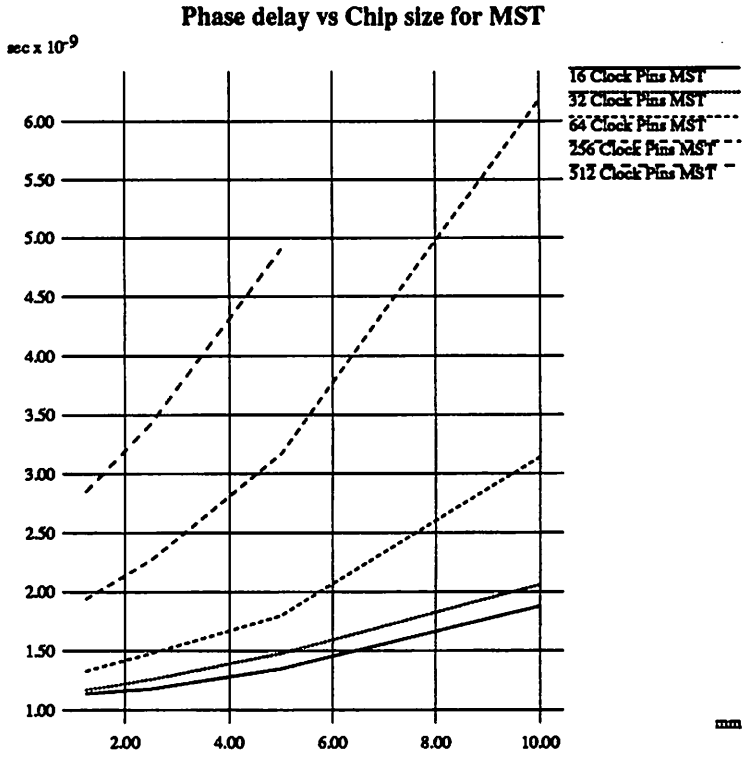

Figure 13: Phase delay for MST, varying chip size

Chip Size(mm)	MST Phase Delay(ns)	MMM Phase Delay(ns)
1.25	1.94	1.25
2.50	2.27	1.5
5.0	3.17	2.0
10.0	6.18	3.4

Figure 14: Phase delay table for 2 micron feature size, 256 points, varying chip size

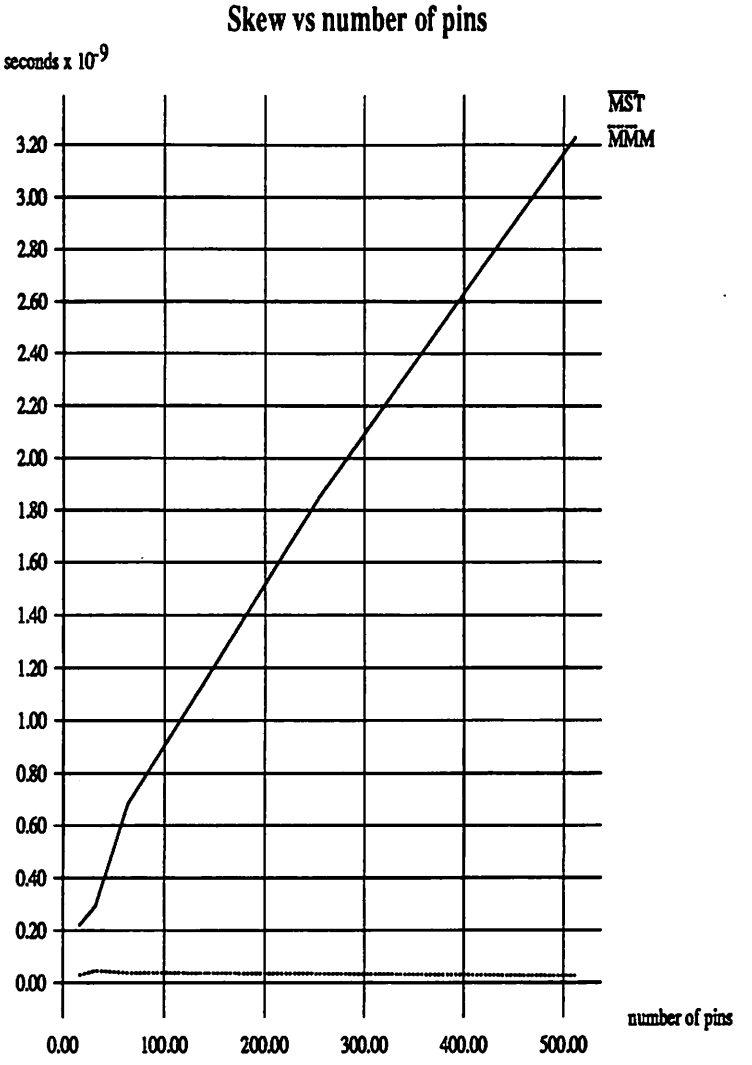

Figure 15: Skew comparison for varying no. of points

Number of pins	MST Skew(ns)	MMM Skew(ns)
16	0.22	0.033
32	0.29	0.047
64	0.68	0.039
256	1.86	0.035
512	3.23	0.027

Figure 16: Skew comparison table for 2 microns, varying number of points

No. of points	Δ
64	0.301
256	0.253
1024	0.174
4096	0.122

Figure 17: Δ is the maximum difference in length to endpoints

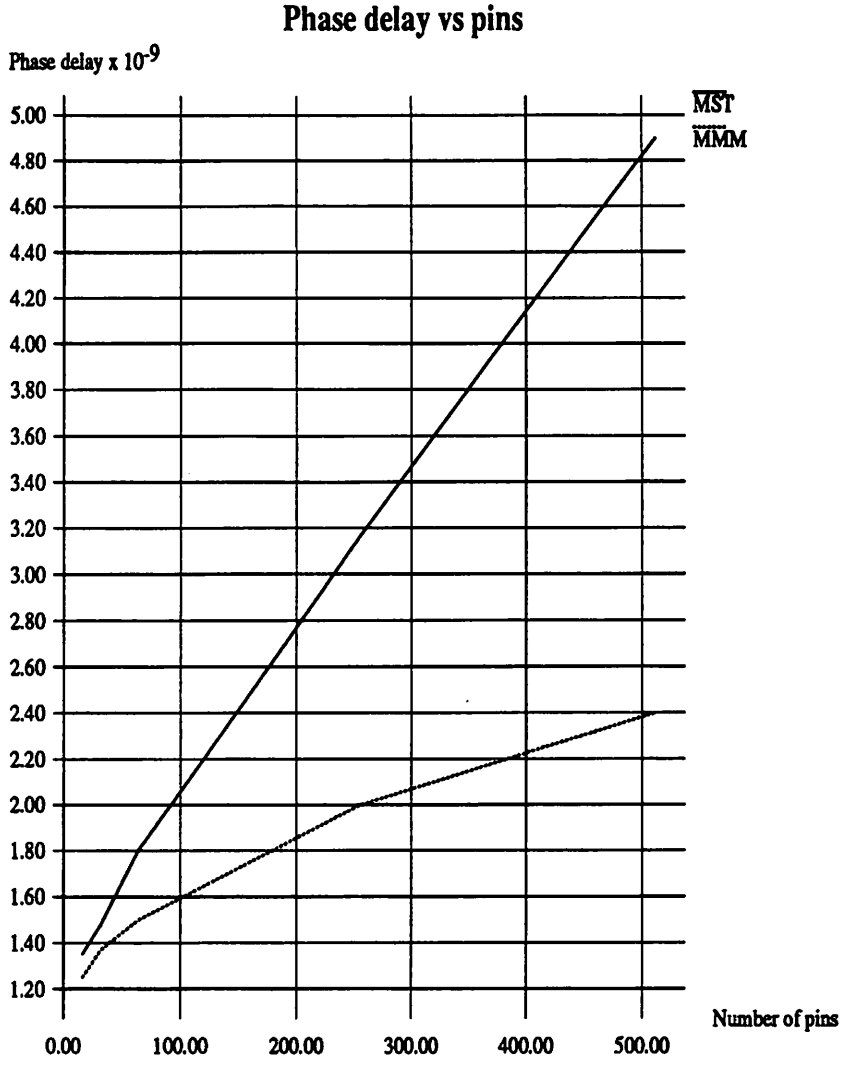

Figure 18: Phase delay comparison for varying no. of points

Chip Size(mm)	MST Phase Delay(ns)	MMM Phase Delay(ns)
16	1.35	1.25
32	1.48	1.37
64	1.8	1.5
256	3.17	2.0
512	4.9	2.4

Figure 19: Phase delay table for 2 micron feature size, varying no. of points

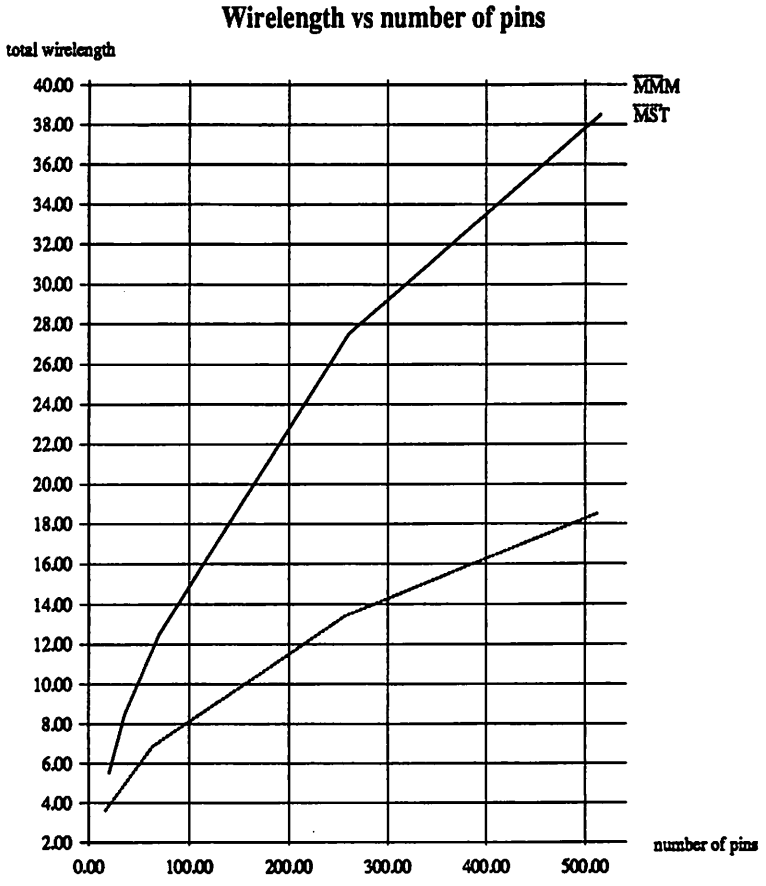

Figure 20: Wirelength comparison for varying number of pins

No. of points	MST	MMM	$\frac{3}{2}\sqrt{n}$	$\frac{9}{4}\sqrt{n} - \frac{3}{2}$
16	3.6	5.5	6.0	9.0
32	5.2	8.5	8.48	12.73
64	7.1	12.5	12.0	18.0
256	12.2	27.3	24.0	36.0
512	18.5	38.5	33.9	50.9

Figure 21: Wirelength comparison table

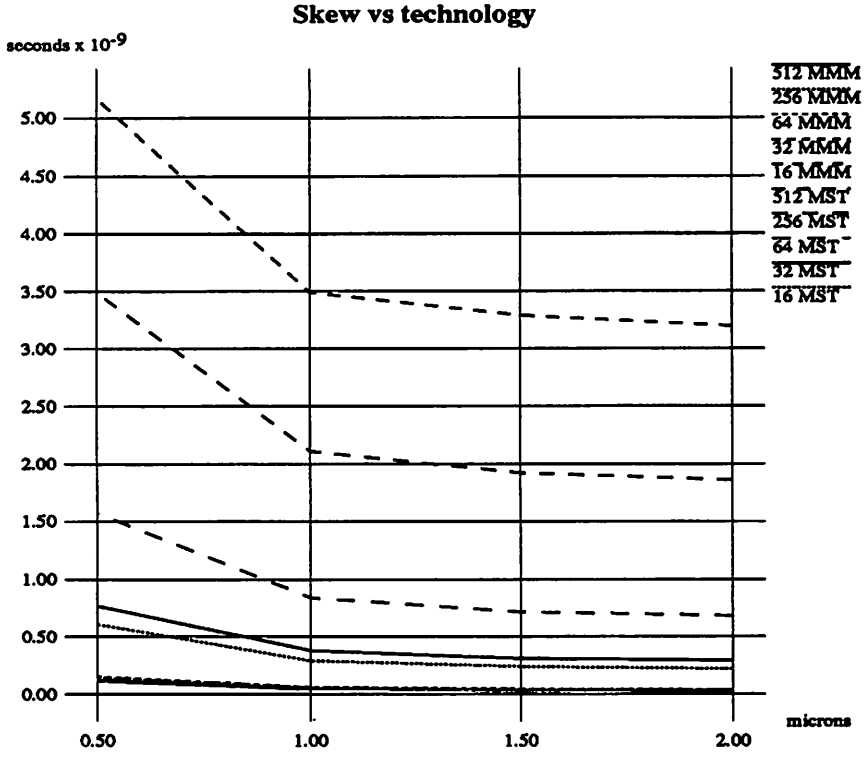

Figure 22: Skew variation with feature size

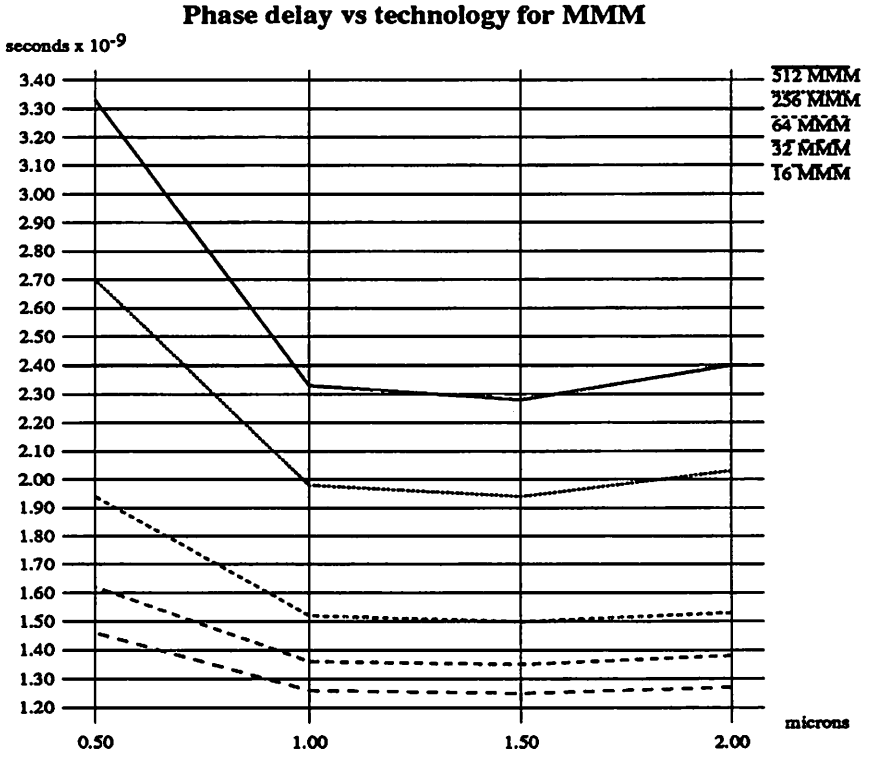

Figure 23: Phase delay vs feature size for MMM

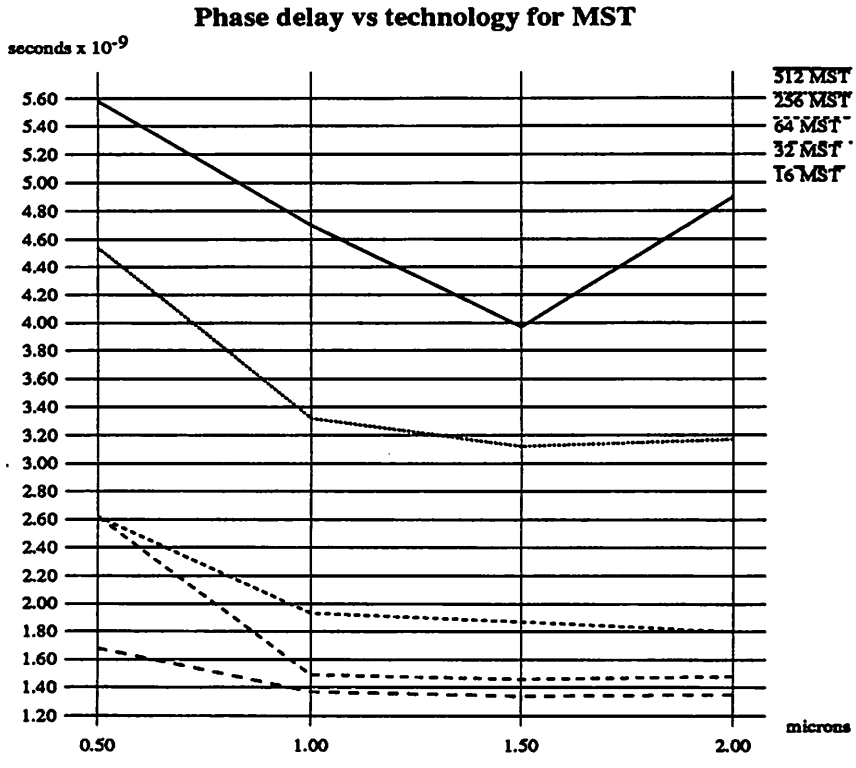

Figure 24: Phase delay vs feature size for MST

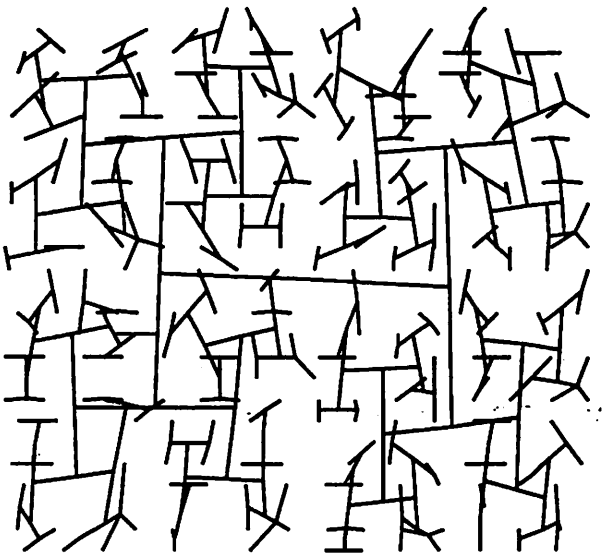

Figure 25: Clock tree for example Primary1

Phase delay is plotted against feature size in Figures 23 and 24. As geometries are reduced, the phase delay worsens for both MMM and MST. However, for all feature sizes the phase delay generated by MMM is less than MST in all examples with a comparable number of clock pins.

Figure 25 and 26 show MMM's routing results for the MCNC Primary1 and Primary2 benchmarks respectively. The skew introduced for each of these examples was 31 ps and 260 ps respectively. Primary1 had 269 clock pins and Primary2 had 603 clock pins. Both placements were obtained using PROUD [TKH88]. It is interesting to note that Primary2's placement exhibited an asymmetric clock pin distribution while Primary1's remained relatively uniform. However, the asymmetry was not enough to deter MMM from yielding excellent results. Figures 27 and 28 show the voltage waveforms at the furthest and closest pins from the clock driver for Primary1 when routed using MST and MMM respectively. The skew introduced by MST was 4.7 ns, and the routing to the furthest point was so poor (in terms of timing behavior) that the pin was unable to charge to the supply voltage. As mentioned the skew generated by MMM is 31 ps and is barely visible in Figure 27.

7 Conclusions and Future Work

We have presented an approach to clock routing that is clearly superior to simple minded clock routing based on a minimum spanning tree. While high-performance industrial designs are unlikely to have clock routing performed using such a simple approach as MST, the quality of the results generated by MMM are exceptional. The approach has all but eliminated clock skew and yielded excellent phase delay results for widely ranging chip sizes, net sizes (pin count), technologies, and pin distributions on both randomly created and industrial benchmarks.

Future work will address clock tree buffer optimization and blockage and congestion consideration

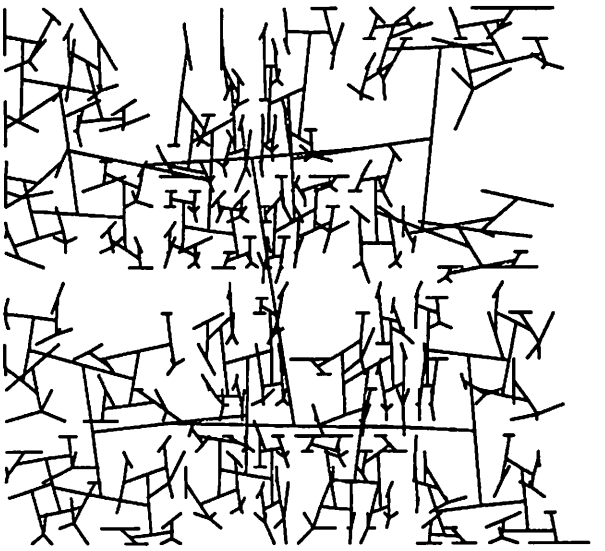

Figure 26: Clock tree for example Primary2

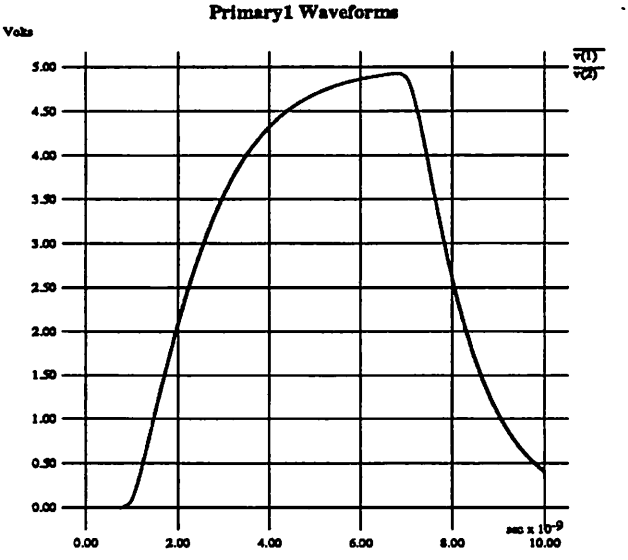

Figure 27: Clock waveforms for Primary1(MMM)

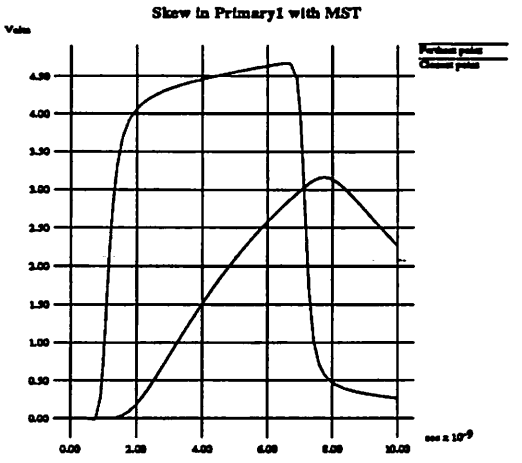

Figure 28: Clock waveforms for example Primary1 routed with MST

during the growth of the clock tree. Additionally, the impact of the approach on wirability and chip area will be investigated.

References

[BBB+89] S. Boon, S. Butler, R. Byrne, B. Setering, M. Casalanda, and Al Scherf. High performance clock distribution for cmos asic's. *IEEE Custom Integrated Circuit Conference*, pages 15.4.1–15.4.4, 1989.

[BWM86] H. B. 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. *IEEE Int. Conference on Computer Design: VLSI in Computers and Processors (ICCD-86)*, pages 118–122, October 1986.

[CH79] F. K. Chung and F. K. Hwang. The largest minimal rectilinear steiner trees for a set of n points enclosed in a rectangle with given perimeter. *Networks*, 9(1):19–36, Spring 1979.

[DFW84] S. Dhar, M. A. Franklin, and D. F. Wann. Reduction of clock delays in vlsi structures. *IEEE Int. Conference on Computer Design: VLSI in computers (ICCD)*, pages 778–783, 1984.

[DS80] R. L. M. Dang and N. Shigyo. A two-dimensional simulation of lsi interconnect capacitance. *IEEE Electron Device Letters*, EDL-2:196–197, August 1980.

[Elm48] W. C. Elmore. The transient response of damped linear networks with particular regard to wideband amplifiers. *Journal of Applied Physics*, 19(1):55–63, January 1948.

[FK82] A. L. Fisher and H. T. Kung. Synchronizing large systolic arrays. *Proceedings of SPIE*, 341:44–52, May 1982.

[Han66] M. Hanan. On steiner's problem with rectilinear distances. *SIAM Journal of Applied Math*, 14:255–265, 1966.

[KGE82] S. Y. Kung and R. J. Gal-Ezer. Synchronous versus asynchronous computation in vlsi array processors. *Proceedings of SPIE*, pages 53–65, May 1982.

[Ko89] Ping Keung Ko. 1989. Private Communication.

[MC80] Carver A. Mead and Lynn A. Conway. *Introduction to VLSI Systems*. Addison-Wesley, Reading, Massachusetts, 1980.

[Mij87] D. Mijuskovic. Clock distribution in application specific integrated circuits. *Microelectronics Journal*, 18(4):15–27, 1987.

[Nag75] W. Nagel. Spice2, a computer program to simulate semiconductor circuits. *University of California, Berkeley, Memo No. ERL-M520*, May 1975.

[RPH83] Jorge Rubinstein, Paul Penfield, and Mark A. Horowitz. Signal delays in rc tree networks. *IEEE Trans. Computer-Aided Design*, CAD-2:202–211, July 1983.

[TKH88] R. S. Tsay, E. S. Kuh, and C. P. Hsu. Proud: A sea-of-gates placement algorithm. *IEEE Design and Test of Computers*, pages 318–323, December 1988.