On Clock Routing Techniques for VLSI Synchronous Systems

Wasim A. Khan
Western Michigan University

Follow this and additional works at: http://scholarworks.wmich.edu/masters_theses
Part of the Computer Sciences Commons

Recommended Citation
http://scholarworks.wmich.edu/masters_theses/837
ON CLOCK ROUTING TECHNIQUES FOR VLSI SYNCHRONOUS SYSTEMS

by

Wasim A. Khan

A Thesis
Submitted to the
Faculty of The Graduate College
in partial fulfillment of the
requirements for the
Degree of Master of Science
Department of Computer Science

Western Michigan University
Kalamazoo, Michigan
June 1993
The clock signal is vital in maintaining proper dataflow in a synchronous system and thus the total throughput of a high performance system depends on the frequency of the clock. The clock frequency is determined mainly by clock skew, clock delay, planarity and total wire length of the clock tree. As a result, clock routing has received significant attention in recent years. Most of the research has concentrated on elimination of clock skew while minimizing the total wirelength of the clock tree. The problem of maintaining planarity and minimizing clock delay has received little or no attention.

In this thesis, we develop a clock distribution scheme for high performance systems which maximizes the operating clock frequency. We develop an algorithm which routes a planar clock tree with zero skew, minimum source to sink pathlength, and minimal total wirelength. The algorithm also provides a smooth tradeoff between maximum source to sink pathlength and total wirelength while keeping the clock skew at zero.

In many microprocessor designs, multi-phase clocks are used for improved system design. The problem of routing multiple clock is more complicated than a single clock because in multiple clock systems we need to minimize not only the clock skew within a clock but we also minimize cross skew between different clocks. In this thesis, we present a clock routing technique for routing two phase clock.
ACKNOWLEDGMENTS

First and foremost, I would like to express my gratitude to my advisor, Professor Naveed A. Sherwani, from whom I have learned many valuable skills, both research and otherwise. His consistent support and accessibility to his students, often in the early morning hours, has encouraged me in particular, and ‘nitegroup’ in general. Our professional working relationship has been so productive and enjoyable that I expect it to continue for many years to come.

Professor Donald Nelson, the Department Chair, has my gratitude for his continued support in providing research facilities that has been a great help in finishing this thesis work and conducting other research.

I am grateful to the members of my thesis committee, Professor Alfred Boals and Professor Ajay Gupta, who accepted this additional task with high enthusiasm.

I would like to thank the ‘nitegroup’ members, Jahangir Hashmi, Moazzem Hossain, Sidharth Bhirarde, Arun Shanbhag, and Qiong Yu, for their constant help, cooperation, and friendship. I would also like to thank all my friends at WMU, who, in one way or the other, made my study here an enjoyable one.

Special thanks to Suzanne Moorian and Phyllis Wolf from Computer Science Department, who helped me in many administrative situations: their pleasant characters and helpful personalities are assets to us all.

Finally, I thank my parents for their encouragement and guidance, which has always helped me accomplishing my goals.

Wasim A. Khan
INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps. Each original is also photographed in one exposure and is included in reduced form at the back of the book.

Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6" x 9" black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.
On clock routing techniques for VLSI synchronous systems

Khan, Wasim A., M.S.
Western Michigan University, 1993
To
My Parents
# TABLE OF CONTENTS

ACKNOWLEDGMENTS ................................................. ii  
LIST OF TABLES ...................................................... iv  
LIST OF FIGURES ....................................................... v  
I. INTRODUCTION ..................................................... 1  
   1.1 Clock Delay in Unbuffered Clock Trees ......................... 4  
   1.2 Clock Delay in Buffered Clock Trees ......................... 5  
II. MINIMUM DELAY ZERO SKEW PLANAR CLOCK ROUTING .... 7  
   2.1 Overview of the Algorithm ..................................... 7  
   2.2 Details of Algorithm k-PCT ................................... 9  
III. MULTIPLE PHASE CLOCK ROUTING ........................... 20  
   3.1 Algorithm for Uniformly Distributed Clock Pairs .......... 25  
IV. EXPERIMENTAL RESULTS .................................... 31  
V. CONCLUSION ...................................................... 33  
REFERENCES ......................................................... 34
### LIST OF TABLES

1. Result of Algorithm $k$-PCR for Randomly Generated Examples . . . 33

2. Comparison of $k$-PCR with the Existing Algorithms . . . . . . . . . . . 34
LIST OF FIGURES

1. (a) A Clock Tree Generated by RGM, (b) An Optimal Clock Tree. ........................................ 3
2. Clock Layouts for Different Values of $k$. ........................................................................ 11
3. A Bounding Box in Rectilinear Geometry. .................................................................. 12
4. Effect of $k$ on Total Wirelength $\Gamma$. ...................................................................... 14
5. Algorithm $k$-PCT for Eight Clock Pins. .................................................................. 15
6. Wirelength Minimization. ......................................................................................... 16
7. A Separable Clock Tree. ......................................................................................... 16
8. Steiner Wirelength Minimization. .............................................................. 17
9. (a) The Single Phase Clocking. (b) The Two-phase Clocking. .......................... 21
10. (a) A Clock Buffer. (b) An Equivalent Model. ................................................. 24
11. Example of Layout Generated by Algorithm TCR. ......................................... 30
12. Layout for Primary 2, Generated by $k$-PCT ($k$) for $k=3$. ................................ 31
CHAPTER I

INTRODUCTION

Over the past few years, clock routing has gained much attention and extensive research on clock skew optimization has led to significant development [2, 5, 9, 11, 13]. The major objective is to achieve the design objective of running the chips at highest possible clock frequency. The clock frequency depends on four factors: the clock skew, maximum delay in the clock tree, the total wirelength, and the planarity of the clock tree. The clock skew is caused by the difference in the arrival times of the clock signal at different circuit elements. Clock skew reduces the clock frequency since clock period must be increased to account for late arrival of clock signal at certain circuit elements. The maximum delay in the clock tree also plays a key role in determining the clock frequency, since clock signal must arrive at all unbuffered segments of the clock tree within the clock period. Thus, the delay in the longest source-to-sink path forms a lower bound on the clock period. The maximum delay in the clock is increasing due to increased chip size and higher RC constant due to thinner metal lines. Total wirelength also plays a major role in determining the clock frequency. Larger total wirelength leads to higher loading capacitance at the clock driver. Finally, Planarity is another factor which effects the clock frequency. Planarity of clock tree guarantees that the clock can be routed in the same metal layer thereby maintaining the uniformity of the electrical parameters. All previous researches considered each of these factors independently.

Routing a clock is based on the tradeoff between wirelength and longest path while keeping the zero skew in the planar clock tree. Jackson, Srinivasan, and Kuh [9] presented a clock routing algorithm for circuits with small cells. Their algorithm recursively partitions a circuit into two equal parts and then connects
Cong, Kahng, and Robins [2] presented a binary tree based routing scheme. In this approach clock routing is achieved by constructing binary trees using recursive geometric matching. All of these schemes attempt to minimize the skew and total wirelength, while ignoring the clock delay caused by the longest source-to-sink path in the clock tree. The clock delay is, however, equally important in determining the clock frequency. The clock trees with longer paths have larger clock periods and, therefore, operate on lower frequency. For example, consider the clock tree, as shown in Figure 1(a). This clock tree, generated by recursive geometric matching-based (RGM) algorithm [2], has zero skew but the maximum delay is 7 units. Figure 1(b) shows another clock tree for the same example and yields zero skew but the maximum delay is 5 units. Also note that the total wirelength in Figure 1(a) is 27 units as compared to that in Figure 1(b), which is just 18 units. In addition to the wirelength, the schemes listed above do not achieve planar routing. It is extremely difficult to achieve zero skew in a non-planar clock tree, because of the non-uniform electrical parameters in different layers. Recently, Dai and Zhu [11] has proposed a new scheme which routes the clock with zero skew in a planar fashion and minimizes the longest source to sink path. However, this result is obtained at the cost of large wirelength. Thus we need to develop an algorithm which minimizes total wirelength, minimizes longest path, and completes the routing in a planar fashion with zero skew. In this thesis, we present a clock routing algorithm which routes the clock in planar fashion with minimum clock delay.

The clock signal is generated external to the chip and provided to the chip through the clock entry point. Each functional unit which needs the clock is interconnected to the clock entry point by the clock net. Each functional unit performs a series of logic functions and waits for the clock signal to pass its results to another unit before the next processing cycle. The clock controls the flow of...
information within the system. In high performance systems, the clock frequency depends not only on skew but also on the time to communicate between functional elements and the time to distribute a clock signal over all elements. The clock distribution network is usually represented as a rooted tree, called a clock tree. Therefore, the clock period is determined by the maximum difference in the arrival times at two different leaves \((t_{\text{skew}})\), the time to distribute the clock signal in clock tree \((d_{\text{max}})\) and the time taken by one leaf to propagate its output to another leaf \((\Delta)\). Therefore, for a high performance system, the clock period is determined by

\[ t_p \geq \max(d_{\text{max}}, \Delta, t_{\text{skew}}) \]  

(1.1)

For a combinational logic, however, the clock period, \(t_p\), should satisfy the following inequality [6]

\[ t_p \geq t_d + t_{\text{skew}} + t_{\text{su}} + t_{\text{ds}} \]  

(1.2)

where \(t_d\) is the delay on the longest path through combinational logic, \(t_{\text{skew}}\) is the clock skew, \(t_{\text{su}}\) is the set up time of the synchronizing elements (assuming

Figure 1. (a) A Clock Tree Generated by RGM, (b) An Optimal Clock Tree.
that synchronizing elements are edge-triggered), and \( t_{ds} \) is the propagation delay within the synchronizing elements.

The communication time between two functional elements is inversely proportional to the feature size. However, the time to distribute a clock event in a clock tree, which is determined by the longest path (longest root-to-leaf path) in the clock tree, should be given due consideration in high performance system. This is due to the fact that as the chip size increases, critical nets like clock nets become more critical. Longer clock nets lead to large RC delay which causes in low rise time and clock pulse fails to attain sufficient amplitude at high frequency. Therefore, the length of the longest path determines the clock period and hence the frequency. To overcome this problem, two major approaches have been used. One, referred to as structural approach, uses higher level design methodologies (such as systolic arrays [7]). The other, referred to as circuit design approach, improves the delay by inserting buffers in clock lines. The clock delay minimization is necessary in both the unbuffered clock trees and buffered clock trees, as discussed below.

### 1.1 Clock Delay in Unbuffered Clock Trees

As mentioned above, since the length of the longest path in a clock tree determines the system performance, it is necessary to minimize the longest path. If the clock tree has zero skew and is unbuffered, the minimum clock period is determined by the delay in the longest path.

\[
\frac{1}{t_p} \geq \frac{1}{d_{max}} \geq l \times r \times c
\]

where \( l \) is the length of the longest path, \( r \) is the per unit resistance and \( c \) is the per unit capacitance. Thus, in order to maximize the frequency, length of the longest path has to be minimized. The issue of longest path is important in large designs because it not only contributes to the larger clock delays but also causes
some electrical problems due to damping and reflection phenomena. In today's
chips of 25 x 25 mm, the longest path becomes very critical.

1.2 Clock Delay in Buffered Clock Trees

In case of large designs, it becomes difficult to drive the clock lines at the
desired speed, because of the time needed to bring long wires to an equipotential
state. In such cases, long clock lines are buffered. Buffer insertion in clock lines can
be realized in two ways. One way is to insert equally spaced minimum-size buffers
in long clock lines. Another way is to insert a chain of buffers called as cascaded
buffers. The buffers restore the clock signal and prevent noise propagation. At the
same time buffer occupy area and consume significant power (and hence dissipate
heat). It is, therefore, desirable that the number of buffers, $w$, placed in clock line
be minimum. If buffers are equally placed in the clock lines, the clock period is
given by

$$t_p = \frac{d_{\text{max}}}{w}$$

(1.3)

The number of buffers in a clock tree can, therefore, be minimized by minimizing
the length of the longest path in the clock tree.

Before we present our proposed clock routing algorithm, let us formulate
the clock routing problem.

**Definition 1** Given a clock source, $s_0$, and a set of clock terminals (sinks), $S =
\{s_1, s_2, \ldots, s_n\}$. A c-path from clock source $s_0$ to a clock terminal $s_i$, denoted as
$d_c(s_0, s_i)$, is a sequence of edges in a steiner tree.

$$d_c(s_0, s_i) = e(s_0, p_1) + \sum_{j=1}^{i-1} e(p_j, p_{j+1}) + e(p_i, s_i)$$

Where $e(p_j, p_{j+1})$ is an edge from steiner point $p_j$ to $p_{j+1}$, which lie on the
unique path $d_c(s_0, s_i)$.
The cost of a c-path is the sum of the lengths of all intermediate edges which lie on a unique c-path. The cost or delay in a c-path is denoted as \( \text{delay}(s_0, s_i) \) and is given as:

\[
\text{delay}(s_0, s_i) = \text{delay}(s_0, p_1) + \sum_{j=1}^{i-1} \text{delay}(p_j, p_{j+1}) + \text{delay}(p_l, s_i)
\]

The total cost (or total delay), \( \Delta T \), in a clock tree \( T \) is given as:

\[
\Delta T = \sum_{i=1}^{n} \text{delay}(s_0, s_i)
\]

**Definition 2** A clock tree \( T \) rooted at clock source \( s_0 \) is a tuple of c-paths from \( s_0 \) to each \( s_i \), \( 1 \leq i \leq n \). The rectilinear distance from \( s_0 \) to each \( s_i \) is denoted as \( \text{dist}(s_0, s_i) \). Then the radius of clock tree, \( \sigma_T \), is defined as

\[
\sigma_T = \max(\text{dist}(s_0, s_i), \ 1 \leq i \leq n)
\]

**Definition 3** Given a clock source \( s_0 \) and a set of clock terminals, \( S = \{s_1, s_2, \ldots, s_n\} \). The Planar Clock Routing Problem (PCRP) is to find a planar clock tree such that

1. \( \sigma_T \) is minimized.
2. \( \text{delay}(s_0, s_i) = \sigma_T \)
3. \( \Delta T \) is minimized.

**Definition 4** Given a set of clock sinks (terminals), \( S = \{s_1, s_2, \ldots, s_n\} \), each clock terminal is defined by a unique xy coordinate, \( s_i = (x_i, y_i) \). A bounding box, \( B \) is formed such that

\[
x_1 = \min(x_i) \quad 1 \leq i \leq n \quad \& \quad y_1 = \min(y_i) \quad 1 \leq i \leq n
\]

\[
x_2 = \max(x_i) \quad 1 \leq i \leq n \quad \& \quad x_2 = \max(y_i) \quad 1 \leq i \leq n
\]

\( B \) is then defined by \( (x_1, y_1) \) & \( (x_2, y_2) \). The center of the bounding box is the clock source, \( s_0 \). The clock terminal, which is farthest from \( s_0 \) determines \( \sigma_T \).
CHAPTER II
MINIMUM DELAY ZERO SKEW PLANAR CLOCK ROUTING

As mentioned earlier, non-planar clock trees are routed in different metal layer which makes clock skew and longest source-to-sink path worse. The existing approaches for routing clock nets can be classified as: the bottom-up approach and the top-down approach. Both approaches achieve zero skew, but the layouts obtained by each of the approaches are different in terms of wirelength and longest source-to-sink path. The bottom up approach generates clock layouts with minimal total wirelength, but the layout is non-planar and the longest source-to-sink path is more. In top down approach, on the other hand, the clock layouts have shorter maximum source-to-sink path, are planar, but have large total wirelengths. Therefore, our approach is a Combined Bottom-up And Top-down (COMBAT) approach. The COMBAT approach achieves zero skew planar clock routing with minimal wirelength and shortest maximum path. Following is an overview of the proposed algorithm.

2.1 Overview of the Algorithm

In this section, we present an overview of the proposed algorithm for zero skew planar clock routing with minimal delay and reduced wirelength. The proposed algorithm has four key phases: (1) Defining and routing the regions (Inter-region routing), (2) Routing each region to get planar clock tree, (3) Steiner wirelength minimization, and (4) Delay reduction by inserting buffers.

In the first phase, we partition the region into different regions. The partitioning of the plane is achieved by the horizontal-vertical (H-V) partitioning. The layout area is partitioned into $2^k$ different regions, for any value of $k$, $1 \leq k \leq \log n$ (typically, $k$ ranges from 1 to 7). The second phase deals with the planar routing
in each region. After all regions are routed in a planar fashion, we further mini-
mimize the total wirelength in the third phase. The third phase deals with merging
of leaves of the clock tree produced in second phase. Finally, in the fourth phase,
we minimize $\sigma_T$ - the maximum clock delay, by inserting buffers in the clock tree.
The formal algorithm $k$-Planar Clock Tree, denoted as $k$-PCT, is given as follows.
The details of the algorithm are discussed in the subsequent sections.

Algorithm $k$-PCT()

\textbf{Input:} A set of clock sinks (terminals), $S = \{s_1, s_2, \ldots, s_n\}$ and $k$

\textbf{Output:} Planar clock tree $T$ with minimal total wirelength

\begin{algorithm}
\textbf{Begin Algorithm}

$B \leftarrow \text{Find\_Bounding\_Box}(S)$ /* $B$ is bounding box */

set $d$ to $x$-direction /* $d$ represents $x$ or $y$ direction.

for $k = 1$ to $\log(n)$

$R \leftarrow \text{Define\_Regions}(S, k, d)$; /* $R = \{R_1, R_2, \ldots, R_m\}$ */

for each $R_i \in R$ do

$T_i = \text{Planar\_Wire\_Min}(R_i, \sigma_T)$;

output $T^k$;

select $k$ such that $\Gamma_{T^k} = \min(\Gamma_{T_l}), \quad 1 \leq l \leq \log(\sigma_T)$

for each $R_i \in R$ do

if $T_i$ is non-separable

$T_i' \leftarrow \text{Steiner\_Wire\_Min}(T_i)$;

\textbf{Buffer\_Insertion}(T);

output $T$;

\textbf{End Algorithm}

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
2.2 Details of Algorithm $k$-PCT

In this section, we present details of our proposed algorithm $k$-PCT. The details include the description of the key procedures involved in the algorithm $k$-PCT.

2.2.1 Defining and Routing the Regions

The procedure *DefineRegions* partitions the layout area into different regions i.e. \( R = \{R_1, R_2, \ldots, R_m\} \), where \( 1 \leq m \leq 2^k \). Note that the bounding box \( B \) is defined by the farthest point in the region, therefore, any clock terminal which resides in the bounding box, can always achieve \( \sigma_T \) (Theorem 2). Routing to the region centers is performed by constructing an *H-tree*, recursively. Let \( k \) be the number of levels of H-tree. Then, the total number of regions formed by H-V partitioning is \( 2^k \). Each region \( R_i \) has a leaf of H-tree as a center \((x'_c, y'_c)\) to that region. Symmetric nature of H-tree guarantees zero skew clock routing upto the centers of each of the regions. Let \( S = \{s_1, s_2, \ldots, s_n\} \) be the set of clock sinks. Each clock sink \( s_i \) is represented by \((x_i, y_i)\). Assume that \( S \) is first divided into \( R_1 \) and \( R_2 \) in \( x \)-direction. The center of \( R_1 \) and \( R_2 \) are then routed. The regions \( R_1 \) and \( R_2 \) are then recursively split in \( y \)-direction (the direction orthogonal to the previous). This procedure is continued \( k \) times. The pseudo-code of the procedure is shown below.

/* Procedure DefineRegions(S, k, d) */

*Input:* A set of clock sinks (terminals), \( S \), \( k \), and \( d \)

*Output:* \( R = \{R_1, R_2, \ldots, R_m\} \), where \( 1 \leq m \leq 2^k \).

\( R=\emptyset; \)

*Begin Procedure*

if \( k \leq 0 \) return(\( R \));

else  \quad \( x_0 = x_c(S); \quad y_0 = y_c(S); \)

\( x_1 = x^1_c(R_1); \quad y_1 = y^1_c(R_1); \)

\*Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.*
\[ x_2 = x_2^2(R_1); \quad y_2 = y_2^2(R_2); \]
connect \((x_0, y_0)\) with \((x_1, y_1)\) and \((x_2, y_2)\)
\(k = k - 1;\)
\(d = \bar{d};\) /* the direction reversed */
Define_Region\((S, k, d)\)

End Procedure

Lemma 1 The procedure Define_Regions runs in \(O(2^k)\) time, where \(k\) is user-defined number of levels.

Consider an example shown in Figure 2. The layout is shown for different values of \(k\). When \(k\) is set at 0, we get the layout as shown in Figure 2(b). The planar routing is performed as shown in Figure 2(c). If we set \(k=1\), we get two \((2^k)\) regions (Figure 2(d)) and each region has an X-tree. The final layout achieved after planar routing is shown in Figure 2(f)). Note that the algorithm reported in [11] is a special case of our algorithm for \(k=0\).

2.2.2 Planar Wirelength Minimization Within a Region

The output of procedure Define_Regions is \(2^k\) regions routed in an H-tree structure. In this way, we guarantee that the clock signal reaches to each of the regions at the same time. The H-tree is constructed in a bottom-up fashion and in each region, we perform planar routing in a top-down fashion such that the maximum delay \(\sigma_\tau\) is achieved and all paths have equal delay to the regional center. Let \(d_i\) be the delay from the clock source to the regional center of \(R_i\). As a result, we must route all clock pins within \(R_i\) with atmost \(d'_i\), where \(d_i + d'_i = \sigma_\tau\).

The center of each region partitions that region into four different sub-regions. An X-tree, rooted at the center, is formed such that the radius \((r_x)\) is the diagonal line joining the center to the corner of that region. To perform planar routing in a region \(R_i\), we route the clock pin which has maximum delay from center \(c_i\). Next,
Figure 2. Clock Layouts for Different Values of $k$.

we consider the clock pin which is farthest to the regional clock tree and route it such that its path length is equal to $d'$. In this manner, all clock pins in region $R_i$ are routed with zero skew and maximum total delay of $\sigma_T$. The pseudo-code of this procedure is shown below.

/* Procedure Planar_Wire_Min$(R_i, \sigma_T)$ */

*Input:* Region $R_i$ and $\sigma_T$.

*Output:* $T_i$ - a planar clock tree of region $R_i$.

Begin Procedure

if $|R_i| \leq 0$ return($T_i$)

else

for all $s_i \in R_i$ do
route $s_i$ such that $\text{dist}(s_0, s_i) = \sigma_T$

$R_i = R_i - s_i$

$\text{Planar.Wire.Min}(R_i, \sigma_T)$

end Procedure

Lemma 2 The procedure $\text{Planar.Wire.Min}(\cdot)$ runs in $O(n^2)$, where $n$ is the number of clock sinks (terminals).

Consider an $L \times L$ area shown in Figure 3, having $s_0$ as the center. Let clock sink $s_i$ be the farthest from $s_0$. Therefore, the delay from $s_0$ to $s_i$ is $\sigma_T$. Since, in rectilinear geometry, the distance is always measured on $x$-$y$ axes, therefore

$$\text{dist}(s_0, s_i) = \text{dist}(s_0, p, s_i) = \text{dist}(s_0, q, s_i)$$

It is, therefore, clear from the Figure 3 that no matter where a point $p'$ lies on the boundary of the box, $s_i$ remains at $\sigma_T$ from $s_0$, i.e., for any point $p'$ on the boundary

$$\text{dist}(s_0, p', s_i) = \sigma_T$$

We, now state the following Lemma.

Lemma 3 Given a bounding box, let $s_0$ and $s_i$ be its opposite corners. In rectilinear geometry, $\text{dist}(s_0, s_i) = \text{dist}(s_0, p', s_i)$, where $p'$ be any point on the boundary of the box.
Theorem 1 For any $L \times L$ area, the procedure RegionTree() generates total wirelength

$$\Gamma \leq \left( k - \frac{1}{2} \right) L + \begin{cases} \frac{nL}{2^{k-1}} & \text{If } k \text{ is even} \\ \frac{3nL}{2^k} & \text{If } k \text{ is odd} \end{cases}$$

where $k$ is the number of levels of $H$-tree and $n$ is the total number of clock pins.

Proof: Notice that procedure DefineRegions partitions the area such that the total number of partitions (regions) for a $k$-level $H$-tree is $2^k$. As stated above that the maximum length from center of each region $R_i$ is $d_i'$. It can be shown that

$$d_i' \leq \begin{cases} \frac{L}{2^{k-1}} & \text{if } k \text{ is even} \\ \frac{3L}{2^k} & \text{if } k \text{ is odd} \end{cases}$$

In the worst case, $n$ clock terminals, in all $2^k$ regions are routed with $d_i'$ each. Furthermore, the regions are routed in an $H$-tree structure, therefore, the length of a $k$-level $H$-tree contributes towards total wirelength. The length of a $k$-level $H$-tree is $(k - \frac{1}{2})$. Therefore, the total wirelength is bounded as

$$\Gamma \leq \left( k - \frac{1}{2} \right) L + \begin{cases} \frac{nL}{2^{k-1}} & \text{If } k \text{ is even} \\ \frac{3nL}{2^k} & \text{If } k \text{ is odd} \end{cases}$$

Above inequality shows that the total wirelength is determined by the length of $k$-level $H$-tree and the length of $X$-tree in each region. The effect of $k$ on total wirelength is depicted in Figure 4. The total wirelength decreases as $k$ increases. For $k > \log(n) + 1$, the total wirelength increases, therefore, $k$ should be selected such that $1 \leq k \leq \log(n)$.

Consider an example of eight clock terminals (sinks) in an $L \times L$ layout area as shown in Figure 5. If we set $k=0$ (Figure 5(a)), the layout obtained is
similar to [11] and total wirelength is 6L. However, if \( k \) increase, our algorithm produces the layout as shown in Figure 5(b) and (c) for \( k=1 \) and 2, respectively. The total wirelength for \( k=2 \) is 5.5L. Experimental results (Section 5) show that up to 10% reduction in total wirelength is achieved as compared to [11].

**Theorem 2** Given a region \( R_i \) with a center \( (x_{c}, y_{c}) \). Each clock sink \( s_{j} \in R_i \) can always be routed such that \( \text{dist}(s_{0}, s_{j}) = \sigma_{T} \).

**Proof:** As the bounding box \( B \) is formed such that the distance from the center to any corner of that box is \( \sigma_{T} \), therefore, as stated in Lemma 3, each region will have an X-tree such that \( r_{x} = \sigma_{T} \). Since, each region has an X-tree of radius \( \sigma_{T} \), all clock terminals in that region can be routed such that for each clock sink \( s_{i} \), \( \text{dist}(s_{0}, s_{i}) = \sigma_{T} \).

\[ \square \]

### 2.2.3 Steiner Wirelength Minimization

In this section, we present details of steiner wirelength minimization method which further reduces the wirelength, while converting the part of layout into rectilinear steiner tree. The wirelength minimization phase is performed in a bottom
up fashion starting at the lowest level (leaves) of the clock tree. Consider edges, $a, b, c, d$ as shown in Figure 6(a). The layout is generated from the planar wire-length minimization phase, as discussed in above section. Two edges $d$ and $e$ are merged first such that the layout is changed to rectilinear and the reduction in wirelength is achieved (Figure 6(b)). Secondly, the segment $c$ and $f$ are merged as shown in Figure 6(c).

If any two edges $e_1$ and $e_2$ lie in non-adjacent (or opposite) quadrants, we call such edges as separable edges and non-separable edges otherwise. If all edges in a tree are separable, the clock tree is called as separable tree. A separable clock tree is shown in Figure 7. We present the following lemma.

**Lemma 4** The Steiner Wire Min procedure results in minimization of total wire-length if the clock tree is non-separable.

---

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
Figure 6. Wirelength Minimization.

Figure 7. A Separable Clock Tree.
As an explanation of above lemma, consider two leaves $l_1$ and $l_2$ (Figure 8(a)). The edges of these leaves are represented by $e_1$ and $e_2$. Let us analyze the position of the leaves with respect to the quadrants. We have following cases:

Case I: Suppose both $e_1$ and $e_2$ lie in the same quadrant (Figure 8(b)). We examine the wirelength minimization in four quadrants. Without the loss of generality, we assume that the parent edge of the leaves lie in the second quadrant. Let $L_1$ be the wirelength before minimization phase and $L_2$ be the wirelength after minimization. If $e_1$ and $e_2$ lie in the 1st quadrant, the reduction in the wirelength (Figure 8(c)) is given by

$$L_1 = x_1 + y_1 + x_2 + y_2$$

$$L_2 = x_1 + y_2 + (y_1 - y_2) + (x_1 - x_2)$$

$$= y_1 + x_2$$
The total reduction $\delta L$ is $L_1 - L_2 = x_1 + y_2$

Similarly, the wirelength can be minimized when $l_1$ and $l_2$ lie in other quadrants. However, the percentage reduction in wirelength reduces as the angle ‘$\varphi$’ between $e_1$ & $e_2$ increases (Figure 8(d)).

Case II: Suppose both $e_1$ and $e_2$ lie in adjacent quadrants. In this case, we assume that parent edge lies in second quadrant (Figure 8(e)). First consider that $e_1$ and $e_2$ lie in first and second quadrant, respectively. Then wirelength minimization is given as

$$L_1 = x_1 + y_1 + x_2 + y_2$$

$$L_2 = y_1 + x_2 + (y_2 - y_1) + x_1$$

Therefore, $\delta L = y_1$

In most of the cases, the wirelength minimization deals with merging two leaf nodes and converting their edges into rectilinear edges. But, in a more general case, we might have $i$ number of leaf nodes and we have to decide, out of these leaf nodes, a sequence combination of leaf nodes, when merged, gives maximum reduction in wirelength. Let us analyze the number of possible combinations.

Let $f_i$ be the number of valid sequences for combination of $i$ leaf nodes. Let a sequence be called odd (even) if $f_i$ is odd (even). Also assume that $g(i)$ be the number of valid sequences of $i$ nodes and $h(i)$ be the odd valid sequences. Thus, $g(i)$ will start with two leaves and followed by a sequence of $(i - 2)$ leaf node sequence. Similarly, $h(i)$ will start with one leaf (which does not take part in the sequence) and followed by a sequence of $(i - 1)$ leaf node sequence. It is easy to see that $g(2) = 2, h(2) = 0$ and $g(3) = 1, h(3) = 1$.

**Theorem 3** The number of valid sequences of $i$ number of leaf nodes is given by

$$f_i = \left(\frac{1 + \sqrt{5}}{2}\right)^{i-1} - \left(\frac{1 - \sqrt{5}}{2}\right)^{i-1} \quad n > 2$$
Proof: In order of construct a sequence of \( i \) leaf nodes, we have two choices.

1. To construct an even sequence of length \( i \), we take first two nodes and append all valid even sequences of of length \((i - 2)\) and all valid odd sequence of length \((i - 2)\).

2. We construct an odd sequence by taking one leaf node and appending all valid even sequences of length \((i - 1)\).

These two conditions are expressed as

\[
g(i) = g(i - 2) + h(i - 2)
\]

\[
h(i) = g(i - 1)
\]

Since,

\[
f_i = g(i) + h(i)
\]

\[
= g(i - 2) + h(i - 2) + g(i - 1)
\]

\[
= f_{i-2} + g_{i-1}
\]

Also \(g(i) = f_{i-2}\), therefore

\[
f_i = f_{i-2} + f_{i-3}
\]

Solving this recursion, we get

\[
f_i = \left(\frac{1 + \sqrt{5}}{2}\right)^{i-1} - \left(\frac{1 - \sqrt{5}}{2}\right)^{i-1}
\]

Note that for large values of \(i\), second term of the above solution tends towards zero, even for small values of \(i\), as is the case in our problem, second term still plays no dominant role.
CHAPTER III
MULTIPLE PHASE CLOCK ROUTING

As discussed in Chapter I, the role of clock in digital systems is to synchronize the orderly and controlled flow of information by holding up the signal until it is time for it to begin to move through the next stages of logic. In routing clock signal, key parameters are (i) delay which a signal incurs while going from source to sink and (ii) the skew between the arrival times of different clock terminals. If $t_i$ is the arrival time of clock signal as it travels from source to terminal $i$ (sink), then clock skew, $\sigma$, is defined as

$$\sigma = \max \{t_i \mid \forall i = 1, 2, \ldots, n\} - \min \{t_i \mid \forall i = 1, 2, \ldots, n\}$$

Single-phase clocking scheme, as shown in Figure 9(a) is the most transistor-efficient of all clocking schemes, but at the same time it is insidiously complex and imposes unnecessary constraints on the speed of the logic elements in the block of combinational logic. Let $T_p = T_L + T_H$, where $T_p$ is the clock period, $T_H$ is the time the clock is high, and $T_L$ is the time the clock is low. Signals propagate through the combinational logic at different speeds depending on their logical value. We are concerned with two extremes in time i.e., first, the time that it takes the slowest valid logic signal to propagate through the logic block (for the clocking to be data-dependent, this propagation delay must be less than $T_p$) and second, the shortest time it might take a signal to reach the output of the logic block. This time must be greater than $T_H$. Let $\tau_{CL}$ represent the range of combinational logic delays, from the shortest invalid delay to the longest valid delay. We then have

$$T_H < \tau_{CL} < T_p$$

20
which means that every delay through the combinational logic block must be greater than $T_H$ and less than $T_P$. It is the two-sided nature of this constraint that makes single-phase clocking so difficult to implement. Not only do we need to worry about the critical slow path, but we must also find the critical fast path. Circuit designing would then be very difficult because we not only have to make the slow case fast enough, but also the fast case slow enough. In multiphase clocking environment, however, we can be free of the two-sided constraint. Consider the two-phase clocking scheme (Figure 9(b)), two clocks are represented by $\phi_1$ and $\phi_2$. $T_1$ is the time that clock $\phi_1$ is high, $T_3$ is the time that $\phi_2$ is high, $T_2$ is the low time between $\phi_1$ and $\phi_2$, and $T_4$ is the low time between $\phi_2$ and $\phi_1$. Using the
same analysis, we find

\[ \tau_{CL} < T_1 + T_3 + T_4 \]

or

\[ \tau_{CL} < T_P - T_2 \]

where

\[ T_P = T_1 + T_2 + T_3 + T_4 \]

A signal starts to propagate through the combinational logic block at the time that \( \phi_2 \) goes high, and must finish before \( \phi_1 \) goes low. Note that we reduced the constraint, which was two-sided, to a single-sided constraint. The price we pay is twofold: an increase in the latch circuitry and the requirement to bus many more clock wires around the chip. In VLSI design, we are more concerned with the added clock wires than with the added transistors, but even doubling the number of clock wires is a small price to pay for making the design problem merely hard.

We consider a system with multiple clocks. Let \( \Phi = \{ \phi_1, \phi_2, \ldots, \phi_k \} \) be the set of \( k \) clocks, where \( \phi_j = \{ c_{j1}, c_{j2}, \ldots, c_{jn} \} \). For any clock \( \phi_j \), we denote the skew of \( \phi_j \) i, intra clock skew \( \sigma_j \), as

\[ \sigma_j = \max \{ t_i \mid \forall i = 1, 2, \ldots, n \} - \min \{ t_i \mid \forall i = 1, 2, \ldots, n \} \]

Multiple clocks entering in a subblock are usually very close but still considered as separate entry points. For \( k \) clocks entering in a subblock, we define a superset \( P \) of \( k \) sets of terminals as \( P = \{ p_1, p_2, \ldots, p_n \} \), where each \( p_i \) is \( k \)-tuple defined as \( p_i = (c_{i1}, c_{i2}, \ldots, c_{ik}) \), for \( i = 1, 2, \ldots, n \) and \( c_{ij} \in \phi_j, j = 1, 2, \ldots, k \).

Let \( t_{ji} \) be the arrival time of clock \( \phi_j \) at terminal \( c_{ji} \), for \( j = 1, 2, \ldots, k \), \( i = 1, 2, \ldots, n \). Then the inter clock skew of \( \forall p_i \in P \) is defined as

\[ \chi_i = \max \{ t_{ji} \mid \forall j = 1, 2, \ldots, k \} - \min \{ t_{ji} \mid \forall j = 1, 2, \ldots, k \} \]

We also define cross skew as
\[ x^* = \max\{x_i \mid \forall i = 1, 2, \ldots, n\} - \min\{x_i \mid \forall i = 1, 2, \ldots, n\} \]

The objective of multiple clock routing system is, thus not only to minimize \( \sigma_j \) for each clock \( \phi_j \), but also to minimize \( x_i \) between each set of clocks and \( x^* \), the cross skew, of multiple clock system. But this is not easy to do, because routing multiple clocks definitely introduces intersections between clock wires. In this case, we can make use of VLSI design style. Each time clock signals intersect, one signal crosses under other signal using polycide. Crossunders are always placed in polycide because of its low resistance. Crossunders should be minimized, subject to the constraint that the number of crossunders should be equalized for multiple clock phases, in order to equalize the signal delays.

Now we briefly discuss the delay characteristics of clocking system to form a basis for our two clock routing algorithms. Given a clock layout scheme, the point from which clock signal enters is termed as clock entry point (CEP). The delay from CEP (root) to any synchronizing device (sink) depends on the wire length from CEP to the device and the RC constants of the wire segments. The exact computation of the delay of a clock tree is very difficult. However, it is not quite difficult to calculate the delay approximately by using the Elmore delay [14]. The Elmore delay is defined as the first order moment of the impulse response \( g(t) \), also known as the inertia:

\[ T_d = \int_0^\infty g(t) \, dt \]

Considering an interconnection delay model, the time delay \( T_D \) required for the output voltage of distributed and lumped \( RC \) networks to rise from 0 to 90 percent of their final values are \( 1.0RC \) and \( 2.3RC \) respectively [6]. Accordingly, a very good approximation for delay is obtained by combining the resistive and capacitive terms and weighting them by the appropriate factors as described earlier.
Using this Elmore delay model, the total clock signal delay $TD_0$ can easily be computed recursively:

$$TD_0 = \sum_{i=1}^{n} TD_i$$

In order to reduce phase delays (skew) and supply sufficient driving currents, we usually use several levels of clock buffers as shown in Figure 10. The rectangle in Figure 10 represents a delay element having $b_d$ as buffer internal delay, $b_r$ as buffer output resistor and $b_c$ is the buffer input capacitor. A buffer is basically used to introduce stages such that the subtree capacitance is not carried over i.e., the equivalent total subtree capacitance as seen at the buffer input is only $b_c$. Since, the buffer driving resistance and capacitance are small, delay is reduced.

In the next section, we present the routing algorithm for gate array architecture. If we have gate arrays, it is reasonable to assume that clock pairs are distributed in a uniform fashion.
3.1 Algorithm for Uniformly Distributed Clock Pairs

In this section, we present an efficient algorithm for H-tree layout of uniformly distributed clock pairs (UDCP). This algorithm is suitable for regular structures, such as systolic arrays. Before we formally state the algorithm, we develop a lower bound on number of crossunders needed to layout multiple clocks on the same layer. Crossunders are used to avoid the crossing of two different clock signals. As we have already mentioned that the crossunders increase the signal propagation delay, therefore, it is desirable to use a minimum number of crossunders.

In single phase clock systems, the best way to represent clock distribution scheme is H-shaped structure (H-tree). When it comes to multiple phase clock distribution, we can still use H-tree but the resulting H-tree has crossunders at each level whenever they intersect. We have two options at this stage; either we distribute the signals in such a way that they donot intersect at all or allow intersections and have crossunders. Second option is clearly a better option because first option will involve a longer wire length and thereby greater delay. The second option has the advantage that crossunders do not increas the chip area, but they increase the delay. Therefore, it is desirable to minimize the number of crossunders used to layout the multiple phase clock system.

We now develop a lower bound on the number of crossunders in uniformly distributed two-phase clock system, then we generalize it for uniformly distributed k-phase clock system. Let \( \{p_1, p_2, \ldots, p_n\} \) be the set of \( n \) pairs, where \( n = 2^h \) for \( h \geq 2 \) and \( p_i = (c_{1i}, c_{2i}) \).

**Lemma 5** Let \( P = \{p_1, p_2, \ldots, p_n\} \) be the set of terminal pairs for \( \phi_1 \) and \( \phi_2 \), where \( p_i = (c_{1i}, c_{2i}) \). Let \( H_1 \) and \( H_2 \) be two trees formed by \((c_{11}, c_{12}, \ldots, c_{1n})\) and \((c_{21}, c_{22}, \ldots, c_{2n})\) respectively. Then \( H_1 \) and \( H_2 \) will intersect exactly at \((n - 1)\) different points.
Proof: We proof above lemma by induction on $h$, where $h = \log_2 n$. It is easy to show that for $h = 2$, i.e., four pairs of clock terminals, $H_1$ and $H_2$ intersect at exactly three points.

We assume that the lemma is true for $h = (l - 1)$, we have to show that it is also true for $h = l$.

Note that H-tree of $2^l$ number of terminals can be formed by merging two H-trees of $2^{l-1}$ number of terminals. Let $p_1 = (c_{11}, c_{21})$ and $p_2 = (c_{12}, c_{22})$ be two pairs of clock terminals such that $c_{11}$ and $c_{21}$ be the roots of two H-trees (running in parallel) of $2^{l-1}$ terminals. Similarly, $c_{12}$ and $c_{22}$ be the roots of two H-trees (running in parallel) of $2^{l-1}$ terminals. Note that when we connect $c_{11}, c_{12}$ and $c_{21}, c_{22}$, they intersect at three places (from above argument). Therefore, the total number of intersections is given by

$$x = 2(2^{h-1} - 1) + 3 = 2^h - 1 = n - 1 \square$$

Note that every time two clock signals cross at a point, we need to place a crossunder on one of the clock signals. Therefore, using the previous lemma, we conclude the following lemma:

Lemma 6 Let $P = \{p_1, p_2, \ldots, p_n\}$ be a set of uniformly distributed clock terminals for $\phi_1$ and $\phi_2$ where $p_i = (c_{1i}, c_{2i})$ for $1 \leq i \leq n$. Then we need at least $n - 1$ crossunders to effectively route $\phi_1$ and $\phi_2$.

Using the similar argument, we can generalize the above lemma for $k$ clocks ($k > 1$).

Lemma 7 Let $P = \{p_1, p_2, \ldots, p_n\}$ be a uniformly distributed set of clock pins. Each $p_i$ is a $k$-tuple $(c_{i1}, c_{i2}, \ldots, c_{ik})$. Then we need at least $\frac{3n(k-1)}{2}(= o(n))$ crossunders to effectively layout $\phi_1, \phi_2, \ldots, \phi_k$ clocks.
We now present our proposed algorithm in detail. We begin our routing algorithm by dividing the region recursively, in top down fashion, until each sub-region has two pairs. Our algorithm works in two phases. In first phase it builds the clock tree $H_1$ and $H_2$, simultaneously, using H-tree model in a bottom up fashion recursively. At each recursive call, we merge two path-balanced subtrees to get a new tree. We start our algorithm with clock pins (leaf nodes in the tree) which serve as initial clock entry points. Each time two subtrees are merged, a new clock entry point is determined.

Let $P = \{p_1, p_2, \ldots, p_n\}$, where $p_i = (c_{i1}, c_{i2})$, and $n = 2^h$ for $h \geq 1$. Each of $c_{i1}$ and $c_{i2}$ represent clock entry point of $\phi_1$ and $\phi_2$ respectively. Each entry point, $c_{i1}$ and $c_{i2}$, is represented by $x$- and $y$-coordinate, which we denote as $(x_{c_{i1}}, y_{c_{i1}})$ and $(x_{c_{i2}}, y_{c_{i2}})$. Associated with each pair of entry points, $p_i$, is a unique level $l$, $1 \leq l \leq h$ where $h$ is the height of the clock tree. The level of clock terminal pairs is $h$ i.e $l = h$. We denote the clock terminal pairs $p_1, p_2, \ldots, p_n$ as $p_1^h, p_2^h, \ldots, p_n^h$. We start by merging two subtrees at level $h$ to get new entry point (new pair) which is at level $h - 1$ and so on. We assume, without the loss of generality, that $p_1^h, p_2^h$ are paired together and $p_3^h, p_4^h$ are paired together. Each time two clock trees intersect, we need to place crossunder in one of the trees.

In second phase, we balance the number of crossunders as we come from top to down. In this phase, we consider the segments formed in bottom-up procedure. Each segment is divided into two halves and each half is assigned a flag, depending on whether a particular half has a crossunder or not. A segment obtained by connecting $c_{ti}$ and $c_{ti+1}$ is represented by $S^{l}_{(i,i+1)}$ and the midpoint of this segment is $\frac{c_{ti} + c_{ti+1}}{2}$. This point divides the segment into two halves; $\frac{c_{ti}}{1 + \frac{1}{2}c_{ti+1}}$ and $\frac{c_{ti+1}}{1 + \frac{1}{2}c_{ti+1}}$. 

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
A flag, set to 1, is used to indicate that a particular half has a crossunder and flag, set to 0, is used to indicate that a particular half does not have a crossunder. We represent flag of $c_{1\frac{1}{2}}c_{1\frac{1}{2}+1}$ as $\mathcal{F}_{1\frac{1}{2}+1}$ and the flag attached to $c_{1\frac{1}{2}+1}c_{1\frac{1}{2}+1}$ is represented as $\mathcal{F}_{1\frac{1}{2}+1}$. We now formally present algorithm denoted as $TCR()$.

**Algorithm TCR()**

*Input:* $n$ pairs of clock pins.

*Output:* $H_1$ and $H_2$ with zero skew.

/* Phase 1: */

for $l = h$ to 1 do

i = 1

for $k = 1$ to $2^h$ do

$p_1 = p_i^l; p_2 = p_{i+1}^l$;

/* $p_1 = (c_{11}, c_{21})$ and $p_2 = (c_{12}, c_{22})$ */

$S_{1(i+1)}^l = \sqrt{(x_{c_1} - x_{c_1(i+1)})^2 + (y_{c_1} - y_{c_1(i+1)})^2};$

$S_{2(i+1)}^l = \sqrt{(x_{c_2} - x_{c_2(i+1)})^2 + (y_{c_2} - y_{c_2(i+1)})^2};$

$c_{1\frac{1}{2}+1}^{l-1} = S_{1(i+1)}^l/2;$

$c_{2\frac{1}{2}+1}^{l-1} = S_{2(i+1)}^l/2;$

$p_{i(\frac{1}{2}+1)}^{l-1} = (c_{1\frac{1}{2}+1}^{l-1}, c_{2\frac{1}{2}+1}^{l-1});$

if $(l \neq h)$ then

if $S_{1(i+1)}^l||r$ then

Place.crossunder($H_1, (x_1, y_1)$);

else

Place.crossunder($H_2, (x_1, y_1)$);

else

Place.crossunder($H_2, (x_1, y_1)$);

Place.crossunder($H_1, (x_2, y_2)$);

i = $i + 2$;
/* Phase 2: */
for $l = 2$ to $h$
do 
    $i = 1$
    for $k = 1$ to $2^h$
do 
        for $j = 1$ to $2$
do 
            if ($F^l_{i(\frac{i+1}{2})} = 1$ and $F^l_{(i+1)(i+1)} = 0$) then 
                let $(x_1, y_1)$ be a point on $c_{j(\frac{i+1}{2})}^{-1}c_{j(i+1)}^l$
                Place_crossunder($H_j, (x_1, y_1)$);
            else if ($F^l_{i(\frac{i+1}{2})} = 0$ and $F^l_{(i+1)(i+1)} = 1$) then 
                let $(x_1, y_1)$ be a point on $c_{j(i+1)}^{-1}c_{j(\frac{i+1}{2})}$
                Place_crossunder($H_j, (x_1, y_1)$);
        end.
    end.
end.

**Theorem 4** Algorithm TCR routes two phase clock with zero skew in $O(n)$ time.

**Proof:** Notice that algorithm TCR is based on H-tree layout model, which is linear in nature [13]. At each level of recursion, H-tree is formed and later on crossunder placement is decided. All these operations take linear time, hence our algorithm is linear in $n$, where $n$ is the number of clock pairs.

Figure 11 illustrates the algorithm for uniformly distributed pairs of clock terminals. In practice, clock pins are randomly distributed having different capacitive loadings. In next section, we extend above result to handle randomly distributed pins where the capacitive loading of each pins plays a great role in determining the entry points.
Figure 11. Example of Layout Generated by Algorithm TCR.
CHAPTER IV

EXPERIMENTAL RESULTS

The algorithm $k$-PCT( ) was implemented in C programming language on SUN SPARC station 1+. The algorithm was tested for MCNC industrial benchmarks Primary 1 and Primary 2. The algorithm was also tested for some random examples. To compare our results, we also implemented the algorithm PCR [11]. Figure 12 shows the layout generated by $k$-PCT for Primary 2. Note that the results obtained by our algorithm $k$-PCT are listed for $k=3$.

Table 1 shows the results of our algorithm for randomly generated examples for clock pins ranging from 32 to 128. It compares the maximum path and total wirelength (WL) for each example. Notice that the maximum path remains same

![Figure 12. Layout for Primary 2, Generated by $k$-PCT ( ) for $k=3$.](image-url)
Table 1

Result of Algorithm $k$-PCR for Randomly Generated Examples

<table>
<thead>
<tr>
<th></th>
<th>32</th>
<th></th>
<th>64</th>
<th></th>
<th>128</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>max. path WL</td>
<td>max. path WL</td>
<td>max. path WL</td>
<td>max. path WL</td>
<td>max. path WL</td>
<td>max. path WL</td>
</tr>
<tr>
<td>$k=0$</td>
<td>3.56</td>
<td>23.63</td>
<td>5.44</td>
<td>69.73</td>
<td>5.43</td>
<td>122.76</td>
</tr>
<tr>
<td>$k=1$</td>
<td>3.56</td>
<td>24.54</td>
<td>5.44</td>
<td>67.61</td>
<td>5.43</td>
<td>110.53</td>
</tr>
<tr>
<td>$k=2$</td>
<td>3.56</td>
<td>25.48</td>
<td>5.44</td>
<td>61.65</td>
<td>5.43</td>
<td>101.64</td>
</tr>
<tr>
<td>$k=3$</td>
<td>3.56</td>
<td>29.54</td>
<td>5.44</td>
<td>64.75</td>
<td>5.43</td>
<td>94.91</td>
</tr>
<tr>
<td>$k=4$</td>
<td>3.56</td>
<td>32.70</td>
<td>5.44</td>
<td>72.71</td>
<td>5.43</td>
<td>121.51</td>
</tr>
</tbody>
</table>

for every value of $k$. The total wirelength, however, decreases as $k$ increases for Primary 2.

The results by our algorithm $k$-PCT( ) on Primary 1 and Primary 2, when compared with obtained by existing algorithms, are listed in Table 2. The results show that reduction of 10% in total wirelength and 14% reduction in maximum delay is achieved by $k$-PCT().

Table 2

Comparison of $k$-PCR With the Existing Algorithms

<table>
<thead>
<tr>
<th></th>
<th>Primary 1</th>
<th>Primary 2</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>GM MMM PCR $k$-PCR</td>
<td>GM MMM PCR $k$-PCR</td>
</tr>
<tr>
<td>Planarity</td>
<td>no no yes yes</td>
<td>no no yes yes</td>
</tr>
<tr>
<td>Max. path</td>
<td>7.51 7.24 6.03 6.03</td>
<td>11.58 13.05 9.96 9.96</td>
</tr>
<tr>
<td>TWL</td>
<td>154 162 190 162</td>
<td>377 406 470 428</td>
</tr>
</tbody>
</table>
CHAPTER V

CONCLUSION

In high performance systems, clock delay and clock skew are the key issues on which the system performance depends. Existing algorithms for clock routing minimize or eliminate the clock skew and completely ignore the clock delay. In this thesis, an efficient clock router has been developed, which generates planar clock trees with minimal delay, zero skew and minimum wirelength. Thus, our clock routing algorithm is best suited for high performance systems. Since, our algorithm generates planar clock trees, it can easily be implemented in a single metal layer. For large design, the clock is routed in multiple phases. Therefore, multiple clock routing is vital for such designs. We also developed crossunder distribution scheme for such systems.

Experimental results prove that our routing algorithms are practical in nature, and hence we expect that the proposed clock routing algorithms will be widely used for high performance VLSI systems.
REFERENCES


