Clock Routing for High-Performance ICs

Michael A. B. Jackson Arvind Srinivasan E. S. Kuh
Electronics Research Laboratory, University of California, Berkeley, CA 94720

Abstract

In this paper we focus on routing techniques for optimizing clock signals in *small-cell* (e.g., standard-cell, sea-of gate, etc...) ASICs. In previously reported work, the routing of the clock net has been performed using ordinary global routing techniques based on a minimum spanning or minimal Steiner tree that have little understanding of clock routing problems. We present a novel approach to clock routing that all but eliminates clock skew and yields excellent phase delay results for a wide range of chip sizes, net sizes (pin count), minimum feature sizes, and pin distributions on both randomly created and standard industrial benchmarks. For certain classes of pin distributions we have proven theoretically and observed experimentally a decrease in skew with an increase in net size. In practice, we have observed a two to three order magnitude reduction in skew when compared to a minimum rectilinear spanning tree.

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, 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 includes clock distribution logic and interconnects. It serves to unify the physical and temporal design representations by determining the precise instants in time that 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 either of two phenomena: the routing to the chip's synchronizing elements, or in the non-symmetric 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] [WF83]. For regular structures such as systolic arrays the H-tree works well to reduce skew, but in the general case asymmetric distributions of clock pins are common and the H-tree is not as effective 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. More recently, [BWM86] has provided an analysis of the clock lines that considers the transmission line properties of the clock net. [BBB+89] have presented an approach for ASIC clock distribution that integrates buffer optimization into place and route algorithms, while [RS89] and [FP86] have presented approaches that consider macrocell clock distribution. However, in all previous work the routing of the clock net is performed using ordinary routing techniques. This causes non-optimal clock behavior and as region size or the number of pins in the net increases, the undesirable behavior is exacerbated. In this paper, we focus exclusively on routing techniques for optimizing the clock signal in VLSI circuits. We demonstrate the superiority of our algorithm over standard routing techniques for examples of a wide range of ranging size.

In section two the preliminaries necessary for understanding the paper are presented. Following this, the problem is defined in section three. Section four illustrates the algorithm for clock routing and section five discusses theoretical results. Next, in section six, practical considerations are discussed. In section seven the experimental results are presented, and in section eight 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 are often modeled as a Moore finite state machine for the purposes of analysis. The topological requirement imposed on such a finite state machine is that all closed signal paths must contain at least one synchronizing element. Satisfaction of this constraint has several benefits, two of which are: the assurance of deterministic behavior if the physical aspects of the design are correct, and the elmination of 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 be edge-triggered. Furthermore, let CP denote the clock period, d_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

$$CP \geq d_L + t_{SKEW} + t_{SU} + t_{CQ} \qquad (1)$$

This expression demonstrates the important relationship between the clock period, the longest path delay, and the clock skew.

The two timing related clock parameters that one must consider for high-performance design are clock skew and phase delay. Clock skew is defined to be the

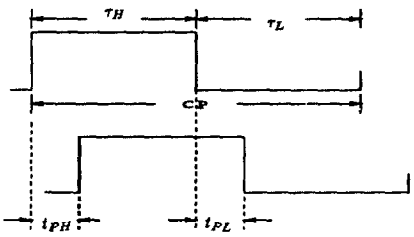

Figure 1: Relationship between τ_H, τ_L, and CP with 50 % duty cycle

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 will be presented. Phase delay may be defined to be the maximum delay to any synchronizing clock pin. The same phenomena causing skew also contributes to phase delay. It is convenient to consider phase delay to consist of two components: an intrinsic cell 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 phase delay may be defined as

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

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

Figure 1 illustrates equations 2 and 3. Phase delay affects chip to chip interfaces by appearing as inter-chip skew and in worst-case scenarios may provide inadequate time to charge and discharge the clock net. For example, for a clock signal with a duty cycle of 50 %, the high and low portions of the clock period τ_H and τ_L, must satisfy the following constraints

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

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

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 is necessary to make provisions for minimizing phase delay.

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 be easily extended to the case of multiple clocks by treating the clock nets independently. When multiple clocks are present, inter-clock and intra-clock skew and phase delay must be considered. Independent treatment of the different clock nets will address intra-clock problems by minimizing the phase delay of each clock net and minimizing the skew between similarly clocked synchronizing elements. Inter-clock phase delay is minimized since intra-clock phase delay is minimized. While inter-clock skew is not explicitly minimized, should it be a problem, circuit techniques that insert delay to equalize arrival times across different clock nets will ameliorate the problem. 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. Traditional ASIC design proceeds from logic to physical design. Physical design consists of three classical steps: placement, global routing, and detailed routing. In our proposed design flow a clock routing step is interposed between placement and global routing. 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 determined routes are constructed so that the clock signal behavior is optimized.

To understand the consequences of decisions made during physical design, one must model the interconnect parasitics that load the clock tree. We make the reasonable assumption that in a high-performance design environment, the interconnects are realized with aluminum due to its excellent conducting 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 given by

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

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 permittivity of the oxide. All lengths used to calculate resistance and capacitance are based on manhattan distances.

Based on estimates for R_{int} and C_{int}, simple and accurate interconnect delay estimates may be calculated using 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 the rectilinear segments comprising the interconnect are modeled using their equivalent T-network representation.

3 Problem Definition

Given the ICs 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 and yields 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 couple (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 sub-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. 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 on 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. 2.

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.

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 2: Pseudo-code for the basic algorithm

Delay equalization look-ahead

Consider the example shown in Figure 3, 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 3(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 3(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

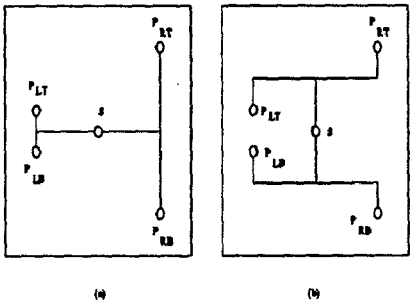

Figure 3: Clock tree (a) without look-ahead, (b) with look-ahead

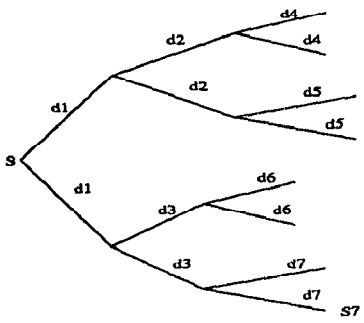

Figure 4: Clock tree representation used for delay calculations

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.

Delay Calculation

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 4. 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

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

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. For a detailed analysis of the complexity, the interested reader is referred to [JSK90]. 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 state some key results that motivate the Method of Means and Medians. For the sake of brevity we have omitted proofs of the propositions. A detailed treatment the proofs leading to the results can be found in [JSK90]. The first result is 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 which claims that increasing the number of points within a region reduces the skew. Finally, we show 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 7).

Theorem 5.1
Given a set of points $S = \{s_1, s_2, \ldots, s_n\}$, where n is an even integer,

$$\begin{aligned}|\, x_c(S) - x_c(S_L(S)) \,| + |\, y_c(S) - y_c(S_L(S)) \,| = \\ |\, x_c(S) - x_c(S_R(S)) \,| + |\, y_c(S) - y_c(S_R(S)) \,|\end{aligned}$$

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 are always equal. Note that as we move deeper into the clock tree, the segments become shorter. 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 is an integer ≥ 1, within a region of side 1.0 unit, the total wirelength of the tree produced by the basic algorithm grows as $\frac{3}{2}\sqrt{n}$.

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.

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

We conjecture that the worst-case wirelength for the Method of Means and Medians is $\frac{9}{4}\sqrt{n} - \frac{3}{2}$ [JSK90]. 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 $\frac{1}{\sqrt{p}}$.

This result indicates that as the number of points within the region is increased, the skew between the endpoints is reduced. 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.

The algorithm is 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 when running the algorithm on practical examples.

6 Practical Considerations

In practice one would like to route the clock net with minimum wirelength while satisfying a prespecified tol-

erance on clock skew and phase delay. We have developed a *hybrid* clock routing algorithm that performs MMM to a certain depth, i.e., until the chip has been divided into a number of regions each containing less than a certain number of clock pins. Then standard routing techniques are applied to each of the remaining sub-regions. The depth at which the transition from MMM to standard techniques occurs and the number of pins within each sub-region that are routed using standard methods are a function of the amount of skew tolerable.

Another practical concern is the degradation in the clock waveform. Usually, buffers are inserted into the clock tree to regenerate the clock waveform. We propose two strategies to deal with the placement of buffer cells. The first is to pre-place buffers at symmetric locations on the chip that coincide with expected locations of the centers of mass of the sub-regions to be driven by the buffers. Then, during the clock routing, detours are made to the pre-placed buffers so that they may drive the clock pins. These buffers then act as centers from which a clock tree is grown using the Method of Means and Medians. The second strategy is to insert buffers after placement at optimal locations determined by the Method of Means and Medians. The expected perturbation to the placement would be small considering the size and number of buffers relative to the total number of cells.

7 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 uniform pin distributions in a square region. For the twenty examples, four equal-sized chips with 16, 32, 64, 256 or 512 pins were generated. For comparative purposes, we routed the same pin distributions using a minimum rectilinear spanning tree (MST) algorithm. As shown in [Han66], the ratio of the length of a minimum rectilinear spanning tree and an optimal rectilinear Steiner tree is bounded by a factor of $\frac{3}{2}$. 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.

We compared the skew, phase delay and wirelength as a function of the number of pins for MMM and MST. Additional experiments comparing skew as a function of chip size and minimum feature size may be found in [JSK90]. An extended evaluation of the results of all experiments is also given in [JSK90].

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 5. Interestingly, the skew decreased with increasing number of pins for MMM and grew linearly for MST. 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 sublinear and MST approximating linear growth. These results can be seen in Figure 6.

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 7. The experimental results corroborate the theoretical $\sqrt{n}$ relationship between wirelength and number of points

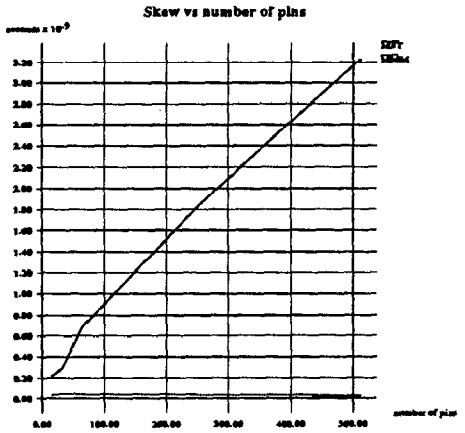

Figure 5: Skew versus no. of pins

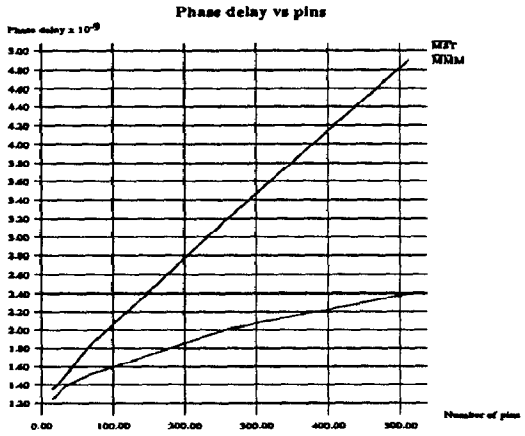

Figure 6: Phase delay versus no. of pins

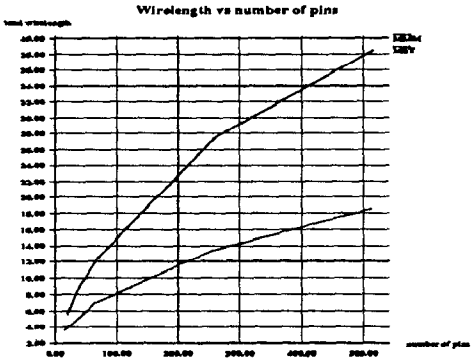

Figure 7: Wirelength versus no. of pins

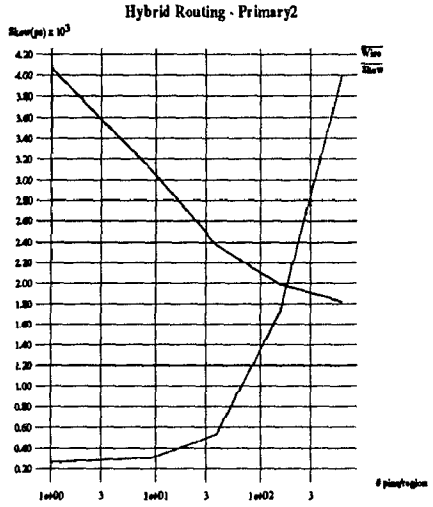

Figure 8: Hybrid routing for Primary2

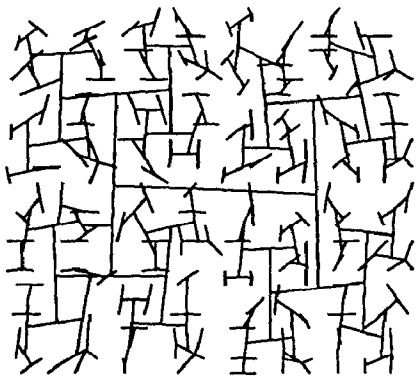

Figure 9: Clock tree for example Primary1

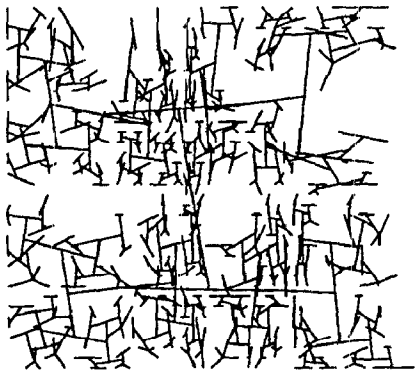

Figure 10: Clock tree for example Primary2

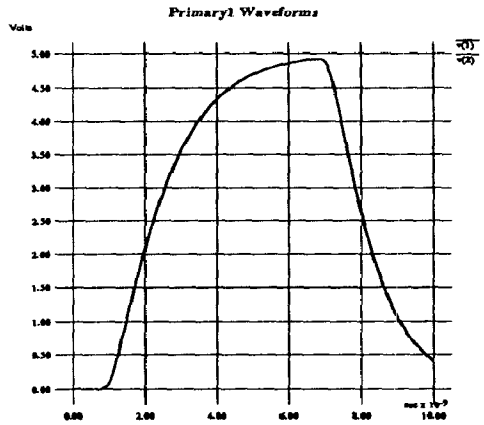

Figure 11: Clock waveforms for Primary1(MMM)

n with the difference appearing as a constant factor. Thus, improvements in clock behavior are accompanied by an increase in the clock net's wirelength.

Figure 8 shows the results of running the hybrid algorithm to varying depths on the MCNC benchmark chip Primary2. On the x-axis we have plotted the number of pins in each region that were routed using standard techniques. The origin of the x-axis corresponds to routing using MMM for all the pins. The rightmost point on the axis corresponds to routing the clock net using a minimum spanning tree. Thus the depth to which MMM was applied decreases as we move towards the right of the figure. The solid line shows decreasing wirelength (normalized) while the dotted line shows increasing skew with decreasing depth. This provides the designer with the opportunity to arrive at a compromise between the excellent skew of MMM and the low wirelength of a minimum spanning tree.

Figure 9 and 10 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 11 and 12 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. Note that the skew generated by MMM is 31 ps and is barely distinguishable in Figure 11.

8 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 a wide range 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 give consideration to blockages and routing congestion during the growth of the clock tree. Ad-

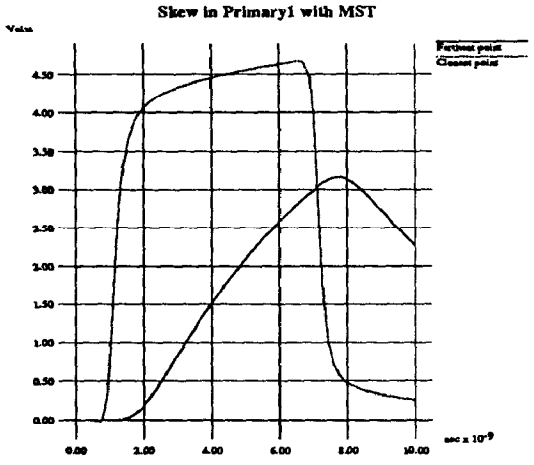

Figure 12: Clock waveforms for Primary1(MST)

ditionally, the impact of the approach on wirability and chip area will be investigated.

[Fis89] has shown that clock skew may be used to decrease the clock period of a system by introducing relative delays between the arrival times of the clock signal at the clocked pins. We are considering techniques to implement this idea.

Acknowledgements

This work was supported by SRC Grant 90-DC-008 and JSEP Grant F49620-87-C-0041.

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.

[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.

[Fis89] J.P. Fishburn. 1989. Private Communication.

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

[FP86] E. G. Friedman and S. Powell. Design and analysis of a hierarchical clock distribution system for synchronous standard cell/macrocell vlsi. *IEEE Journal of Solid-State Circuits*, SC-21(2):240–246, 1986.

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

[JSK90] Michael Jackson, Arvind Srinivasan, and E.S. Kuh. Clock routing methodologies. Technical report, University of California at Berkeley, 1990. Memo ERL.

[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.

[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.

[RS89] P. Ramanathan and K. G. Shin. A clock distribution scheme for non-symmetric vlsi circuits. *IEEE Int. Conference on Computer-Aided Design (ICCAD-89)*, pages 398–401, November 1989.

[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.

[WF83] D. F. Wann and M. A. Franklin. Asynchronous and clocked control structures for vlsi based interconnection networks. *IEEE Transactions on Computers*, C-32(3):284–293, March 1983.