NOTE TO USERS

Tuning and Topology Optimization of the Clock Distribution Network Under Obstacle Constraints

Haydar Saaied

A Thesis
in
The Department
of
Electrical and Computer Engineering

Presented in Partial Fulfillment of the Requirements
for the Degree of Doctor of Philosophy at
Concordia University
Montreal, Quebec, Canada

October 2004

ABSTRACT

Tuning and Topology Optimization of the Clock Distribution Network Under Obstacle Constraints

Haydar Saaied, Ph.D.
Concordia University, 2004

Our research focuses on routing the Clock Distribution Network (CDN). The CDN consumes an increasing portion of all resources in terms of wiring area, power, and design time. Different approaches that have been proposed, such as Deferred-Merge Embedding algorithm (DME) and Greedy-DME (GDME), require the re-calculation of the whole solution when there is a change in the skew constraint or the location or the load capacitance of the clock pins. Redesigning the CDN would be an extremely computation intensive process for a complex system, and very painful with the increase in demand for shorter time to market. For this reason, we have used an incremental routing scheme. The incremental routing or ECO (Engineering Change Order) routing is a new field that has been introduced to meet the demand of high performance and complex designs.

We propose a new method, called Adaptive Wire Adjustment (AWA), to minimize the skew of a given CDN to any given bound by tuning its wire lengths in order to cope with minor modifications in System on Chip (SoC) during the design process. The proposed algorithm outperforms DME and GDME for implementing ECO for large CDNs. Moreover, to speed up AWA's convergence, we propose the use of a Local Topology Modification (LTM) technique. Additionally, LTM helps to enhance the CDN's quality in terms of total wire length and wire elongations. We show that GDME relies heavily on wire elongations and offers a solution that suffers from large Standard Deviation of the Path Lengths (SDPL) between clock pins and the CDN's root. To our knowledge, this thesis is one of the first of its kind that deals with minor CDN's modifications and local topology modification.

In addition to satisfying the timing requirements, the CDN has to be routed with minimum wire length. Moreover, it is often the case that the interconnects of a net must not intersect with some obstacles in the routing plane. We propose a simple model called Shortest Paths Polygon (SPP) to describe the routing area of shortest paths between two points among obstacles. A methodology is proposed to determine the SPP of two points using a gridless graph. The SPP model facilitates a better routing of any multi-terminal net since it determines all shortest paths. Consequently, using the SPP would provide a good correlation between detailed routing and other design phases. This is important because it satisfies the system requirements and supports ECO routing effectively. The SPP model is applied to three problems related to the CDN routing: tuning the CDN under obstacle constraints, designing a ZSCDN under obstacle constraints, and routing the CDN in two layers.

ACKNOWLEDGMENT

I have been very fortunate to have Dr. Dhamin Al-Khalili and Dr. Asim J. Al-Khalili as my supervisors and it is hard to acknowledge them in few words. Dr. Dhamin was always there with his knowledge, good advice and deep insight. His patience and compassion will always be remembered. Dr. Asim made me love the field of VLSI. Ever since, I've been knocking on his door, day and night, seeking his guidance and wisdom.

I wish to express my gratitude to Dr. B. Jaumard from University of Montreal , the optimization centre, for her support and expert advice on optimization.

I am grateful to Dr. Mohamad Nekili for his support and encouragement. I benefited so much from his insightful thoughts on high speed interconnect.

Many thanks to my wife, Mrs. Ibtihal Fadhil, for her kind support and encouragment.

I would like to reserve my deepest thanks for my first teachers: my parents, Mr. and Mrs. Al-Taraihi, and my uncle, Dr. Taki Al-Taraihi. I am extremely indebted to them for their unconditional love, sincerity, and their tradition of encouraging me and my brothers to pursue science.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ACRONYMS

AWA	Adaptive Wire Adjustment
BB	Balance Bipartition
BSCDN	Bounded Skew Clock Distribution Network
CDN	clock distribution network
DME	Differed Merge Embedding
GDME	Greedy DME
GMA	Geometric Matching Algorithm
LTM	local topology modification
ITRS	International Technology Roadmap for Semiconductor
LTM-AWA	Applying LTM approach to AWA algorithm
LTM-DME	Applying LTM approach to DME algorithm
LTM-GDME	Applying LTM approach to GDME algorithm
MMM	Method of Means and Median
SDPL	standard deviation of the path lengths between the root and the clock pins of a CDN
SPP	shortest paths polygon
SPP-AWA	Applying SPP model to AWA algorithm
SPP-DME	Applying SPP model to DME algorithm
ZSCDN	Zero Skew Clock Distribution Network

LIST OF SYMBOLS

α	The first sign of a concave point of a polygon
A	Manhattan arc set
β	The second sign of a concave point of a polygon
C	capacitance of a subtree
c_0	capacitance per unit length
Centre	centre of delay range
D()	Manhattan distance between two arguments
e	edge length
l	location of a node of a CDN in the Manhattan Plane
L	Left poiter of a node of a CDN
m	cost of the shortest path
Max	The Maximum a leaf delay range
Min	The Minimum a leaf delay range
r_0	resistance per unit length
R	Right poiter of a node of a CDN
Range	leaf delay range
SPP()	Shortest Paths Polygo of two arguments
Θ	An operation that determines the end points of a Manhattan arc
Wire()	The wire that connects two points or two arcs
Ψ	An operation that determines the corners of a Rectangle

Chapter 1

Introduction

1.1 System on Chip Design

A System on a Chip (SoC) combines several homogenous or heterogeneous blocks, which vary in complexities and specification, on a single chip in order to carry out numerous integrated operations. The design of a SoC is challenging and can be very complicated compared to the design of Application Specific Integrated Circuits (ASIC) due to the scale of the problem. Accordingly, no single designer will be able to handle this huge design task. To cope with such a challenging task, it is necessary to develop more sophisticated design approaches.

One approach to design the SoC is to have a number of design teams that work concurrently in a common user-friendly design environment. This approach is manifested in the growing trend among designers to import pre-designed macro functions, called Intellectual Property (IP) blocks, from various providers [12]. According to International Technology Roadmap for Semiconductor (ITRS), the complexity and the size of the SoC will continue to drive designers to reuse IPs over the next decade [1]. These third party IP blocks may cover up to 96% of the area of the chip [1]. However, the major challenges posed by the SoC design are system complexity, signal integrity and verification of the implementation. Other two factors influencing the SoC design are the constant

demand for the reduction in the time to market and the power consumption. Therefore, the SoC design imposes new challenges that need to be addressed by the designers and EDA community.

1.1.1 Chip Interconnects

One of the major challenges of the SoC design that have surfaced is the role and performance of the interconnects [2-8]. The role of the interconnects is to distribute data, control signals and system clock and its derivatives across the chip. It also feeds power and ground to various circuits of the system. Due to the size and complexity of a SoC, different metal layers are used to lay out the massive interconnects. It is projected that future SoC will use some 14 metal layers by 2011 [1]. Local interconnects normally use lower level metals, while global interconnects use high level layers. The latter include signals between IPs, power busses and clock distribution networks [7]. The process of laying out the interconnect is very complex; and furthermore, the signal propagation through the interconnect is not linear [2].

The reduction in the feature size renders the device delay smaller. However, the delay of the interconnect increases with the reduction in the feature size due to the increase in the resistivity of the interconnect [2-4]. In addition, due to the impact of the parasitic capacitance, the delay increases exponentially with the length of the interconnect. Figure 1.1 shows a comparison between the impact of feature reduction of the interconnect and gate delay [1]. Local interconnects are relatively unaffected by the shrinking in the technology. However, the delay of the signal is primarily influenced by the global intercon-

nects [1]. In fact, the global interconnect delay will continue to increase with the technology scaling. As a result, one of the most challenging problems of the SoC is to maintain the signal integrity at the targeted performance [2-8].

1.1.2 The Design Methodology

The SoC has to be designed according to a methodology that orchestrates the steps of the design process in order to fulfill the system requirements [12 , 65]. Thus, the design methodology holds the key answer to the challenges posed by the SoC design. Nowadays, different design tasks communicate with each other at different design steps making the process tedious and complex [48]. Indeed, the designer has to go back and forth between different design steps in order to meet the system requirements.

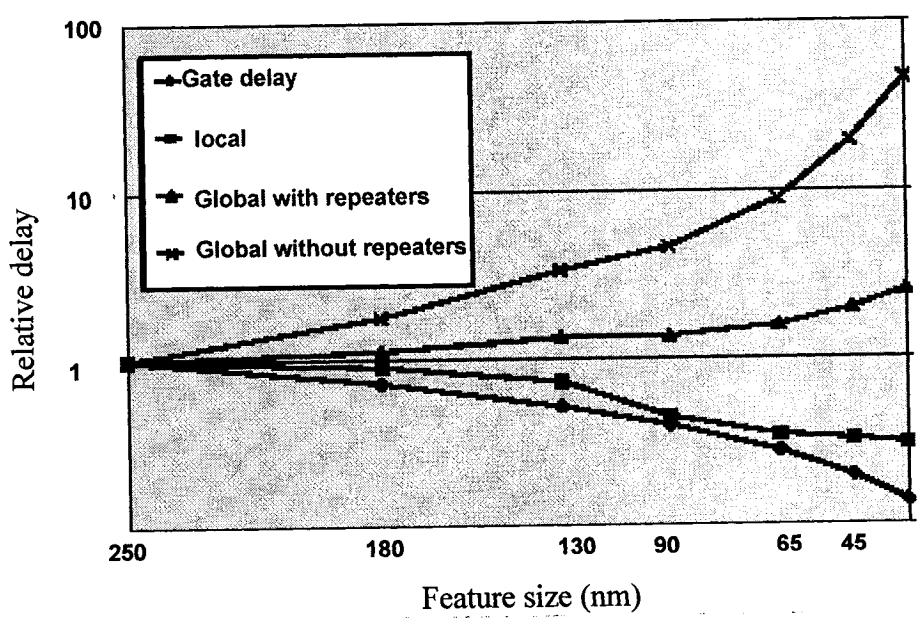


Figure 1.1 Delay for interconnect versus feature size (courtesy ITRS 2003) [1].

In order to cope with the challenges of the SoC design, different design tasks need to

communicate with each other, operate concurrently and share data memory. Figure 1.2 shows, in general, the projected design methodology for feature size less than 65 nm according to ITRS [1]. In addition, different design steps have to be executed incrementally in the case where minor modification are required. Supporting different design steps with efficient Engineering Change (EC) techniques, and executing these steps in parallel would improve engineering productivity and reduce time to market.

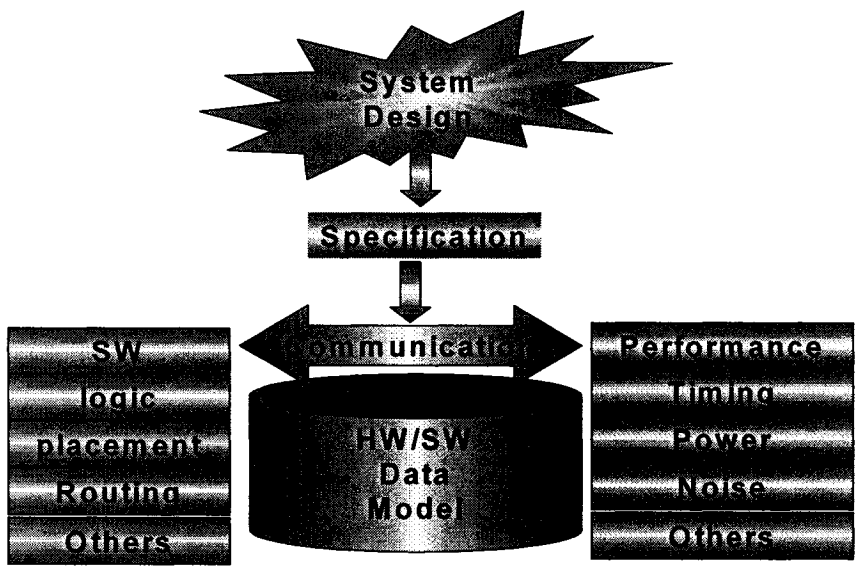


Figure 1.2 Design methodology as projected by ITRS 2003 for technology less than 65 nm [1].

With the dominance of the interconnect role in determining SoC performance, the task of routing becomes more important than the optimization of the devices [2-4]. Thus, there is a definite need for efficient EC techniques that can handle any local modification in the layout of interconnects of the SoC [48]. Kahng et. al. showed that the crucial role of the interconnect promotes a need for interconnect tuning [49]. The rise in SoC complexity and interconnect dominance team up to intensify a specific challenge to the

Clock Distribution Network (CDN)[5-24]. The challenge stems from the CDN's crucial role of integrating different IP cores into a single synchronous SoC.

1.2 Clock Distribution Network

High performance systems may contain hundreds of thousands of combinational logic elements and tens of thousands of registers. All registers, memory elements and dynamic circuits in a synchronous digital system are synchronized by a single clock signal or multiple clock signals. The function of the clock signal is to deliver a time reference so that the correct data propagates throughout the system. The clock signal is normally delivered by a network of interconnects commonly known as a Clock Distribution Network (CDN) [5].

The clock signal is a performance bottleneck for any SoC because of its characteristics that distinguish it from other signals in the system. One of the most important characteristics is that the clock signal must feed all registers around the system. Hence, the CDN is largest net occupying a significant portion of the chip area. The increase in the size of the CDN implies an increase in the power consumption of the CDN. This power is almost dynamic power in nature due to the charge and discharge of the massive interconnection capacitance of the CDN. Furthermore, the CDN with its heavy fanout load has to operate at the highest speed among all other signals. Thus, designing the CDN is a critical step in the design process of a SoC. The approach of designing the CDN would affect the arrival time of the clock signal to the corresponding registers; and consequently, would impact the CDN's quality.

The difference between arrival times of the clock signal is referred to as clock skew. The skew effects on the system performance depend on the data propagation paths between the registers. In general, the clock skew can either enhance or degrade the system performance depending on the magnitude and polarity of the skew. The skew can harm the system performance in terms of speed, power and reliability. Worst than that, if the skew is not properly controlled, it might lead to a system failure.

The skew problem can be tackled by devising a CDN so that the skew is zero between all registers. Such a CDN is called Zero Skew Clock Distribution Network (ZSCDN) [39-43]. Other researchers focused on achieving a Bounded Skew CDN (BSCDN) instead of ZSCDN due to the difficulties of achieving ZSCDN [44-47]. The skew impact on system performance [25-28] motivated researches to explore various techniques of clock scheduling with the objective of optimizing clock skew [29-33].

This thesis focuses on the performance of different CDN design approaches in light of the new challenges of the SoC design process. The run time and the complexity of different CDN design approaches become a concern with the increase in the number of sinks to be connected to the CDN. The new challenges that are imposed by the SoC require different design approaches for the CDN. The main shortcoming of previous approaches is that they are not incremental; and hence, the whole solution has to be recalculated whenever a design change is required. The main challenge is to have a CDN design approach that can handle minor modifications in the system. Other shortcoming of previous approaches is that the locations of the Steiner points of a CDN are deter-

mined regardless of the routabilty of the wires. Indeed, most of the CDN approaches solve the skew problem by determining the required wire lengths between different points of the CDN without considering the realization of such wire lengths. Furthermore, most of these approaches resort to wire elongation in order to achieve the skew requirements. However, it is not always possible to route a wire with a specific length due to the difficulties of the routing task. Also, it is often the case that vias are required in order to avoid routing through obstacles and to route the wire in different layers. Thus, it is necessary to consider the routability of each wire of the CDN during the determination of the CDN in order to meet the timing requirements.

1.3. Overview of the Dissertation

This work provides one of the first studies of CDN redesign, called CDN tuning, which focuses on incremental synthesis of the CDN in order to meet the timing constraints. A novel algorithm, called Adaptive Wire Adjustment (AWA), is proposed here with a main goal to enable a quick Engineering Change (EC) to the CDN [64]. A secondary goal is to reduce the CDN wire length and number of wire elongations under to help with routing and to minimize power consumption [60-63]. This thesis is organized as follows:

Chapter 2 covers background material related to the work presented in this thesis. The chapter presents an overview of the skew problem; and different approaches that have been used to solve it. The chapter further presents different algorithms that are used to generate the initial topology.

Chapter 3 presents a new incremental algorithm, called Adaptive Wire Adjustment, in order to manage small modifications in the CDN. It also details the proof of convergence of the algorithm. The chapter further presents various figures of merit used to evaluate AWA and other algorithms. The issue of small modifications are addressed by devising different scenarios that are similar to the scenarios of small modifications in a SoC design process.

Chapter 4 presents a method of Local Topology Modification (LTM) in order to enhance the convergence of AWA algorithm. A new method to select the topology using a search tree will be presented. The chapter further introduces figures of merit to demonstrate the advantage of using the LTM method into AWA and other CDN synthesis algorithms.

Chapter 5 studies the shape of the routing area of an interconnect of a CDN under obstacle constraints. An obstacle might be a pre-placed macro cell with blocked region or a previously routed wire. The chapter presents a simple model, called Shortest Paths Polygon (SPP), to describe all shortest paths between two points in the presence of obstacles. The chapter also presents the incorporation of the SPP model into AWA algorithm in order to tune the CDN under obstacle constraints.

Chapter 6 focuses on the obstacles that stand for previously routed wires. A method is presented to determine the SPP of each wire of the CDN such that the wires do not intersect with each other. Specifically, the CDN is routed in one and two layers. The chapter presents a comparison between this method and other methods. It shows the advantage

of using the SPP model in terms of reducing the total wire length. Furthermore, the impact of the planarity on the total wire length will be studied.

Finally, Chapter 7 concludes and summarizes the contributions provided in this work. The chapter also discusses future research directions in designing CDNs.

Chapter 2

Clock Distribution Network

2.1 Introduction

The clock skew problem is a direct result of the feature reduction; and it is the main problem that has to be resolved by any CDN synthesis approach in order to meet the system requirements. This chapter discusses this problem and different approaches that address it. The following sections provide the background and the terminology of the issues involved in this thesis. Throughout this thesis, a simple synchronous system is considered using positive edge triggered flip flops under a single phase clocking scheme.

The Clock Distribution Network (CDN) refers to the circuits and interconnections that are used to deliver the clock signal to all registers and dynamic circuits in a synchronous system.

2.2 Clock Skew

A typical scenario for a synchronous system is a data flow between two registers, *i* and *j*, as shown in Figure 2.1. Such registers are called sequentially adjacent since there is a

combinational logic (or just an interconnection) that connects the output of *i* to the input of *j*. If *i* and *j* are properly functioning, then the data latched by *i* must be captured by *j* with the next latching clock edge. However, the delay of the interconnection of the CDN results in different arrival times of the clock signal to the registers *i* and *j*, T_i and T_j respectively. The difference between T_i and T_j is called the clock skew between *i* and *j*, S(i,j); and it is given as:

$$S(i, j) = T_i - T_j \tag{2.1}$$

Note that S(i,j) can be positive or negative depending on whether the T_j lags or leads T_i.

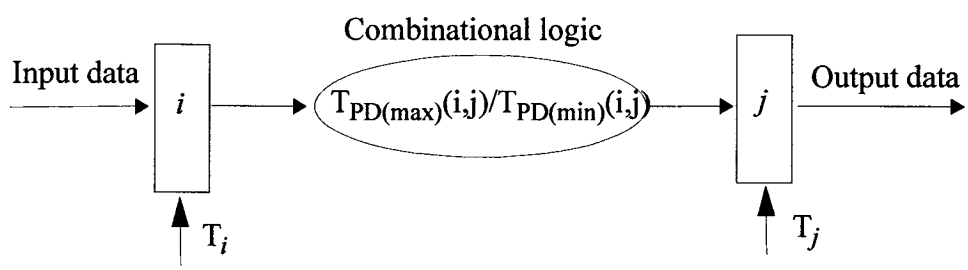


Figure 2.1 Pair of sequentially adjacent registers

On the other hand, data signal encounters a delay when it propagates from *i* to *j*. The data propagation delay between two adjacent registers, $T_{PD}(i,j)$, is due to the registers, combinational logic and the interconnection; and it can be determined as follows:

$$T_{PD}(i, j) = T_Q(i) + T_C(i, j) + T_{int}(i, j) + T_s(j) \tag{2.2}$$

Where:

$T_Q(i)$: the switching delay of register *i*

$T_C(i,j)$: the propagation delay of the combinational logic between the registers *i* and *j*

$T_{int}(i,j)$: the propagation delay of the interconnect between *i* and *j*

$T_s(j)$: the required time for data to be stable in order to be latched by *j*

Note that there are minimum and maximum values for $T_{PD}(i,j)$, $T_{PD(min)}(i,j)$ and $T_{PD(max)}(i,j)$ respectively, depending on the minimum and maximum values of different delay components.

In order to latch the correct data by *j*, the clock skew and the data propagation delay between *i* and *j* have to be controlled. In more details, two constraints must be satisfied:

1) the data that propagates from *i* should not reach *j* before the arrival of the clock signal at *j*, T_j. Otherwise, the data will propagate through *i* and *j* in the same clock cycle (double clocking or race condition). Mathematically, the skew should satisfy the following relationship:

$$S(i,j) \geq T_H(j) - T_{PD(min)}(i,j) \tag{2.3}$$

where $T_H(j)$ is the hold time of register *j*.

2) the data should not reach *j* with a delay greater than the time difference between the next clock edge at *j* and the current clock edge at *i*. Otherwise, the data will not be latched by *j* at the following clock pulse (zero clocking). Mathematically, the skew should satisfy the following relationship:

$$S(i,j) \leq T_{clk} - T_{PD(max)}(i,j) \tag{2.4}$$

where T_{clk} is the clock cycle time

The magnitude and polarity of S(i,j) and $T_{PD}(i,j)$ affect the system performance. For example, the later constraint determines the maximum clock frequency, f, as follows:

$$\frac{1}{f} \geq S(i, j) + T_{PD(max)}(i, j) \tag{2.5}$$

Based on the above discussion, Equations 2.3 and 2.4 define, respectively, the lower ($S_{min}(i,j)$) and upper ($S_{max}(i,j)$) bounds of the permissible range of S(i,j) as shown in Figure 2.2.

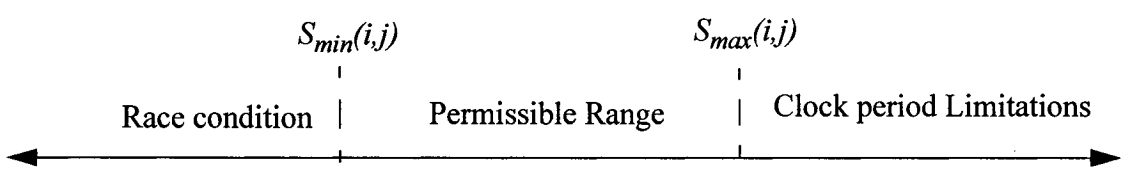


Figure 2.2 The permissible clock skew range for two registers.

Two registers may be connected by more than one data path as shown in Figure 2.3, where the system is modeled as a directed graph. Each vertex refers to a register; and each edge refers to a combinational logic path (or just a wire) between two registers. In addition, each edge has two weights that correspond to the maximum and minimum data propagation delay between its two vertices. For example, the registers 1 and 3 shown in Figure 2.3(a and b) are connected by two data paths; and each path has a different permissible skew range. Thus it is important to determine the permissible skew range properly in order to ensure that *3* latches the correct data.

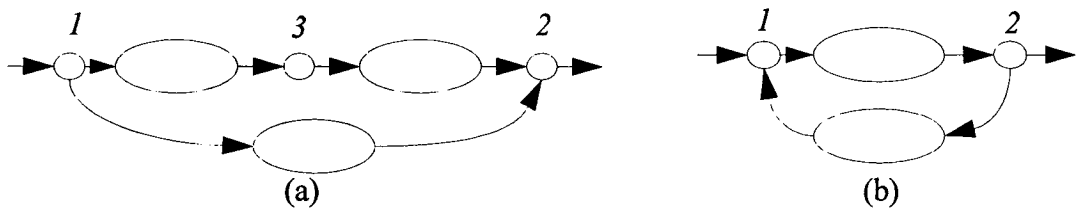


Figure 2.3 Global data path (a) A Parallel data path (b) A feedback data path

2.3 Preliminaries of The CDN Synthesis

2.3.1 Graphical Model of the CDN

Consider a set of registers R={1, 2,., n} to be connected by a CDN. Each register $i \in$ R, is associated with a load capacitance, C_i, and a location in the Manhattan plane l_i. The set R can be connected by different CDNs depending on the approach used to generate the CDN. The objective of the early CDN design approaches was the minimization of the total wire length. This is due to the fact that the interconnect delay was insignificant compared to the device delay. As such, the CDN design problem is reduced to the problem of connecting a set of registers with minimum wire length, which is known as the Steiner tree problem. Indeed, the CDN would have a tree topology since there is a unique path from the clock source to each register (the topology here refers to the graph that represents the connection between registers and the clock source). Furthermore, the Steiner tree connects the set R through intermediate nodes, called Steiner points, so that the total wire length is minimized.

The connection between the set R and the Steiner points can be depicted by a topology tree T. The leaves of T correspond to the registers and the internal nodes in T correspond

to the Steiner point in the CDN as shown in Figure 2.4. Any node k, k∈T, is connected to its parent by an edge e_k. For any two nodes w and k, where w is the ancestor of k, there is a unique path from w to k in T; which is denoted as *Path(w,k)*.

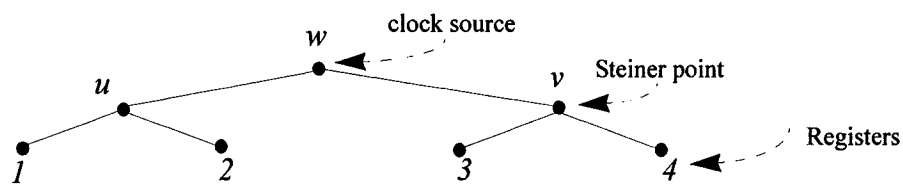


Figure 2.4 The topology tree of a CDN

2.3.2 Delay Model of the CDN

The edge of node k, e_k, in the CDN can be modeled as a π-type circuit with a resistor r_k and two capacitors $c_k/2$, where r_k and c_k are the resistance and capacitance of e_k respectively. Consequently, the CDN can be modeled as an RC tree as shown in Figure 2.5. Under the Elmore delay model, the arrival time of the clock signal to a node, k, is given by [3]:

$$\mathrm{Delay}(w, k) = \sum_{i \in \mathrm{Path}(w,\,k)} r_i\left(\frac{c_i}{2} + C_i\right) \tag{2.6}$$

where: C_i is the total capacitance of the sub-tree rooted at node i

w is the source of the clock signal.

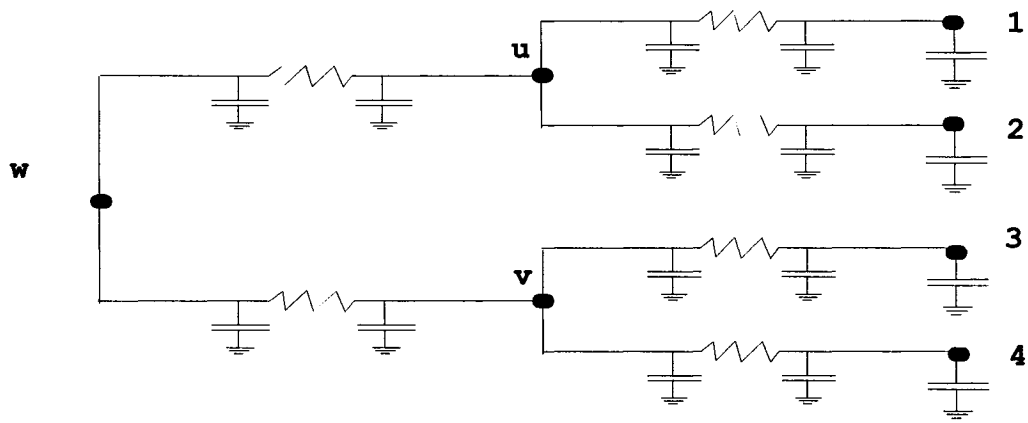


Figure 2.5 The RC-Tree of the topology tree shown in Figure 2.4

2.3.3 Power Dissipation of the CDN

Each transition of the clock signal changes the state of each node in the CDN. Consequently, the primary component of the power dissipated by the CDN is the dynamic power due to charging and discharging the capacitance load of the CDN. The dynamic power dissipated by CDN depends on the capacitance of the CDN, swing voltage and clock frequency as follows [24-29]:

$$P = C_L V_{DD}^2 f \tag{2.7}$$

Where: C_L is the CDN capacitance

V_{DD} is clock signal swing

Normally, the CDN consumes around 40% of the system power consumption [24-29]. Minimizing the power of the CDN can be achieved by minimizing the signal swing and/ or CDN capacitance. Minimizing the CDN capacitance requires the reduction of the

total wire length of the CDN. Thus, it is important to design the CDN with minimum wire length in order to reduce the power dissipation.

2.3.4 Problem Definition

For a given set of registers R, where each register is associated with a location and a load capacitance, let the skew constraint between every pair of registers be given by the set $S=\{S(i,j)=(S_{min}(i,j),S_{max}(i,j))\}$. The CDN design problem can be formulated as follows: *Find the CDN that satisfies the skew constraints S with minimum wire length.*

The CDN synthesis requires the determination of the location and the edge length of each node in the CDN. If there is no skew constraints, then the problem becomes a Rectilinear Steiner Tree problem. This problem is NP-hard, and much work has been devoted to designing good heuristics and approximation algorithms [3, 71, 73, 77]. Nevertheless, such an assumption is not true, and the skew constraints must be carefully determined and satisfied.

A close look into the relation 2.3, which determines $S_{min}(ij)$, reveals that it is usually $S_{min}(i,j)<0$ since $T_H<T_{PD}$. On the other hand, according to the relation 2.4, it is usually true that $S_{max}(i,j)>0$ since $T_{PD} < T_{clk}$. Thus, one may deduce that a zero skew CDN (ZSCDN), or a bounded skew CDN (BSCDN), would satisfy the skew requirements.

2.4 CDN Synthesis Using Geometric Approaches

Early approaches of designing a ZSCDN are based on linear delay model. According to

this delay model, the clock signal down a path in the CDN is assumed to increase linearly with the length of the path. In general, there are three geometric approaches considered in this thesis: H-tree, method of means and median and geometric matching algorithm.

2.4.1 H-Tree

Let the set of registers, R, to have a symmetric distribution such that the set R can be divided into two symmetric sets recursively and alternatively by vertical and horizontal lines until each set has one register only. Then, it is possible to connect the set R using recursive H-shapes; and the resulting CDN is called H-tree as shown in Figure 2.6. The clock source is connected to the center of the first H structure; and the four corners of each H-structure provide inputs to the next level of the H tree, and so on. The final destination points of the H-tree are used to drive the local registers.

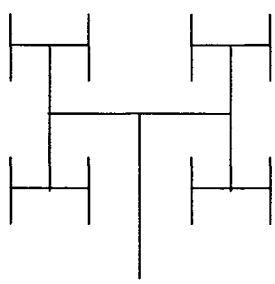


Figure 2.6 The H-tree

The H-tree increases the path lengths from the center of the first H to all registers so that all paths become equal. That is, the H-tree method trades of the total wire length for zero skew. In addition, the H-tree requires that the registers to have a symmetric distribution.

Also, all registers must have equal capacitance load in order to achieve a zero skew between all registers.

2.4.2 Method of Means and Medians (MMM)

At each iteration, The method sorts a set of registers R according to their x-coordinates (or y-coordinates). Then the method finds the median of the x-coordinate (or y-coordinate) of all elements of R; and groups the elements into two sub-sets based on whether each element is located to the left or right of the median. This process continues recursively for each sub-set from a previous iteration with alternation between the x and y coordinates until each sub-set has two registers at most as shown in Figure 2.7.

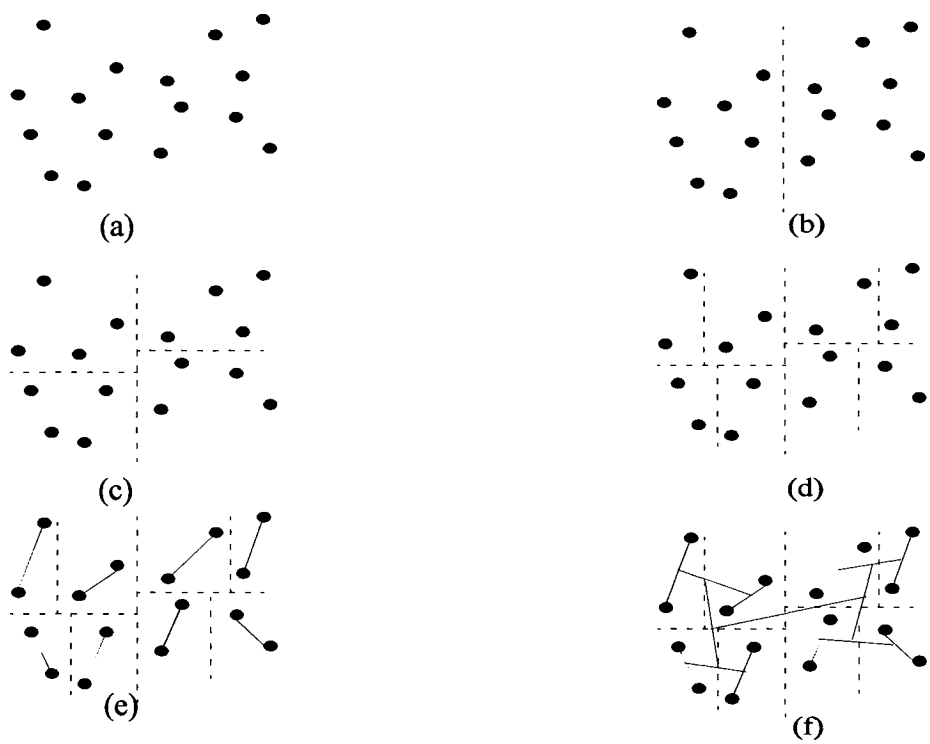


Figure 2.7 The MMM algorithm

2.4.3 Geometric Matching Algorithm (GMA)

This algorithm constructs the CDN by matching the closest nodes iteratively in a bottom up manner. In the first iteration, the algorithm finds the pair of registers that are closest to each other in a given set of registers R. The pair of registers are replaced by a node in R; where this node becomes the parent of the pair of registers. At each iteration, the algorithm reduces the number of nodes in R by one as shown in figure 2.8. The algorithm stops when R is reduced to one node that stands for the root of the CDN. Due to the process of selecting the two nodes that are closest to each other, the GMA algorithm takes longer time to reach its solution as compared to the MMM. However, the GMA produces a better CDN in terms of total wire length.

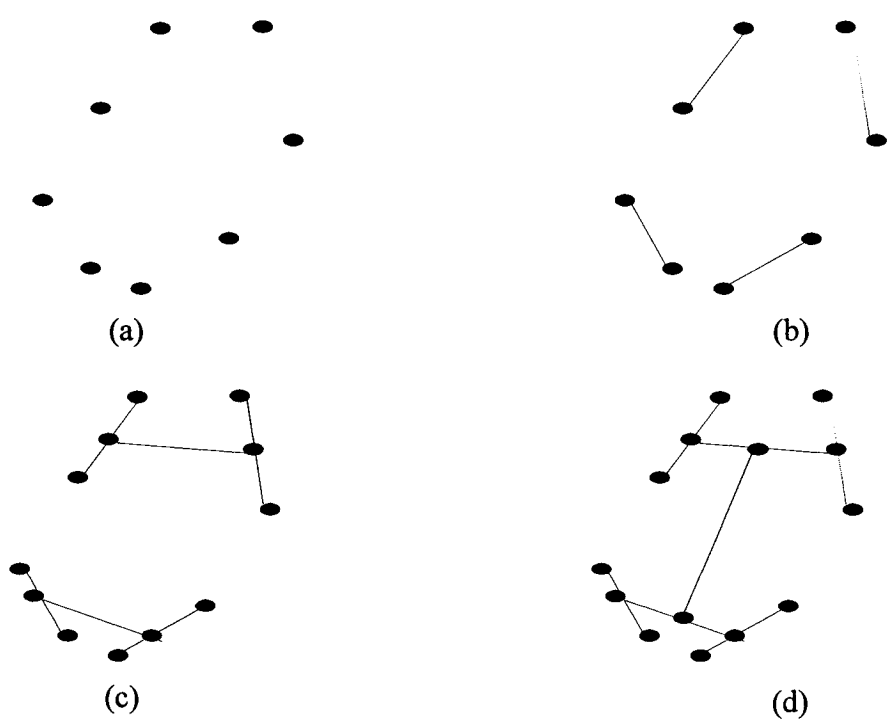


Figure 2.8 The GMA algorithm

2.5 CDN Synthesis Using Elmore Delay Model

Various CDN synthesis algorithms were developed based on Elmore delay model to generate relatively accurate solutions [3]. Some of these approaches require an initial connection topology of the CDN. Such a topology can be generated using the MMM or GMA approaches; and it is traditionally a tree since there is a single path from the root to each leaf.

2.5.1 Tsay's Method

Tsay proposed a method that produces a ZSCDN based on Elmore delay model [39]. It requires the topology tree of the CDN to be defined initially. Then, the method determines the edge lengths of the tree such that the clock signal arrives to all leaves, registers, at the same time. Specifically, the Steiner points of the tree are selected iteratively in a bottom-up manner; and the two child edge lengths of the selected Steiner node are determined. Assume that during the progression, Steiner point w is reached, where u and v are the children of w as shown in Figure 2.9. This implies that the method has determined the edges of the sub-tree rooted at u (and v) such that the clock delays from u (and v) to all its leaves are equal. Let t_u and t_v be the clock delays from u and v to their leaves respectively. In order to achieve a zero skew between all the leaves that are connected to w, the edge lengths of u and v, e_u and e_v respectively, have to satisfy the following equation:

$$e_u r_0\left(\frac{e_u c_0}{2} + C_u\right) + t_u = e_v r_0\left(\frac{e_v c_0}{2} + C_v\right) + t_v \tag{2.8}$$

where r_0 and c_0 are the resistance and capacitance per unit length of wire respectively

Let the rectilinear distance between u and v be D. Also, let x be a factor, $0 \le x \le 1$, that denotes the partitioning of D between e_u and e_v such that:

$$e_u = xD \qquad e_v = (1-x)D \tag{2.9}$$

Then, Equation 2.8 can be written as follows:

$$xDr_0\left(\frac{xDc_0}{2} + C_u\right) + t_u = (1-x)Dr_0\left(\frac{(1-x)Dc_0}{2} + C_v\right) + t_v \tag{2.10}$$

Solving Equation 2.10 for x yields:

$$x = \frac{(t_v - t_u) + r_0 D\left(\frac{Dc_0}{2} + C_v\right)}{r_0 D(c_0 D + C_u + C_v)} \tag{2.11}$$

If the resulting x is $0 \le x \le 1$, then it is possible to balance the two subtrees by a wire of length D; and e_u and e_v are determined according to Equation 2.9. However, if $x \le 0$ (or $x \ge 1$) then a wire of length D cannot balance the skew between the two subtrees. The case $x \le 0$ implies that $t_u >> t_v$; and hence, the node w must coincide on node u. That is $e_u = 0$; and e_v can be calculated from Equation 2.8 as follows:

$$e_v = \frac{\sqrt{(r_0 C_v)^2 + 2r_0 c_0 (t_u - t_v)} - r_0 C_v}{r_0 c_0} \tag{2.12}$$

Similar equation can be derived for the case $x \geq 1$. The cases $x \leq 0$ and $x \geq 1$ imply that $e_v > D$ and $e_u > D$ respectively. Consequently, there would be a wire elongation as shown in Figure 2.10. Such a wire elongation results in an increase in the chip area and power consumption; and in addition, it is cumbersome for the CDN routing.

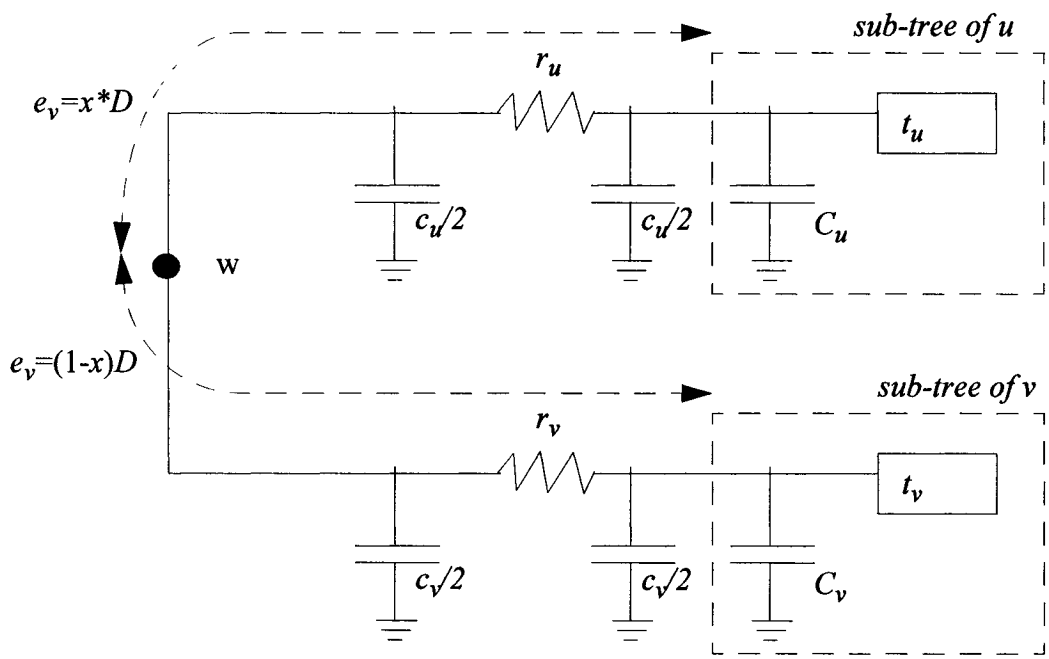


Figure 2.9 The determination of the edge lengths using Tsay approach

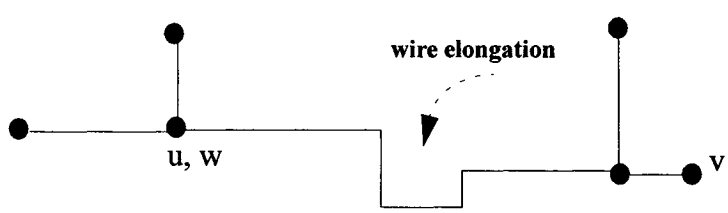


Figure 2.10 The wire elongation

2.5.2 Differed Merge Embedding (DME) Algorithm

The Tsay method determines the edge lengths of the tree such that the skew become zero

between all registers. However, this method does not consider all locations of the Steiner points in the routing plane. For example, node w shown in Figure 2.9 can be located in different locations for a given e_u and e_v. Considering all possible locations of each Steiner node would help to minimize the total wire length of the CDN. This shortcoming was addressed by the Differed Merge Embedding (DME) algorithm [40-42]. The DME algorithm uses the Tsay method, which is based on Elmore delay model, in order to determine the edge lengths. In addition, it determines all possible locations of each Steiner point in the Manhattan plane, which will be called the merging segment. The merging segment of a node is a Manhattan arc whose slope is ±1 as shown in Figure 2.11. The DME reaches its solution in two phases: a bottom-up phase and top-down phase as will be described next.

The first phase determines the merging segment of each Steiner node in bottom-up mannar. Assume that the algorithm has reached the Steiner node w whose child nodes are u and v as shown in Figure 2.9. The edge lengths e_u and e_v are determined using Tsay method. Then the merging segment of w, *ms(w),* is determined such that *ms(u)* and *ms(v)* can be connected with the edge lengths e_u and e_v. Note that the points that have equal distances form a Manhattan arc constitute a *Tilted Rectangle*, TR, as shown in Figure 2.11. The Manhattan arc of a TR is called the core of the TR; and the distance between the core and the border of the TR is called the radius of the TR.

Let TR_u be the TR of node u such that the core is *ms(u)* and the radius is e_u as shown in

figure 2.11. Similarly, let TR_v be the TR of node *v*. Then, *ms(w)* would be the intersection of TR_u and TR_v:

$$ms(w)=TRR_u \cap TRR_v \qquad 2.13$$

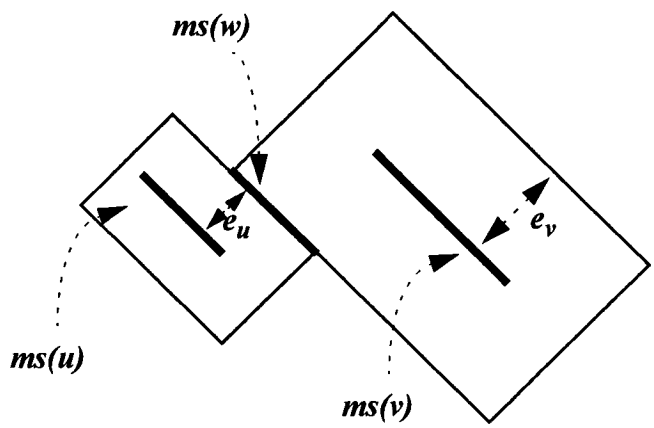


Figure 2.11 Calculating the merging segment of *w* during the first phase when 0<x<1

The output of the first phase of the DME is a tree of merging segments that will be used as input for the second phase. The second phase selects the exact location of each internal node from its merging segment in a top-down manner. Assume that the procedure has reached node *u*, where the location of its parent *w*, l_w, has already been decided. Then, l_u is determined by constructing a TR_w whose core is l_w and radius is e_u as shown in Figure 2.12. Since the *ms(w)* is constructed from the intersection of tilted regions of

its children, a portion of *ms(u)* will lay in TR_w, which can be denoted as $ms(u) \cap TR_w$. The final location of *u* is a point that belongs to $ms(u) \cap TR_w$ [41].

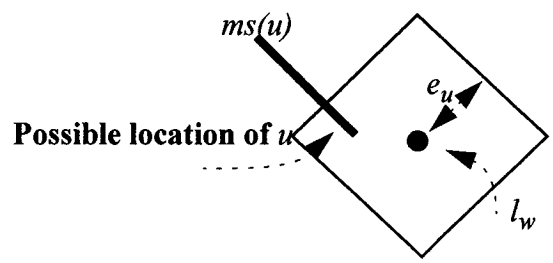


Figure 2.12 Selecting the exact location of a node from its merging segment

The DME algorithm also requires an initial topology. The type of the initial topology has a significant impact on the total wire length of the resulting ZSCDN. In general, using a topology obtained by GMA would yield less total wire length. Since, the initial topology is generated without considering the capacitance load of the CDN, the total wire length of the synthesized tree can be minimized further by generating the topology and the ZSCDN simultaneously as will be described next.

2.5.3 Greedy Differed Merge Embedding (GDME) Algorithm

The Greedy DME is the best known ZSCDN design approach in terms of total wire length [42]. This algorithm generates a ZSCDN in two phases. The first phase determines, in a greedy bottom-up manner, the edge lengths, the merging segments and the topology. The edge lengths are determined using Tsay method; and the merging segments are determined using the DME algorithm. The edge length and the merging seg-

ment of a Steiner node are determined by defining the child nodes of the Steiner node as will be described next.

Consider a set of registers, R. Each register stands for a node in the CDN, where each node is associated with a merging segment of zero length. Let the set of nodes be Q, where Q=R at the beginning of the algorithm. At each iteration, the GDME algorithm finds the pair of nearest neighbors in Q. Let these two nodes be *u* and *v*. That is, the distance between *ms(u)* and *ms(v)* is the minimum. The two selected nodes, *u* and *v*, will be replaced by a new node, *w*, in Q. In the final solution, the node *w* becomes the parent of *u* and *v*. The merging segment of *w*, *ms(w)*, is determined from *ms(u)* and *ms(v)* as described in Section 2.5.2.

The bottleneck of GDME is the selection of the pair of nodes at each iteration. The complexity of GDME is $O(n^2)$ [42]. The computation time of GDME can be reduced by finding several nearest neighbor pairs simultaneously; and the complexity is enhanced to $O(n \log n)$. This can be achieved using an efficient implementation of the nearest neighbor computation based on the Delauny triangulation[43].

2.6 Shortcomings of Previous Approaches

The main shortcoming of the previous CDN synthesis approaches is that they are not incremental. As a result, the solution has to be re-calculated whenever a modification is required. Such a modification is expected often in the course of designing a SoC. Also,

to improve system performance, some level of tuning the synthesized CDN is required. Therefore, the need for incremental algorithms to tune the network or to implement Engineering Changes (ECs) in CDN is critical in the SoC design cycle.

In addition, previous approaches satisfy the time requirements by determining the location of each Steiner node of the CDN in the Manhattan plane. However, these approaches cannot guarantee the routablity of the wires that connect the registers through the Steiner nodes to the source of the clock tree. In fact, routing the DME or GDME solutions result in many intersections between the wires. The intersections might be resolved by using vias or lengthening the wires, but the solution would not necessarily meet the time budget. This problem becomes advent due to the increase in the difficulties of the routing problem in complex systems. Thus, it is important to produce a solution that solves both the timing and routing problems.

In this thesis, an incremental synthesis for small design modifications will be presented first. Then, the issue of routability of the CDN in the presence of obstacles will be discussed later.

Chapter 3

Tuning the Clock Distribution Network

3.1 Introduction

Designing a Bounded Skew CDN (BSCDN) in a SoC has become increasingly challenging, as the clock network consumes an increasing portion of all resources in terms of wiring area, power and design time. Furthermore, it is sometimes required to insert or remove IPs or to perform localized modifications without affecting the performance of the CDN. Hence, there is a need for implementing the EC by tuning the BSCDN instead of redesigning it.

The approaches that are presented in Chapter 2 produce Zero Skew CDN (ZSCDN) based on Elmore delay model. These approaches control the skew by using Tsay's approach, which may resort to wire elongation [40-47]. Such a wire elongation compounds the complexity and increases the power consumption of the CDN. Furthermore, a change in the locations or the load capacitance of clock pins would require a recalculation of the whole CDN solution as in Deferred Merging Embedded (DME) algorithm

[40-42]. This will be an extremely computation intensive process for a complex system. In fact, it is often that IPs are inserted or removed, which changes the CDN's topology and the loading and placement of the clock pins [48]. Such changes may affect the entire CDN and may require a redesign of the clock network. However, in complex system, it is only practical to implement these minor modifications through an incremental algorithm to tune the CDN. This specific process has received little attention in the literature. Elboim et. al. [54] have reported that the repetitive redesign of the CDN can be eliminated by inserting programmable delay circuits in the CDN in order to tune the clock delay. Indeed, these delay circuits enable a quick integration of the IPs into SoC by avoiding the redesign of the CDN itself. However, the same goal can be achieved by enabling a quick redesign of the CDN and thus avoiding the penalty of introducing extra circuitry, which may result in an increased area and power dissipation. This work provides one of the first studies of redesigning CDN, called CDN tuning, using the physical information of the CDN so that it better meets the timing constraints. A novel algorithm, called Adaptive Wire Adjustment (**AWA**), is proposed with a main goal to enable a quick EC to the CDN [55, 64]. A secondary goal is to reduce the CDN wire length and number of wire elongation to help with routing and to minimize power consumption.

3.2 Clock Distribution Network Tuning by Adaptive Wire Adjustment

The Adaptive Wire Adjustment (AWA) algorithm [55] takes a CDN as an input in the form of a linked tree data structure and checks for the maximum skew in the tree then minimizes it by adjusting the wires of the tree. Whenever a skew exists between two

leaves, the paths to these two leaves must be adjusted so that the skew becomes smaller. To adjust these two paths, the algorithm finds the node that can be shifted, by adjusting its child edges, so that the skew is minimized. Such a node, called the **Balancing Node, BN**, is the first ancestor of the same two leaves. For example, consider the CDN shown in Figure 3.1. In order to minimize the skew between leaves *1* and *3*, their BN, *w,* is shifted by adjusting its child edges e_u and e_v. If the leaf *1* suffers higher delay than leaf *3*, then *w* must be shifted by Δ towards node *u* and away from node *v*. Thus, the edge e_u decreases and edge e_v increases in the following manner:

$$e'_u = e_u - \Delta \quad , \quad e'_v = e_v + \Delta \tag{3.1}$$

where, e'_u and e'_v are the new values of e_u and e_v.

In order to determine the edge adjustment, Δ, the delay from node *w* to its leaves have to be considered. In fact, the leaf delays of the subtrees rooted at *u* and *v* span ranges in the time domain as shown in Figure 3.2. Note that the leaf delays of a sub-tree are calculated from the root of the sub-tree. Consequently, in order to minimize the skew between the two sub-trees, the ranges of *u* and *v* must be shifted in opposite directions. Let us define the leaf delay range of a sub-tree rooted at node α, $\boldsymbol{Range_\alpha}$, be the maximum and the minimum delays in that range calculated from the node α as follows:

$$Range_\alpha = Max_\alpha - Min_\alpha \tag{3.2}$$

where Max_α=Maximum{Delay(α,β)}

Min_α=Minimum{Delay(α,β)}

for every leaf β∈subtree rooted at α

In general, the balancing node can be defined as the node that has the maximum range and belongs to the lowest level in the tree. Let the center of the range of a node α, ***Centre***$_{\alpha}$, be the average of maximum and minimum delays of that range as follows:

$$\text{Centre}_{\alpha} = \frac{\text{Max}_{\alpha} + \text{Min}_{\alpha}}{2} \tag{3.3}$$

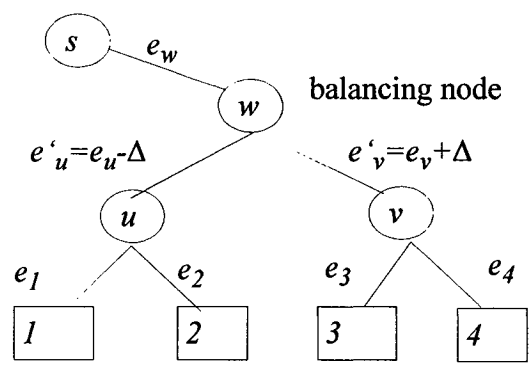


Figure 3.1 Selecting the balancing node in a tree

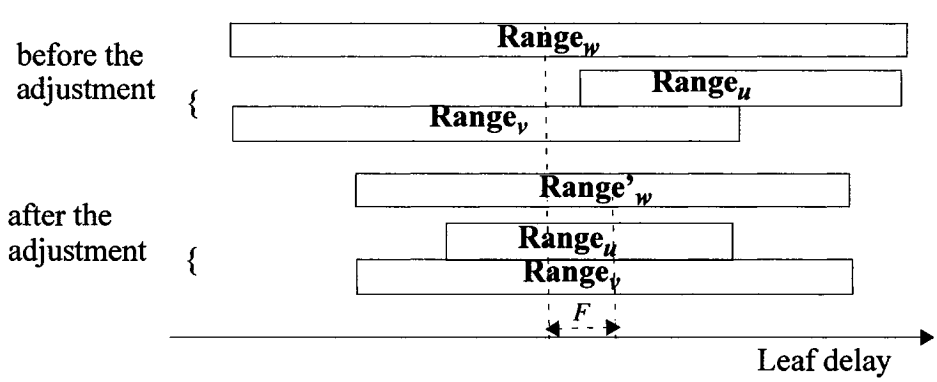


Figure 3.2 The leaf delay ranges of the sub-trees rooted at nodes *u, v* and *w* of Figure 3.1

Using the range centre as a time reference, the skew between a pair of internal nodes, *u* and *v*, is defined as:

$$S(u, v) = \left[e_u r_0\left(\frac{e_u c_0}{2} + C_u\right) + Centre_u\right] - \left[e_v r_0\left(\frac{e_v c_0}{2} + C_v\right) + Centre_v\right] \tag{3.4}$$

Equation 3.4 can be used to determine a BN's edge adjustment, Δ, so that the skew between the BN's children, *u* and *v*, becomes zero by substituting e_u and e_v by e'_u and e'_v respectively in Equation-6 and equating it to zero as follows:

$$\left[(e_u - \Delta)r_0\left(\frac{(e_u - \Delta)c_0}{2} + C_u\right) + Centre_u\right] - \left[(e_v + \Delta)r_0\left(\frac{(e_v + \Delta)c_0}{2} + C_v\right) + Centre_v\right] = 0 \tag{3.5}$$

$$\Delta = \frac{\frac{r_0 c_0}{2}\left(e_u^2 - e_v^2\right) + (Centre_u - Centre_v) + r_0(e_u C_u - e_v C_v)}{r_0(c_0(e_u + e_v) + C_u + C_v)} \tag{3.6}$$

The value of Δ refers to the required adjustments for the child edges of the BN, e_u and e_v, in order to minimize the skew.

Adjusting e_u and e_v by Δ would shift the BN along a **Balancing Segment, BS**, which represents all possible locations of the BN. The BS of *u* and *v*, BS_{uv}, can be determined from the intersection of two tilted rectangles, TR_u and TR_v, whose centers are nodes *u* and *v* and whose radii are e'_u and e'_v respectively as shown in Figure 3.3(a). Note that this is true only when

$$e'_u + e'_v = d(u,v) \tag{3.7}$$

where *d(u,v)* is the distance between the locations of *u* and *v*.

If $e'_u + e'_v > d(u,v)$, then the intersection of TR_u and TR_v is a TR as shown in Figure 3.3(b). For such a case, it might be possible to shorten both e'_u and e'_v while minimizing the skew between u and v. But, this is not always the case. Specifically, if there is a need for wire elongation, then $e'_u + e'_v > d(u,v)$. For such cases, one of the edges, e'_u or e'_v, would be zero; and the BN coincides on the nodes whose edge is zero.

For the general case, the final location of the BN, w, is determined by the intersection of BS_{uv} and TR_s whose centre is s and its radius is the BN edge e_w as shown in Figure 3.3(a). The BN's edge, e_w, has not been considered yet, and its length would affect the total wire length. In fact, the total wire length can be minimized by determining the BN's edge as the minimum length between the BS and the BN's parent.

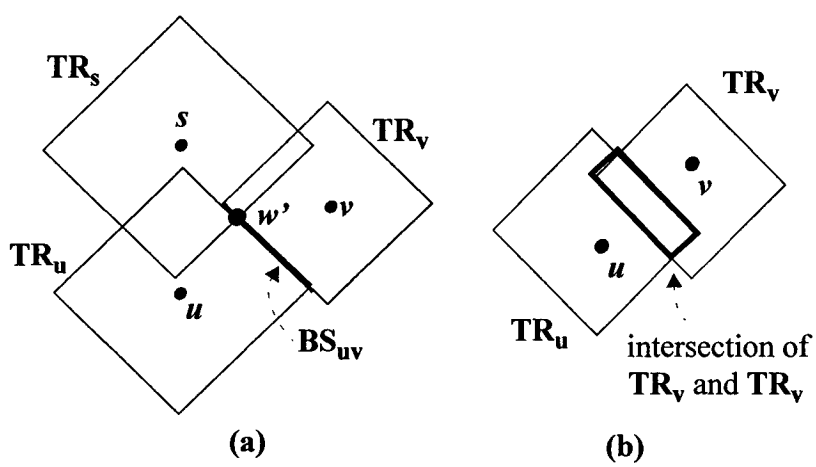


Figure 3.3 (a) Determination of BS_{uv} and the new location of the balancing node, w'. (b) The intersection of TR_u and TR_v is a tilted rectangle when $e_u + e_v > Rect_{uv}$

Adjusting the BN's child edges, as described before, would minimize the maximum

skew between the leaves of the BN. As such, the BN's range gets smaller as the child ranges are shifted in opposite directions so that their centres coincide on each other as shown in Figure 3.2. However, if the centres of the child ranges have already coincided, then it is impossible to shorten the BN's range by adjusting its child edges. Also, if the BN's range is equal to one of its child ranges, then shifting the child ranges would not help to minimize the BN's range. In such a case, the BN's child that has the greater range have to be selected as the balancing node. From the previous discussion, we can draw the AWA algorithm to achieve a bounded skew for a given tree by minimizing the maximum skew in the tree iteratively. At each iteration, a BN is selected and its child edges are adjusted according to Equations 4-8. The description of AWA algorithm is shown in Figure 3.4.

Input: *Initial CDN, as a linked tree T, for a set of clock pins R*
Output: *Bounded Skew CDN by B*

while *maximum skew >B:*
 Find *the BN that has the maximum skew in* ***T****, and it belongs to the lowest level in T*
 Calculate the required adjustment, Δ, to balance the child nodes of the BN, u and v.
 Construct $\boldsymbol{TR_u}$ *and* $\boldsymbol{TR_v}$ *as following:*
  $core(\boldsymbol{TR_u}) = location\ of\ u,\ radius(\boldsymbol{TR})=e'_u$
  $core(\boldsymbol{TR_v}) = location\ of\ v,\ radius(\boldsymbol{TR_v})=e'_v$
 Calculate the $\boldsymbol{BS_{uv}}$ *as* $\boldsymbol{TR_u} \cap \boldsymbol{TR_v}$
 Locate ***BN*** *at the closest point of* $\boldsymbol{BS_{uv}}$ *to the parent of BN, s.*
 Find the maximum skew in T

Figure 3.4 The AWA algorithm

The AWA algorithm can narrow the skew in a given tree iteratively; and hence the whole solution need not be recalculated when a smaller skew is required. In fact, the iterative nature of the AWA differentiates it from other algorithms, such as Tsay's approach or DME algorithm. This feature of AWA is important when a minor modification in the clock pins is required. Indeed, AWA can save the calculation time required for the generation of the CDN while managing a small modification during the design iterations of the digital system [55]. However, minimizing the BN's range at each iteration does not necessarily lead immediately to a smaller skew in the tree, and this will impact the convergence of AWA as will be described in the next section.

3.3 Convergence of AWA Algorithm

At each iteration of AWA, the child edges of the BN are adjusted so that the range of the BN gets smaller. Unfortunately, such an adjustment affects the skew between the leaves that belong to the BN and other leaves in the whole tree, which may result in an increase in the skew in the tree. This section shows that selecting an ancestor node as a BN in later iterations will ensure the minimization of the skew further; and AWA can achieve the desired bound of skew ultimately.

As described in the previous section, the adjustment of the child edges of the BN minimizes the range of the BN by shifting the child ranges in opposite direction. However, the child ranges cannot be shifted in opposite direction if the centres of the child ranges have already coincided, or if the BN's range is equal to one of its child ranges. This can be stated as follows:

Lemma 3.1: Let $\boldsymbol{Range_u}$, $\boldsymbol{Range_v}$ *and* $\boldsymbol{Range_w}$ *be the ranges of u, v and w respectively such that* $\boldsymbol{Range_w} > \boldsymbol{Range_u}$ *and* $\boldsymbol{Range_w} > \boldsymbol{Range_v}$ *where u and v are the children of w. If* $\boldsymbol{Centre_u}$ *and* $\boldsymbol{Centre_v}$ *are not coincided on each other, then shifting* $\boldsymbol{Range_u}$ *and* $\boldsymbol{Range_v}$ *in opposite direction so that* $\boldsymbol{Centre_u}$ *and* $\boldsymbol{Centre_v}$ *get closer would result in* $\boldsymbol{Range'_w} < \boldsymbol{Range_w}$, *where* $\boldsymbol{Range'_w}$ *is the range of w after the shift.*

Proof: Let D and D' be the difference between $\boldsymbol{Centre_u}$ and $\boldsymbol{Centre_v}$ before and after shifting the ranges respectively. Shifting $\boldsymbol{Range_u}$ and $\boldsymbol{Range_v}$ so that $\boldsymbol{Centre_u}$ and $\boldsymbol{Centre_v}$ get closer implicitly gives D'<D. Thus, the case that $\boldsymbol{Range'_w} > \boldsymbol{Range_w}$ is impossible as it implies that D'>D. Furthermore, the case that D'<D and $\boldsymbol{Range'_w} = \boldsymbol{Range_w}$ would happen only if $\boldsymbol{Range_w}$ is equal to either $\boldsymbol{Range_u}$ or $\boldsymbol{Range_v}$, which is contradictory to the statement of the Lemma. Hence $\boldsymbol{Range'_w} < \boldsymbol{Range_w}$ must be true.

Indeed, the minimum range of a BN can be achieved only by shifting the ranges of the BN's children in opposite directions till their centres coincide as stated below:

Lemma 3.2: Let $\boldsymbol{Range_u}$, $\boldsymbol{Range_v}$ *and* $\boldsymbol{Range_w}$ *be the ranges of u, v and w respectively, such that u and v are the children of w. If* $\boldsymbol{Centre_u}$ *and* $\boldsymbol{Centre_v}$ *coincide on each other, then* $\boldsymbol{Range_w} = Maximum(\boldsymbol{Range_u}, \boldsymbol{Range_v})$.

Proof: Let $\boldsymbol{Range_u} > \boldsymbol{Range_v}$. The case that $\boldsymbol{Range_w} > \boldsymbol{Range_u}$ implies that either $\boldsymbol{Range_u} < \boldsymbol{Range_v}$ or $\boldsymbol{Centre_u}$ and $\boldsymbol{Centre_v}$ have not coincided on each other, which con-

tradicts the given statement. Also, it is impossible that $\boldsymbol{Range_w}<\boldsymbol{Range_u}$ since u is the child of w. Hence $\boldsymbol{Range_w}=\boldsymbol{Range_u}$ must be true. A similar argument can be deduced when $\boldsymbol{Range_v}>\boldsymbol{Range_u}$.

Previous discussion is centered on getting the BN's range smaller, which implies that the minimization of the maximum skew between the BN's leaves. Unfortunately, getting the BN's range smaller may result in a greater skew between the BN's leaves and other leaves in the tree as described before. However, minimizing the range of a node in a tree will lead to the minimization of the range of the parent of that node in a later adjustment as stated in the following two lemmas:

Lemma 3.3: Let $\boldsymbol{Range_w}$ *be the range of w when its child ranges are* $\boldsymbol{Range_u}$ *and* $\boldsymbol{Range_v}$ *and they are coincided on each other. Also, let* $\boldsymbol{Range'_w}$ *be the range of w when its child ranges are* $\boldsymbol{Range'_u}$ *and* $\boldsymbol{Range'_v}$ *and they are coincided on each other. If* $\boldsymbol{Range'_u}<\boldsymbol{Range_u}$ *and* $\boldsymbol{Range'_v}<\boldsymbol{Range_v}$ *then* $\boldsymbol{Range'_w}<\boldsymbol{Range_w}$.

Proof: Since the child ranges are coincided on each other, then, from Lemma 3.2, $\boldsymbol{Range_w}$=Maximum($\boldsymbol{Range_u}$,$\boldsymbol{Range_v}$) and $\boldsymbol{Range'_w}$=Maximum($\boldsymbol{Range'_u}$,$\boldsymbol{Range'_v}$). Thus, $\boldsymbol{Range'_w}<\boldsymbol{Range_w}$, since $\boldsymbol{Range'_u}<\boldsymbol{Range_u}$ and $\boldsymbol{Range'_v}<\boldsymbol{Range_v}$.

Lemma 3.4: Let u and v be the children of w, and the child edges of u and v are adjusted separately such that $\boldsymbol{Range'_u}<\boldsymbol{Range_u}$ *and* $\boldsymbol{Range'_v}<\boldsymbol{Range_v}$ *where* $\boldsymbol{Range'_u}$ *and*

Range'$_v$ *are the ranges of u and v after the adjustments respectively. If the new range of w,* ***Range'$_w$****, increases due to the previous adjustments, then the node w can be selected as the BN to adjust its child edges so that* ***Range"$_w$<Range$_w$****, where* ***Range"$_w$*** *is the range of w after adjusting child edges of w.*

Proof: In this statement, there are three individual adjustments such that the adjustments of nodes *u* and *v* lead to the adjustment of node *w*. Before any adjustment, and according to Lemma 3.2, the minimum possible value of ***Range$_w$*** is Maximum(***Range$_u$***,***Range$_v$***) which would happen when ***Range$_u$*** and ***Range$_v$*** are coincided on each other. If the last adjustment, for node *w*, shifts the child ranges of *w*, ***Range'$_u$*** and ***Range'$_v$***, till they coincide on each other, then ***Range'$_w$***=Maximum(***Range'$_u$***,***Range'$_v$***) according to Lemma 3.2. Hence, according to Lemma 3.3, ***Range"$_w$*** will be less than the minimum possible value of ***Range$_w$*** since ***Range'$_u$<Range$_u$*** and ***Range'$_v$<Range$_v$***. Selecting a BN and adjusting its child edges would minimize the skew between the BN's leaves. If such an adjustment results in a larger skew in the tree, then selecting an ancestor node as a BN will ensure the minimization of the skew further. Note that the condition that the BN belongs to the lowest level is intended to satisfy Lemma 3.1 as the BN's range would not equal to any of its child ranges. Further, the fact that BN's range is greater than its child ranges implies that the child centres are not coincided on each other. Also, note that Lemma 3.4 implies that the maximum skew in the tree would bounce, and hence it would slow down the convergence of the algorithm. However, the algorithm would converge ultimately to the required bounded skew as stated in the following theorem:

Theorem 3.1: applying the AWA algorithm to any given tree of a CDN would ultimately minimizes the maximum skew to any bound.

Proof: The algorithm, at each iteration, selects a BN that has the maximum skew in the tree. According to Lemma 3.1 and Lemma 3.2, adjusting the BN's child edges would minimize the maximum skew in the subtree rooted at the BN. If the adjustment results in a larger skew in the whole tree, then at a later iteration, an ancestor of that BN will be selected as the BN and the new maximum skew will get smaller according to Lemma 3.4. Getting a larger skew would stop when the BN is the root of the tree according to Lemma 3.1. This can be visualized as the BN bounces between different levels of the tree.

An increase in a tree skew, that may evolve from adjusting BN's child edges, can be determined from the shifting in the BN's range. In fact, the BN's range is shifted due to the alteration in the clock delay along the path of the BN, which may evolve from adjusting the BN's child edges and/or the BN's edge. Adjusting BN's child edges would shift the BN's centres by F, as shown in Figure 3.2, and it can be calculated for a BN *w* as:

$$F_w = Centre_w - \left[e'_u r_0\left(\frac{e'_u c_0}{2} - C_u\right) + Centre_u\right] \tag{3.8}$$

Adjusting BN's child edges so that $F>0$ leads to the reduction of the clock delay from the source to the BN's leaves. On the other hand, the clock delay would be increased if $F<0$.

The alteration of the BN's edge is a consequence of relocating the BN, and that will affect the clock delay from the source to the BN's leaves, and as a result, it will affect the BN's range. For a BN *w*, an adjustment to its edge by Δe_w will lead to shifting $\boldsymbol{Centre_w}$ by:

$$G_w = \Delta e_w r_0\left(\frac{\Delta e_w c_0}{2} + C_w\right) \tag{3.9}$$

In fact, the shifting in the BN's range due to the shifting in the BN's child edges, ***F***, can be compensated theoretically by adjusting the BN's edge so that:

$$G_w = -F_w \tag{3.10}$$

Such an adjustment for the BN's edge will prevent any bouncing in the skew, and AWA would converge smoothly. Figure 3.5(a) shows an example of the convergence of AWA algorithm as it was described previously, and Figure 3.5(b) shows the convergence of AWA when the BN's edge is adjusted so that Equation 3.10 is satisfied. It is obvious that such an adjustment helps to speed up the convergence. However, it is not always possible to adjust the BN's edge to any value since it is constrained by the geometric locations of the BN's parent and its children. Further, satisfying Equation 3.10 may elongate the BN's edge, and that means to trade off the wire length for the computation time. The spikes shown in Figure 3.5(a) are due to the bouncing in the maximum skew, which results from the bouncing of the BN between levels of the CDN's tree **T**. The maximum skew in **T** can be a result of any combination of two leaves, where N leaves have N(N-1)/2 combinations. As a tree of N leaves has $\log_2 N$ levels, the BN may bounce $\log_2 N$ times for each maximum skew in **T**. Thus, the order of AWA is $N^2 \log_2 N$, which is not linear. The increase in the frequency and quantity of spikes delays the convergence. In

the next section a new method is used to get faster convergence.

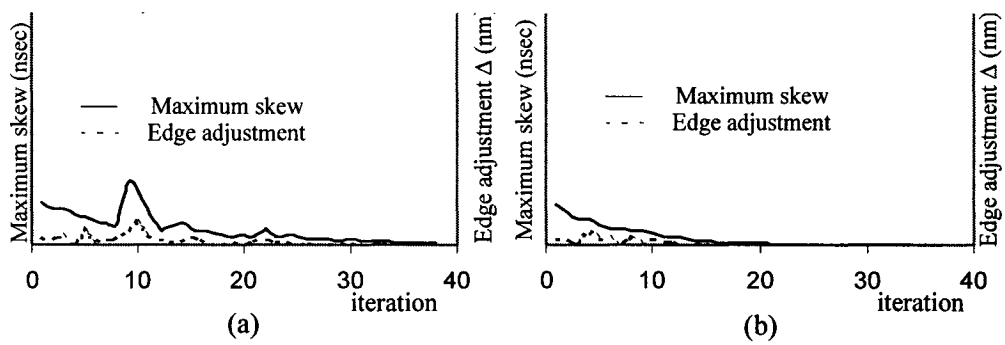


Figure 3.5 For a set of 32 clock pins (a) the skew convergence of AWA algorithm (b) the skew convergence of AWA algorithm when the BN's edge is adjusted theoretically as in Equation 3.10.

3.4 Implementation and Results

The AWA, DME and GDME algorithms were implemented using C++ on a SUN Ultra 10 machine in the Unix environment. Various benchmarks were tested for various scenarios, to provide assessment of different metrics such as wire length, clock latency, wire elongation, run time and the standard deviation of the distance between clock pins and the CDN's root. The benchmarks consist of two sets i) five benchmarks r1-r5 from [39] (downloaded from [56]); and two benchmarks were generated randomly based on 0.18 um technology [57]. The die size is assumed to be 4cmx4cm, and the parameters of these two sets; number of clock pins, load capacitance of a clock pin, unit length wire resistance and unit length wire capacitance are described by Table 3.1. Each benchmark is a netlist that provides the location and capacitance of the clock pin of a specific digital system.

For each benchmark, the initial topology is generated by both MMM and GMA algorithms. Each node in the tree is represented by a tuple of the following 9 parameters:

i: tag number of the node

x,y: coordinate of node *i* in the Manhattan plane respectively

e,s: edge and detour length that connect node *i* to its parent

c: the capacitance of the subtree connected to node *i*

l,r: pointers to the left and right children of node *i* respectively

Range, Centre: range and centre of node *i* as described by Equation 3.2 and Equation 3.3 respectively

In order to investigate the initial topology impact on the final solution, the AWA and DME algorithms were applied for initial topologies obtained by MMM and GMA algorithms, and they are labeled as MMM-AWA, GMA-AWA, MMM-DME and GMA-DME respectively.

Table 3.1. Parameters of the benchmarks that are used in the experiments

benchmark	r1	r2	r3	r4	r5	b5k	b10k
number of clock pins	267	598	862	1903	3101	5000	10000
capacitance of a clock pin	30 fF - 80 fF					23.4 fF	
unit length wire resistance	0.003 Ω					0.076 Ω	
unit length wire capacitance	0.02 fF					0.118 fF	

The results are presented in two subsections. Initially, AWA performance is compared to other algorithms, then the performance of AWA is investigated in terms of CDN's tuning.

3.4.1 Comparing AWA's performance to DME and GDME

The initial tree of each benchmark was used as an input for the AWA algorithm. AWA relocates the internal nodes iteratively and reduces the skew till it achieves the bounded specification B. If B is set to zero, then AWA will end up with a ZSCDN. Figure 3.6 shows the skew convergence of AWA for the benchmark r3, where the initial topology was generated by MMM. It is obvious that the number of required iterations decreases as the required skew bound, B, increases. The spikes are resulting from the bouncing of the BN between levels of T. Note that the previous bounded skew algorithm given in [45] requires the recomputation of the whole solution whenever a change in the CDN requirements is made. Further, the reduction of skew comes at the expense of extra wire length. On the other hand, for AWA, the total wire length is reduced while the skew is being minimized incrementally as shown in Figure 3.7. Additionally, and in order to show the impact of the initial topology, two initial topologies were used; one is obtained by MMM and the other by GMA as shown in Figure 3.8. It is obvious that the GMA topology renders less wire length. Table 3.2 provides comparisons of total wire length, clock latency, number of wire elongations and run time for different benchmarks tested by different algorithms.

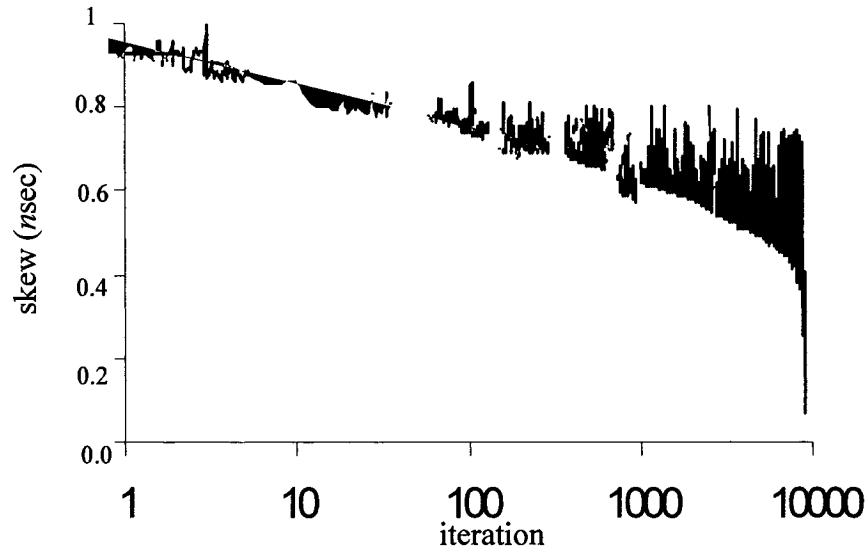


Figure 3.6 The AWA skew convergence for the benchmark r3.

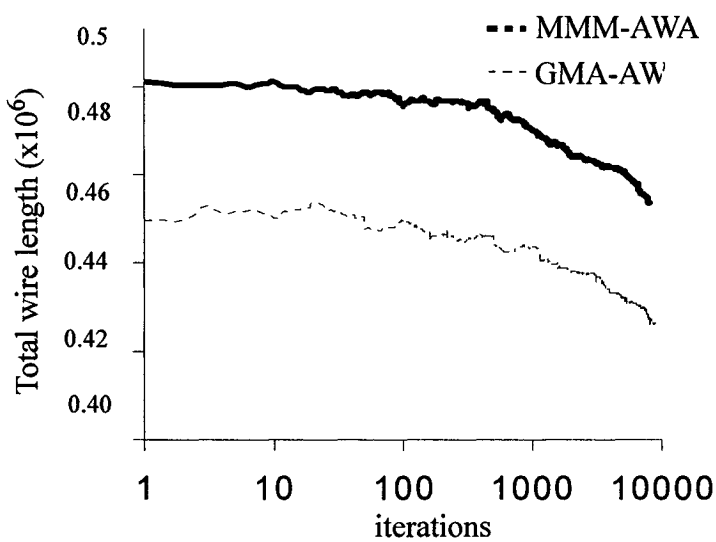


Figure 3.7 Relation between AWA's iteration and the total wire length for the benchmark r3.

Table 3.2.Different metrics of DME, AWA and GDME for different benchmarks. Both DME and AWA are tested with two different initial topology; MMM and GMA.

benchmark		r1	r2	r3	r4	r5
number of clock pins		267	598	862	1903	3101
Total wire length ($x10^6$ unit length)	MMM-DME	1.73	3.61	4.69	9	13.75
	GMA-DME	1.59	3.31	4.34	9.1	13.69
	GDME	1.47	3.05	4.05	7.28	12.89
	MMM-AWA	1.73	3.59	4.68	9	13.86
	GMA-AWA	1.57	3.32	4.35	9.2	13.7
Number of wire elonga-tions	MMM-DME	2	19	36	41	179
	GMA-DME	11	38	49	98	183
	GDME	20	37	67	115	726
	MMM-AWA	2	20	36	42	453
	GMA-AWA	11	38	48	96	571
Clock latency (*psec*)	MMM-DME	2	5.5	7.95	22.31	36.18
	GMA-DME	2	4.1	5.87	14.38	43.47
	GDME	1.6	4.4	4.61	11.54	35.38
	MMM-AWA	2	4.6	7.66	22.48	40.62
	GMA-AWA	1.5	4.4	6.67	18.56	38.59
run time (sec)	MMM-DME	0.01	1	1	1	1
	GMA-DME	0.01	1	1	1	1
	GDME	22	120	721	7432	33081
	MMM-AWA	2	4	16	39	27
	GMA-AWA	3	4	16	42	27

Figure 3.8 and Table 3.3 show the relationship between the number of leaves for different random sets and the number of required iteration to reach ZS. It is obvious that the relationship is not linear, however, the main advantage of AWA is its iterative approach of minimizing the skew whereas a small modification in the input data would not require the recalculation of the whole solution as in DME. This feature is important when size of T is very large. To show this advantage, different random ZSCDNs of size N=1024,

4096 and 8192 were tested by altering leaf locations then tested with AWA. The bounded skew was set to zero, B=0, in order to compare number of the iterations to the DME required iterations (DME is a linear zero skew algorithm). As one would expect, the number of iterations increases as the number of altered leaves increases. Figure 3.9 shows the relationship between the number of iterations and number of altered leaves for the mentioned sets. The number of iterations for each number of altered leaves is the average of numbers of the iterations, where each number of iterations corresponds to altering different leaves but of the same number of leaves. The alteration in the leaf locations was 6.25% of the whole Manhattan plane. The benefit of AWA is elevated as the size of the tree increases. For example, when the number of leaves N=1024, and number of altered leaves is 9, the AWA's iteration is 1693 while using DME requires 1024 iterations only. However, when the number of leaves is N=8192, and number of altered leaves is 9, the AWA's iteration is 2410 while using DME requires 8192 iterations.

Table 3.3 Number of iterations required by AWA for different benchmarks

	Benchmark				
Algorithm	r1	r2	r3	r4	r5
MMM-AWA	4356	17410	28838	85963	162549
GMA-AWA	4995	17572	28348	99554	183772

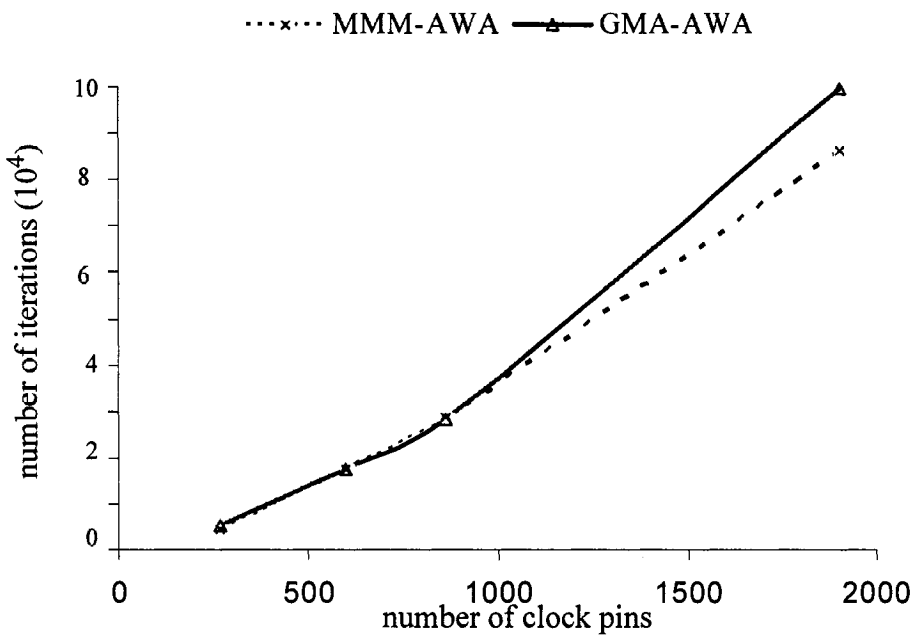


Figure 3.8 Relation between the number of clock pins and the number of iterations of AWA algorithm.

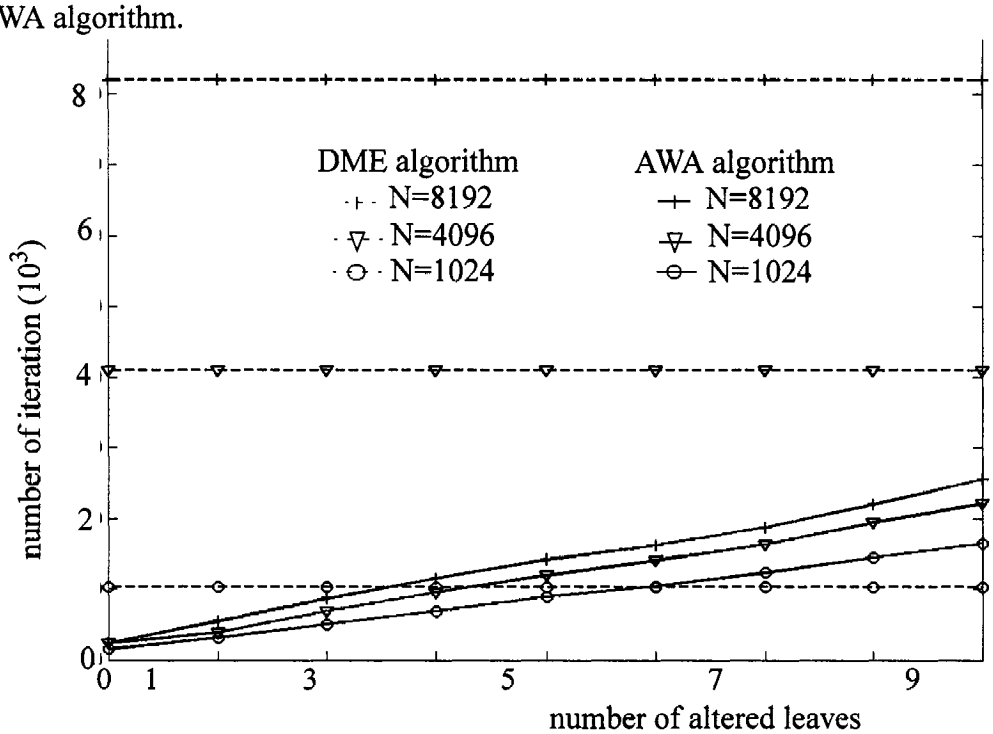


Figure 3.9 Relation between AWA's iteration and the number of altered leaves for ZSCDNs of different benchmarks of clock pins.

3.4.2 CDN's Tuning using AWA

The main advantage of AWA is to handle minor modifications. Three experiments were set to study the performance of AWA in scenarios similar to minor modifications required in a design cycle. The three experiments are designed to study the sensitivity of AWA to the number of the shifted clock pins, the shifting size in the clock pins and to removing and inserting an IP block. Note that the previous algorithms require the recalculation of the whole solution when there is a minor modification. The benefit of AWA is more prominent as the number of clock pins increases. Different ZSCDNs of benchmarks r3, b5k and b10k were generated, and then tested for different scenarios. For each benchmark, two ZSCDNs were generated by using MMM and GMA for initial topology, and then the DME algorithm is applied.

Experiment 1: AWA's Sensitivity to the Number of Shifted Clock Pins

Whenever a number of clock pins are shifted, the CDN does not hold the zero skew status anymore, and AWA is applied to minimize the skew. Figure 3.10 shows the relationship between the number of iterations and number of shifted clock pins when AWA is applied to different sets. In this experiment, the shifted clock pins, n, are selected from dispersed locations in the presumable die of the ZSCDN (the die area is divided into 16 equal size squares, and the n nodes are selected from different squares). Further, in order to have a realistic estimation of AWA performance, the experiment was repeated ten times with different sets of n nodes. The number of iterations for each number of shifted clock pins shown in Figure 3.10 is the average number of iterations required by different tests, where each test is repeated for the two ZSCDNs of each benchmark. The shifting

in the clock pins locations was set to 1% of the whole Manhattan plane. It is obvious that AWA tends to have more iterations when the number of shifted clock pins is greater. The shifting in the clock pins results in an increase in the total wire length of the final solution as shown in Figure 3.11. This increase can be attributed to the topology that has not been modified to capture the change in the clock pins location.

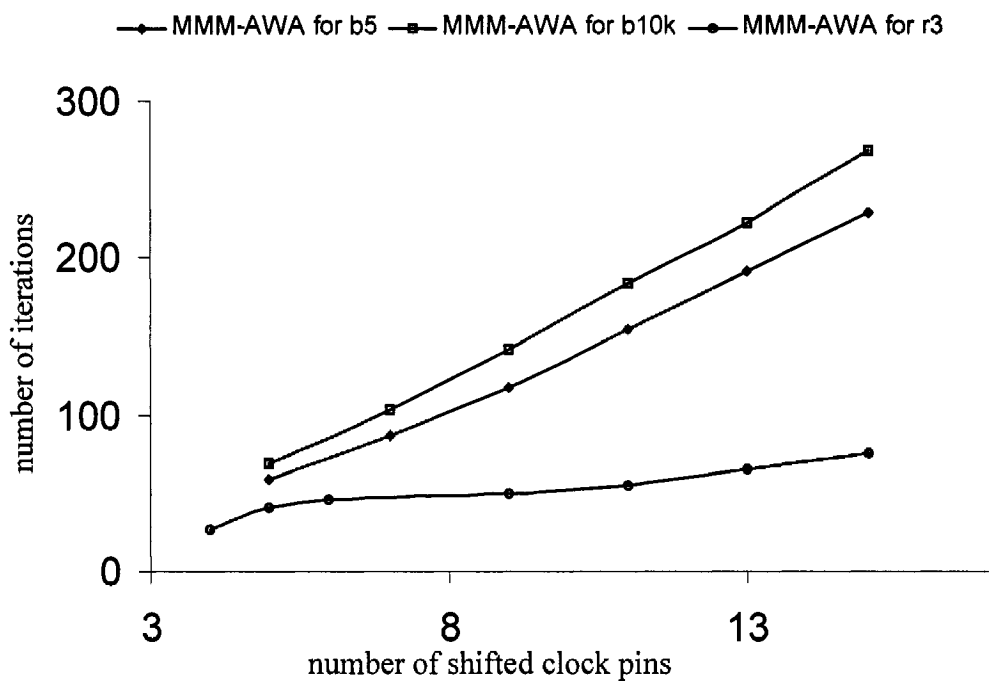


Figure 3.10 Relation between AWA's number of iterations and the number of shifted clock pins for benchmarks r3, b5k and b10k.

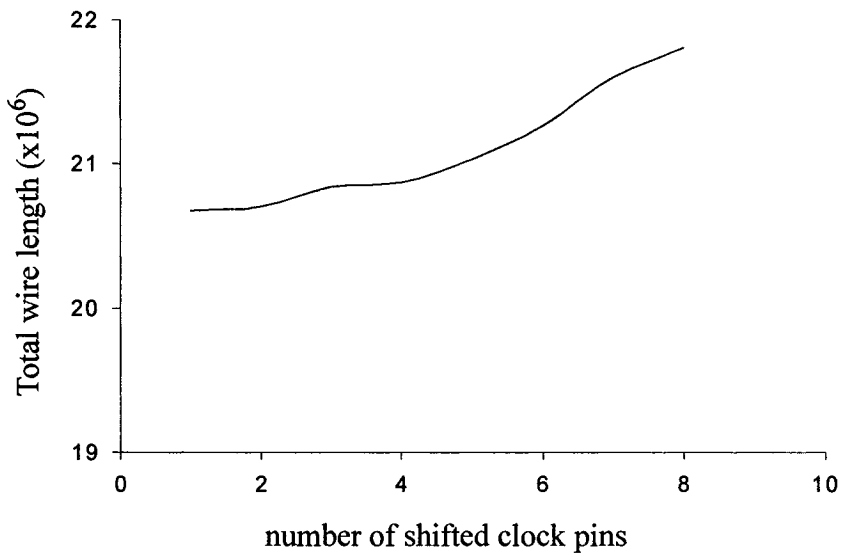


Figure 3.11 Relation between the total wire length and the number of shifted clock pins for benchmark b5k.

Experiment 2: AWA's Sensitivity to the Size of the Shifting in the Clock Pins

In this experiment, the shifting in the clock pins is varied for a fixed number of clock pins. The two ZSCDNs of the benchmarks r3, b5k and b10k were tested by shifting the locations of ten clock pins by different amounts. Figure 3.12 shows the relationship between the amount of shifting in the clock pins' locations and the required iterations to achieve ZS for the benchmarks. By comparing Figure 3.10 to Figure 3.12, one can notice that the number of iterations is less sensitive to the shifting value than to the number of shifted clock pins. This means that the increase in the number of shifted clock pins results in an increase in the bouncing of the BN between different levels of the CDN as described in section IV. Similar to the previous experiment, the shifting in the clock pins results in an increase in the total wire length as shown in Figure 3.13.

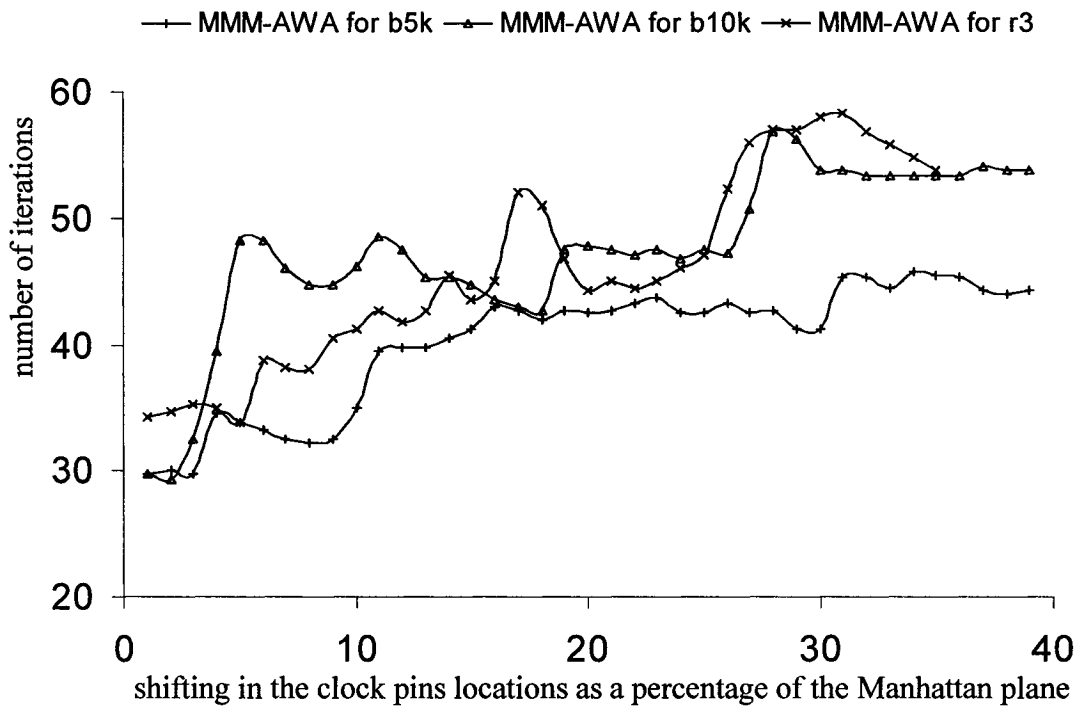


Figure 3.12 Relation between the number of iteration and the shifting in the location of ten clock pins in benchmarks r3, b5k and b10k.

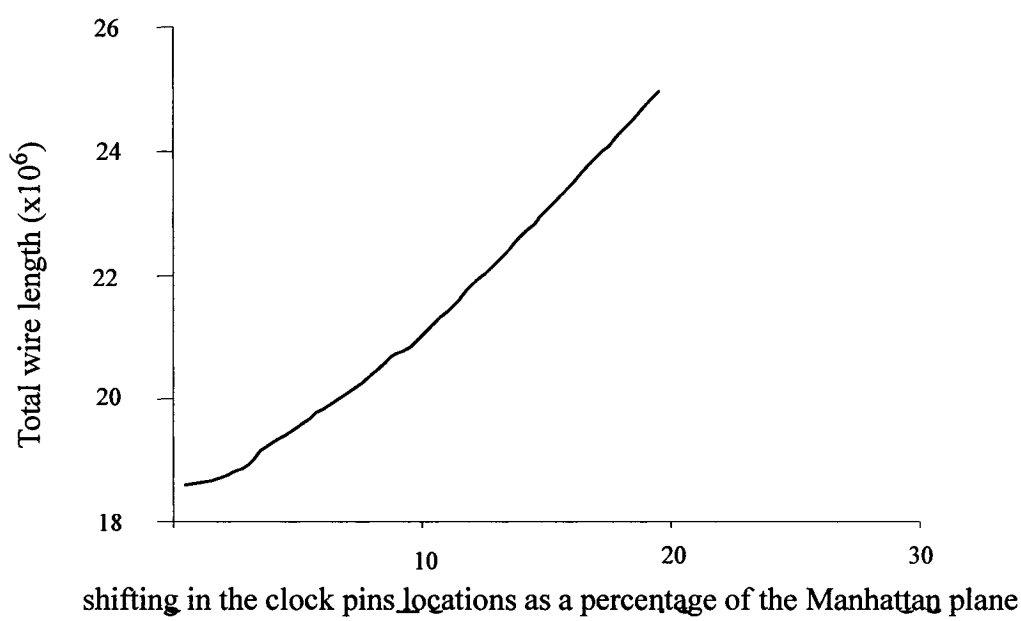


Figure 3.13 Relation between the total wire length and the shifting in the clock pins for the benchmark b5k.

Experiment 3: AWA's Sensitivity to Removing and Inserting an IP

The IP block is assumed to be a rectangular whose size is varied between 0.5% to 20% of the whole die area, and it is assumed to be located at the centre of the die. At first, all the clock pins located inside the IP area are deleted, then a similar number of clock pins are inserted randomly in the area of the IP. Table 3.4 shows the size and the number of clock pins of each IP for the benchmarks bk5 and b10k. The new clock pins stand for the inserted IP, and whenever a new clock pin is inserted, it is connected to the closest clock pin in the CDN. Figure 3.14 shows the relationship between the size of the IP for b5k and the number of required iterations to achieve ZSCDN using AWA algorithm. As one may expect, the number of iterations increases with the size of the IP. Indeed, the increase in the number of iterations is a function of the number of the clock pins in the IP rather than the size of the IP itself. However, these nodes are neighbors to each other, and hence such a minor modification requires less number of iterations than the minor modifications described in Sections 8.3.1 and 8.3.2 where the nodes are scattered all around the die. For example, the change in the locations of 6 nodes in the benchmark b5k requires 273 iterations by AWA when these nodes are distributed all around the die, however the change in 21 nodes in the same benchmark requires 45 iterations by the same algorithm when the nodes are neighbors to each other.

Table 3.4 Number of clock pins per IP for benchmarks b5k and b10k.

size of the IP as a percentage of the die area	number of clock pins	
	b5k	b10k
0.5	21	38
2	107	382
10	493	975
15	668	1318
20	845	1683

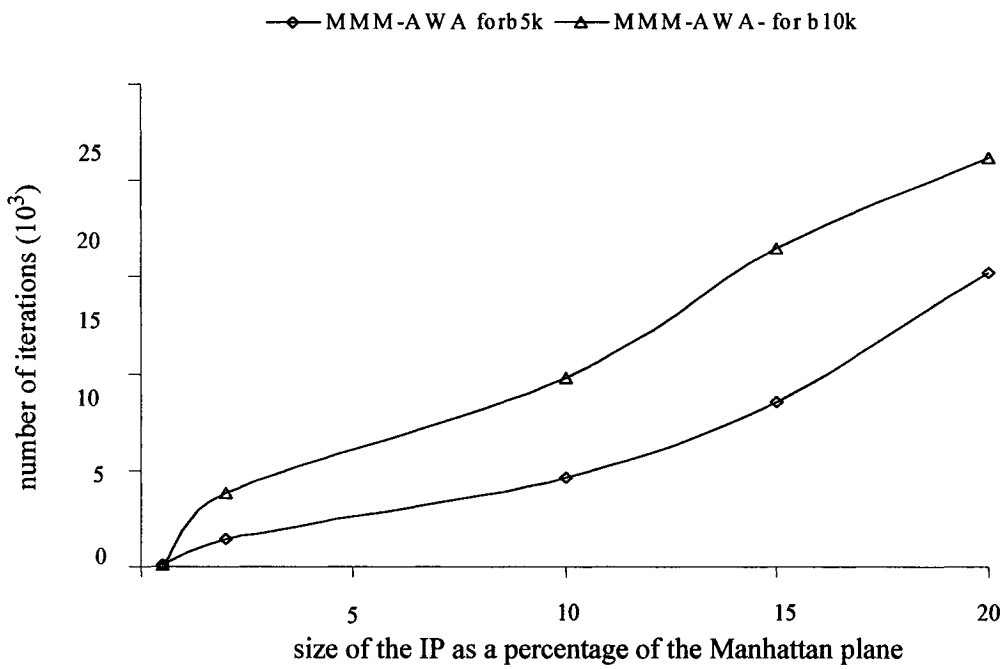


Figure 3.14 Relation between AWA's iteration and the size of the deleted and inserted IP. The size is given as percentage of the whole die area.

3.5 Summary

The topic of minor modifications of the CDN is an important concept in complex synchronous systems; and hence, there is a need for an algorithm to cope with these modifications in the CDN. The AWA algorithm was proposed to handle such scenarios. Whenever there is a skew between any two clock pins, AWA algorithm can reduce the skew to any required bound, and ultimately to zero, though zero skew is an over constraint sometimes. Simulations performed on CDNs have shown that using AWA algorithm can:

(i) reduce the bounded skew requirement further without recalculating the whole solution.

(ii) achieve ZS for a given initial CDN.

(iii) achieve ZS for a modified ZSCDN at less computation compare to linear methods

The AWA algorithm, similar to DME, requires an initial topology to reach to its solution. Such a topology has an impact on the quality of the CDN as well as on the convergence of AWA algorithm. In the next chapter, a new method of local topology modification will be proposed in order to speed up the convergence of AWA algorithm, as well as to improve the quality of the CDN.

Chapter 4

Local Topology Modification

4.1 Introduction

This chapter focuses on the topology impact on the performance of AWA as well as on previously developed algorithms, DME and GDME, in terms of the total wire length, wire elongations and standard deviation of the path lengths, SDPL, between the root and the clock pins. These algorithms require an initial topology, which can be generated by MMM, GMA or Balanced Bipartition (BB) algorithms. Lillis et. al. proposed to search for the topology by searching for a good permutation of the sinks in order to minimize the total wire length and to satisfy the time constraints[59].

Researchers have tested different combinations of the DME with different topology generation algorithms [3, 58]. For example, the combination of GMA and DME offers a solution with less wire length than that one obtained from MMM and DME combination. However, using a fixed initial topology may result in an unnecessary increase in wire length. On the other hand, GDME does not require an initial topology; and it gener-

ates the topology that leads to less wire length. However, GDME may connect the clock pins to different levels of the topology; and hence, the CDN is compounded intensively by wire elongations. In fact, the same problem incurs with DME depending on the initial topology. For example, the GMA may connect the sinks to any level in the tree while MMM keeps all the sinks in the lowest level of the tree.

Another consequence of connecting the clock pins to different levels of the topology, as in GDME and GMA, is that some clock pins are close to the CDN's root while others are distant from the root; and consequently, the CDN' SDPL would be large. A large SDPL implies an inaccuracy in the timing of the clock signal due to the resistive shielding effect, in which Elmore model overestimates the delay for the sinks that have shorter connection to the source. Another implication of a large SDPL is the vulnerability of the CDN to temperature and process variations, where a longer path would be more susceptible to these variation than a shorter path.

In this chapter, the CDN is treated as a quadratic data structure instead of binary in order to have more flexibility of locating the steiner nodes, and so, the total wire length can be minimized further [60]. A novel method, called Local Topology Modification (LTM), is proposed to optimize the topology using a search tree or a quadratic optimization technique [62].

4.2 Quadratic Tree vs. Binary Tree

The DME and AWA algorithms are based on the binary data structure. Considered a simple binary tree of four leaves (1, 2, 3 and 4) as shown in Figure 4.1(a). According to this topology, a ZSCDN can be generated as shown in Figure 4.1(b) where node 1 is connected to node 2 and node 3 is connected to node 4. The tree topology governs the pairing between child nodes, and hence, it may result in an unnecessary increase in the total wire length. For example, by using the topology shown in figure 4.1(c), another ZSCDN with less wire length can be generated for the same leaves as shown in Figure 4.1(d). However, the topology of the latter CDN is different from the topology of the first CDN. Hence, in order to avoid unnecessary increase in the total wire length due to the tree topology, the pairing between nodes needs to be more flexible. Some flexibility can be gained by using a quadratic data structure instead of a binary structure as will be described next.

Quadratic tree means that each internal node has four children. For example, in Figure 4.2, node s is connected to four children (1, 2, 3 and 4). The grouping between the four child nodes is not decided by the tree topology; and hence, the topology imposes no constraints on connecting each internal node to its children. Comparing a binary tree to a quadratic one, the first method connects w to its children (1, 2, 3 and 4) through two Steiner nodes (u and v) which are explicit in the tree. On the other hand, the quadratic tree may introduce up to two additional Steiner nodes in order to connect a node to its four children as shown in Figure 4.2. The additional Steiner nodes are not defined by the topology tree, which means that there is a flexibility in connecting a node to its four chil-

dren. This flexibility in choosing between two topologies can be exploited by adapting the DME, GDME and AWA algorithms for local topology modification in order to enhance the quality of the CDN [64].

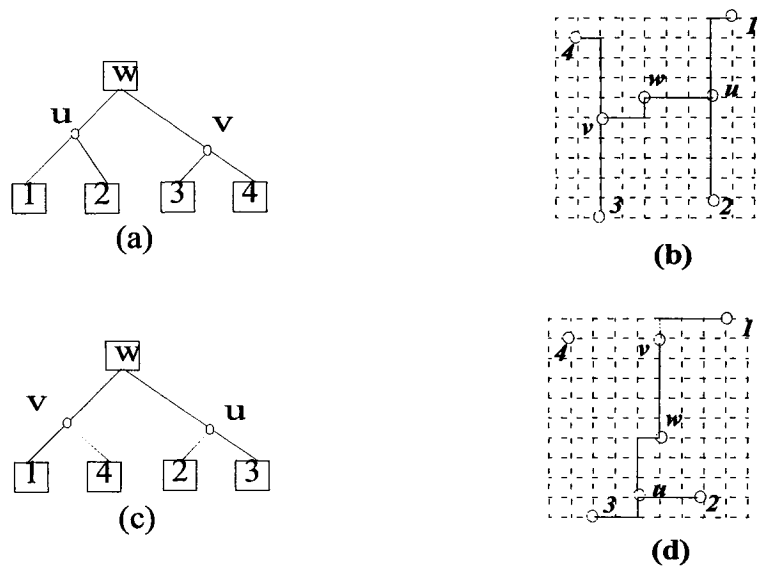


Figure 4.1 An example of minimizing the total wire length by modifying the topology

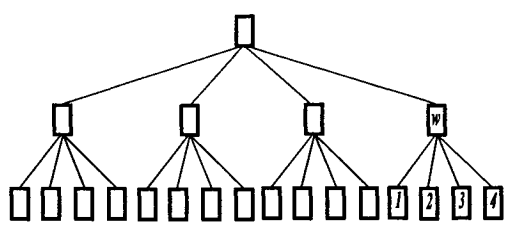


Figure 4.2 The Quadratic tree.

Figure 4.3 lists 15 possible topologies of connecting a node to its four children; and these topologies will be called the primary topologies. There are other topologies as shown in Figure 4.4, which will be called non-primary topologies. The non primary topologies can be derived from the primary topologies. For example, the topology of Figure 4.4(a) implies that the four children are connected directly, without Steiner

points, to a single node. This topology can be derived from any topology shown in Figure 4.3 when the steiner points coincide on node *w*. Similarly, the topology shown in Figure 4.4(b) can be derived from one of the topologies shown in Figure 4.3(a4, b4 or c4). Thus, only primary topologies will be considered since the non primary topologies can be derived from the primary topologies.

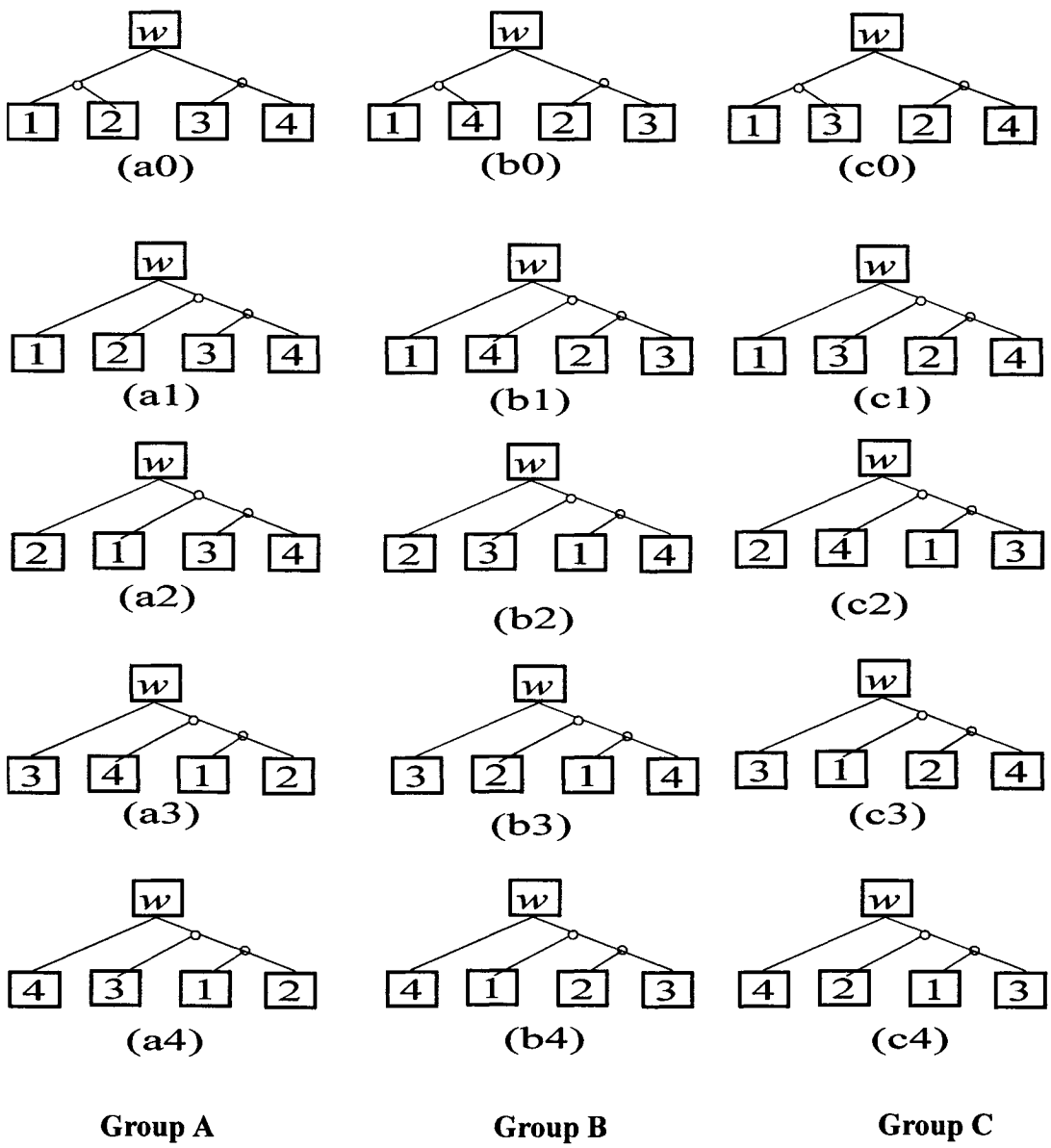


Figure 4.3 The primary topologies of connecting a node to its four children.

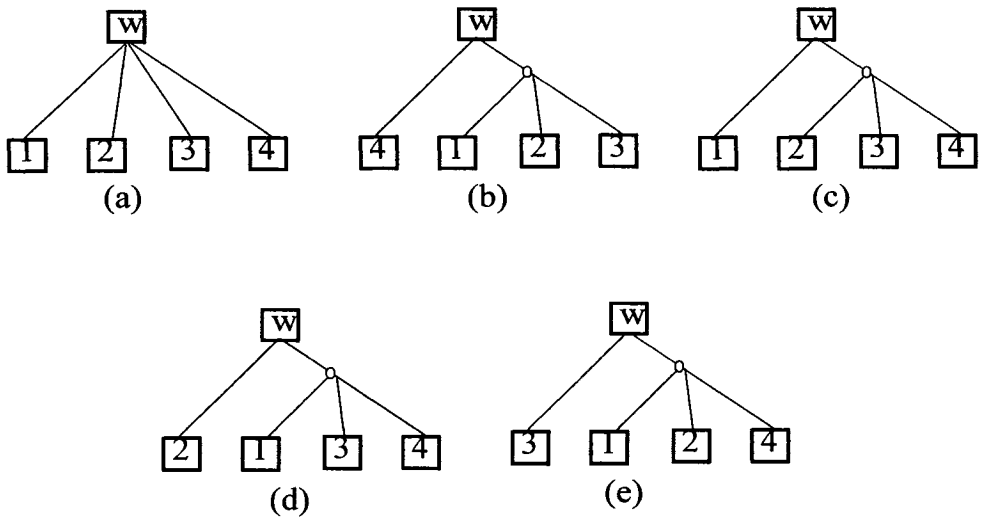


Figure 4.4 The non primary topologies of connecting a node to its four children.

The primary topologies shown in Figure 4.3(a0, b0 and c0) imply that two nodes are balanced with the other two nodes. Thus, these topologies are used most often since the load difference between child nodes is small. Indeed, these three topologies emulate the binary topology in terms of connecting the source node to two Steiner nodes and each Steiner node is connected to a pair of child nodes. On the other hand, the rest of the primary topologies shown in Figure 4.3 imply that one child node is balanced with the other three child nodes. This indicates a large load difference between the child nodes. Such cases are less frequent than the cases of balancing two nodes with the other two nodes. The choice between the primary topologies is decided based on which one renders less wire length. Two methods are developed to search for the near optimal topology. The first method is based on a search tree and the other method is based a branch and bound method.

4.3 Local Topology Modification Using a Search Tree

The problem of connecting a set of nodes in a Manhattan plane with minimum wire length is a Rectilinear Steiner Tree (RST) problem [3]. There are many heuristic approaches that produce the RST for a set of points, particularly, the RST that connects the points as terminals, which is called full RST [13]. In the case of four node example, a full RST would connect two nodes by a backbone and the other two nodes are connected by segments incident alternately to the backbone. For example, in Figure 4.5(a), nodes 1 and 3 are connected by a backbone, and nodes 2 and 4 are connected by segments incident to the backbone. In other words, the RST of Figure 4.5(a) pairs between nodes 1 and 2 and between nodes 3 and 4, where such RST has a topology as shown in Figure 4.3(a0). Note the difference between the topology of the clock signal flow and the RST's topology, because they are not necessarily the same. For example, Figure 4.5(b) represents a case in which wire elongation is introduced between nodes u and w, where node w coincides on node v, in order to achieve zero skew. Such a wire elongation can be avoided by using different topology for the clock signal flow as shown in Figure 4.5(c). The RST's topology of both CDNs shown in Figure 4.5(b and c) is Figure 4.3(a0), however, the topologies of the clock signal flow are Figure 4.3(a0 and a4) respectively.

In general, a full RST of four nodes may have one of the three topologies shown in Figure 4.3(a0, b0 or c0) based on pairing between the nodes. On the other hand, the clock signal flow can be any topology shown in Figure 4.3. For each RST's topology, the clock signal flow may follow one of five possible topologies. Thus, the topologies of the

clock signal flow can be grouped in three groups, Group A, Group B and Group C, as shown in Figure 4.3. These groups are classified based on the topology of the RST that is required for each group. Specifically, when RST's topology is Figure 4.3(a0), Figure 4.3(b0) or Figure 4.3(c0), the clock signal flow may follow the topologies of Group A, Group B or Group C respectively.

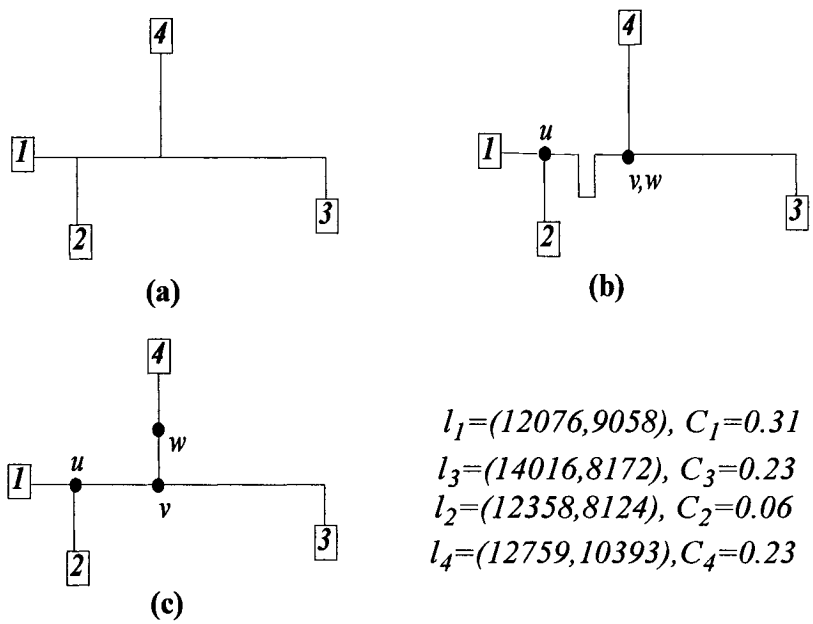


Figure 4.5 Avoiding wire elongation by modifying the topologies of the RST and/or the clock signal flow (the drawing is not to scale).

Based on this observation, wire length can be minimized by searching for the different topologies of RST. Then further minimization can be achieved by investigating different topologies for clock signal flow. The selected RST's topology, say Figure 4.3(a0), will be considered for the clock signal flow, as well as the other four clock signal flow topologies that correspond to the same RST topology. In the previous example of Figure 4.5(b), the first solution, based on Figure 4.3(a0), results in e_v=0 and $e_3<e_4$, where node

w coincides with node v. This means that node 4 needs to be connected directly to w according to Figure 4.3(a4). Modifying the topology between a node and its four children is referred to as Local Topology Modification, LTM, which takes into consideration the children capacitance. In this example, modifying the topology of the clock signal flow reduces the total wire length from 5569 to 5274 units. The search for the topology of RST and clock signal flow is summarized in the searching tree shown in Figure 4.6. The description of LTM procedure is given in Figure 4.7. In the LTM procedure, it is assumed that node w has four children, and a similar procedure can be written when node w has three children. For each selected topology, the merging segments are determined in bottom-manner as described in Section 2.7.2.

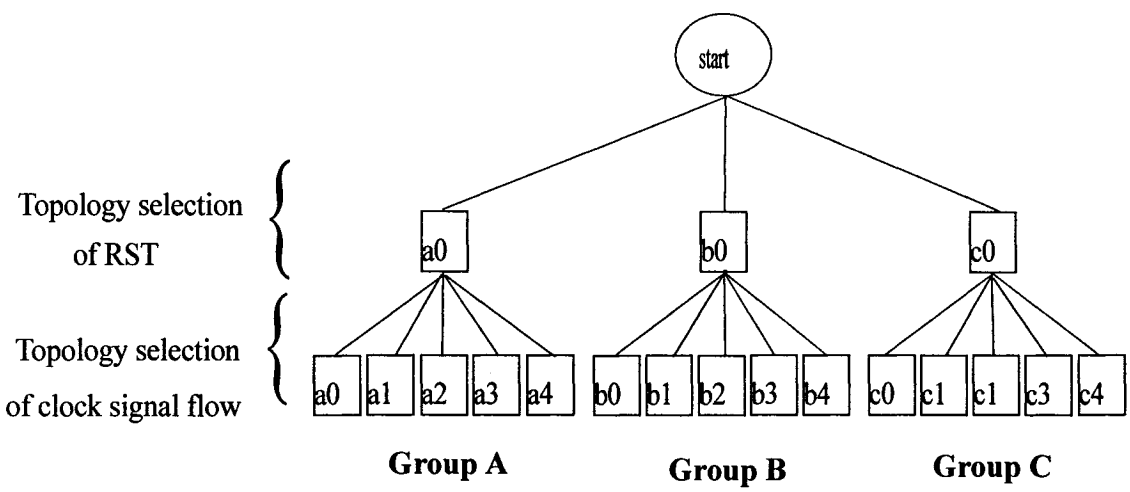


Figure 4.6. The searching tree for the optimal topology. The labels refer to topologies labeling in Figure 4.3.

Input: Any node, *w*, its two child nodes, *u* and *v*, and their children, 1,2, 3 and 4, that are connected to *u* and *v*. (Each child node, 1, 2, 3 and 4, has a merging segment)
Output: The merging segments of *u*, *v* and *w*
Procedure: Local Topology Modification For each RST topology of Figure 4.6(a0, b0, c0): For each node *u*, *v* and *w*: calculate the merging segment in a bottom-up manner according to the specified topology. Select: the topology that has the minimum wire length, let it be x0 For each topology of Figure 4.6 (x1, x2, x3, x4): For each node *u*, *v* and *w*: calculate the merging segment in a bottom-up manner according to the specified topology. Select: the topology that has the minimum wire length Adapt: the left and right links of nodes *w*, *u* and *v* according to the selected topology.

Figure 4.7 The LTM procedure.

Besides modifying the topology locally by LTM between a node and its four children, LTM allows nodes to be swapped in the tree. For example, assume that LTM procedure has been called for nodes *z* and *w* as shown in Figure 4.8, where the merging segments of *x*, y, *z*, *u*, *v* and *w* have been calculated. Then, calling LTM at node *s* will modify the topology that connects nodes *x*, *y*, *4* and *v*. As such, node *4* can be swapped with nodes *y* (or *x*) if still it cannot be balanced with nodes 1, 2 and 3 only.

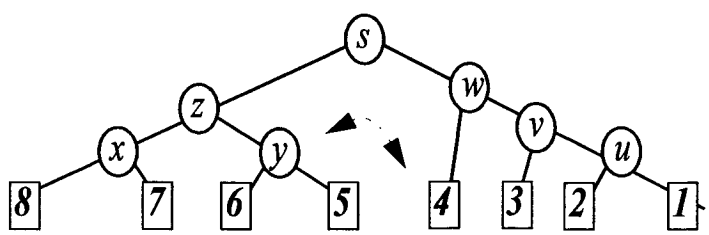


Figure 4.8 Swapping the nodes in the tree when LTM is applied.

4.4 Applying the LTM Method to AWA, DME and GDME Algorithms

In order to use the LTM concept with AWA algorithm, the procedure of balancing two nodes becomes a procedure of balancing four nodes. Whenever, a node is selected as BN, the local topology is selected as described in the LTM procedure. Assume that the balancing node is node *w* in Figure 4.3(a0) where its children are *1, 2, 3* and *4*. The selected topology requires two Steiner nodes (will be referred as *u* and *v)*, and hence, three balancing segments have to be calculated: $\mathbf{BS_s}$, $\mathbf{BS_u}$ and $\mathbf{BS_v}$ in a bottom-up manner according to the selected topology. The balancing segment $\mathbf{BS_u}$ is calculated from the intersection of two tilted rectangles, $\mathbf{TR_1}$ and $\mathbf{TR_2}$, whose centers are nodes *1* and *2* and whose radii are e'_1 and e'_2 respectively, where e'_1 and e'_2 are calculated as in Equation 3.1 and Equation 3.5. In a similar manner, $\mathbf{BS_v}$ and $\mathbf{BS_w}$ can be calculated. Note that $\mathbf{BS_w}$ must be calculated after the calculation of $\mathbf{BS_u}$ and $\mathbf{BS_v}$ where the centres of $\mathbf{TR_u}$ and $\mathbf{TR_v}$ are the segments $\mathbf{BS_u}$ and $\mathbf{BS_v}$ respectively. Then, nodes *w, u* and *v* can be located exactly at their balance segments in top-down manner according to the selected topology. Indeed, node *w* can be located at the closest point of $\mathbf{BS_w}$ to the parent of *w*. On the other hand, locations of nodes *u* and *v* can be determined from the intersections of $\mathbf{BS_u}$ and $\mathbf{BS_v}$ with the tilted rectangles whose centres are the parents of *u* and *v*, and whose radii are e'_u and e'_v respectively.

Finally, an algorithm can be drawn to achieve bounded skew for a given tree by minimizing the maximum skew in the tree iteratively. At each iteration, a BN is selected; a

topology of connecting BN to its children is selected where the edge lengths and node locations are determined as described earlier. The description of LTM-AWA is given in Figure 4.9.

Input: Initial CDN for a set of clock pins R in form of a tree **T**
Output: CDN with a skew bounded by **B**
while *Maximum skew >**B**:* *Find the **BN**, that has the maximum skew in **T** (let s be the **BN** and its children are 1, 2, 3 and 4, and the two Steiner nodes u and v.* *Call the LTM procedure* *Locate w at the closest point of $\boldsymbol{BS_w}$ to its parent and calculate e_w* *for node β= u or v in top-down manner according to the selected topology* *Let p be the parent of β* *Construct $\boldsymbol{TR_p}$ whose centre in node p and its radius is e_β* *locate β at the intersection of $\boldsymbol{BS_\beta}$ and $\boldsymbol{TR_p}$* *Calculate the Maximum skew in **T***

Figure 4.9 The LTM-AWA algorithm.

It is obvious that selecting a BN and balancing its four children means the division of the leaf delay plane of that BN into four quarters and balancing them. Figure 4.10 shows the leaf delay plane of a tree of 16 leaves before running the LTM-AWA. It also shows the delays after the first iteration of LTM-AWA and after the last two iterations where the plane becomes flat. Note that LTM-AWA requires 10 iterations only to achieve ZSCDN, while AWA requires 79 iterations for the same set of clock pins.

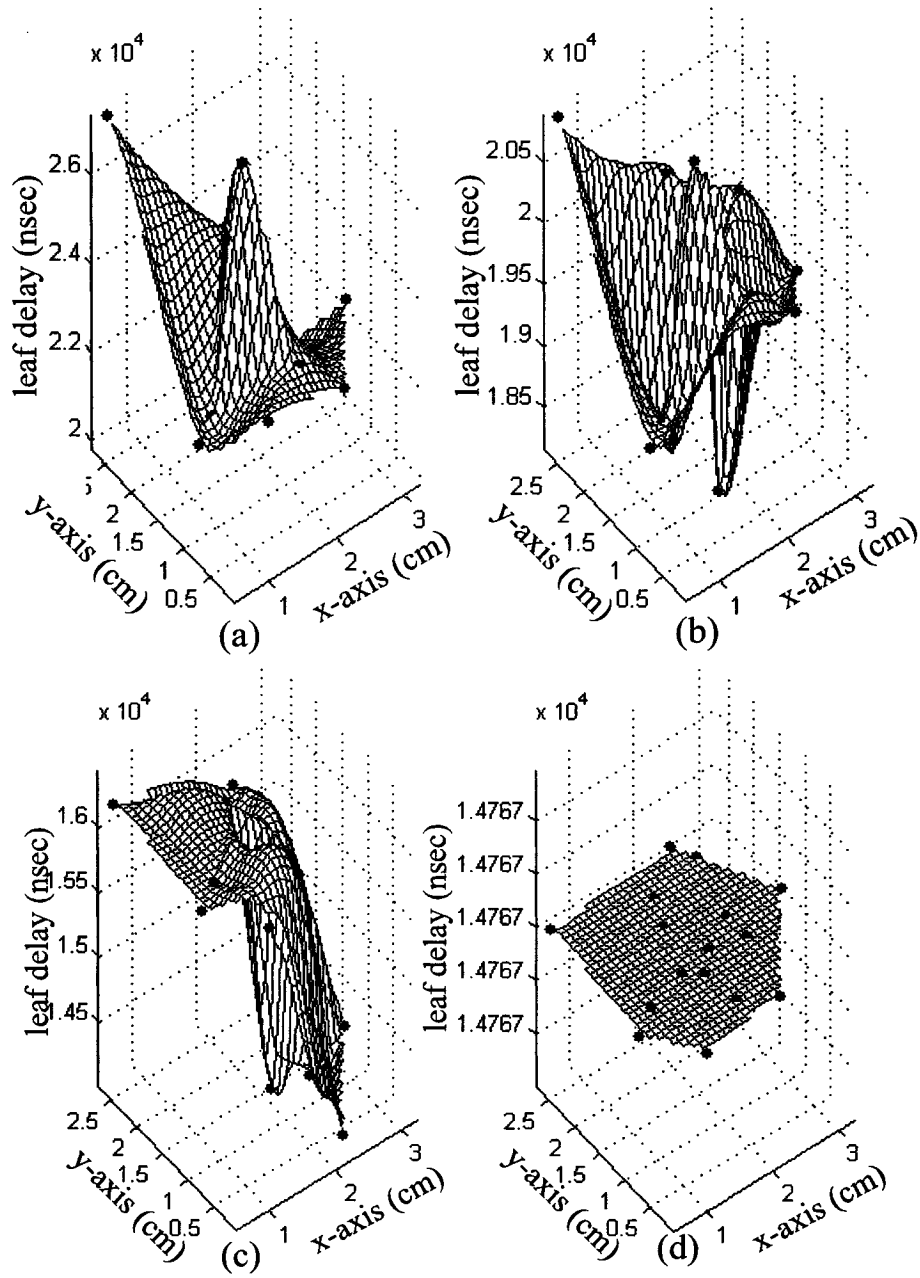


Figure 4.10 Leaf delays plane of a tree of 16 leaves during the running of LTM-AWA at: (a) the beginning (b) iteration 1 (c) iteration 9 (d) iteration 10.

To apply LTM to DME and GDME, the procedure of merging segment calculation in the bottom-up phase of DME and GDME is replaced with LTM procedure, and so they will

be called LTM-DME and LTM-GDME respectively. Applying LTM method will increase the calculation time, but it does not affect the order of both algorithms. Also note that using LTM allows nodes to be swapped in the tree.

4.5 Implementation and Results

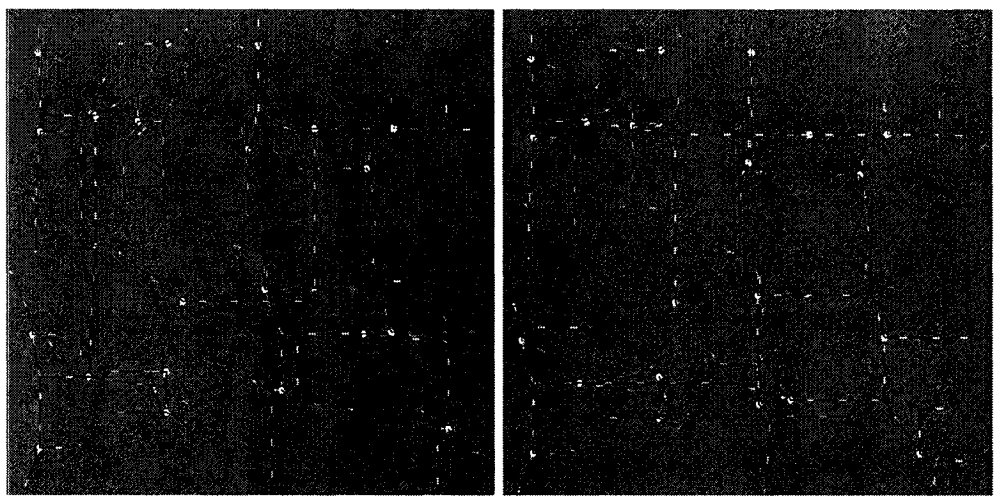
The LTM procedure was implemented using C++ language and applied to AWA, DME and GDME algorithms on a SUN Ultra 10 machine under Unix environment. For LTM-DME and LTM-AWA the initial binary topology was generated using MMM and GMA algorithms [36,37].

4.5.1 Applying LTM to AWA, DME and GDME Algorithms

The main advantage of using LTM is the topology flexibility that helps minimizing the total wire length and wire elongation. Figure 4.11 shows the resulting ZSCDNs of DME and LTM-DME for a set of 64 clock pins distributed randomly on 0.5cm x 0.5cm silicon device. Note the difference between the two resulting CDNs where the one that results from LTM-DME has both H and flipped-H connections while the other one has H-connection only. The resulting total wire length of applying DME and LTM-DME are 6.6cm and 5.8 cm respectively.

Next, the benchmarks that were described in Section 3.4 are tested by LTM-AWA, LTM-DME and LTM-GDME to provide assessment of different metrics such as wire length, clock latency and wire elongations, run time and the standard deviation of the distance between clock pins and the CDN's root. Table 4.1 provides comparison of total

wire length, clock latency, number of wire elongations, SDPL and run time when LTM is applied to different algorithms and tested for different benchmarks. Figure 4.12 shows the impact of applying LTM approach by comparing the metrics obtained for the benchmark r3. From Table 4.1 and Figure 4.12, it is obvious that using LTM always leads to a reduction in the total wire length. The minimum wire length can be achieved by using LTM-GDME, but with only 7.7% increase in run time as compared to GDME. One can notice that using GMA's topology with either LTM-DME or LTM-AWA results in a CDN whose total wire length is close to the one obtained by GDME.



(a) (b)

Figure 4.11 The resulting ZSCDN for 64 clock pins by using (a) DME (b) LTM-DME.

From Figure 4.12, one may notice that applying LTM would most likely reduce the clock latency, but this is not always the case. One of the drawbacks of GDME is its high SDPL as well as the wire elongations are most prominent. By applying LTM to GDME, the SDPL and the number of wire elongations are reduced effectively, but with a slight increase in the running time as shown in Figure 4.12. The reduction in the number of

wire elongations is a potential benefit of applying LTM approach that will ease the burden of the CDN's layout. In general, from Table 4.1 it can be seen that by using LTM with DME the total wire length is reduced by 7.83%, and by using LTM with GDME the total wire length is reduced by 9.77%.

Table 4.1 Different metrics of LTM-AWA, LTM-DME and LTM-GDME for different benchmarks.

benchmark		r1	r2	r3	r4	r5
Total wire length ($x10^6$ unit length)	MMM-LTM-DME	1.53	3.25	4.18	8.04	12.53
	GMA-LTM-DME	1.46	2.94	3.9	8.05	11.98
	LTM-GDME	1.41	2.72	3.44	6.85	10.83
	MMM-LTM-AWA	1.54	3.19	4.16	8.13	12.59
	GMA-LTM-AWA	1.4	2.89	3.8	7.81	11.67
Number of wire elongations	MMM-LTM-DME	6	20	24	57	152
	GMA-LTM-DME	4	15	20	41	64
	LTM-GDME	5	19	27	39	108
	MMM-LTM-AWA	5	19	32	51	129
	GMA-LTM-AWA	2	12	18	35	67
Clock latency (*psec*)	MMM-LTM-DME	1.69	5.07	6.93	20.07	35.6
	GMA-LTM-DME	1.56	3.84	6.29	11.98	32.92
	LTM-GDME	1.74	3.56	4.69	14.64	30.08
	MMM-LTM-AWA	1.71	3.27	6.59	19.09	37
	GMA-LTM-AWA	1.36	4.57	6.58	17.67	31.7
SDPL	MMM-LTM-DME	5577	4474	3420	3797	3996
	GMA-LTM-DME	9235	8216	7332	18412	12512
	LTM-GDME	5951	10365	8558	8785	7209
	MMM-LTM-AWA	5208	10008	2746	5912	4428
	GMA-LTM-AWA	4821	12268	11002	11149	9386
run time (sec)	MMM-LTM-DME	1	1	3	6	6
	GMA-LTM-DME	1	1	3	7	7
	LTM-GDME	23	122	772	8084	38659
	MMM-LTM-AWA	5	10	29	74	95
	GMA-LTM-AWA	6	10	29	77	93

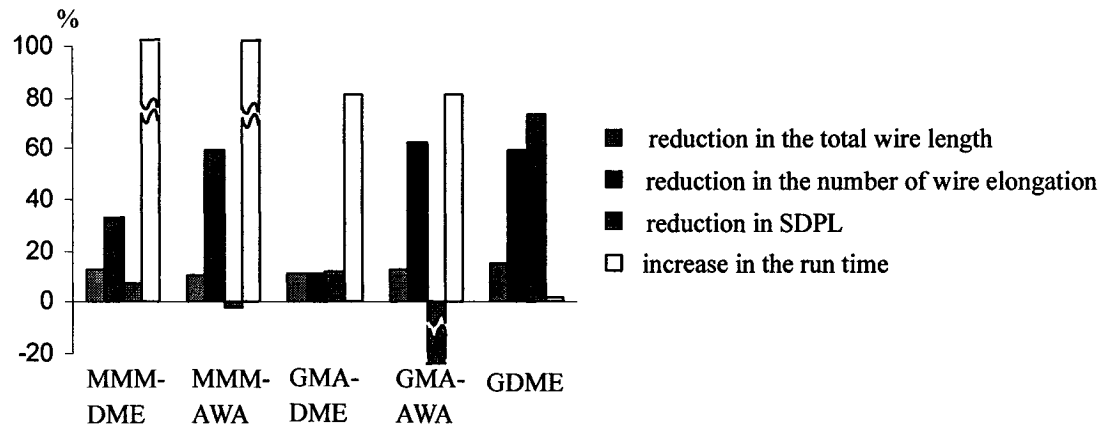


Figure 4.12 The comparisons of different parameters for benchmark r3 when the LTM is applied to different algorithms.

4.5.2 Impact of LTM on the Convergence of AWA Algorithm

Figure 4.13 shows the convergence of LTM-AWA for the benchmark r3. It is apparent that LTM-AWA requires less number of iterations than AWA as shown by Table 4.2. The main reason for such a reduction is that LTM-AWA balances four nodes at each iteration. Figure 4.14 shows the relationship between the number of clock pins for different benchmarks and the number of required iteration to reach ZS by using both AWA and LTM-AWA. Note that the reduction in the iterations does not necessary lead to a reduction in the run time since the run time depends on the number of iterations as well as on the time of each iteration. Similar to AWA, LTM-AWA minimizes the skew and the total wire length iteratively, and hence, getting smaller skew does not come at the price of an increase in the total wire length. Figure 4.15 shows the relationship between the total wire length and the iteration for the benchmark r3 using two initial topologies: one obtained by MMM and the other by GMA.

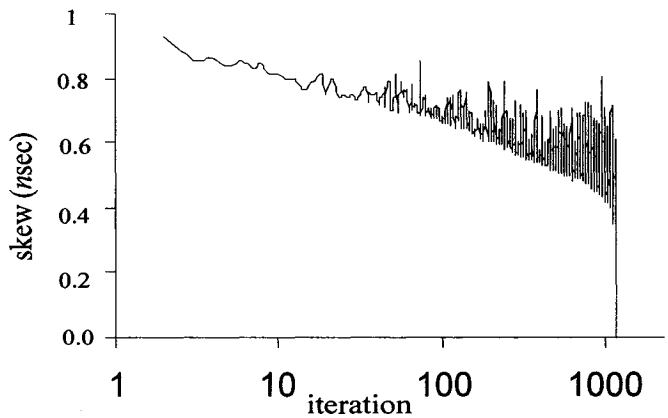


Figure 4.13 The skew convergence of LTM-AWA for benchmark r3.

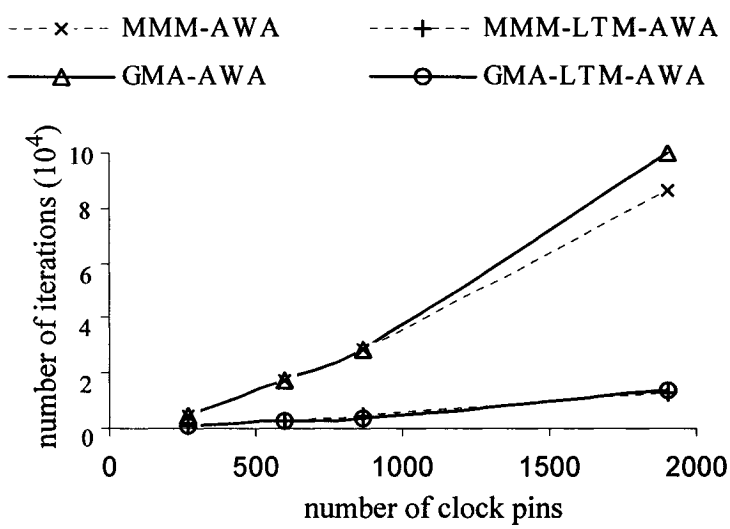


Figure 4.14 Relation of number of clock pins and number of iterations of AWA and LTM-AWA.

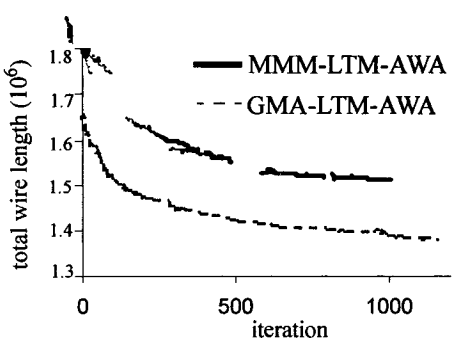


Figure 4.15 The relation of LTM-AWA's iteration and total wire length for benchmark r3.

Table 4.2 Number of iterations required by AWA and LTM-AWA for different benchmarks.

	benchmark	r1	r2	r3	r4	r5
Algorithm	MMM-AWA	4356	17410	28838	85963	162549
	MMM-LTM-AWA	842	2773	4304	12997	24713
Reduction (%)		80.67	84.1	85.1	84.88	84.8
Algorithm	GMA-AWA	4995	17572	28348	99554	183772
	GMA-LTM-AWA	871	2963	4105	13439	20684
Reduction (%)		82.56	83.14	85.52	86.5	88.74

4.6 Summary

In this chapter, a method of Local Topology Modification (LTM) is proposed in order to reduce the total wire length of the CDN. Such a reduction implies less power consumption. The reduction in total wire length is attributed to the topology flexibility that is offered by using LTM, however it comes at the cost of increase in the run time. Also, the experiments showed that GDME relies heavily on wire elongations, as well as it offers a solution that suffers from high SDPL. A high SDPL implies that the skew determination of the CDN is inaccurate, as well as the CDN is vulnerable to environment and process variations. By using LTM with GDME, SDPL and wire elongations can be reduced effectively. The concept of LTM can be considered for other versions of DME algorithm, such as the bounded skew DME and planar DME [44,80]. It may be also considered for different approaches that handle the presence of obstacles or when buffer insertion is required.

Previous algorithms, including the AWA algorithm, deal with the determination of the edge lengths of the CDN in order to meet the timing requirements. However, such wire

lengths are not necessarily realizable due to the difficulties of the routing step. Thus, it is important to consider the routability of the wires during the determination of the CDN. One major problem with routing a wire is the presence of obstacle in the routing plane as will be described in the next chapter.

Chapter 5

Tuning the Clock Distribution Network Under Obstacle Constraints

5.1 Introduction

The DME, AWA or GDME algorithms produce a CDN by determining the merging segments of the Steiner point in the tree. A merging segment stands for all possible locations of a Steiner node in the routing plane. The determination of a merging segment is accomplished with the assumption that the Steiner node can be located any where in the routing plane. Unfortunately, the routing plane may suffer from the presence of blockages, called obstacles, with which wires must not intersect. An obstacle might be a preplaced macro cell with blocked region or a previously routed wire. The obstacle interfere with the determination of the merging segments and with the routing of the wire of the CDN. The problem of routing a wire between two terminals under obstacle constraints is well addressed in the literature, where a shortest path can be determined to route such a wire. However, the obstacles affect the routing of the CDN, or any multi-

terminal net. In conclusion, the obstacles have to be considered when a Steiner node is embedded in the routing plane so that the system performance is optimized in terms of total wire length or the system frequency to name few.

In general, the problem of finding the shortest path has received considerable attention due to the wide variety of its applications, such as in robotics motion, VLSI design or geographical information systems to name few [71-78]. Different algorithms have been proposed to find the shortest path that may connect two points among rectilinear obstacles [71-78]. The solution is given as a set of vertical and horizontal segments. Indeed, such a solution is a specific routing for a specific shortest path. However, the shortest path is not unique, neither the routing of each shortest path. Different shortest paths, and their different routings, have different qualities, such as the number of turns or the separation from other paths or obstacles, although they have a minimum wire length. Thus, it is important to find the set of all solutions instead of finding a single solution. Having a simple representation of all shortest paths would help the designer to choose the appropriate path in response to other limitations.

In [70], a method is proposed to determine the merging region of a Steiner node in a routing plane under obstacle constraints. The procedure was incorporated into the first phase of DME to synthesize a BSCT. However, the limitation of such a procedure is that it considers only parts of the merging region of each Steiner node. The procedure is based on simple determination of the shortest paths; and whenever a shortest path is found, a part of the merging region is immediately determined. In fact, only five possible

parts of the merging region are explored at most. Consequently, some possible locations of the Steiner node are neglected, which would limit the capability of DME to reduce the total wire length further. In [69], the obstacle is treated by devising a set of rules to go around it. These rules are applied for a planar CT. However, the obstacles are not considered during the CT calculation. This implies that the impact of the obstacles on the total wire length has been neglected.

In addition to the limitations of the approaches in [69] and [70] in minimizing the total wire length, these approaches require the re-calculation of the whole solution when there is a change in the obstacles constraints, the skew constraint or the location or the load capacitance of the clock pins. However, it is often true that, during design iterations, IPs are inserted or removed. Consequently, the CDN's topology and the loading and placement of the clock pins are changed; and thus, the CDN has to be redesigned. This will be an extremely computation intensive process for a complex system. In view of the fact that the design convergence has become increasingly critical with the pressure for shorter time to market [12], what is required is an incremental algorithm to perform minor modification and to tune the CDN as opposed to redesigning it [49-54]. The AWA algorithm, as described in Chapter 3, enables a quick Engineering Change, EC, to the CDN without considering the obstacle constraints. In this chapter, a new methodology is proposed to determine the merging segment under obstacle constraints. Then the method is incorporated into AWA algorithm in order to tune the CDN under obstacle constraints.

This chapter focuses on the shape of the routing area of an interconnect that connects

two points. A simple and compact model is proposed, called Shortest Paths Polygon (SPP) to describe all shortest paths between two points in the presence of obstacles. This model, which is based on gridless graph, facilitates a better routing of any multi-terminal net. The SPP model is applied to the problem of tuning the CDN under obstacle constraints. A new method is proposed, based on the SPP model, and incorporated into AWA algorithm. Nevertheless, the method is general and applicable to other CDN algorithms as will be described in Chapter 6, as well as to other multi-terminal nets. The results show the capability of the proposed method in handling engineering changes efficiently.

5.2 Definitions

5.2.1 Manhattan Arc

A point, p, in a Manhattan plane is defined by its x and y coordinates (p_x,p_y). A Manhattan arc (hereafter referred to as arc), U, is a set of points that lie on a segment whose slope, U_{slope}, is ±1 as shown in Figure 5.1. An arc, U, can be defined by its two end points that will be referred to as head and tail, U^h and U^t respectively, such that:

$$U^h_y \leq U^t_y \tag{5.1}$$

where U^h_y and U^t_y are the y-coordinates of U^h and U^t respectively.

In addition, an arc has a specific displacement on the y-axis or x-axis. The displacement on the y-axis will be referred to as U_{disp}. Note that the case $U^h_y=U^t_y$ (or $U^h_x=U^t_x$) implies that U is a single point; and U_{disp} and U_{slope} would become undefined.

Figure 5.1 The Manhattan arc

5.2.2 The notations Ψ and Θ

Two points, say s and t, define a rectangle R(s,t) such that s and t are, respectively, either the bottom-left and top-right corners or the bottom-right and top-left corners of R(s,t) as shown in Figure 5.2. For a given rectangle R(s,t), which is abbreviated to R, two notations, Ψ and Θ, are defined as follows:

Definition 5.1: The notation Ψ(R), or Ψ(s,t), refers to the other two corner points of R. One corner, Ψ^+(R), is located at (s_x, t_y); and the other corner, Ψ^-(R), is located at (t_x,s_y).

Definition 5.2: The notation Θ(R,τ), or Θ(s,t,τ), refers to the two points on the border of R, which are located at a distance τ from s. One point, Θ^+(R,τ), is above or on the diagonal of R that passes through s and t. The other point, Θ^-(R,τ), is below or on the same diagonal.

Figure 5.2 shows the points Ψ^+(R), Ψ^-(R), Θ^+(R,τ) and Θ^-(R,τ) for the rectangle R(s,t). For the sake of simplicity, if a rectangle is defined by the end points of an arc U, s and t, then the notation Ψ(s,t) is abbreviated to Ψ(U). The notation $d(.)$ denotes the (Manhattan) distance between two arguments: two points, a point and an arc or two arcs.

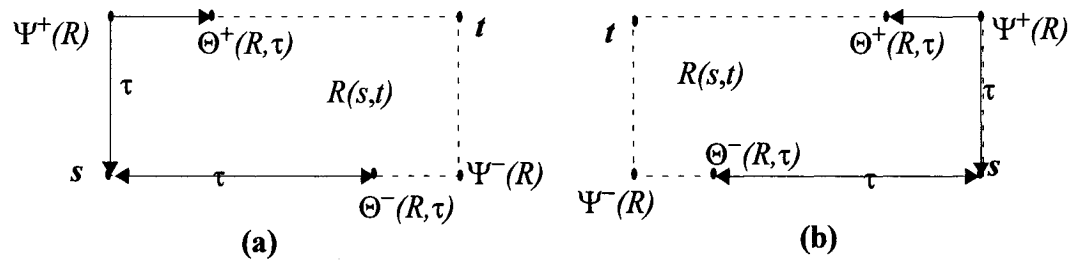


Figure 5.2. The notations Ψ and Θ

5.2.3 Problem Formulation

The problem of routing a two-terminal net with obstacle constraints is well addressed in the literature. This chapter addresses the problem of routing a multi-terminal net with obstacles in the routing plane. Given a set of obstacles and a set of sinks, where each sink is associated with a location and a load capacitance, the problem can be formulated as follows: tune the wires of a given CDN so that the skew between any pair of sinks is less than a specified bound where each Steiner node in the CDN is associated with a location in the Manhattan plane, a load capacitance and a signal delay from this node to its leaves. Tuning a CDN means adjustment of wire lengths and relocation of the Steiner nodes of the CDN, while minimizing the skew, so that no wire or Steiner point lies in any obstacle.

All obstacles are assumed to be of a rectangular shape and the wire can run around the edge of the obstacle. The AWA algorithm was proposed to tune the CDN without obsta-

cle constraints. In this chapter, AWA algorithm will be extended to tune the CDN under obstacle constraints. At first, the concept of Shortest Paths Polygon will be introduced next.

5.3 Shortest Paths Polygon

Let nodes *s* and *t* be the children of node *u* in the CDN as shown in Figure 5.3(a). Also, let the wire that connects *s* and *t* with minimum wire length in the Manhattan plane be *Wire(s,t)* as shown in Figure 5.3(b). If there is no obstacle constraints, then the length of *Wire(s,t)* is *D(s,t)*. Such a wire can be routed in different layouts in the routing rectangle. On the other hand, if R(*s,t*) intersects with obstacles as shown in Figure 5.3(c), then neither the routing area of *Wire(s,t)* is R(*s,t*), nor the length of *Wire(s,t)* is *D(s,t)*. In such a case, the length of *Wire(s,t)* can be determined using different shortest path algorithms, such as the maze, line search or graphical algorithms [71-77]. Any of these algorithms produces a solution that stands for a possible layout of *Wire(s,t)*. However, there is no algorithm that determines all solutions. Indeed, the problem of determining all solutions means the determination of the routing area of *Wire(s,t),* which has not been addressed in the literature. For example, the routing area of *Wire(s,t)* shown in Figure 5.3(c) is the polygon *s,a,b,c,t,d,e,f,g,h,s*. Thus, this polygon represents a complete solution for all shortest paths between *s* and *t*, and it will be called the *Shortest Paths Polygon* of *s* and *t*, *SPP*(*s,t*). Such a complete solution would have a great advantage for the routing phase of the system design process. For example, a detail router that has all solutions for *Wire(s,t)* can pick the optimum layout of *Wire(s,t)* in response to different design rules. Also, the router can modify the layout of *Wire(s,t)* without going back to a previous

design stage. Nevertheless, it is important to have a simple representation of the SPP in order to incorporate it into current routing algorithms without a significant increase in the computation time or the memory usage.

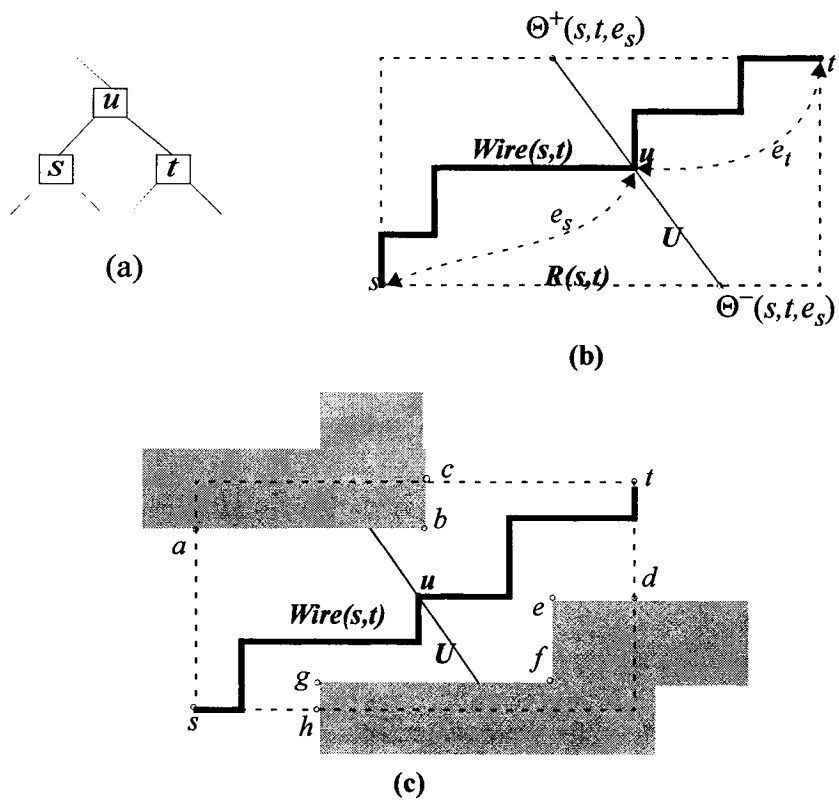


Figure 5.3 An example of connecting two nodes (a) the connection topology (b) the routing area of the connection without obstacle constraints (c) the routing area of the connection with obstacle constraints.

Definition 5.3: The Shortest Paths Polygon of two nodes, s and t, $SPP(s,t)$, is the smallest polygon that includes all possible shortest paths between s and t without intersecting with any obstacle.

The SPP of two points has some interesting properties that will help in developing a concise representation and simple determination of the SPP. Let node u be the parent of the pair of nodes s and t in the CT; and e_s and e_t be the edge lengths of s and t respectively as shown in Figure 5.4(a). Accordingly, node u can be any tapping point on *Wire(s,t)* such that $D(s,u)=e_s$ and $D(t,u)=e_t$ as shown in Figure 5.3(b). Indeed, there would be different possible locations for u in the Manhattan plane since there are different layouts for *Wire(s,t)*. The set of all possible locations of u constitutes the merging segment of u, U; which is a Manhattan arc [41-43]. If there is no obstacle constraints, then *Wire(s,t)* can be routed in *R(s,t)*; and consequently, U is located inside *R(s,t)* as shown in Figure 5.3(b) (see Appendix B for the details). However, if *R(s,t)* intersects with an obstacle, as shown in Figure 5.3(c), then *Wire(s,t)* can be routed in *SPP(s,t)*; and U is located inside *SPP(s,t)*.

Consider the case of a single obstacle intersecting with *R(s,t)* as shown in Figure 5.4. Accordingly, *SPP(s,t)* is the polygon *s,a,b,c, t,d,s*. Depending on e_s and e_t, there are different possible arcs for U, which are shown as solid and dotted arcs in Figure 5.4. These possible arcs of U are grouped according to the following:

Definition 5.4: A set of arcs, $\mathcal{A}$, includes all arcs that have the same slope; and the end points of all arcs lie on the border of a single rectangle R (referred to as the rectangle of $\mathcal{A}$).

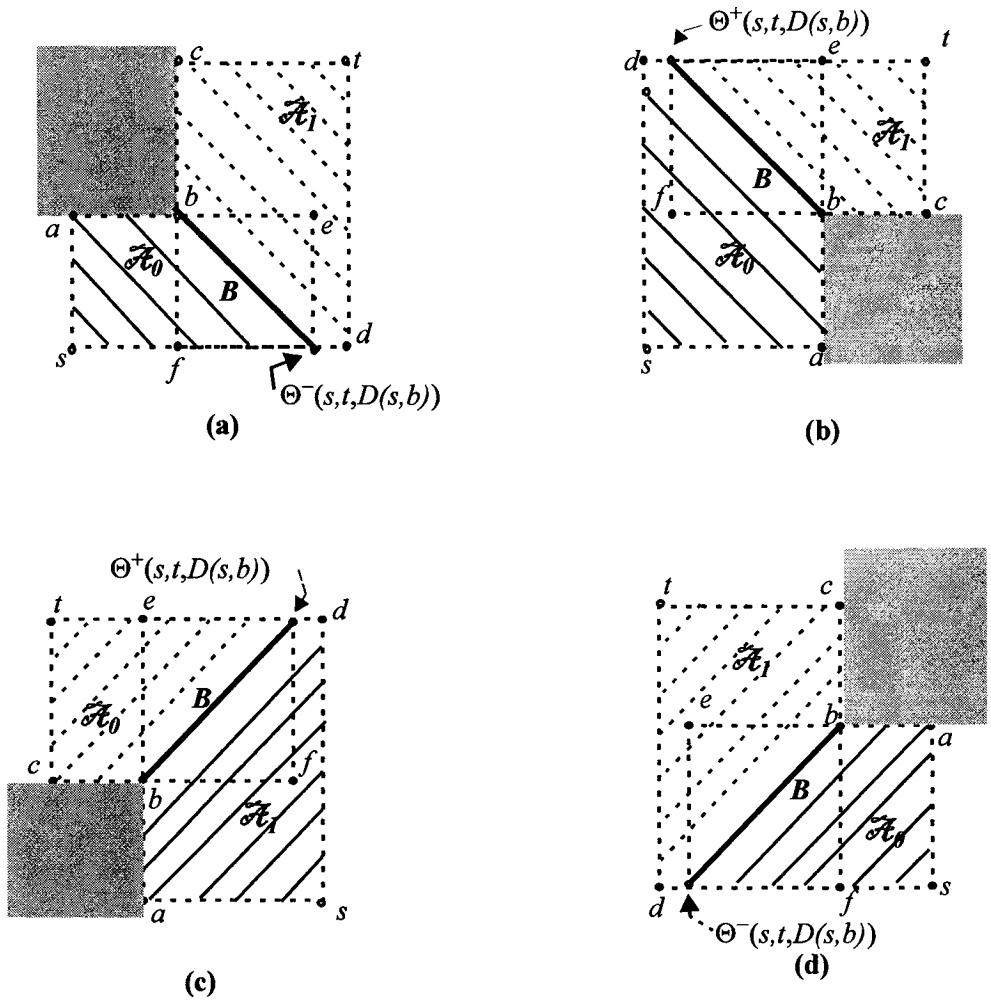


Figure 5.4 The signs of the corner b (a) $\alpha=\beta=-1$ (b)$\alpha=-1$ and $\beta=+1$ (c)$\alpha=\beta=-+1$ (d)$\alpha=+1$ and $\beta=-1$.

The sets of the solid and dotted arcs are labeled as $\mathcal{A}_0$ and $\mathcal{A}_1$ respectively as shown in Figure 5.4. The end points of the arcs of $\mathcal{A}_0$ lie on the border of $R(s,e)$. Similarly, the end points of the arcs of $\mathcal{A}_1$ lie on the border of $R(f,t)$. That is $R(s,e)$ and $R(f,t)$ are the rectangles of $\mathcal{A}_0$ and $\mathcal{A}_1$ respectively. The arc that has the minimum displacement in $\mathcal{A}_0$ is the

point s, and the arc that has the maximum displacement in $\mathcal{A}_1$ is B as shown in Figure 5.4. Similarly, the arcs that have the minimum and maximum displacement in $\mathcal{A}_1$ are B and t respectively. The arcs that have the minimum and maximum displacement in arc set, $\mathcal{A}$, will be referred to as the head ($\mathcal{A}^h$) and tail ($\mathcal{A}^t$) of $\mathcal{A}$ respectively. One may notice that $\mathcal{A}_0$ and $\mathcal{A}_1$ are separated by B, where $\mathcal{A}^t_0 = \mathcal{A}^h_1 = B$.

Definition 5.5: The *boundary arc* between two arc sets is the head of one arc set and the tail of the other.

***Corollary 5.1*:** The rectangle of an arc set does not intersect with any obstacles.

Proof: If there is an obstacle O intersects with the set's rectangle, R, then O would eliminate portions of some arcs. As such, the end points of these arcs do not lie on the border of R, which is a contradictory to the definition of the arc set.

5.3.1 Polarity of a Convex Point of an Obstacle

A SPP or an obstacle is defined by a set of points; where each point is either a concave or convex point. A convex point of an obstacle would be a concave point of an SPP, and vice versa. Without losing generality, we assume that all obstacles have rectangular shapes or combinations of rectangular shapes. A rectangle can be defined by its four corners which are convex points. A corner of a rectangle can be top-left, top-right, bottom-right or bottom-left corner. For example, consider the point b shown in Figure 5.4 which is a concave point in the SPP and a convex point of the obstacle. In addition, the point b is an end point of the boundary arc B. Indeed, the arc that is originated at a concave point

of a SPP would be a boundary arc between two arc sets in the same SPP. Such a boundary arc has a slope of ± 1 as shown in Figure 5.4. Thus, every convex point of an obstacle, p, is assigned a sign, α, that stands for the slope of the boundary arc, which is originated at p. For example, in the case that the obstacle is a rectangle, α is positive if p is a bottom-left or top-right corner; and negative otherwise.

One end point of each boundary arc is a concave point of the SPP, while the other end point can be determined from s, t and other concave points of the SPP. For the case shown in Figure 5.4(a), the other end point of B is $\Theta^{-}(s, t, D(s,b))$. The sign of the operation Θ can be deduced from the fact that a shortest path passes each boundary arc by moving around a convex point of an obstacle where the boundary arc is originated. In general, the sign of the operation Θ is used as another sign, β, for each convex point p of an obstacle such that β is positive if p is the top-left or top-right corner of the obstacle, and negative otherwise as shown in Figure 5.4. In conclusion, each convex point, p, of an obstacle has two signs, $p^{\alpha,\beta}$, as described in Table 5.1.

Table. 5.1 Signs of a corner of an obstacle

corner	α	β
bottom-right	-	-
bottom-left	+	-
top-left	-	+
top-right	+	+

5.3.2 SPP's Vertices

It is important to have a simple model of the SPP in order to incorporate it effectively into current routing algorithms. Consider the case of two obstacles intersecting with $R(s,t)$ as shown in Figure 5.5(a), where $SPP(s,t)= s,a,o_4, b, t, c, o_6, d,s$. The possible arcs of the parent node, U, are grouped into three sets: $\mathcal{A}_0$, $\mathcal{A}_1$ and $\mathcal{A}_2$. These arc sets are separated by two boundary arcs, B_4 and B_6, as shown in Figure 5.5(a). The convex points a, b, c and d of $SPP(s,t)$ can be determined from s, t and the concave points of $SPP(s,t)$ as follows:

$$a=\Psi^{+}(s,o_4),\ \ b=\Psi^{+}(o_4,t),\ \ c=\Psi^{-}(o_6,t)\ ,\ d=\Psi^{-}(s,o_6)$$

Thus, it is possible to represent $SPP(s,t)$ by its source, destination and concave points only as will be described next.

Each boundary arc of a SPP is originated at a concave point of the SPP. In addition, the wire that connects s and t, $Wire(s,t)$, passes through these boundary arcs in a certain sequence. Thus, the concave points of a SPP can be ordered according to the order of passing through their boundary arcs by $Wire(s,t)$. For example, the SPPs shown in Figure 5.5(a) and (b) are s,o_4^-,o_6^+,t and $s,o_1^{+,-},o_5^-,o_7^+,o_4^-,t$ respectively. One may notice that two sequential concave points of a SPP have to have the same α sign unless there is a detour around an obstacle. For example, $SPP(s,t)$ shown in Figure 5.5(b) detours around the corners o_1 and o_5. Accordingly, o_1 and o_5 appear in sequence in $SPP(s,t)$ though their α signs are different. For such a case, the boundary arcs that are originated at these two points, B_1 and B_5, have zero lengths since there is a detour around these two points. Also, the arcs of the set between these two boundary arcs, $\mathcal{A}_1$, have zero length. Otherwise, if

there is no detour around an obstacle, then the sequence of the points of the SPP have the same α sign (their boundary arcs have the same slope). Consequently, such points are ordered according to the displacement of their boundary arcs. For example, the points o_5, o_7 and o_4 of *SPP(s,t)* shown in Figure 5.5(b) are listed in a descending order by the displacement of B_5, B_7 and B_4. In general, the concave points of a SPP have the following properties:

Property 1: two sequential points in a SPP have the same α sign unless there is a detour around an obstacle.

Property 2: sequential points are listed in descending or ascending order by the displacement of their boundary arcs.

In conclusion, the SPP can be modeled by its source, destination and the concave points, where each concave point has two signs α and β. All other convex points of the SPP can be determined from the set of points that represent the SPP.

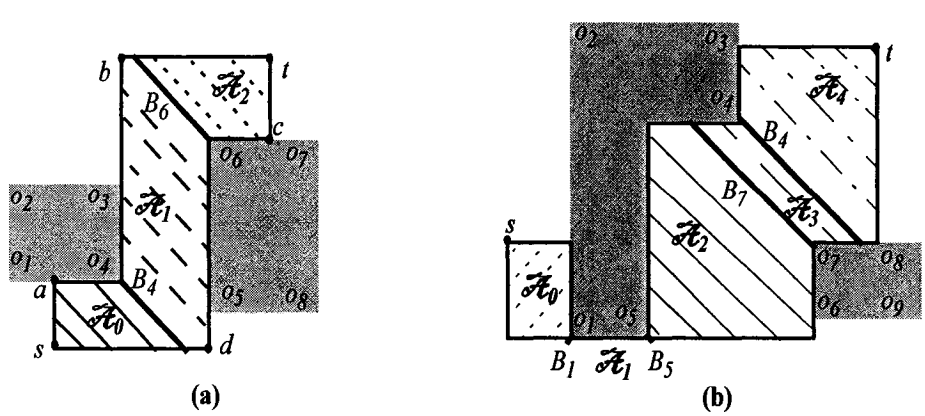


Figure 5.5 The Shortest Paths Polygon.

For a SPP whose set of points are $p_0, p_1, .. p_n$, the cost of a wire that connects p_0 to p_n, $Wire(p_0, p_n)$, is given as:

$$m(p_0, p_n) = \sum_{k=0,n-1} d(B_k, B_{k+1}) \tag{5.3}$$

where: B_i is the boundary arc that is originated at point p_i of the SPP.

5.4 SPP Determination

The SPP of two points, s and t, can be determined using a visibility graph, G, whose vertices are s, t and the obstacle corners. Pair of vertices in G are connected by an edge if they correspond to a pair of points whose rectangle does not crossover any obstacle. Figure 5.6 shows the visibility graphs of the cases shown in Figure 5.5. In order to simplify the graph, two vertices are connected only if they have the same slope, except if the two vertices belong to the same obstacle. This is due to the fact that two sequential points in SPP have to have the same α sign unless if there is a detour around an obstacle.

A greedy approach can be used to find shortest paths between s and t in the graph G, where there can be more than one shortest path. Such an approach would find the shortest paths in order $O(n^2)$, where n is the number of obstacles. However, some paths are redundant, and can be neglected according to the following observations:

Observation 1: A path $p_0, .., p_{n-1}, p_n$ becomes redundant if one of the points $p_0, .., p_{n-2}$ is connected to p_n in G

Observation 2: A path $p_0,..,p_{n-1},p_n$ becomes redundant if it has two sequential points, p_i and p_{i+1}, that have different slopes and belong to one obstacle such that the later obstacle does not intersect with the rectangles $R(p_{i-1},p_{i+1})$ or $R(p_i,p_{i+2})$.

The first observation helps in preventing an unnecessary point from appearing in the shortest paths. For example, the path s,o_1,o_5,o_6,o_4 (Figure 5.5(b) and Figure 5.6(b)) becomes redundant since o_5 is connected to o_4. The second observation helps in preventing an unnecessary detour from appearing in the shortest path. For example, the path s,o_5,o_6 (Figure 5.5(a) and Figure 5.6(a)) becomes redundant since the rectangles $R(s,o_6)$ does not intersect with the obstacle whose corners are o_5,o_6,o_7 and o_8. Neglecting the redundant paths, coupled with reducing the number of edges in G, would speed up the procedure of finding the shortest paths. In order to use the set of points of a non-redundant path as a set of points of the SPP, the interrelation between shortest paths need to be investigated at first.

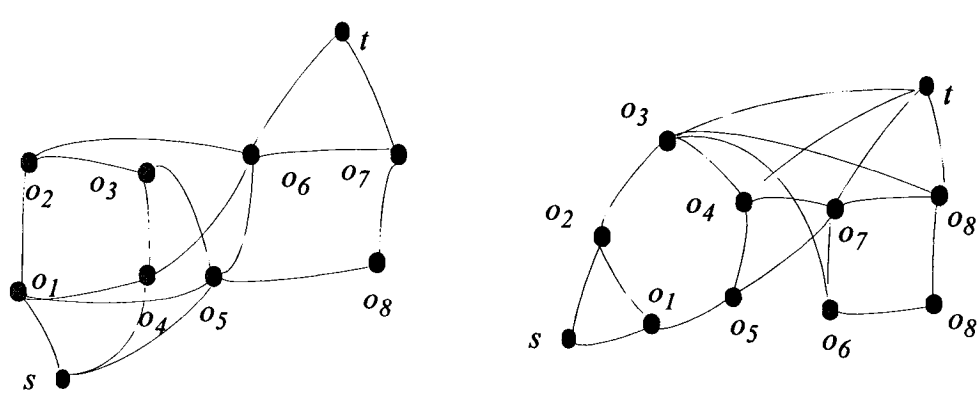


Figure 5.6. The graph representation.

5.4.1 The Interrelated Shortest Paths

Unfortunately, a non-redundant shortest path does not necessarily trace all points of the SPP were shortest path would be routed. For example the path s,o_1,o_5,o_4,t does not trace o_7 as shown in Figure 5.5(b). Similarly, the path s,o_1,o_5,o_7,t does not trace o_4. However, a wire connecting s and t according to the two paths would pass the boundary arcs B_4 and B_7, since the two paths are routed in the same SPP. Hence, in order to determine the set of the points of the SPP, the two paths have to be *merged* into one path. Consequently, the points of the new path are ordered by the displacement of their boundary arcs. For example, the interrelated paths shown in Figure 5.5(b) are merged into s,o_1,o_5,o_7,o_4,t, which represent the SPP of s and t.

Definition 5: let $p_0,...p_i,...p_j...p_n$ and $q_0,...q_k...q_l,...q_m$ be two shortest paths in the visibility graph G, such that $p_0,..,p_i$ and $p_j,..,p_n$ are similar to $q_0,..,q_k$ and $q_l,..,q_m$ respectively. The two paths are interrelated if the boundary arcs of $p_i,..,p_j$ and $q_k,..,q_l$ have the same slope and if they can be ordered in one path according to corollary 5.1.

5.4.2 Boundary Arcs Determination

In order to determine the merging segment, it is necessary to determine first the end points of the boundary arcs. As described in Section 5.4, one end point of a boundary arc is a corner of an obstacle. The other end point of the boundary arc would be located on the border of the SPP. For example, consider the case shown in Figure 5.7, where $R(s,t)=$ s,o_2,o_6,t. The end point of B_6 are o_6 and $\Theta^+(o_2t,m(o_2o_6))$. This implies that, the immedi-

ate left and right neighbors of the origin o_6, o_2 and t respectively, are used to determine the other end point of B_6. However, determining a boundary arc from the immediate neighbors of its origin may result in eliminating a portion of that arc. For example, determination the other end point of B_2 from s and o_6 implies that the other end point lies on the border of $R(x_1,o_6)$, where $x_1=\Psi^+(s,o_2)$. But, B_2 can be extended further into the rectangle $R(x_2,t)$, where $x_2=\Psi^+(x_1,t)$, since this rectangle does not intersect with any obstacles. In fact, the other end point of B_2 is $\Theta^+(s,t,m(s,o_2))$.

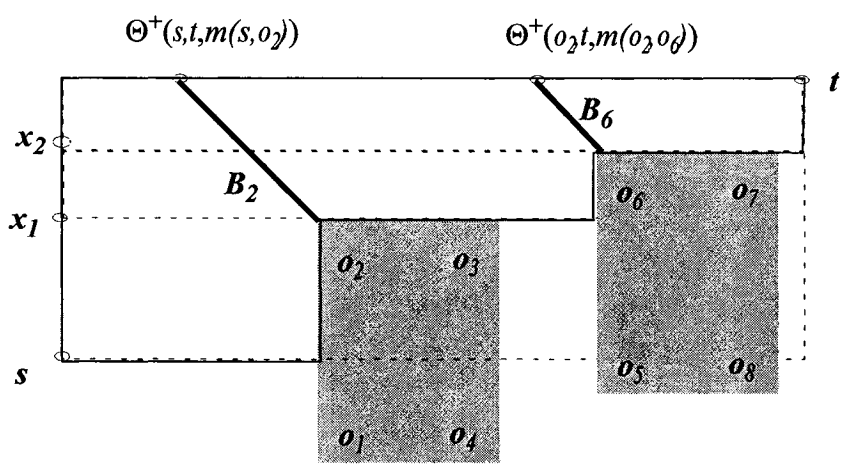


Figure 5.7 The extension of the boundary arcs.

In general, for a boundary arc B, which originates at p, the other end point of B can be determined by applying the operation Θ to two points: one point from the left neighbors of p, p_l which will be referred to as the left associate of p ; and one point from the right neighbors of p, p_r, which will be referred to as the right associate of p.

Definition 5.6: The left and right associates, p_l and p_r respectively, of a boundary arc B, whose origin is p, are points from the left and right neighbors of p in the SPP that determine the other end point of B as follows:

$B^h=p$ and $B^t=\Theta^{\beta}(p_l p_r, m(p_l p))$ if $\alpha=+1$

$B^h=p$ and $B^t=\Theta^{\beta}(p_l p_r, m(p_l p))$ O.W

where α and β are the signs of p

For the example shown in Figure 5.7, the left associate of o_6 becomes the point s instead of o_2 because both o_2 and o_6 have the same β sign. Similarly, the right associate of o_2 becomes the point t instead of o_6 because both o_2 and o_6 have the same β sign. Accordingly, the right associate of B_2 becomes t; and the end points of B_2 are o_2 and $\Theta^{+}(s,t,\mathrm{d}(s,o_2))$. In the same manner, the left associate of B_6 becomes point s; and the end points of B_6 are o_6 and $\Theta^{+}(s,t,\mathrm{d}(s,o_6))$. In general, when there are more than three points in the shortest path, the left and right associates of a boundary arc have to move respectively towards the source and destination of the shortest path as follows:

Lemma 5.1: For the SPP $p_0,\ldots p_n$, let p_i, $0<i<n$, be the origin of a boundary arc B_i. Also, let α and β be the signs of p_i. The right associate of p_i traces the points to the right of p_i to become as p_r , $r>i$, such that p_r is the first point whose β sign is different from the β sign of p_i. Similarly, the left associate of p_i, moves towards the source p_0 and becomes p_l, $l<i$, such that p_l is the first point whose β sign is different from the β sign of p_i.

Proof: According to the Lemma, the right associate of p_i, p_r has to trace the points of SPP to the right of p_i. Let k be the index of tracing the points to the right of p_i. Also, let χ_1 be defined as follows:

$\chi_1=\Psi^{\beta}(p_{i-1},p_i)$

where β is the β sign of p_i.

At each iteration k, let χ_k be defined as follows:

$\chi_k=\Psi^{\beta}(\chi_{k-1},p_{i+k-1})$

where β is the β sign of p_i.

At first iteration, k=1 and $p_r=p_{i+1}$, where $r=i+k$. Accordingly, B_i can be extended into the rectangle $R(\chi_1,p_r)$. If the β sign of p_r is different from the β sign of p_i, then the rectangle $R(\chi_2,p_{r+1})$ intersects with the obstacle of p_r. Thus, B_i cannot be extended into the rectangle $R(\chi_2,p_{r+1})$; and the right associate of p_i is p_r. On the other hand, if the β signs of p_r and p_i are the same, then the rectangle $R(\chi_2,p_{r+1})$ must not intersect with any obstacle; and B_i can be extended into this rectangle. Consequently, k=2; and $p_r=p_{i+2}$. The index k can be increased as long as the β sign of p_i and p_{i+k} are the same; till $i+k=n$ (n is the index of the last point in the SPP). Similarly, the left associate of p_i traces the points to the left of p_i till the β sign of p_{i-k} is different from the β sign of p_i or $i-k=0$.

For the cases discussed previously, there was a single shortest path. But, for the general case, there might be more than one shortest path as shown in Figure 5.8, where there are two shortest paths: s,o_2,t and s,o_4,t. The obstacle O may break down the merging segment B into two parts; therefore, by considering one shortest path, one part of B would be neglected. In the case shown in Figure 5.8, the possible arcs of different parts of B can be located into four arc sets $\mathcal{A}_1$, $\mathcal{A}_2$, $\mathcal{A}_3$ and $\mathcal{A}_4$. The shortest path s,o_2,t provides only the part of B that is located in $\mathcal{A}_1$ or $\mathcal{A}_2$. On the other hand, the shortest path s,o_4,t provides the part of B that is located in $\mathcal{A}_3$ or $\mathcal{A}_4$. The two parts that are produced by the two shortest paths may overlap on each other. For example, if $e_s<m(s,o_1)$, then the two parts are located in $\mathcal{A}_1$ and $\mathcal{A}_3$, and they overlap with each other. But, this is not always the case. For example, if $m(s,o_1)<e_s<m(s,o_4)$, then the obstacle O breaks down the merging segment into two non-overlapped parts, where each part can be obtained from one of the shortest paths. In general, all the shortest paths have to be explored in order to calculate all possible parts of a merging segment.

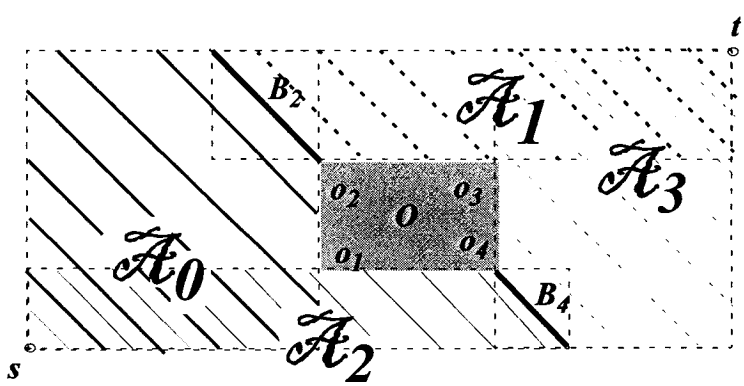


Figure 5.8. The overlapping between different parts of a merging segment.

5.5 Balancing Segment Determination

In Section 2.5.2, the merging segment of node u, B_u, is determined from its child arcs B_s and B_t by constructing tilted rectangles without considering the obstacle constraints. A simpler method is presented in Appendix A to determine the merging segment without obstacle constraints by using the notations Ψ and Θ. However, due to the obstacle constraints, the merging segment has to be determined differently. Let the child segments be the points s and t; and they have to be merged by the merging segment B_u, where $SPP(s,t)= s,o_1,o_2,o_3,o_4,t$ as shown in Figure 5.9.. Thus, there are four boundary arcs, B_1, B_2, B_3 and B_4, that separate five arc set. Assume that the merging segment B_u belongs to the arc set $\mathcal{A}_2$, which is located between B_2 and B_3. That is, the head and tail of $\mathcal{A}_2$ are the arcs B_2 and B_3 respectively. Note that B_u merges the two points s and t; and the wire length from s and t to B_u are e_s and e_t respectively as shown in Figure 5.9. By using the edge lengths of s and t, e_s and e_t respectively, the arc set of a merging segment can be decided as follows.

***Lemma 5.2*:** For a SPP $p_0,...p_n$, the merging segment, B, belongs to the arc set $\mathcal{A}_k$, whose head and tail are $\mathcal{A}^h_k$ and $\mathcal{A}^t_k$ if:

$$e_0 \geq m(p_0,\mathcal{A}^h_k) \text{ and } e_n \geq m(\mathcal{A}^t_k p_n) \tag{5.4}$$

Proof: The wire lengths from the source, p_0, and the destination, p_n, to the merging segment B are e_0 and e_n respectively. Also, according to the given SPP, the shortest wire length from p_0 to the head of $\mathcal{A}_k$ is $m(p_0,\mathcal{A}^h_k)$; and the shortest wire length from p_n to the

tail of $\mathcal{H}_k$ is $m(p_n, \mathcal{H}_k)$. Thus, if the points p_0 and p_n are to be connected with minimum wire length; and $e_0 \geq m(p_0, p_k)$ and $e_n \geq m(p_{k+1}, p_n)$, then $B \in \mathcal{H}_k$ must be true.

For the case shown in Figure 5.9, $B_u \in \mathcal{H}_2$, since B_u is located between B_2, which is the head of $\mathcal{H}_2$, and B_3, which is the tail of $\mathcal{H}_2$. Consequently, the end points of B_u are located on the border of the rectangle $R(\Psi^-(B_2), \Psi^+(B_2))$. In general, to determine a merging segment B_u from a SPP, $p_0, \ldots p_n$, such that $B_u \in \mathcal{H}_k$, the head and tail of B_u, B_u^h and B_u^t respectively, are calculated as follows:

$$B_u^h = \Theta^-(\Psi^-(\mathcal{H}^h_k), \Psi^+(\mathcal{H}^t_k), l)$$

$$B_u^t = \Theta^+(\Psi^-(\mathcal{H}^h_k), \Psi^+(\mathcal{H}^t_k), l) \qquad (5.5\ A)$$

where l is defined as follows:

$$l = e_0 - m(p_0, \mathcal{H}^h_k) + d(\mathcal{H}^h_k, \Psi^-(\mathcal{H}^h_k)) \qquad (5.6\ A)$$

With the assumption that $m(p_0, \mathcal{H}^h_k) \leq m(p_0, \mathcal{H}^t_k)$ as shown in Figure 5.9. On the other hand, if $m(p_0, \mathcal{H}^t_k) \leq m(p_0, \mathcal{H}^h_k)$, then B_u^h and B_u^t are calculated as follows:

$$B_u^h = \Theta^-(\Psi^+(\mathcal{H}^t_k), \Psi^-(\mathcal{H}^h_k), l)$$

$$B_u^t = \Theta^+(\Psi^+(\mathcal{H}^t_k), \Psi^-(\mathcal{H}^h_k), l) \qquad (5.5\ B)$$

where l is defined as follows:

$$l = e_n - m(p_0, \mathcal{H}^h_k) + d(\mathcal{H}^h_k, \Psi^+(\mathcal{H}^h_k)) \qquad (5.6\ B)$$

Figure 5.9 shows the determination of the head and tail of the merging segment B_u, $\Theta^-(\Psi^-(B_2),\Psi^+(B_3), l)$ and $\Theta^+(\Psi^-(B_2),\Psi^+(B_3), l)$ respectively. Note that the Equation 5.5.A and the Equation 5.6.A are used to determine B_u since $m(s,B_2) < m(s,B_3)$.

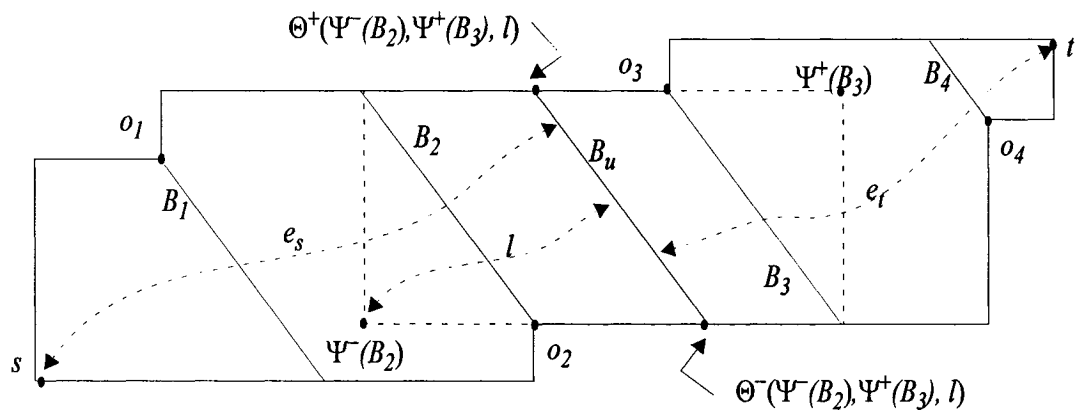


Figure 5.9 Merging segment determination

Based on the Lemmas and Corollaries presented in previous sections, a procedure can be drawn to determine the merging segment of two points as described in Figure 5.10.

Input: locations, load capacitances and time delays of child nodes s and t, and a set of obstacles.
Output: merging segment B of the parent node and its time delay and capacitance

Construct the visibility graph for s and t and the set of obstacles
Find the shortest paths in the visibility graph
Merge interrelated paths according to Corollary 5.1
For each shortest path
  Find the arc set, $\mathcal{A}_k$, of B according to Lemma 5.1.
  Determine B according to Equation 5.5
Eliminate the overlapping between different parts of the merging segment

Figure 5.10 The *Merging Segment Determination* procedure.

Theorem: the procedure *Merging Segment Determination* can find all possible locations of connecting a node to its children with minimum wire length.

Proof: Two child nodes can be connected with minimum wire length if the wire follows a shortest path among the obstacles. By constructing a visibility graph for the child nodes and the obstacles, all shortest paths between the child nodes can be found. For each shortest path, all possible locations of the parent node, merging segment, can be determined according to Equation 5.5. Thus, the determination of all shortest paths gives all possible locations of the parent node.

5.6 Incorporating the SPP methodology into AWA algorithm

In the previous discussion, every node in the visibility graph is assumed to correspond to a point in the routing plane. The link between a pair of nodes is established whenever the rectangle of the two points, that correspond to the pair of nodes, does not intersect with any obstacle as described in Section 5.3. The procedure shown in Figure 5.10 determines the merging segment of a node based on the SPP of the child nodes, s and t, with the assumption that s and t correspond to single points. Thus, the proposed method can be incorporated into AWA algorithm, which will be called SPP-AWA, since the child nodes of the balancing node are single points. At each iteration, the procedure is called to determine the merging segment of the balancing node, which is the parent node in the procedure. Then the final location of the balancing node is selected to be the closest point of the resulting merging segment to its parent. According to this procedure, the visibility graph is built at each iteration for the children of the balancing node, s and t,

and all obstacles. However, the merging segment, B, is mostly located inside $R(s,t)$; and only the obstacles that intersect with $R(s,t)$ would intersect with B. Thus, only the obstacles that intersect with $R(s,t)$ are considered during the visibility graph building in the procedure so that the visibility graph becomes smaller and the shortest path is found faster. But, it is necessary to check if the resulting B intersects with other obstacles that do not intersect with $R(s,t)$. If there is an intersection, then the procedure has to be called again with consideration of the obstacles that intersect with the resulting merging segment in the first call.

Note that the slope of arc sets of a SPP can be ± 1. But the slope of two sets that are separated by a boundary arc would be either the same or the arcs of one set are of zero length. For example, a wire connecting s and t, as shown in Figure 5.11(a), would be routed in $SPP(s,t)=s,o_1,o_8,o_7,o_6,t$, which is the result of merging the paths s,o_1,o_7,o_6,t and s,o_8,o_7,o_6,t, as shown in Figure 5.11(b). Accordingly, there are four boundary arcs that separate the possible locations of the merging segment into five arc sets: $\mathcal{A}_1$, $\mathcal{A}_2$, $\mathcal{A}_3$, $\mathcal{A}_4$ and $\mathcal{A}_5$. The slope of $\mathcal{A}_1$, $\mathcal{A}_2$ and $\mathcal{A}_3$ is +1; while the slope of $\mathcal{A}_5$ is -1. Thus, the arcs of the set $\mathcal{A}_4$, including its boundary arcs $B_6$and B_7, have zero length.

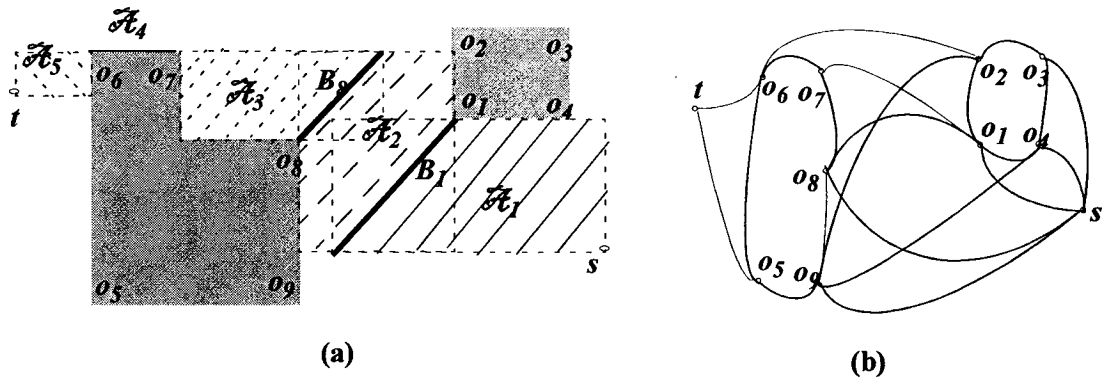


Figure 5.11 (a) The possible arcs of the merging segment have ±1 slope. (b) The visibility graph.

5.7 Implementation and Results

The procedure *Merging Segment Determination* was implemented using C++ language on a SUN Ultra 10 machine in Unix environment. Different experiments are performed in order to show the capability of the proposed approach to handle obstacles and to minimize total wire length simultaneously. To provide assessments of different metrics, the experiments were performed using the same five benchmarks, r1-r5, presented in Section 3.4. Different number of obstacles of different sizes were added randomly to each benchmark so that there is no intersection between any two obstacles as well as no sink is located within an obstacle. The size of different obstacles were set randomly between 5% and 20% of the die area. The obstacles were located randomly in the die area such that the obstacles do not cross each other and no clock pin is located inside an obstacle. A modified version of Mean and Median Method (MMM) [3]was used in order to generate the initial topology.

5.7.1 Building ZSCDN Using SPP-AWA

Different benchmarks were used as inputs to the SPP-AWA in order to study the impact of the obstacles on the convergence and the total wire length. Figure 5.12 shows the skew convergence of SPP-AWA to achieve ZSCDN for r1 when there are 40 rectangular shape obstacles as described previously in the die area. Investigating the large spikes in the skew convergence turned out that these spikes ensue when the shortest path is lengthened by detouring around obstacles. Note that the previous approaches in [69 and 70] require the recomputation of the whole solution whenever a change in the CDN is made. Furthermore, for the approach in [70], the reduction of skew comes at the expense of extra wire length. On the other hand, SPP-AWA reduces the total wire length and the skew simultaneously as shown in Figure 5.12. Similar to the approach in [70], the total wire length somehow increases linearly with the increase in the number of obstacles as shown in Figure 5.13. Table 5.2 provides comparisons of total wire length, number of wire elongations, number of iteration and run time for various benchmarks. By comparing Table 5.2 to Table 3.2, one may notice that 40 obstacles result in only 5% increase in the total wire length, where by using the approach in [70], the same number of obstacles result in 12% increase. The capability of SPP-AWA to reduce the obstacle impact on the total wire length is attributed to the determination of all possible locations of the merging segment of a Steiner node. This was achieved by using the SPP modeling of the routing area of each wire in the CDN.

Also, from Table 5.2, one may notice that the increase in the number of obstacles does not necessarily result in an increase in the number of iterations. However, the run time

increases somehow linearly with the increase in the number of obstacles as shown in Figure 5.14. This is due to the fact that the size of the visibility graph increases linearly with the increase in the number of obstacles, and searching the visibility graph for the shortest paths accounts for the increase in the run time though the number of iterations is less. Note that SPP-AWA searches for the shortest paths locally as described in Section 5.5.

Table 5.2 Different metrics for different benchmarks using SPP-AWA. Each benchmark is compounded with 40 obstacles.

benchmark	r1	r2	r3	r4	r5
Total wire length (cm)	17.1	36.1	47.2	93.5	139.5
Number of wire elongations	233	530	794	1767	2925
number of iterations	4653	17478	27603	83892	201232
run time (sec)	5	18	32	80	244

5.7.2 CDN Tuning Under Obstacle Constraints

The main advantage of SPP-AWA is to handle minor modifications. Three experiments were carried out to study the sensitivity of the proposed approach to the number of shifted sinks, the size of the shifting of the sinks and shifting in the obstacle location. Note that other approaches require the recalculation of the whole solution when there is a minor modification. These experiments will be described next.

Experiment 1: Sensitivity to the Number of Shifted Clock Pins

Figure 5.15 shows the relationship between the number of shifted sinks and the required number of iteration to achieve zero skew. Note that in order to have a realistic estimation of the sensitivity of SPP-AWA to the increase in the number of shifted sinks, the experiment is repeated four times for each number of shifted sinks. The size of the shifting was set to 5% of the routing area. Further, the sinks were shifted in different directions to avoid placing a sink in an obstacle.

Experiment 2: Sensitivity to the Size of the Shifting in the Clock Pins

Figure 5.16 shows the relationship between the size of a shift of a fixed number of sinks (was set to five) and the required number of iterations to achieve zero skew. In order to have a realistic estimation of this sensitivity measure, the experiment is repeated four times for each size of shift, where the shift of the sink are varied between 5% to 30% of the routing area.

Experiment 3: Sensitivity to the Size of the Shifting in the Obstacles

The purpose of this experiment is to investigate the impact of the shift of a routing obstacle on the ZSCDN. Three obstacles were selected randomly and shifted by different amount so that no obstacle intersects with each other. If a shifted obstacle overlaps a sink or a Steiner node, the sink or the Steiner node would be moved to a location outside the obstacle. Such a modification would impact the skew between different sinks. Figure 5.17 shows the relationship between the shifting in the obstacles and the required number of iteration to achieve ZSCDN.

5.8 Summary

In this chapter, a new method was proposed to determine all possible locations of a merging segment with obstacle constraints. The method is based on building a visibility graph to find the shortest paths around obstacles. Two new concepts were introduced: the boundary Manhattan arc and the interrelated shortest paths. The method was incorporated into the adaptive wire adjustment algorithm, SPP-AWA, to build A BSCDN by minimizing the skew incrementally. The results show the capability of SPP-AWA to design a BSCDN for any given skew bound. Additionally, SPP-AWA enables a quick engineering change (EC) to the BSCDN. The method can be incorporated into other CDN routing algorithms, as well as it can be used to route any multi-terminal nets with obstacle constraints.

In this chapter, the SPP-AWA is used to build the CDN from the start. However, the SPP-AWA, similar to AWA, would outperform other algorithms in the course of minor modifications of the CDN. Thus, it is required to synthesize the CDN at first using other algorithms. This can be achieved by incorporating the SPP model into the DME algorithm as will be described in the next chapter. In addition, the next chapter will also address the use of the SPP model to synthesize the CDN with other constraints, such as a limited number of routing planes.

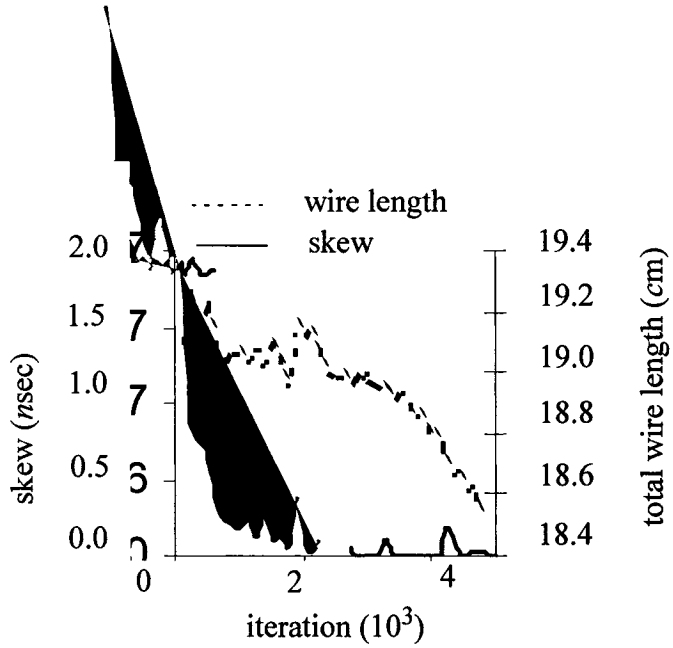


Figure 5.12 Skew convergence and total wire length for r1 with 40 obstacles

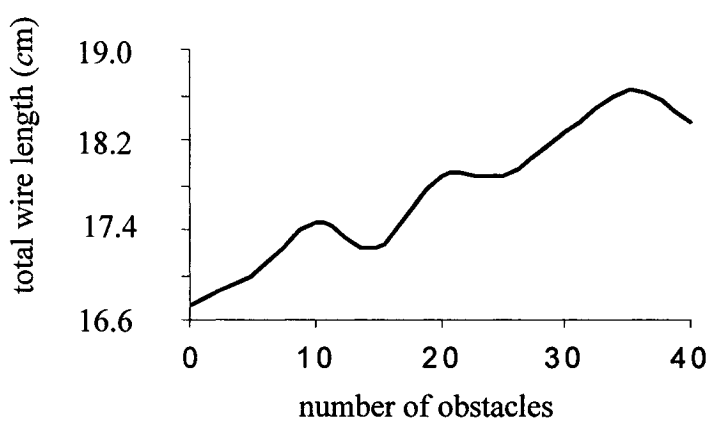


Figure. 5.13 Total wire length vs. number of obstacles for r1

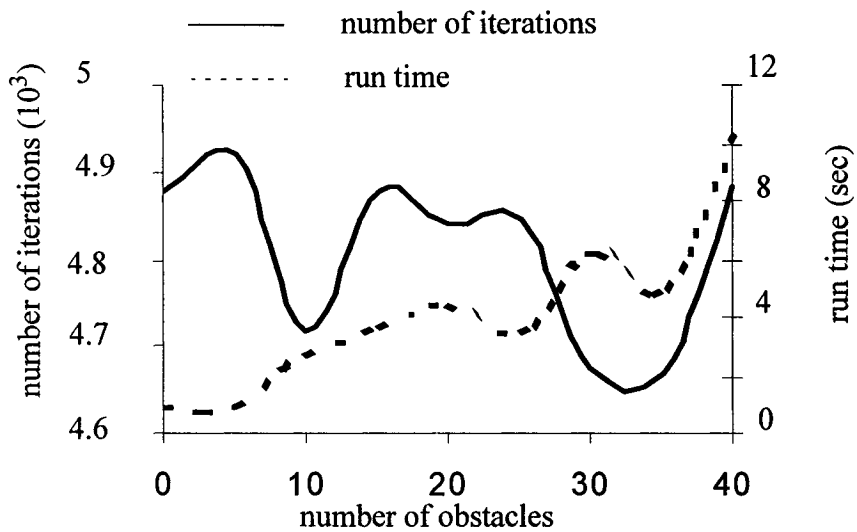


Figure 5.14 Number of iterations and run time vs. number of obstacles for r1

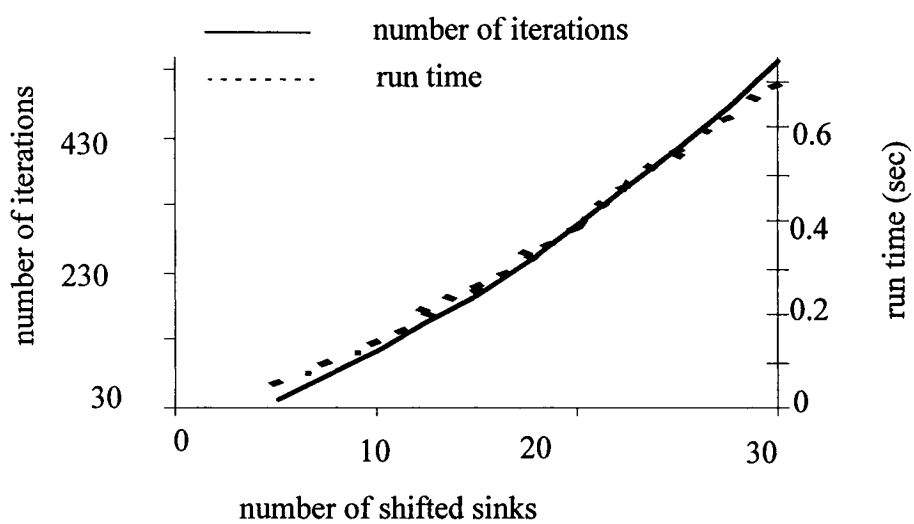


Figure 5.15 Iterations and run time vs. number of shifted sinks for r5

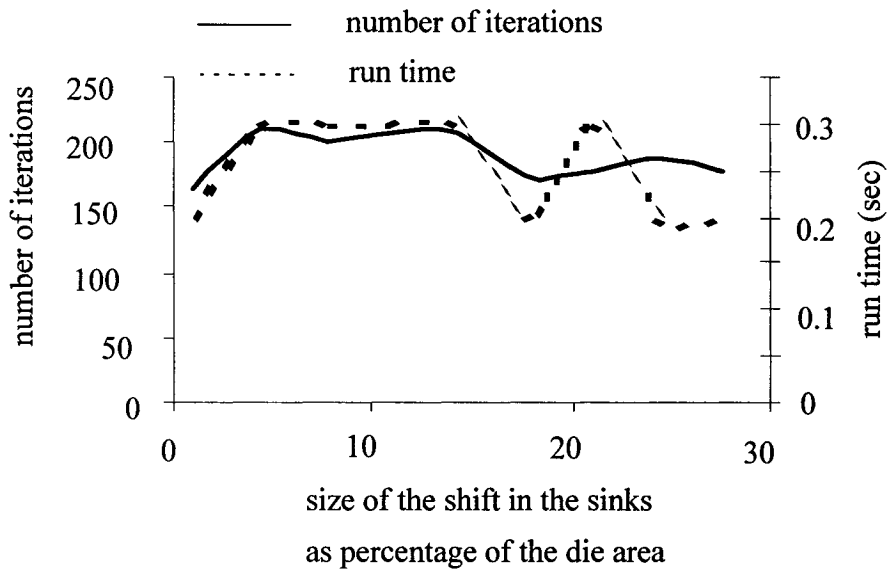


Figure 5.16 Iterations and run time vs. the size of the shift of five sinks for r5.

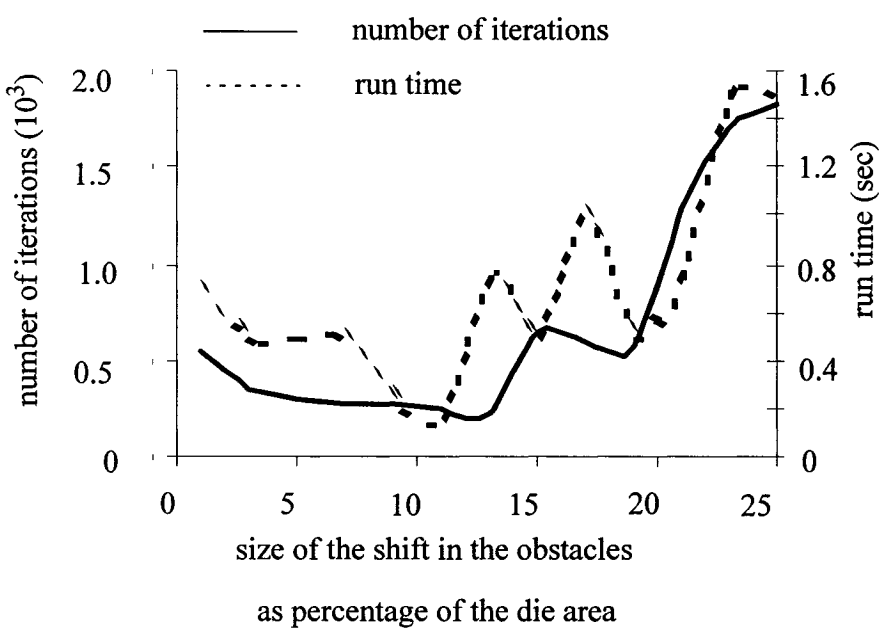


Figure 5.17 Iterations and run time vs. the size of the shift of three obstacles for r5.

Chapter 6

Incorporation of SPP Model for Special Cases

6.1 Introduction

In general, previous shortest path algorithms, such as the maze, line search or graphical algorithms, determine a single shortest path between source and destination points, *s* and *t* respectively. Such a solution dictates the layout of the wire that connects *s* and *t*, *Wire(s,t)*. However, the shortcomings of these algorithms is that not all possible solutions are determined. Indeed, the determination of all solutions means the determination of the routing area of *Wire(s,t)*. If there is no obstacle constraint, then the routing area of *Wire(s,t)* is the rectangle *R(s,t)*. On the other hand, if there are obstacle constraints, then *SPP(s,t)* is the routing area of *Wire(s,t)*. The problem of determining the routing area of two points, instead of the shortest path, has not been addressed in the literature. It is obvious that the determination of the routing area would help the router to produce a better layout solution in terms of total wire length or other performance criteria. In this chapter, the SPP model is applied to solve two CDN routing problems: producing a ZSCDN under obstacle constraints and a planar ZSCDN.

Chapter 2 described the synthesis of a ZSCDN using DME algorithm, which does not consider the obstacle constraints. The obstacle constraints were introduced in Chapter 5 by incorporating the SPP model into the AWA algorithm. However, AWA algorithm is geared towards implementing an Engineering Change (EC) into the CDN. In this chapter, the SPP model will be incorporated into the DME algorithm, which will be called SPP-DME, to design a ZSCDN under obstacle constraints. This is achieved by determining the SPP between two arcs instead of two points. Also, in this chapter, a new approach is proposed, based on the SPP concept, to route the CDN such that all wires of the CDN do not cross each other.

Consider the determination of the merging segment of node u, U, whose child nodes are s and t as shown in Figure 6.1. In Chapter 5, a procedure was proposed to determine U by determining the SPP of its child nodes, $SPP(s,t)$, by constructing a visibility graph G. The links of G are determined with the assumption that the child nodes s and t are represented by points. However, s and t can be Manhattan arcs. Thus, the links of G and $SPP(s,t)$ have to be determined differently.

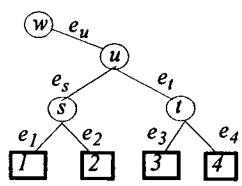


Figure 6.1. An example of a CDN tree

6.2 Preliminary

A point p, whose x and y coordinates are p_x and p_y respectively as shown in Figure 6.2, divides the Manhattan plane into four quarters as follows:

$Q^0(p)=\{q: q_x \geq p_x, q_y \geq p_y\}$

$Q^1(p)=\{q: q_x < p_x, q_y \geq p_y\}$

$Q^2(p)=\{q: q_x < p_x, q_y < p_y\}$

$Q^3(p)=\{q: q_x \geq p_x, q_y < p_y\}$ (6.1)

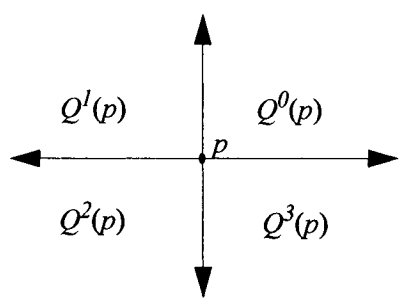


Figure 6.2. The four quarters of a point in the Manhattan plane.

If a point is the head or tail of an arc, say S^h or S^t, then specific quarters of S^h and S^t, are of particular interest as shown in Figure 6.3. Specifically, the indices of the quarters of S^h and S^t will be omitted as follows:

$$Q(S^h)=\begin{cases} Q^2(S^h) & \text{if } S_{slope} = +1 \\ Q^3(S^h) & \text{O.W.} \end{cases} \quad (6.2.a)$$

$$Q(S^t)=\begin{cases} Q^0(S^t) & \text{if } S_{slope} = +1 \\ Q^1(S^t) & \text{O.W.} \end{cases} \quad (6.2.b)$$

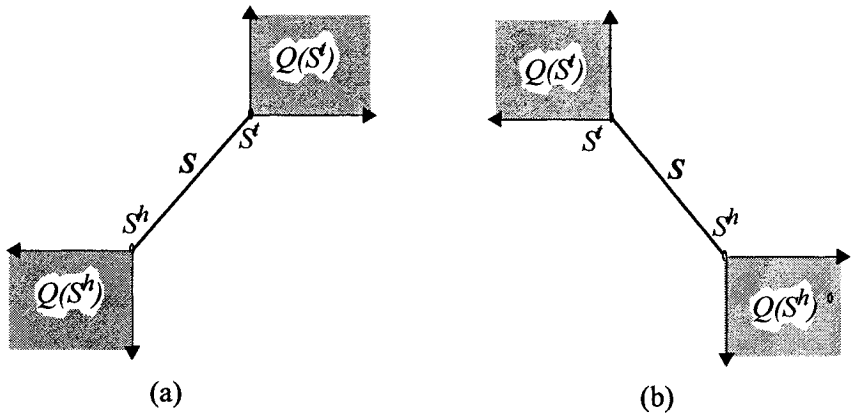


Figure 6.3. The quarters of the head and tail of an arc, S (a) S_{slope} =1 (b) S_{slope} =-1.

A single point, p, is located above or below an arc, S ($p\uparrow S$ or $p\downarrow S$), if the line that passes through p is in parallel with S has a displacement greater or less than S_{disp} respectively (the displacement of an arc is defined in Section 5.2). Figure 6.4(a and b) show two cases where $p\uparrow S$ or $p\downarrow S$.

Figure 6.4. The relationship between a Manhattan arc and a point (a) $p\uparrow S$ (b) $p\downarrow S$.

The wire that connects a point, p, to an arc, S, with minimum length, *Wire(p,S)*, may not be able to connect any point from S to p. Indeed, only a portion of S can be connected by *Wire(p,S)*. This portion will be called the *projection*, which is defined as follows:

Definition 6.1(a) The projection of a point, p, on an arc, S, denoted as Projection(p,S); is a set of points of S such that:

$$Projection(p,S)=\{q\in S: D(q, S) = D(p,S)\} \tag{6.3}$$

For example, *Projection*(p,S) is shown as a solid thick segment in Figure 6.5. Note that the resulting projection can become a single point (see Appendix B for details).

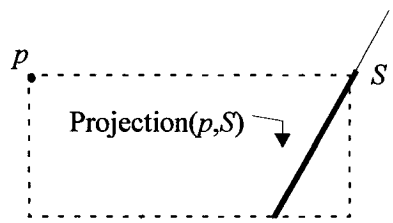


Figure 6.5. The projection of a point on an arc.

An arc, S, is said to be above another arc, T, $S\uparrow T$, if and only if:

$$\exists\, p\in S: p\uparrow T \tag{6.4.a}$$

Similarly, we say that S is below T ($S\downarrow T$) if and only if:

$$\exists\, p\in S: p\downarrow T \tag{6.4.b}$$

Additionally, two arcs would be either in parallel or perpendicular ($S//T$ or $S\perp T$), if $S_{slope}=T_{slope}$ or $S_{slope}\neq T_{slope}$ respectively, as shown in Figure 6.6(a and b). Note that, when $S//T$, the case $S\uparrow T$ necessitates that $T\downarrow S$ and vice versa. But, when $S\perp T$, the case $S\uparrow T$ does not necessitate that $T\downarrow S$.

For two arcs, S and T, *Wire(S,T)* may connect a portion of S with a portion of T. The portion of T that can be connected by *Wire(S,T)* will be called the *projection* of S on T; and it is defined as follows:

Definition 6.1(b) The projection of an arc, S, on another arc, T, denoted as Projection(S,T); is a set of points of S such that:

$$Projection(S,T)=\{ p \in T: D(p, S) = D(S,T)\} \qquad (6.5)$$

Figure 6.6 shows examples of *Projection*(S,T) and *Projection*(T,S) (see Appending B for details).

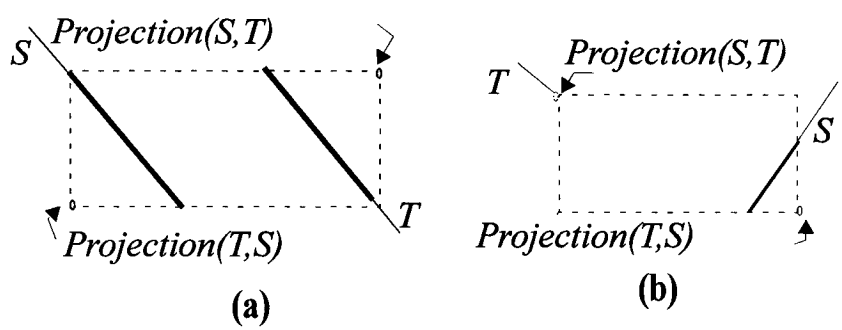


Figure 6.6. The projections of two arcs on each other when they are (a) parallel and (b) perpendicular.

6.3 Determination of a link in the Visibility Graph

6.3.1 Link Determination Between a Point and an Arc

Consider connecting an arc, S, to a point, p, such that S_{slope}=-1 and $S \subset Q^0(p)$ as shown in Figure 6.7. Thus, $p \downarrow S$ and *Projection(p,S)*=S; and consequently, *Wire(p,S)* would pass through the rectangle R(p,$\Psi^+(S)$). Let the edges of R be labeled with index 0, 1, 2 and 3

as shown in Figure 6.7. An obstacle O intersecting with R may cross over one or more edges of R, where o_0, o_1, o_2 and o_3 are the bottom-left, top-left, top-right and bottom-right corners of O respectively. The intersection between R and O can be expressed as a binary number, N, that consists of four bits (N=$n_3n_2n_1n_0$); a bit for each edge of R. The value of the bit n_i represents the intersection between O and the edge i of R such that n_i=*true* if O intersects with that edge. The impact of O on the link between p and U can be determined by the value of N, which will be called the intersection number. Note that some values of N implies impossible intersection between R and O. For example, the case N=1xx1 implies that O is crossing over p, which is impossible. Also, it is impossible to have an intersection that results in N=x111 or N=111x since such intersections imply that O crosses over S. Accordingly, only ten values of N will be considered in order to determine the link between p and S. The rules of determining the link and the projection of p on S, where S_{slope}=-1 and $S \subset Q^0(p)$, can be summarized in a Look Up Table (LUT), as shown Table 6.1.

In general, the impact of an obstacle on the link between a point and an arc depends on the relationship between the point and the arc. Specifically, the arc may belong to one of the four quarters of the point as shown in Figure 6.8. For the cases $S \subset Q^1(p)$ and $S \subset Q^3(p)$, as shown in Figure 6.8(b and d), it is assumed that S_{slope}=+1. And, if S_{slope}=-1, then *Projection(p,S)* would be a single point; where the link between S and p can be determined as described before. Similarly, S_{slope} is assumed to be -1 for the cases $S \subset Q^0(p)$ and $S \subset Q^2(p)$ shown in Figure 6.8(c and d). In order to determine the link

between S and p, a rectangle R is constructed by p and either $\Psi^{+}(S)$ or $\Psi^{-}(S)$ for each case, where the edges of R are labeled as shown in Figure 6.8.

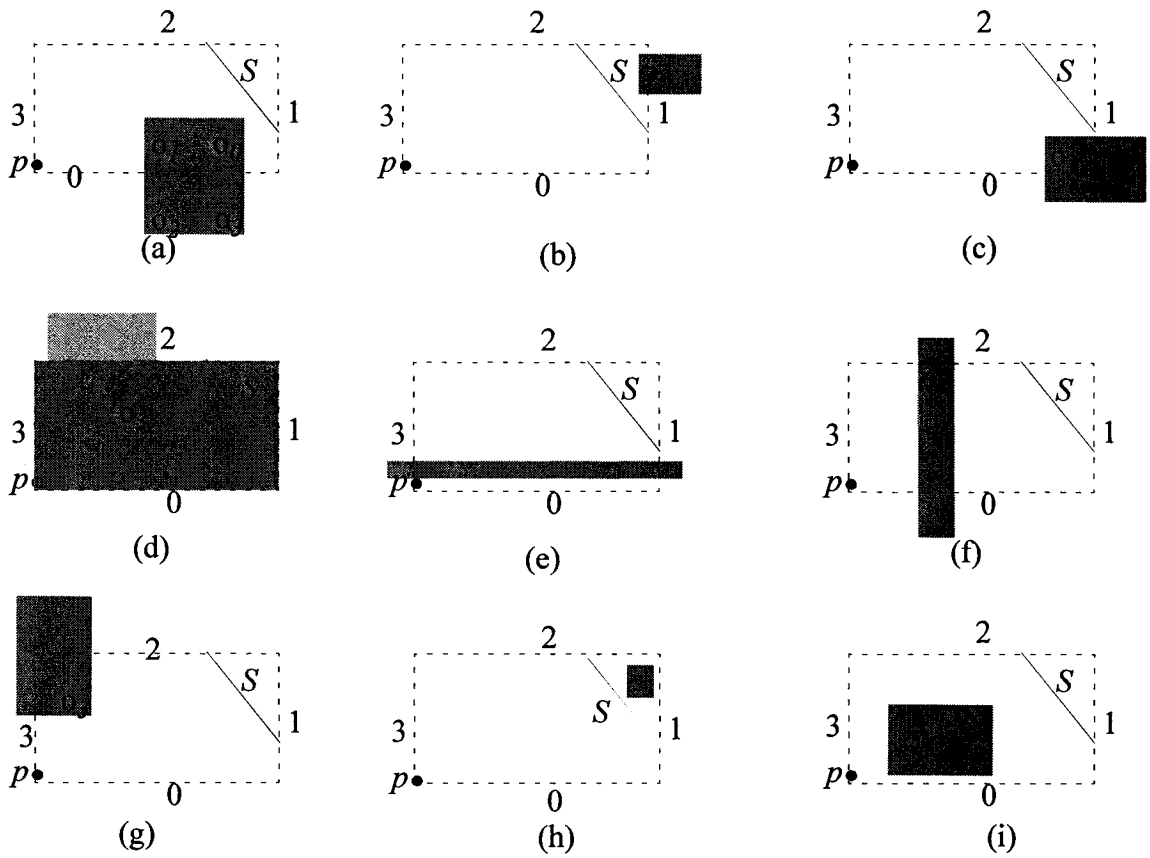


Figure 6.7. The impact of an obstacle on connecting an arc S to a point p.

For the case shown in Figure 6.8(a), S_{slope}=-1 and $S \subset Q^{0}(p)$, the link can be determined using the LUT shown in Table 6.1. Similarly, three LUTs can be generated for the cases $S \subset Q^{1}(p)$, $S \subset Q^{2}(p)$ and $S \subset Q^{3}(p)$, where the edges of R and the corners of O can be labeled as shown in Figure 6.8. A procedure can be used to determine the link between a point and an arc as described in Figure 6.9.

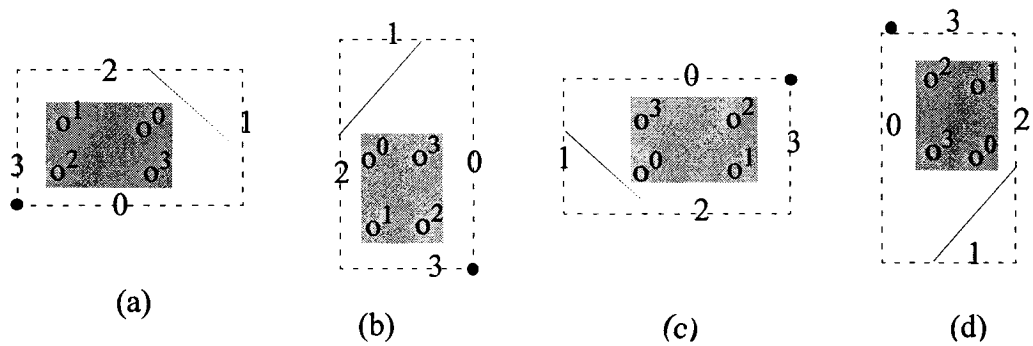


Figure 6.8. Different possible cases the relationship between an arc and a point.

Initially, the above procedure determines the rectangle R for S and p and selects a LUT based on the relationship between S and p as shown in Figure 6.8. Then, for each obstacle O intersecting with R, the *Projection(p,S)* is updated according to the selected LUT. The link between S and p is established if *Projection(p,S)* is non empty such that the rectangle formed by *Projection(p,S)* and p does not intersect with any obstacle. Note that the link between an arc and a corner from obstacle in the visibility graph would hold extra information in reference to the projection of that corner on the arc.

Recall that having a link between two nodes, s and t, in the visibility graph implies that *Wire(s,t)* does not cross over any obstacle. At first, the projections of the arcs of the two nodes on each other, *Projection(S,T)* and *Projection(T,S)*, are determine. With the assumption that $S \uparrow T$, *Wire(s,t)* would not cross over any obstacle if the rectangle $R(\Psi^{-}(T),\Psi^{+}(S))$ does not intersect with any obstacle. Alternately, if O intersects with R, then *Projection(S,T)* and *Projection(T,S)* have to be re-determined.

Input: *an arc S, a point p and a set of obstacles* ***Output:*** *the link between S and p and Projection(p,S)*

Determine *Projection(p,S) as described in Appendix B.*
Construct *a rectangle R from p and Projection(p,S) as shown in Figure 6.8.*
Select *the LUT*
    For *each obstacle O intersecting with R*
        if *Projection(p,S)≠ϕ* ***Then Update*** *Projection(p,S) according to LUT*
if *Projection(p,S)≠ϕ* ***Then*** *establish a link between S and p*

Figure 6.9. Procedure of *Link Determination between an arc and a point*

Table 6.1. The look Up Table for the case S_{slope}=-1 and $S \subset Q^0(p)$

N	impact of an obstacle O on the link between a point *p* and an arc *S*	
0	No impact on the link	if $o^2 \uparrow s$
	There is a link, but $Projection(p,S)^h = r{:}r \in S$, $r_x = o^1{}_x$	if $S^l \in Q^1(o^1)$
	There is a link, but $Projection(p,S)^t = r{:}r \in S$, $r_y = o^3{}_y$	if $S^h \in Q^3(o^3)$
	$Projection(p,S)=\phi$	O.W.
1, 2, 3	No impact on the link	if $o^2 \uparrow S$
	$Projection(p,S)=\phi$	if $S^l \notin Q^1(o^1)$
	There is a link, but $Projection(p,S)^h = r{:}r \in S$, $r_x = o^1{}_x$	O.W.
4, 8, 12	No impact on the link	if $o^2 \uparrow S$
	$Projection(p,S)=\phi$	if $S^h \notin Q^3(o^3)$
	There is a link, but $Projection(p,S)^t = r{:}r \in S$, $r_y = o^3{}_y$	O.W.
5,10	$Projection(p,S)=\phi$	
O.W.	No Impact on the Link	

6.3.2 Link Determination Between Two Arcs

For the case that the two arcs, S and T, are perpendicular, $S_{slope} \neq T_{slope}$, then *Projection(S,T)* and/or *Projection(T,S)* will be a point (see Appendix B for the details). As such, the link can be determined as described in Section 6.3.1, where one of the four LUTs has to be selected. On the other hand, if $S_{slope}=T_{slope}$ as shown in Figure 6.10, then other LUTs have to be used as will be described next.

Consider two arcs, S and T, such that $S \uparrow T$ and $S_{slope}=T_{slope}=-1$, where the edges of R($\Psi^{-}(T)$,$\Psi^{+}(S)$) are labeled as shown in Figure 6.10(a). If an obstacle, O, whose corners are labeled as shown in Figure 6.10(a), intersects with R, then the intersection can be expressed by a binary number, N, of four bits, where each bit corresponds to the intersection between O and an edge from R as described in Section 6.3.1. Some values of N imply impossible intersection between O and R. For example, N=x111 or 111x implies that O crosses over T; and N= 1x11 or 11x1 implies that O crosses over S. Thus, only eleven values of N will be considered. Two LUTs, similar to LUT shown in Table 6.1, can be deduced for the two cases shown in Figure 6.10.

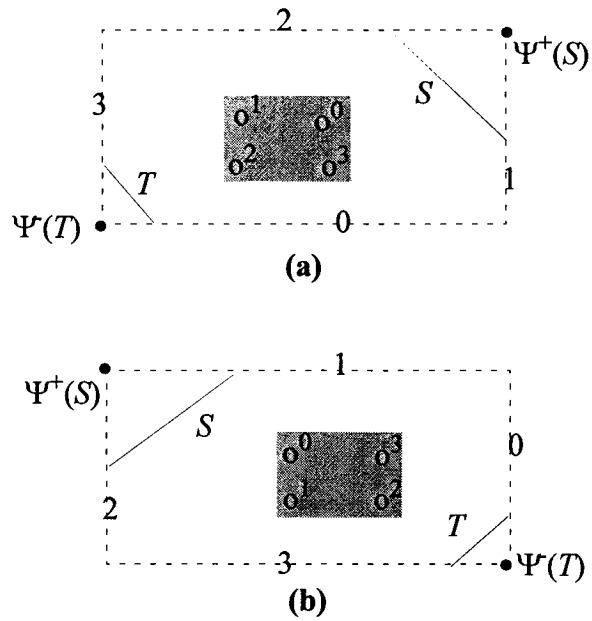


Figure 6.10. The impact of an obstacle on connecting two arcs.

Based on the previous discussion, a procedure can be drawn to determine the link between two arcs as described in Figure 6.11.

Input: *two arcs, S and T, and a set of obstacles* ***Output:*** *the link between S and T, Projection(S,T) and Projection(T,S).*

Determine *Projection(S,T) and Projection(T,S) as described in Appendix B*
Construct *a rectangle R from Projection(S,T) and Projection(T,S) as shown in Figure 6.10.*
Select *the LUT*
    For *each obstacle O intersecting with R*
        if *Projection(S,T)≠ϕ or Projection(T,S)≠ϕ*
        Then Update *Projection(S,T) and Projection(T,S) according to the LUT*
if *Projection(S,T)≠ϕ and Projection(T,S)≠ϕ* ***Then*** *establish a link between S and T*

Figure 6.11. Procedure of *Link Determination between two arcs, S and T.*

6.4 Applying the SPP Model to the DME Algorithm

The procedure of *Merging Segment Determination*, which is shown in Figure 5.10, was applied to the AWA algorithm in order to tune the clock tree under obstacle constraints [79]. However, AWA algorithm is geared towards implementing an Engineering Change (EC) into the CDN. In this section, the proposed method is applied to the DME algorithm, which will be called SPP-DME, in order to design a ZSCDN under obstacle constraints.

The original version of DME produces a ZSCDN in two phases. In the first phase, the merging segment of each Steiner node in the CDN is determined in a bottom-up manner. In the second phase, the exact location of each Steiner node is determined in a top-down manner. In order to incorporate the proposed method into DME algorithm, the merging segment has to be determined differently due to the obstacle constraints. For example, consider the node u and its children s and t as shown in Figure 6.1. The merging segments of u, U, can be determined from $SPP(s,t)$ by applying the procedure *Merging Segment Determination* as described in Chapter 5. This is achieved by constructing a visibility graph, G, with the assumption that all nodes, including s and t, correspond to points in the routing plane. Specifically, the link between a pair of nodes in G is established whenever the rectangle of their corresponding pair of points does not intersect with any obstacle. However, s and t correspond to merging segments which are not necessarily single points in the routing plane. Thus, the step of constructing G in the *Merging Segment Determination* procedure needs more investigation as will be described next.

Let the arcs of the nodes s and t be S and T respectively. The obstacles may break down S and/or T into several arcs. Thus, s or t may correspond to arcs in the Manhattan plane. Each arc of s, or t, has to be represented as a node in G. Henceforth, G becomes a multi-source and multi-destination graph. In addition, the links between the corners of the obstacles and the arcs of s or t have to be established as described in Section 6.2.1; while the links between the arcs of s and t have to be established as described in Section 6.2.2.

Constructing the visibility graph is the first step in the procedure of *Merging Segment Determination*. The rest of the steps of the procedure, as described in Figure 5.10, can be applied without modification in order to determine the merging segment of a node from the merging segments of its child nodes. The incorporation of the proposed procedure into the DME algorithm is achieved by replacing the merging segment procedure of the first phase of DME with the proposed procedure. The merging segments that have been calculated in the first phase of DME algorithm are used to locate the exact location of each internal node in the tree in the top down phase of DME algorithm. Also, the information of the SPP of each pair in the tree has to be preserved too in order to use them in the second phase as described in Figure 6.12

Input: *Tree of merging segments, the edge length of each node (except the root) and the SPP between every pair of nodes in the tree.* ***Output:*** *location of every Steiner point in the tree.*

For *each node u in T in top down manner*
    If *u is the root*
        Locate *u at any location of any part of U*
Else
    Let p_0 *and* p_n *be the children of u, where* $SPP(p_0,p_n)=p_0,\ldots,p_n$ *such that* $U\in\mathcal{T}_k$
    Locate p_0***:***
        If $k=0$ ***Then*** *let* $q=u$ ***Else*** *let* $q=p_1$
        Construct *a tilted rectangle, R, such that the centre of R is q and the radius of R is* e_0
        Locate p_0 *at the intersection of R and the arc of* p_0
    Locate p_n***:***
        If $k<n-1$ ***Then*** *let* $q=p_{n-1}$ ***Else*** *let* $q=u$
        Construct *a tilted rectangle, R, such that the centre of R is q and the radius of R is* e_n
        Locate p_n *at the intersection of R and the arc of* p_n

Figure 6.12 The Top-Down Procedure of the SPP-DME.

6.5 Planar Clock Distribution Network

The CDN that can be routed without any via is called a planar CDN since it can be routed in one plane. It is obvious that vias are undesirable since they complicate the routing process. In addition, the vias affect the reliability of the CDN in terms of introducing extra resistance as well as they render the CDN more vulnerable to the process variation. Authors of [78] proposed an algorithm, called Max-Min algorithm, that results in a planar CDN in order to avoid vias as well as to make the CDN more tolerant to the process variations. The Max-Min algorithm does not need an initial topology; and it starts with the source of the clock signal as the initial CDN. At each

iteration, Max-Min algorithm selects an unconnected register and connects it to the CDN. In order to select an unconnected register, the algorithm determines where every unconnected register can be connected to the CDN so that the delay from the source to that register is equal to the delay from the source to the connected registers. The connection point to the CDN is called balance point, and it is obvious that there are many balance points for each unconnected register. Hence, for each unconnected register, the algorithm considers only the balance point that has the minimum distance from that register. Then, the algorithm selects the unconnected register whose minimum distance is the maximum one among other unconnected registers. The selected register is connected to the CDN at the balance point that has minimum distance from that register. This algorithm results in a CDN that has x-shape, which is undesirable. In fact, if the registers are symmetrically distributed, the CDN will be X-tree and the planarity is achieved by increasing the total wire length. The Max-Min algorithm, similar to MMM and GMA, uses a linear path length delay model in order to produce a ZSCDN [78]. Thus, the solution in not a true ZSCDN since the path length delay model is inaccurate.

Khang and Tsao proposed a version of DME that generates a planar CDN called Planar-DME [80]. This approach determines the solution in one phase by determining the merging segments during the top-down phase. Let a node, w, be the root of a sub-tree that contains a set of registers **W**. The diameter of **W**, Diameter(**W**), is defined as the maximum distance between any two registers in **W**. The merging segment of w is

determined from the intersection of TRRs such that each TRR is centered at a register in **W** and the radius of each TRR is Diameter(**W**)/2. In order to construct a planar CDN, at each iteration, the set of registers **W** that are connected to *w*, is divided into two sets such that the future routing will not interfere with the existing routing.

Recently, another planar CDN, called cutting-line embedding routing algorithm [81]. The algorithm starts with partitioning the registers set R recursively in similar manner as MMM. The registers are connected to each other by edges that are allowed on the boundaries of the partitions only. It is obvious that such restriction results in an increase in the total wire length.

Previous approaches achieve the planarity through high routing cost. Indeed, the planarity is achieved by determining the topology such that the wires of the CDN can be routed without intersection. Thus, the core difference between these approaches is for the topology generation. For example, the Planar DME produce an H-tree like CDN, whereas Max-Min produces a X-tree like CDN. The topology impact on the total wire length was detailed in Chapter 4. Thus, one may deduce that the increase in the total wire length is due to the topology that is used by each of these algorithms.

In addition to the increase in the total wire length, previous approaches determine the location of each Steiner node of the CDN in the Manhattan plane. Then, these approaches use different rules to determine the layout of each wire such that the wires do

not intersect with each other. However, each wire can be routed differently in a specific routing area, where the routing area can be represented by the SPP model. In fact, the SPP determination of each wire, instead of a specific layout of the wire, would provide the router with a flexibility that can be used to optimize different design parameters, such as the total wire length. Also, note that, nowadays, there is no need to limit the CDN in one routing plane. Indeed, there is a need to develop new approaches that can determine the routing of the CDN in more than one plane

In the next section, the SPP model will be applied to develop the first method that can achieve the planarity for any given topology. The planarity is achieved by treating the previous routed wires as obstacles. That is, the SPP of each wire are treated as obstacles. Such approach requires the reformation of the SPPs of previously routed wires in order to avoid an increase in the total wire length. In addition, the proposed approach distributes the wires of the CDN between two layers with the objective of minimizing the total wire length. At first, a new operation will be presented to describe the interaction between two SPPs.

6.5.1 The Inner Product Operation

The procedure *Merging Segment Determination* shown in Figure 5.10 describes the determination of a merging segment, U, from the SPP of its children, $SPP(s,t)$. That is, U is located inside $SPP(s,t)$. Let $\overline{U}$ be the SPP where U is located, i.e. $\overline{U}=SPP(s,t)$. For any pair of SPPs, say $\overline{S}$ and $\overline{T}$, the *inner product* is defined as follows:

Definition 6.2: The inner product of the SPPs $\overline{S}$ and $\overline{T}$ $(\overline{S},\overline{T})$, is the overlapped area between $\overline{S}$ and $\overline{T}$.

By using the inner product operation, two SPPs are said to be *orthogonal* if their inner product is zero. Also, by using the inner product operation, each SPP, say $\overline{S}$, is associated with a real number, denoted by $||\overline{S}||$ and called the *norm* of $\overline{S}$, which is given as:

$$||\overline{S}|| = (\overline{S},\overline{S})^{1/2} \qquad (2)$$

6.5.2 The SPP Formation

Consider the case shown in Figure 6.13(a) where a merging segment U has to be determined from its child segments S and T. At first, the SPP that connects S and T, $\overline{U}$, has to be determined. Let $\overline{S}$ and $\overline{T}$ be the SPPs of connecting S and T to their children respectively. Connecting S to T by a wire means that a single point from S is connected to a single point from T. Consequently, $\overline{S}$ and $\overline{T}$ have to be modified, where concave points are inserted properly into the border of $\overline{S}$ and $\overline{T}$ as shown in Figure 6.13(b).

Selecting single points from S and T affects dramatically the norms of $\overline{S}$, $\overline{T}$ and $\overline{U}$. Furthermore, the norm of $\overline{U}$ impacts the length of U, where a large value of $||\overline{U}||$ implies (most likely) an increase in the length of U. Consequently, U can be connected with less wire length to its parent later on. Additionally, inserting points into $\overline{S}$ and $\overline{T}$ lead to minimizing $||\overline{S}||$ and $||\overline{T}||$. Consequently, there will be less space of maneuvering for the designer to route *Wire(S,T)*. Note that the points of S and T are inserted into $\overline{S}$ and $\overline{T}$ such

that *Wire(S,T)* does not validate the design rule of the minimum distance between wires as shown in Figure 6.13(b).

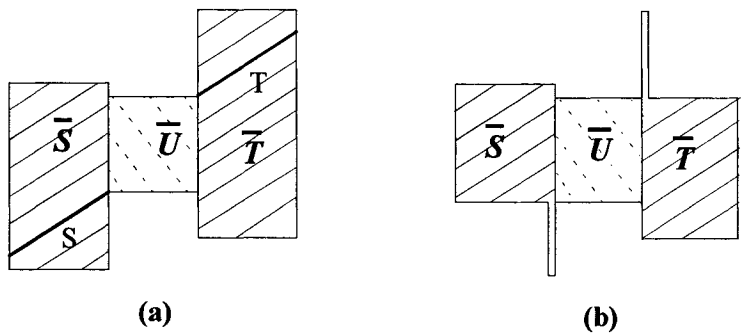


Figure 6.13 Modifying the SPP $\overline{S}$ and $\overline{T}$

In order to simplify the determination of $\overline{U}$, let U be connected only to the end points of its child segments S and T. Hence, $\overline{U}$ can be one of four SPPs that result from the different combinations of the end points of S and T: $SPP(S^h,T^h)$, $SPP(S^h,T^t)$, $SPP(S^t,T^h)$ and $SPP(S^t,T^t)$. In order to minimize the total wire length, only the SPPs that result in minimum length for *Wire(S,T)* will be considered. For the case shown in Figure 6.13, the SPP that connect the S^t to T^h has the minimum cost. If there are more than one SPP that have the minimum cost, then the SPP that has the maximum norm is selected in order to minimize the total wire length later on.

Even though the points that are selected from the child segments S and T lie on the border of $\overline{S}$ and $\overline{T}$, the SPP of their parent, $\overline{U}$, may overlap with $\overline{S}$ and $\overline{T}$. For example, $(\overline{U},\overline{T})>0$ for the case shown in Figure 6.14(a). However, it is important to keep $\overline{S}$, $\overline{T}$ and $\overline{U}$ orthogonal so that $||\overline{U}||$ is maximized. If the inner product of U with its child SPP, say $\overline{T}$, is not zero, then a point is inserted into $\overline{T}$ so that $(\overline{U},\overline{T})= 0$. Such a point would be the

vertex of $\overline{U}$ that is located inside the polygon of $\overline{T}$ as shown in Figure 6.14(a). As such, another point is inserted into $\overline{T}$ as shown in Figure 6.14(b). Although this is true, inserting a point into $\overline{T}$ may eliminate the source or destination point of $\overline{T}$ as shown in Figure 6.14(c). Such elimination must not be allowed since the wire that is supposed to be routed inside $\overline{T}$ cannot connect the source and destination points of $\overline{T}$. In such a case, the graph of determining the shortest path would have two sources (head and tail of S) and two destinations (head and tail of S); and the child SPPs, $\overline{S}$ and $\overline{T}$, are treated as obstacles as shown in Figure 6.14(d). In conclusion, if the inner product of $\overline{U}$ and one of its child SPPs, say $\overline{T}$, is not zero, then the vertex of $\overline{U}$ that is located inside the polygon of $\overline{T}$ would be added to $\overline{T}$ given that the source or destination point of $\overline{T}$ are not eliminated from $\overline{T}$.

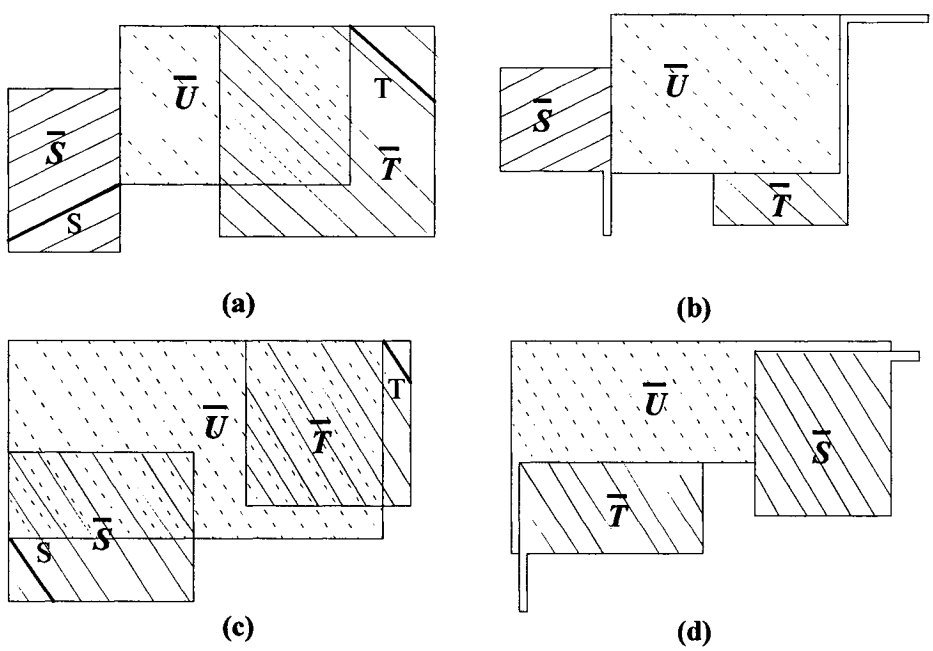


Figure 6.14 Other scenarios of modifying the SPP.

6.5.3 Applying SPP to Planar CDN Determination

The SPP can be used to route the CDN in one or more planes. Each wire of the CDN is routed by dedicating a non-uniform routing cell for it, i.e. SPP. Whenever the SPP of a wire is to be determined, the SPPs of previously routed wires are treated as obstacles. The wires of the CDN can be determined in a bottom-up manner as follows.

For each Steiner node, u, in the CDN the child segments, S and T, are reduced to points, s and t, as described in Section 6.5.2. The SPP of u, $\overline{\mathbf{U}}$ can be determined by finding the shortest paths in the graph that corresponds to $\overline{\mathbf{S}}$, $\overline{\mathbf{T}}$ and any other SPP intersecting with R(s,t). Then the merging segment of u, B_u, can be determined. The procedure of determining the merging segment is described in Figure 6.15.

Note that if a part of the resulting merging segment is located outside $R(s,t)$, then that part has to be checked for intersection with all obstacles that do not intersect with $R(s,t)$. The proposed procedure can be incorporated into existing algorithms that route the clock tree. In this section, the proposed procedures incorporated into the DME algorithm, and will be called Planar-SPP-DME. This is achieved by replacing the merging segment procedure of the first phase of DME with the proposed procedure. Since the proposed procedure determines the exact location of the child nodes of each Steiner point, there would not be a need for the second phase of DME. Furthermore, the levels of the clock tree are divided equally into higher and lower levels. The Planar-SPP-DME selects the Steiner node in a bottom-up manner from the lower levels, then from the higher levels. The Steiner points of the lower levels are routed in the first plane. Then,

the Steiner nodes of the higher levels are routed in the second plane without considering the SPPs of Steiner nodes of the lower levels.

6.6 Results

Different procedures are implemented using C++ language on a SUN Ultra 10 machine in Unix environment. The experiments were performed using the same five benchmarks, r1-r5, presented in Section 3.4. The initial topology of the CDN is generated using Mean and Median Method [3]. Two sets of experiments are set in order to study the synthesis of a CDN under obstacle constraints by applying SPP-DME, and the synthesis of a planar CDN by applying Planar-SPP-DME.

inputs: *S, T,* $\overline{S}$, $\overline{T}$ *and a set SPPs*
output: *U and* $\overline{U}$
Select *points s and t from S and T respectively*
Modify $\overline{S}$ *and* $\overline{T}$ *as described in Section 6.4.2*
Find *the SPPs that intersect with R(s,t)*
Construct *the graph G*
Find *the non redundant shortest paths in G*
Merge *interrelated paths*
Select $\overline{U}$ *from the shortest paths such that:*
$m(s,t)$ *is minimum ,* $\|\|\overline{U}\|\|$ *is maximum*
Determine *U in* $\overline{U}$ *according to Eq. 5.4*

Figure 6.15. The procedure of merging segment determination without intersection with other wires.

6.6.1 Clock Distribution Network Under Obstacle Constraints

The SPP-DME algorithms generates a ZSCDN under obstacle constraints as described in Section 6.4. The algorithm is evaluated in an attempt to show the capability of the proposed approach to handle the obstacles and to minimize total wire length simultaneously. Different number of obstacles of various sizes were added randomly to each benchmark so that there is no intersection between any two obstacles as well as no sink is located within an obstacle. The size of different obstacles were set between 5% and 20% of the die area. A modified version of Mean and Median Method (MMM) [3] was used in order to generate the initial topology.

The benchmarks were used as inputs to the SPP-DME algorithm in order to study the impact of the obstacles on the run time and the total wire length. Figure 6.16 shows an example of the resulting CDN for the benchmark r1 when the routing plane is compounded with 20 obstacles. In order to provide a comparison between the proposed approach and the approach presented in [70], all benchmarks are compounded with 40 obstacles. Table 6.2 provides a comparison between DME and SPP-DME for different benchmarks. The metrics considered in this comparison are total wire length and run time. From Table 6.2, one may notice that the obstacles account for an increase of only around 5% in the total wire length for 40 obstacles when the SPP-DME is used to produce a ZSCDN. Alternately, the approach presented in [70] results in an increase of around 12% in the total wire length for the same number of obstacles. For both approaches, the total wire length increases due to the obstacle constraints. Indeed, the increase in the number of the obstacles renders the shortest paths longer, and as a result,

the total wire length increases as shown in Figure 6.17. Nevertheless, the proposed approach has the advantage of reducing the impact of the obstacles on the total wire length. This is due to the fact that all possible locations are considered for each merging segment. Consequently, the SPP-DME results in a less increase in the total wire length as compared to the approach that was presented in [70]. More importantly, in [70], the number of partitions of the merging region were limited to five due to the complexity of their approach. On the other hand, our study of the properties of the SPP model and the visibility graph helps to reduce the number of edges in the graph and to eliminate the redundant paths as described in Section 5.4. Consequently, the proposed approach reduces the computation dramatically as compared to the run time given in [70]. For example, for a set of 555 clock pins and 40 obstacles, the procedure proposed in [70] could take up to 7 hours to reach the final solution. But, the proposed approach takes 1.6 sec to find the solution for 598 clock pins and 40 obstacles (the experiments of this work were carried out on SUN Ultra 10 and the experiments of [70] were carried out on SUN Sparc 5). The significant reduction in the run time is due to the reduction in the number of links in the visibility graph based on the SPP properties. Figure 6.18 shows the relationship between the run time and the number of obstacles for the benchmark r1. One may notice that the total wire length somehow increases linearly with the increase in the number of obstacles. The increase in the number of obstacles render the visibility graph bigger, and as a result, the run time increases.

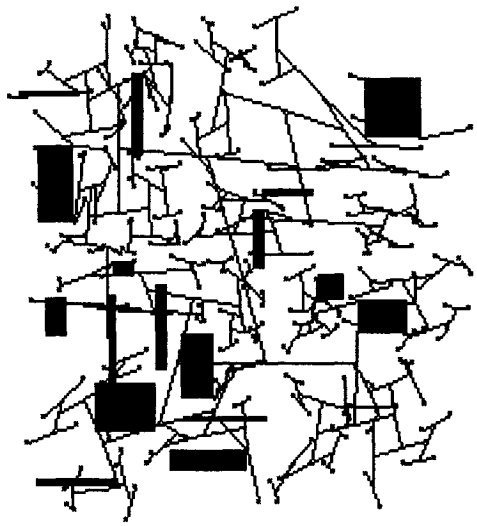


Figure 6.16 The resulting ZSCDN for the benchmark r1 under 20 obstacle constraints.

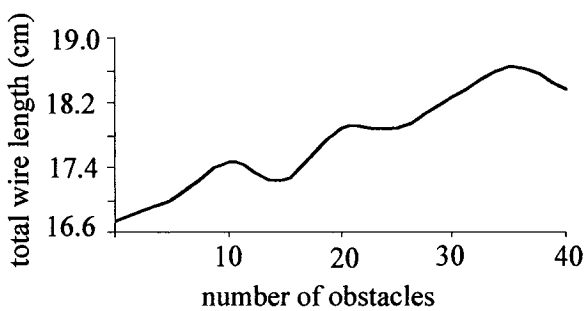


Figure 6.17 Total wire length vs. number of obstacles for r1using SPP-DME

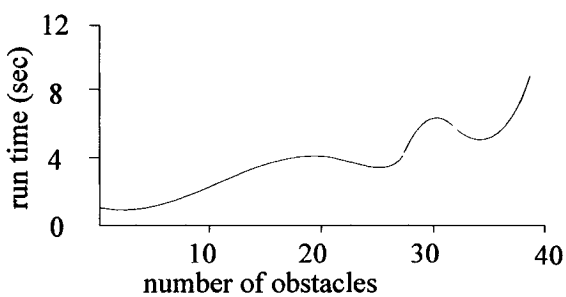


Figure 6.18 Run time vs. number of obstacles for the benchmark r1 using SPP-DME

Table 6.2. Different metrics for different benchmarks using obstacle free DME and SPP-DME with 40 obstacles.

benchmark	r1		r2		r3		r4		r5	
algorithm	DME	SPP-DME	DME	SPP-DME	DME	SPP-DME	DME	SPP-DME	DME	SPP-DME
Total wire length (cm)	16.9	18.2	36.1	38.3	46.9	49.7	90	93.5	137.5	140.6
run time (sec)	0.1	1	0.1	1.6	0.2	2.2	0.6	4.8	1.7	6.3

6.6.2 Planar Clock Distribution Network Results

The initial topology of the CDN is generated using the MMM approach [3]. Figure 6.19 shows the resulting planar ZSCDN for the first 32 clock pins of r1. Table 6.3 provides comparisons of total wire length and run time for different benchmarks. For comparison, the DME algorithm is tested on the same benchmarks. By using Planar-SPP-DME, the planarity results in an increase in the total wire length of only around 6% as shown in Figure 6.20.

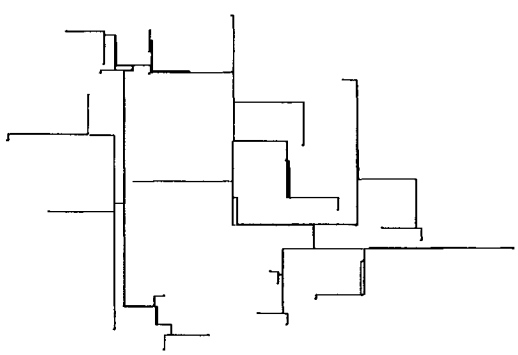


Figure 6.19. The planar ZSCDN for the first 32 clock pins of the benchmark r1.

Table 6.3. Different metrics for different benchmarks using DME and Planar-SPP-DME.

benchmark	r1		r2		r3	
algorithm	DME	Planar-SPP-DME	DME	Planar-SPP-DME	DME	Planar-SPP-DME
Total wire length (cm)	16.9	17.7	36.1	38	46.9	50.2
run time (sec)	0.1	3.5	0.1	8.9	0.2	15.5

In order to study the impact of the planarity on the total wire length, different number of clock pins from r1 were routed in one plane using SPP-DME as shown in Figure 6.21. The results show that the total wire length increases exponentially after a certain number of clock pins. Such a number, which is called the *planarity threshold*, is around 150; and it depends on the geometric distribution of the clock pins and the topology of the CDN. The planarity threshold can be used to estimate the wire load of the CDN, or any multi-terminal net.

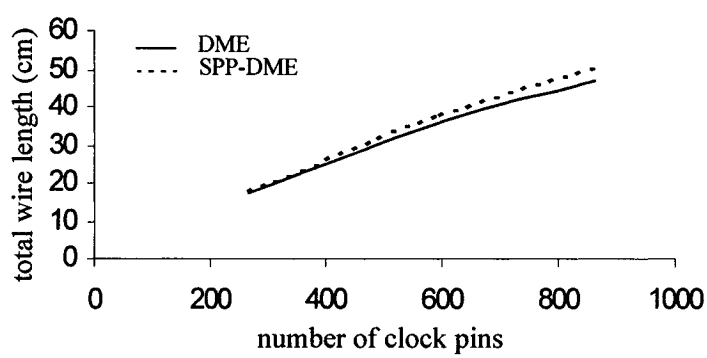


Figure 6.20. Total wire length vs. number of clock pins for different benchmark.

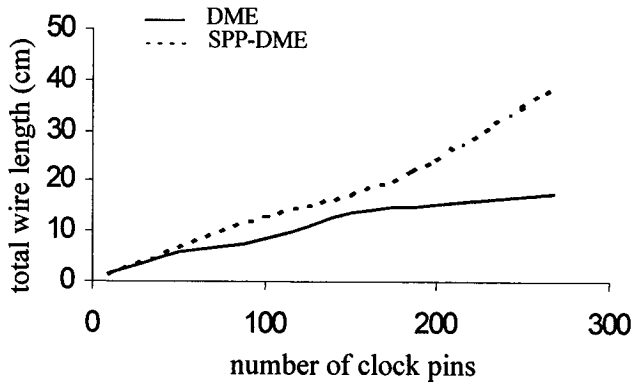


Figure 6.21. Total wire length vs. number of clock pins for the benchmark r1.

Chapter 7 Conclusions and Future Work

7.1 Summary

The motivation for this research is to resolve some of the problems arising from the rapid reduction in the feature size and the growth in the size and complexity of SoCs. One such a problem is the interconnect dominance in determining the system performance. The rise in SoC complexity and interconnect dominance team up to intensify a specific challenge to the CDN design. The challenge stems from the CDN's crucial role in influencing the functional integrity and performance of SoCs. This thesis focused on the problem of routing the CDN since the CDN consumes an increasing portion of all resources in terms of wiring area, power and design time.

Considrable attention was dedicated to the problem of Engineering Change (EC) techniques. A new algorithm is developed, called Adaptive Wire Adjustment (AWA), to handle minor modifications of the CDN. Theoretical analysis and experimental results showed that whenever there is a skew between any two clock pins, AWA algorithm can reduce the skew to any required bound, and ultimately to zero, though zero skew is an

over constraint sometimes. The simulations performed on CDNs have shown that using AWA algorithm can reduce the bounded skew requirement further without recalculating the whole solution. In addition, different experiments showed that AWA has outperformed other algorithms for similar scenarios of minor modifications.

A new method of local topology modification was proposed in order to improve the convergence of AWA algorithm; and to have a topology flexibility that resulted in minimizing the total wire length. The reduction in the total wire length implied a reduction in the power consumed by the CDN. Also, the experiments showed that previous well known algorithms relied intensively on wire elongations, and they offered a solution that suffers from high Standard Deviation between Path Lengths (SDPL). A high SDPL implied vulnerability of the CDN to temperature and process variation and the fidelity of the CDN in terms of satisfying the skew requirements. By using Local Topology Modification with previous algorithms, wire elongations and SDPL were reduced effectively. Reducing the number of wire elongations would be an important advantage for layout tools.

The presence of obstacles in the routing plane of the CDN was studied in detail. In general, terminals and Steiner points of a CDN have to be interconnected by vertical and horizontal wires (Rectilinear Steiner Tree), which must not intersect with some obstacles in the routing plane. Routing a multi-terminal net under obstacle constraints is NP-hard; and all existing algorithms would find the detailed routing of an interconnect of a net by finding the shortest path for the interconnect. However, there are different shortest paths that have different attributes, such as the number of turns or the separation from other

paths or obstacles. Henceforth, it is important to have a routing cell that includes all solutions so that the designer can choose the appropriate path in response to pre-defined constraints. A new model, called Shortest Paths Polygon (SPP), was proposed to describe all shortest paths between two points in the presence of obstacles. This model, which is based on gridless graph, facilitates a better routing of any multi-terminal nets since it determines all shortest paths. The SPP model is incorporated into AWA algorithm in order to tune the CDN under obstacle constraints. The SPP model facilitates the determination of all possible locations of a Steiner node of the CDN under obstacle constraints.

To test the effectiveness of the SPP model, it was incorporated into other CDN synthesis algorithms. At first, the SPP model was incorporated into DME algorithm to produce a ZSCDN under obstacle constraints. This is achieved by developing procedures to determine the links of the visibility graph that is required to determine the SPP of each wire in the CDN. A comparison is held between the proposed approach of producing ZSCDN under obstacle constraints and other approaches. The main advantage of the proposed approach is the determination of all shortest paths; and consequently, the total wire length can be minimized further as compared to other approaches. This advantage does not increase the computation time since the SPP model requires the minimum memory requirements to represent the routing area of a wire. In addition, by considering the characteristics of the SPP model, the edges of the visibility graph is reduced effectively. Consequently, the time of determining the SPP in the visibility graph is reduced effectively.

Finally, the problem of limiting the number of planes that are used to route the CDN was discussed. Usually, the CDN is routed in the upper metal layers; and it is important to have an algorithm that can route the CDN within a specific number of layers. Most of the CDN synthesis approaches do not consider this practical issue at all due to its difficulties. In fact, there are two algorithms that resolve this problem by routing the CDN in one plane; and thus, such a CDN is called planar. Indeed, these algorithms generated their own topology in order to reach to the final solution. Consequently, there is an increase in the total wire length due to the topology of the planar CDN. However, such an increase in the total wire length can be avoided by using different topologies, as was described in Chapter 4. In addition, it is not practical to rout the CDN in one plane due to the increase in the size of the CDN of modern SoCs. Thus, there is a need to have an algorithm that can route the CDN in a specific number of layers instead of just one layer. In Chapter 6, the first approach that can route the CDN in a specific number of layers was developed. This is achieved by incorporating the SPP model into DME algorithm such that the routed wires of the CDN are treated as obstacles at any instant. The approach is tested by routing the CDN in one and/or two layers. The proposed approach achieved less total wire length as compared to other algorithms since the SPP model considers all shortest paths between two points. In addition, it was found that the total wire length of the CDN increases exponentially after a specific number of clock pins. Such a number called the planarity number, depends on the geometric distribution of the CDN and the topology of the CDN.

7.2 Contributions

The major contributions of this thesis include the followings:

- Development of AWA, a new algorithm that implements an ECO for the CDN.
- Formulation of the topology variation impact on the qualities of the CDN, such as the total wire length, clock latency, number of wire elongations and number of intersections between the wire of the CDN.
- Introduction of a new performance parameter called the Standard Division of Path Length (SDPL) of CDN to reflect the quality of the CDN topology.
- Introduction of the quadratic tree and the concept of Local Topology Modification (LTM). The LTM approach improves the convergence of AWA algorithm and the quality of the CDNs.
- Development of a search tree to determine the near optimum local topology modification.
- Development of a new routing model, called Shortest Paths Polygon, to represent all shortest paths between two Steiner nodes of a CDN.
- Defining the characteristics of the SPP model. A procedure is developed to determine the SPP model in the visibility graph.
- Incorporation of the SPP model into the AWA algorithms in order to tune the CDN under obstacle constraints
- Incorporation of the SPP model into the DME algorithm in order to produce a ZSCDN under obstacle constraints
- Application of the SPP model to rout the CDN in a specific number of planes.

- Defining a new parameter, called the planarity number, that can be used to describe the impact of the planarity on the total wire length of a CDN.

7.3 Suggestions for Future Works

7.3.1 Incorporation of the LTM and SPP concepts into different algorithms

In this thesis, the LTM and SPP concepts were proposed. The LTM technique was tested with AWA, DME and GDME algorithms. The SPP model was tested with AWA and DME algorithms; to produce a planar CDN. Nevertheless, the SPP model can be incorporated into other CDN synthesis algorithms. In addition, both concepts, the LTM and SPP, can be incorporated into different CDN synthesis algorithms, such as AWA, DME and GDME algorithms in order to improve the qualities of the CDN solution.

7.3.2 Incremental buffer insertion into the CDN

The AWA algorithm was proposed to tune the CDN by adjusting the wires of the CDN without considering buffer insertion into the CDN. Similar approach can be used to tune the CDN by inserting buffers into the CDN incrementally. The buffer have to be inserted in order to minimize the power consumption of the CDN and to minimize the clock signal latency from the clock source to the clock sinks [90-91]. Inserting the buffers incrementally has the advantage of improving the CDN quality when there is a need to modify the CDN during the system design process.

7.3.3 Incremental link insertion into the CDN

The variation in the clock skew presents a great challenge for the CDN due to the process variation, power/ground noise and temperature variations [83-87]. The clock variation is very harmful to the system performance and functional integrity. In addition, it is very difficult to control and estimate the skew variation. One approach of resolving this problem is to design a non-tree CDN. However, such a CDN consumes excessive amount of wire length. Instead, links can be inserted into a tree CDN. The only known scheme of inserting the links into the CDN is based on inserting the links intensively and systematically into a tree CDN [83]. Instead, the links can be inserted incrementally into the CDN with the objective of minimizing the skew variations.

7.3.4 Differential CDN

Another approach of reducing the power/ground noise impact on the quality of the clock signal is to use a differential signaling instead of the traditional single ended clock signaling. This approach requires the routing of two interconnects between every pair of Steiner nodes of the CDN. In addition, differential buffers have to inserted in order to minimize the power consumed by the CDN and the clock signal latency.

7.3.5 Applying RLC delay model to the proposed approaches

Different approaches that have been discussed in this thesis are based on the Elmore delay model. This model does not consider the inductance impact. Neglecting the induc-

tance impact would affects the quality of the CDN [88-89]. Thus, more accurate delay model have to be considered in future works.

7.3.6 Wire length estimation under obstacle constraints

It is important to have an accurate estimation of the wire length of any multi-terminal net in the system as early as possible during the design process of the system. Indeed, the accuracy of the wire length estimation affects dramatically the quality of the placement task [52]. However, the wire length can be determined only after the placement and routing phases. Thus, it is important to have an advance approach to estimate the wire length of different nets in the system based on the number of the sinks of the net. In Chapter 6, a new parameter, called Planarity number, was proposed; and it can be used to estimate the total wire length based on the number of sinks, topology of the net and number of layers that are used to rout the net.

7.3.7 SPP router

Due to the increase in the size and complexity of SoC, the run time and memory requirements for placement and routing is increasing rapidly. In addition, there is an increasing number of objectives to be considered during the placement and detailed routing [65-68]. For example, signal integrity and cross talk to name a few. Consequently, there is an increase in the iterations between placement and detailed routing. In this regard, the gridless routing approaches attract more attention since they require less memory requirments. Consequently, these approaches can handle larger problem sizes with less run time. Most of these approaches are based on tile base data structure. There are many

works that address the impact of the size and shape of the tile on the performance of the router. One approach of improving the router performance is to use non-uniform tiles, such as the L-shape tile. Instead, the SPP model can be used since it can represent complex shapes with minimum memory requirements. In addition, the SPP model is developed specifically to capture the routing area of the wire, which usually connects two points with minimum wire length. Thus, developing a router based on the SPP model would have the advantage of minimizing the memory/time requirements. In addition, the SPP model provides the flexibility for the detail router to chose the appropriate layout of a wire in response to other design rule conditions.

7.3.8 Incremental place and route

The increase in the iterations between the placement and detailed routing requires that both, the placer and router, have to be implemented using incremental algorithms [49]. The AWA algorithm is dedicated to perform incremental modification to the CDN. Nevertheless, the AWA algorithm can be generalized to develop an incremental router that works with an incremental placer.

References

[1] National Technology Roadmap for Semiconductors. Semiconductor Industry Association, San Jose, California, 2003.

[2] D. Sylvester and C. Hu, "Analytical Modeling and Characterization of Deep Submicron Interconnect", Proceedings of IEEE, special issue on interconnect, vol. 89(5), 2001, pp. 634-664.

[3] J. Cong, L. He, C. Koh, and P. Madden, "Performance Optimization of VLSI Interconnect Layout", VLSI Journal, INTEGRATION, Vol. 21, No. 8, 1996, pp. 1-94.

[4] C.-K. Cheng, J. Lillis, S. Lin and N. Chang, "Interconnect Analysis and Synthesis", Wiley-Interscience Publication, 2000.

[5] E. G. Friedman, "Clock Distribution Networks in VLSI Circuits and Systems", Piscataway, New Jersey, IEEE Press, 1995.

[6] J.-P Schoellkopf, "Impact of Interconnect Performances on Circuit Design", Proceedings of IEEE Intl. Interconnect Technology Conf., 1998, pp. 53 55.

[7] J. Cong, L. He, K. Khoo, C. Koh, and Z. Pan, "Interconnect Design for Deep Submicron IC's", Proceedings of ACM/IEEE Intl. Conf. on Computer-Aided Design, 1997, pp. 478 485.

[8] M. Zhao, R. V. Panda, S. S. Sapatnekar, T. Edwards, R. Chaudhry, and D. Blaauw, "Hierarchical Analysis of Power Distribution Networks", Proceedings of ACM/IEEE Design Automation Conf., 2000, pp. 150 155.

[9] R. Escovar and R. Suaya, "Transmission Line Design of Clock Tree", Proceedings of ACM/IEEE Intl. Conf. on Computer-Aided Design, 2002, pp. 334 340.

[10] S. Hasson and C. Alpert, "Optimal Path Routing in Single-an and Multiple-Clock Domain Systems", IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems, vol. 22, no. 11, 2003, pp. 1580-1588.

[11] X. Huang, P. Restle, T. Bucelot,Y. Cao and T.-J. King, "Optimization Approach for Multigigahertz Clock Network Design", IEEE Journal of Solid-State Circuits, Vol. 38, No. 3, 2003, pp.457-463

[12] C. C.-P. Chen and E. Cheng, "Future SOC Design Challenges and Solutions", Proceedings of the Intl. Symp. on Quality Electronic Design, 2002, pp. 534-537

[13] A. Kahng and G. Robins "On Optimal Interconnections for VLSI", Kluwer Academic Publishers 1995

[14] H. B. Bakoglu, "Circuits, Interconnections and Packaging for VLSI", Addison-Wesley, Reading, MA, 1990.

[15] B. A. Gieseke, R. L. Allmon, D. W. Bailey, B. J. Benschneider, S. M. Britton, J. D. Clouser, H. F. Fair III, J. A. Farrell, M. K. Gowan, C. L. Houghton, J. B. Keller, T. H. Lee, D. L. Leibholz, S. C. Lowell, M. D. Matson, R. J. Matthew, and V. Peng, "A 600 MHz Superscalar RISC Microprocessor with Out-of-Order Execution", Proceedings of IEEE Intl. Solid- State Circuits Conf., 1997, pp. 176 177.

[16] F. Ishihara, C. Klristain and K.-I. Agawa, "Clock Design of 300MHz 128-bit 2 Way Superscalar Microprocessor", Proceedings of Asia and South Pacific Design Automation Conf., 2000, pp. 647 652.

[17] J. D. Warnock, J. M. Keaty, J. Petrovick, J. G. Clabes, C. J. Kircher, B. L. Krauter, P. J. Restle, B. A. Zoric, and C. J. Anderson, "The Circuit and Physical Design of the

POWER4 Microprocessor", IBM Journal of Research and Development, Vol. 46, No. 1, 2002, pp. 27 51.

[18] G. Geannopoulos and X. Dai, "An Adaptive Digital Deskewing Circuit for Clock Distribution Networks," Proceedings of IEEE Intl. Solid-State Circuits Conf., 1998, pp. 400 401.

[19] Restle, P. McNamara, T. Webber, D. Camporese, P. Eng, K. Jenkins, K. Allen, D. Rohn, J. Quaranta, M. Boerstler, D. Alpert, C. Carter, C. Bailey, R.N. Petrovick, J. Krauter and B. McCredie, "A Clock Distribution Network for Microprocessors," IEEE Journal Solid-State Circuits, Vol. 36, 2001, pp. 792 799.

[20] S. Tam and S. Rusu, "Clock Generation and Distribution for the First IA-64 Microprocessor", IEEE Journal on Solid-State Circuits, Vol. 35, No. 11, 2000, pp. 1545 1552.

[21] J. Wood, T. C. Edwards, and S. Lipa, "Rotary Traveling-Wave Oscillator Arrays: a New Clock Technology", IEEE Journal on Solid-State Circuits, Vol. 36, No. 11, 2001, pp. 1654 1665.

[22] E. Friedman, "Clock Distribution Network in Synchronous Digital Integrated Circuits", Proceedings of IEEE special issue on interconnect, vol. 89(5), 2001, pp. 665-692.

[23] R. Escovar and R. Suaya, "Transmission Line Design of Clock Tree", Proceedings of ACM/IEEE Intl. Conf. on Computer-Aided Design, 2002, pp. 334 340.

[24] D. Duarte, V. Narayanan, and M. J. Irwin, "Impact of Technology Scaling in the Clock System Power", Proceedings of IEEE Symp. VLSI, Apr. 2002, pp. 52 57.

[25] F. HajAli Asgari and M. Sachdev, "A Low-Power Reduced Swing Global Clocking Methodology", IEEE Trans. on VLSI Systems, Vol. 12, No. 5, 2004, 538-545

[26] H. Kojima, S. Tanaka, and K. Sasaki, "Half-Swing Clocking Scheme for 75% Power Saving in Clocking Circuitry", IEEE Journal of Solid-State Circuits, vol. 30, 1995, pp. 432 435.

[27] J. Pangjun and S. Sapatnekar, "Low-Power Clock Distribution Using Multiple Voltages and Reduced Swings," IEEE Trans. VLSI Systems, vol. 10, No. 6, 2002, pp. 309 318.

[28] M. Donno, A. Ivaldi, L. Benini and E. Macii, "Clock-Tree Power Optimization Based on RTL Clock-Gating", Proceedings of the ACM/IEEE Design Automation Conf., 2003, pp. 622 627.

[29] A. Farrahi, C. Chen, A. Srivastava, G. Tellez and M. Sarrafzadeh, "Activity-Driven Clock Design", IEEE Trans. on Computer-Aided design of Integrated Circuits

[30] M.-S. Jang, J.-H. Park, Y.-N. Yeon, J.-Y. Lee, K.-M. Choi and J.-T. Kong, "Clock Network Analysis at the Pre-Layout Stage for Efficient Clock Tree Synthesis", Proceedings of IEEE ASIC/SOC Conf., 2002, pp. 363-367

[31] C.-W. Tsao and C.-K Koh, "UST/DME: A Clock Tree Router for General Skew Constraints", Proceedings of ACM/IEEE Intl. Conf. on Computer-Aided Design, 2000, pp 400-405

[32] J. Xi and W. Dai, "Useful-Skew Clock Routing with Gate Sizing for Low Power Design", Proceedings of ACM/IEEE Design Automation Conf., 1996, pp.383 388.

[33] R. B. Deokar and S. S. Sapatnekar, "A Graph-Theoretic Approach to Clock Skew Optimization", Proceedings of the IEEE International Symp. on Circuits and Systems, 1994, pp. 1407 1410.

[34] J. P. Fishburn, "Clock Skew Optimization", IEEE Trans. on Computers, vol 39, 1990, pp. 945 951.

[35] S. Dhar, M. A. Franklin and D. F. Wann, "Reduction of Clock Delays in VLSI Structures", Proceedings of ACM/IEEE Intl. Conf. on Computer-Aided Design, 1984, pp. 778-783.

[36] A. B. Kahng, J. Cong, and G. Robins, "High-Performance Clock Routing Based on Recursive Geometric Matching", Proceedings of ACM/IEEE Design Automation Conf., 1991, pp. 322 327.

[37] J. Cong, A. B. Kahng and G. Robins, "Matching-Based Methods for High-Performance Clock Routing", IEEE Trans. on Computer-Aided Design, 12(8), 1993, pp. 1157-1169.

[38] M. A. B. Jackson, A. Srinivasan and E. S. Kuh, "Clock Routing for High Performance ICs", Proceedings of the ACM/IEEE Design Automation Conf., 1990, pp. 573-579.

[39] R. S. Tsay, "Exact Zero Skew", Proceedings of ACM/IEEE Intl. Conf. on Computer-Aided Design, 1991, pp. 336-339.

[40] T.-H. Chao, Y.-C. Hsu, J.-M. Ho, K. Boese and A. Khan, "Zero Skew Clock Routing with Minimum Wire Length", IEEE Trans. on Circuits and systems, Vol. 39, 1992, pp. 799-814.

[41] K. D. Boese and A. B. Kahng, "Zero-Skew Clock Routing Trees With Minimum Wirelength", Proceedings of IEEE Intl. Conf. on ASIC, 1992, pp. 111 - 115.

[42] M. Edahiro, "Delay Minimization for Zero-Skew Routing", Proceedings of ACM/IEEE Intl. Conf. Computer-Aided Design, 1993, pp. 563 566.

[43] M. Edahiro, "An Efficient Zero-Skew Routing Algorithm", Proceedings of the ACM/IEEE Design Automation Conf., 1994 pp.375-380

[44] D. Huang, A. Kahng, and C.-W. Tsao, "On the Bounded-Skew Clock and Steiner Routing Problems", Proceedings of the ACM/IEEE Design Automation Conf., 1995, pp.508 513.

[45] J. Cong, A. Kahng, C. Koh and C.-W. Tsao, "Bounded-Skew Clock and Steiner Routing", Trans. on Design Automation of Electric Systems, 1998, pp. 341-388.

[46] J. H. Huang, A. B. Kahng and C.-W. A. Tsao, "On the Bounded-Skew Clock and Steiner Routing Problems", Technical Report TR-940026, University of California LA, Computer Science Dept., 1994.

[47] C. P. Chen, Y. W. Chang, and D. F. Wong, "Fast Performance-Driven Optimization for Buffered Clock Trees Based on Lagrangian Relaxation", Proceedings of ACM/IEEE Design Automation Conf., 1996, pp. 405 408.

[48] O. Coudert, J. Cong, S. Malik and M. Sarrafzadeh, "Incremental CAD", Proceedings of ACM/IEEE Intl. Conf. on Computer-Aided Design, 2000, pp. 236-243.

[49] A. Kahng, S. Muddu and E. Sarto "Tuning Strategies for Global Interconnects in High-Performance Deep-Submicron", VLSI Design, vol. 10(1), 1999, pp. 21-34

[50] J. Cong, J. Fang and K.-Y. Khoo, "An Implicit Connection Graph Maze Routing Algorithm for ECO Routing", Proceedings of ACM/IEEE Intl. Conf. on Computer-Aided Design, 1999, 163-167

[51] Y. Liu, X. Hong, Y. Cai and w. Wu, "CEP: A Clock-Driven ECO Placement Algorithm for Standard-Cell Layout", Proceedings of IEEE Intl. Conf. on, 2001, pp. 118-121.

[52] S. Zhang, w. Dai, "TEG: A New Post-Layout Optimization Method", IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems, vol. 14, no. 4, 2003, pp.

446 456.

[53] W. Choi and K. Bazargan, "Incremental Placement for Timing Optimization", Proceedings of ACM/IEEE Intl. Conf. on Computer-Aided Design, 2003, pp. 463-466.

[54] Y. Elboim, A. Kolodny and R. Ginosar, "A Clock Tuning Circuit for System-on-Chip", IEEE Trans. on VLSI Systems, vol. 11, no. 4, 2003, pp. 616-626

[55] H. Saaied, D. Al-Khalili, A.J. Al-Khalili and M. Nekili, "Adaptive Wire Adjustment for Bounded Skew Clock Distribution Network", Proceedings of IEEE Asia and South Pacific Design Automation Conf., 2003, pp. 243-248

[56] http://vlsicad.ucsd.edu/GSRC/bookshelf/Slots/BST/#IIII

[57] The National Technology Roadmap for Semiconductors, Semiconductors Industry Association 1997.

[58] Y. Liu, X. Hoing and Y. Cai, "TGSCO: An Algorithm for Topology Generation of Clock Tree with Skew Constraint and Optimization", Proceedings of IEEE Int. Conf. on Communications, Circuits and Systems, 2002, pp. 1444-1448

[59] J. Lillis and P. Buch, "Table-Lookup Methods for Improved Performance Driven Routing", Proceedings of ACM/IEEE Design Automation Conf., 1998, pp. 368 373.

[60] H. Saaied, D. Al-Khalili, A. J. Al-khalili, M. Nekili, "Quadratic Deferred-Merge Embedding Algorithm for Zero Skew Clock Distribution Network", Proceedings of ACM/IEEE Workshop on Timing Issues in the Specification and Synthesis of Digital Systems (TAU), Monterey, California, 2002, pp. 119-125.

[61] H. Saaied, D. Al-Khalili, A. J. Al-khalili, M. Nekili, "Adaptive Wire Adjustment for Bounded Skew Clock Distribution Network Using Quadratic Tree", Proceedings of the 14th International Conference on Microelectronics, Beirut, 2002, pp. 19-23.

[62] H. Saaied, D. Al-Khalili and A.J. Al-Khalili, "Area Minimization of Clock Distribution Networks Using Local Topology Modification", Proceedings of IEEE SoC Conf., Portland, Oregon, 2003 pp. 227-230.

[63] H. Saaied , D. Al-Khalili, A. J. l-Khalili, "Adaptive Wire Adjustment and Local Topology Modification for Tuning Bounded Skew Clock Distribution Networks", IEEE Northeast Workshop on Circuits and Systems, Montreal, Quebec, 2003, pp. 193 -198.

[64] H. Saaied, D. Al-Khalili and A.J. Al-Khalili, "Simultaneous Adaptive Wire Adjustment and Local Topology Modification for Tuning a Bounded Skew Clock Distribution Network", IEEE Transactions on CAD of Integrated Circuits and Systems (to appear).

[65] N. Shenoy and W. Nicholls, "An Efficient Routing Database", Proceedings of the ACM/IEEE Design Automation Conf., 2002, pp.590-595.

[66] S-P Lin and Y-W Chang "A Novel Framework for Multilevel Routing Considering Routability and Performance", Proceedings of ACM/IEEE Intl. Conf. on Computer-Aided Design, 2002, pp. 44-50

[67] D. Marple, M. Smulders and H. Hegen, "Tailor: A Layout System Based on Trapezoidal Corner Stitching", IEEE Trans. On Computer-Aided Design, Vol. 9, 1990, pages 66-90

[68] J. K. Ousterhout, "Corner Stitching: a Data Structuring Technique for VLSI Layout Tools", IEEE Trans. on Computer-Aided Design, Vol 3, No 1, 1984, pp. 87-100.

[69] W. Li, D. Zhou, H. Kim and X. Zeng, "Automatic Clock Tree Design with the IPs in the System", Proceedings of conf., 2001, pp. 387-390.

[70] A. Kahng and C.-W. Tsao, "More Practical Bounded-Skew Clock Routing", Proceedings of the ACM/IEEE Design Automation Conf., 1997, pp.627-632.

[71] J. Liu, Y. Zhao, E. Shragowitz and G. Karypis, "A Polynomial Time Approximation Scheme for Rectilinear Steiner Minimum Tree Construction in the Presence of Obstacles", Proceedings of IEEE Intl. Conf. on Electronics, Circuits, and Systems, 2002, pp. 781-784

[72] S. Q. Zheng, J. S. Lim and S. S. Iyengar, "Finding Obstacle-Avoiding Shortest Paths Using Implicit Connection Graphs", IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems, Vol. 15, No. 1, 1996, pp. 103-125

[73] D. Lee, C. Yang, C. Wong and H. Kong, "Rectilinear Paths Among Rectilinear Obstacles", Discreet Applied Mathematics 70, 1996, pp. 185-215.

[74] Y. Wu, P. Widmayer, M. Schlag, C. Wong, "Rectilinear Shortest Paths and Minimum Spanning Trees in the Presence of Rectilinear Obstacles", IEEE Tran. on Computers, Vol 36, No. 3, 1987, p.321-331

[75] J. Jaja and S. Wu, "On Routing Two-Terminal Nets in the Presence of Obstacles", IEEE Trans. on Computer-Aided Design, vol. 8, no. 5, 1989, pp. 563- 570

[76] T. Ohtsuki, "Gridless Routers-New Wire Routing Algorithms Based on Computational Geometry", Proceedings of Intl. Conf. of Circuits and Systems, 1985, pp. 802-809.

[77] J. L. Ganley, "Computing Optimal Rectilinear Steiner Trees: A Survey and Experimental Evaluation", Discrete Applied Mathematics 89, 1999, pp. 161-171.

[78] Q. Zhu, and W. Dai, "Planar Clock Routing for High Performance Chip and Package Co-Design", IEEE Trans. on VLSI Systems, Vol., 4, No. 2, 1996.

[79] H. Saaied, D. Al-Khalili and A.J. Al-Khalili, "Clock Tree Tuning using Shortest Paths Polygon", Proceedings of IEEE SoC Conference, Santa Clara, California, 2004, pp. 59-62.

[80] A. Kahng and C.-W. Tsao, "Planar-DME: A Single-Layer Zero-Skew Clock Tree Router", IEEE Trans. Computer Aided Design, January 1996, pp.8-19.

[81] H. Kim and D. Zhou, "Efficient Implementation of a Planar Clock Routing with the Treatment of Obstacles", IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems, vol. 19, no. 10, 2000, pp. 1220-1225.

[82] H. Kim and D. Zhou, "Automatic Clock Tree Design System for High-Speed VLSi designs: A Planar Clock Routing with the Treatments of Obstacles", Proceedings of IEEE Intl. Symp. on Circuits and systems, Vol. 6, 1999, pp. 258-261.

[83] A. Rajaram, J. Hu and R. Mahapata, "Reducing Clock Skew Variability via Cross Links", Proceedings of the ACM/IEEE Design Automation Conf., 2004, pp.18-23.

[84] A. Ajami, M. Pedram and K. Banerjee, "Effects of Non-Uniform Substrate Temperature on the Clock Signal Integrity in High Performance Designs", Proceedings of IEEE Custom Integrated Circuits Conference May 2001 (http://atrak.usc.edu/~aajami/papers/CICC01.pdf)

[85] D. Harris and S. Naffziger, "Statistical Clock Skew Modeling with Data Delay Variations," IEEE Trans. VLSI System, Vol. 9, 2001, pp. 888 898.

[86] Y. Liu, X. Hong, Y. Cai and X. Wei, "Reliable Buffered Clock Tree Routing Algorithm with Process Variation Tolerance", Proceedings of IEEE Intl. Conf., 2003, pp. 344-347

[87] D. Velenis, E. Friedman and M. Papaefthymiou, "A Clock Tree Topology Extraction Algorithm for Improving the Tolerance of Clock Distribution Networks to Delay Uncertainty", Proceedings of IEEE Intl. Symp. on Circuits and Systems, Vol IV, 2001, pp. 422-425.

[88] X. Huang, Y. Cao, D. Sylvester, T.-J. King, and C. Hu, "Analytical Performance Models for RLC Interconnects and Application to Clock Optimization," Proceedings of IEEE Intl. ASIC-SoC Conf., 2002, pp. 353 357.

[89] D. Lehther and s. Sapatnekar, "Moment-Based Techniques for RLC Clock Tree Construction", IEEE Trans. on Circuits and Systems, Analog and Digital Signal Processing, vol. 45, no. 1, 1998, pp. 69-79

[90] J. Cong and L. He, "An Efficient Technique for Device and Interconnect Optimization in Deep Submicron Designs" Proceedings of Intl. Symp. Physical Design, 1998, pp. 45 51.

[91] J. Lillis, C. K. Cheng, and T. T. Y. Lin, "Optimal Wire Sizing and Buffer Insertion for Low Power and a Generalized Delay Model", Proceedings of ACM/IEEE Intl. Conf. on Computer-Aided Design, 995, pp. 138 143.

Appendix A

Local Topology Modification Using Quadratic Optimization

The problem of finding the near optimum topology was solved using a search tree as described in Section 4.3. In this appendix, an optimization approach will be used to find the optimum topology under obstacle constraints. As described in Section 4.2, there are 15 primary topologies of connecting a node to its four children, which are listed in Figure 4.3. Any one of these topologies is built up of Steiner nodes that are distributed between different levels, where these Steiner nodes connect the leaves to the root. In order to find the optimum topology using an optimization approach, a mathematical model is ought to group leaves between Steiner nodes, as well as to introduce Steiner nodes at any level of the topology. The topologies shown in Figure 4.3(a0, b0 and c0) have two levels similar to the topology shown in Figure A.1(a). On the other hand, the rest of topologies shown in Figure 4.3 are similar to the topology shown in Figure A.1(b) in terms that they have three levels. Counting leaf level as level 0, the topology of Figure

A.1(a) has three levels with two Steiner nodes in level 1, while the topology of Figure A.1(b) has four levels with one Steiner node in level 1 and another Steiner node in level 2. Different topologies can be represented mathematically by a 4x4 binary matrix, **A**, defined as follows:

Property 1: matrix A is symmetric.

Property 2: the diagonal elements are given as:

$$a_{ii} = \begin{cases} \text{if leaf i is connected to a node in levels 1 or 2} \\ \text{O.W.} \end{cases} \tag{A.1}$$

Property 3:The non diagonal elements are given as:

$$\underset{i \neq j}{a_{ij}} = \begin{cases} 1 & \text{if leaves i and j are connected to a node in level 1} \\ 0 & \text{O.W.} \end{cases} \tag{A.2}$$

Property 4: the non diagonal elements satisfy the following constraint:

$$\sum_{\substack{i=1 \\ i \neq j}}^{4} a_{ij} = \begin{cases} 0 \\ 1 \end{cases} \tag{A.3}$$

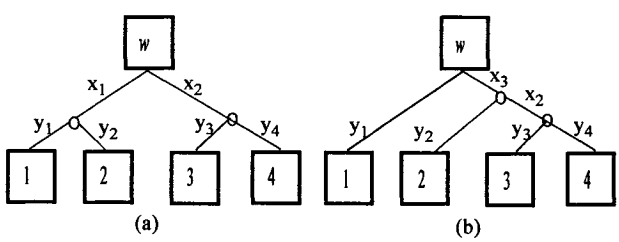


Figure A.1 Different topologies of connecting a node to its four children, where the topology may consist of (a) two levels (b) three levels.

The first property forces the topology matrix to hold a correct topology representation. The second and third properties connect each leaf to a Steiner node in level 1 or 2. Finally, property 4 ensures that a leaf is connected to a single Steiner node. Since the

matrix is symmetric, the binary variables $\bar{a}=a_{11}a_{22}a_{33}a_{44}a_{12}a_{13}a_{14}a_{23}a_{24}a_{34}$ would control the topology. Let y_1, y_2, y_3, and y_4 be the edges that connect leaves 1, 2, 3, and 4 to their parents respectively, x_1 be the edge of a Steiner node in level 1 that may pair *1* with *2*, *1* with *3*, or *1* with *4*, x_2 be the edge of a Steiner node in level 1 that may pair *2* with *3*, *2* with *4*, or *3* with *4*, and x_3 be the edge of a Steiner node that may appear in level 3. Note that for any given topology, there will be two Steiner nodes, and hence one of the variables x_1, x_2 *or* x_3 will be nulled. Using the above notation, the total wire length can be calculated as follows:

$$L = y_1 + y_2 + y_3 + y_4 + (a_{12} \vee a_{13} \vee a_{14})x_1 + (a_{23} \vee a_{24} \vee a_{34})x_2 + (\overline{a_{11} \wedge a_{22} \wedge a_{33} \wedge a_{44}})x_3 \tag{A.4}$$

Using Elmore model, the delay from *w*, as shown in Figure 4.1, to the sinks that are connected to the node *1* can be described as follows:

$$\begin{aligned}\text{Delay}(w,1) = {} & r_0 y_1\left(c_0\frac{y_1}{2} + C_1\right) + a_{12}r_0x_1\left(c_0\frac{x_1}{2} + c_0(y_1+y_2) + C_1 + C_2\right) + a_{13}r_0x_1\left(c_0\frac{x_1}{2} + c_0(y_1+y_3) + C_1 + C_3\right. \\ & + a_{14}r_0x_1\left(c_0\frac{x_1}{2} + c_0(y_1+y_4) + C_1 + C_4\right) + (\overline{a_{22} \wedge a_{33} \wedge a_{44}})r_0x_3\left(c_0\frac{x_3}{2} + (a_{12} \vee a_{13} \vee a_{14})c_0x_1 + \right. \\ & \left.\left.(a_{23} \vee a_{24} \vee a_{34})c_0x_2 + a_{11}(c_0y_1 + C_1) + a_{22}(c_0y_2 + C_2) + a_{33}(c_0y_3 + C_3) + a_{44}(c_0y_4 + C_4)\right) + t_1\right.\end{aligned} \tag{A.5}$$

where t_1 is the required arrival time at leaf *1*.

Similar equations can be deduced for the nodes *2, 3* and *4*. Time requirements of different sinks can be satisfied by equating the delays of the four nodes, i.e:

$$\text{Delay}(w,1) = \text{Delay}(w,2) = \text{Delay}(w,3) = \text{Delay}(w,4) \tag{A.6}$$

Using the above formulation, the process of synthesizing a routing net of four leaves can be considered as a nonlinear optimization problem whose objective functions are Eq. A.4 and Eq. A.5 with a non linear constraint given by Eq. A.6. Such an optimization problem has ten binary variables, $\bar{\mathbf{a}}$, to represent the topology, four variables for the edges of the nodes *1, 2, 3* and *4*; $\bar{\mathbf{y}}=y_1y_2y_3y_4$, and three variables for the edges of the Steiner nodes; $\bar{\mathbf{x}}=x_1x_2x_3$. The boundaries of the variables can be deduced from the fact that the edge lengths have to be positive and the topology matrix has to be binary. However, without considering the distances between the nodes *1, 2, 3* and *4,* the optimization process may result in impractical edge lengths; $\bar{\mathbf{y}}$ and $\bar{\mathbf{x}}$. Hence, the geometric locations of the nodes have to be brought into the scene. Considering four nodes would add six distance constraints on the optimization problem. For example, the wire that connects a pair of nodes, say *1* and *2,* w_{12}, can be determined as follows:

$$w_{12} = y_1 + y_2 + (a_{13} \vee a_{14})x_1 + (a_{23} \vee a_{24})x_2 + \overline{(a_{11} \wedge a_{22})}x_3 \tag{A.7}$$

which must exceed the Manhattan distance between the two nodes, d_{12}, i.e.:

$$w_{12} \geq d_{12} \tag{A.8}$$

Similar constraints can be deduced for w_{13}, w_{14}, w_{23}, w_{24} and w_{34}, which constitute additional constraints on $\bar{\mathbf{y}}$ and $\bar{\mathbf{x}}$, and limits the solution space. Further, these constraints vary as the topology matrix varies during the optimization process.

In order to consider obstacle constraints, consider connecting four points; 1, 2, 3 and 4, in a Manhattan plane as shown in Figure A.2. Let $\bar{1}$, $\bar{2}$, $\bar{3}$ and $\bar{4}$, called the incident points, be the closest points around the border of an obstacle, *O,* to points 1, 2, 3,and 4

respectively. Figure A.2 shows that the points 1, 2, 3 and 4 are connected by a net that traverses the incident points in the order *(2 ,3 ,4 ,1)*. But the net may traverse the incident points in other three orders; *(1 ,2 ,3 ,4)*, *(2 ,3 ,4 ,1)*, *(3 ,4 ,1 ,2)* or *(4 ,1 ,2 ,3)*. As such, there are four possibilities to avoid the obstacle. For each possible order of the incident points, the connection wire between the first and last point in the pertaining order isolates *O* from the other two points.

The obstacle affects the optimization problem by altering the constraints on the wire lengths. Indeed, the constraints on the wire that connects points *i* and *j*, w_{ij}, can be determined by *d(i,i')*, *d(j,j')* and *m(i',j')*. The length of *m(i',j')* depends on the order of traversing the corners of *O* by w_{ij}. For example, according to the routing shown in Figure 4.4, *m(2 ,3)* is equal to *d(2 ,3)*, but *m(2 ,1)* is greater than *d(2 ,1)*. Note that if *m(i',j')* is equal to *d(i',j')*, then *O* does not interfere with the routing of w_{ij}, and w_{ij} is not elongated. In general, the constraint on w_{ij} can be determined as follows:

Lemma 1: The constraint on a wire w_{ij} connecting two points, *i* to *j*, is:

$$w_{ij} \geq \begin{cases} d(i,j) & \text{if } m(i',j') = d(i',j') \\ d(i,i')+m(i',j')+d(j,j') & \text{O.W} \end{cases} \tag{A.9}$$

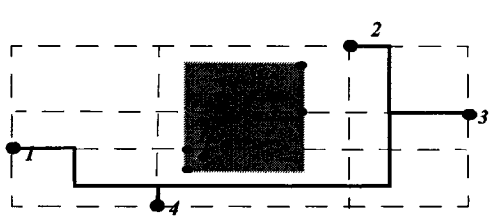


Figure A.2. Connecting four nodes with the presence of an obstacle.

The Relation A.9 generalizes relation A.8 to determine the constraint on the wire length under obstacle constraint. The wire lengths are optimized by generating the objective function and the constraints, then a quadratic optimization package is called to determine the edges y_1, y_2, y_3, y_4, x_1, x_2 and x_3. Figure A.3 shows the description of the optimization procedure.

Whenever the topology, wire length and signal delay are to be optimized locally between a node and its four children, the procedure LTM is called at first to determine the topology without considering the obstacle impact. Then, the QO procedure is called four times in accordance with different possible orders of the incident points. At each time, the objective function is the same, but the constraints on wire lengths are different. The search tree shown in Figure 4.3 can be used to minimize the total wire length or the signal delay. If the total wire length is to be minimized, then at each stage of the search tree, the branch that has less wire length is selected. On the other hand, if the objective is to minimize the signal delay, then the branch that gives minimum signal delay is selected. The QO procedure can be applied to any algorithm that synthesizes the signal's net.

Input: *A node, w, its two child nodes, u and v, and their children, 1,2, 3 and 4, that are connected to u and v by a specified topology and a specified order of the incident points.*
Output: *The edge lengths*
Begin:
  Determine *the objective functin according to Eq. 4*
  Determine *the constraints according to Eq. 6 and Eq. 9*
  Call *the optimization package*

Figure A.4 The Quadratic Optimization (QP) Procedure

Appendix B

Determination of the Arc Projection

Here, a new method is proposed to determine the projection of an arc on another. Let S and T be the arcs of nodes s and t respectively; and let the distance between the two arcs be $D(S,T)$. The two arcs can be either parallel or perpendicular as shown in Figure 6.6, where *Projection*(S,T) and *Projection*(T,S) are shown as solid thick segments.

Eq 6.2 can be used to determine the projection of an arc on another, say *Projection(S,T)*, as follows. Recall that *Projection(S,T)* is the set of points of T that can be connected with minimum wire length to S. Thus, *Projection(S,T)* can be determined by the distance between the two arcs, $D(S,T)$; which depends on the relationship between S and T.

Projections of parallel arcs

The projection of S on T, *Projection(S,T)* is a subset of T; and it can be determined as follows:

$$\begin{aligned}
&\textit{Projection}(S,T)=T\cap \overline{Q}(S^h)\cap \overline{Q}(S^t) && \text{if } T\not\subset Q(S^h) \text{ and } T\not\subset Q(S^t)\\
&\textit{Projection}(S,T)=T^h && \text{if } T\subset Q(S^t)\\
&\textit{Projection}(S,T)=T^t && \text{if } T\subset Q(S^h)
\end{aligned} \qquad \text{(B.1.a)}$$

where $\overline{Q}(.)$ refers to the complimentary of Q in the Manhattan plane.

Note that the case $S \cap T \neq \phi$ is considered a trivial case where *Projection(S,T)*$=S \cap T$. Reducing an arc (T) into its projection *(Projection(S,T))* by another arc (S) implies that T is being *truncated* by *S*. Thus, the end points of the resulting arc, *Projection(S,T)*, can be different from those of *T*. Indeed, the head and tail of *Projection(S,T)* can be determined as follows:

$Projection(S,T)^h = p{:}p \in T$ and $p_x = S^h_x$	if $T^h_x \leq S^h_x \leq T^t_x$ and $S \uparrow T$
$Projection(S,T)^h = p{:}p \in T$ and $p_y = S^h_y$	if $T^h_y \leq S^h_y \leq T^t_y$ and $S \downarrow T$
$Projection(S,T)^h = T^h$	if $T^h \notin Q(S^h)$
$Projection(S,T)^h = T^t$	O.W. (if $T^t \in Q(S^h)$)
$Projection(S,T)^t = p{:}p \in T$ and $p_x = S^t_x$	if $T^h_x \leq S^t_x \leq T^t_x$ and $S \uparrow T$
$Projection(S,T)^t = p{:}p \in T$ and $p_y = S^t_y$	if $T^h_y \leq S^t_y \leq T^t_y$ and $S \downarrow T$
$Projection(S,T)^t = T^t$	if $T^t \notin Q(S^t)$
$Projection(S,T)^t = T^h$	O.W. (if $T^h \in Q(S^t)$) (B.2.a)

Projections of perpendicular arcs

The projection of *S* on *T*, *Projection(S,T)*, is also a subset of *T*; and it can be formulated as follows:

$Projection(S,T) = T \cap (Q((S^h) \cup Q((S^t))$	if $T \cap Q(S^h) \neq \phi$ or $T \cap Q(S^t) \neq \phi$
$Projection(S,T) = T^h$	if $T \subset (\overline{Q}(S^h) \cap \overline{Q}(S^t))$ and $T \uparrow S$
$Projection(S,T) = T^t$	if $T \subset (\overline{Q}(S^h) \cap \overline{Q}(S^t))$ and $T \downarrow S$ (B.1.b)

The end points of *Projection(S,T)* are different from those of *V*; and they can be determined as follows:

$Projection(S,T)^h = p{:}p \in T$, $p_x = S^h_x$	if $T^h_x \leq S^h_x \leq T^t_x$ and $S \uparrow T$
$Projection(S,T)^h = p{:}p \in T$, $p_y = S^t_y$	if $T^h_y \leq S^t_y \leq T^t_y$ and $S \downarrow T$
$Projection(S,T)^h = T^h$	if $T^h \in \{Q(S^h) \cup Q(S^t)\}$ or $T^h \uparrow S$
$Projection(S,T)^h = T^t$	O.W.