Congestion-Driven Clock Tree Routing with Via Minimization

Ali Mohammadi Farhangi

A Thesis
in

The Department
Of
Electrical and Computer Engineering

Presented in Partial Fulfillment of the Requirements
For the Degree of Master of Applied Science at
Concordia University

Montreal, Quebec, Canada

July 2011

©Ali M Farhangi, 2011

CONCORDIA UNIVERSITY
SCHOOL OF GRADUATE STUDIES

This is to certify that the thesis prepared

By: Ali Mohammadi Farhangi

Entitled: "Congestion Driven Clock Tree Routing with Via Minimization"

and submitted in partial fulfillment of the requirements for the degree of

Master of Applied Science

Complies with the regulations of this University and meets the accepted standards with respect to originality and quality.

Signed by the final examining committee:

____ Chair
Dr. R. Raut

____ Examiner, External To the Program
Dr. R. Jayakumar, CSE

____ Examiner
Dr. M. R. Soleymani

____ Supervisor
Dr. A. J. Al-Khalili

Approved by: ____
Dr. W. E. Lynch, Chair
Department of Electrical and Computer Engineering

____________20_____

____
Dr. Robin A. L. Drew
Dean, Faculty of Engineering and Computer Science

ABSTRACT

Congestion Driven Clock Tree Routing with Via Minimization

Ali Mohammadi Farhangi

Physical routability constraints such as legal location checking and excessive number of vias are usually ignored in most of the clock tree algorithms. These Constraints could make an abstract clock tree difficult to route in practice and cause important manufacturability and reliability challenges. Therefore the final clock tree layout specifications can be seriously deviated from the expected ones. Vias have major impact on circuit reliability and manufacturing yield. The variability in via resistance is becoming an increasing concern in nanotechnologies. In this thesis a practical frame work is proposed to construct the clock tree network under via constraint. We propose an algorithm that minimizes the number of bends that is closely related to the number of vias. The proposed algorithm is able to construct a zero skew clock tree with at most one bend branch merging. By performing simultaneous wire sizing and clock tree construction, the algorithm effectively reduces the number of bends at the expense of a small increase in capacitance. Furthermore, the number of vias is also controlled by considering a pre-specified pattern to route the internal clock tree edges. The impact of the pattern routing is taken into account in the early clock distribution design phase. We introduce a probabilistic routing demand estimation method to integrate the expected routing demand of the clock net with other clock tree optimization metrics. A new demand driven cost function is exploited in network topology generation as well as branch point embedding stages of a zero skew clock tree algorithm to reduce the number of vias. Our experiments show considerable improvements in the total number of vias. 28% reduction in the number of vias is obtained while the total clock tree wire length is reduced by an average of 8%. The post-routing induced clock skew is also controlled efficiently.

Acknowledgment

I offer my sincerest gratitude to my supervisor, Dr Asim- Al-Khalili, who has supported me throughout my research with his patience and knowledge. Without his support this thesis, too, would not have been completed or written. To me, he was not only the supervisor but a caring mentor throughout the long road of my graduate study in Concordia University. I will always be grateful to him and will remember his words of wisdom.

Table of Contents

Table of Figures

List of Tables

Abbreviation Table

MMM	Method of Means and Medians
DME	Deferred Merge Embedding
MS	Merging Segment
GMA	Geometric Matching Algorithm
BST	Bounded Skew Tree
ZST	Zero Skew Tree
SPP	Shortest Path Polygon
ARD	Average Routing Demand
NG	Nearest Neighbor Graph
MC	Merging Cost
pd	Probabilistic Demand
Adp	Accumulative Demand Peak
SCCER	Single ended Conditional Capturing Energy Recovery

Chapter 1

Introduction

1.1 Clock distribution network

Clock signal is used as a timing reference to control the flow of data within a synchronous digital system. Almost every signal transition between memory elements is referenced to specific clock edge. If a memory element receives a clock edge at the wrong time it would capture the wrong data and would cause a system malfunction. Therefore the clock waveform must be clean and sharp. As the complexity of system increases, the number of flip flop which are to be clocked increases and the flip flops spans in a larger chip area. Hence, the clock signal in modern synchronous circuits has the largest fan-out and travels over the longest distance, which impose lots of challenges to deliver the clock to all flip flops at certain time. Furthermore, clock signal is affected by technology scaling, in that long global interconnect lines become much more resistive as line dimensions are decreased.

As VLSI circuit feature size continues to shrink, the quality of clock networks becomes more influential upon the circuit timing performance. A poor clock distribution can consume an inordinate amount of power, degrade system performance or prevent correct functionality. A clock distribution network can be characterized by the following criteria:

1. **Clock Skew**

   Clock skew between two flip flops is defined as the deference between clock edge arrival times from the clock source to the flip flops. Minimizing skew is necessary to prevent hold time violation, which can cause flip flops to operate in meta-stable state and provoke random circuit failure. Because hold time violations are independent of clock period, they cannot be avoided by increasing the period. Skew minimization is also important because skew reduces the available positive time slack and could in some cases decrease operating frequency.

2. **Clock phase delay**

   Clock phase delay is the maximum delay of all paths from the clock source to all flip flops. Phase delay is becoming a dominant factor in chip performance as feature size decreases and chip size getting bigger. Phase delay can increase the impact of clock delay uncertainty due to process variation.

3. **Power dissipation**

   Clock distribution network is the largest capacitive interconnect network which also has the largest fan-out. The clock signal operates with the highest switching frequency in a chip which makes the clock network one of the most contributing factors in total dynamic power dissipation of a chip.

4. **Jitter**

   Jitter is the random variation in clock cycle time which is mainly a problem of clock generation source. Also delay variation of clock buffers induced by power supply noise can be a contributing factor of clock jitter seen by a flip flop. Jitter is generally a non-deterministic phenomenon

5. **Clock edge slope (slew rate)**

   Clock edge slope impacts the operating speed of the flip-flop. The clock slope must be sharp enough to ensure proper data capturing in flip-flops and to prevent meta-stability problem. The slope also is a contributing factor in short circuit power dissipation in flip-flops and clock buffers. In clock distribution algorithms, the clock slope is usually considered as a design constraint in optimization process.

6. **Process variation**

   Global systematic and local random variations in interconnect and transistor characteristics induce clock skew even if zero skew is achieved by wiring adjustment and buffer insertion. Since the amount of variation is unpredictable, it is difficult to minimize the variation induced clock skew. Common strategies to deal with process variation problem in clock design include sensitivity minimization or non tree clock routing [1]

All works related to clock networks can be divided into three categories: tree and none-tree (i.e. mesh) networks and hybrid networks that combine both the tree and none-tree approaches [2],[3],[4]. Many high performance designs such as micro processors use grid or hybrid networks or to reduce the impact of manufacturing variations on the clock network performance [5]. Non tree structure is robust against process variation because several paths from the clock source to a flip flop compensate for difference of delay time However, these network configurations consume significantly more power than traditional trees because of their large wiring overhead. Furthermore clock gating technique to turn off a part of network would be more difficult to perform in a non-tree structure since there are multiple redundant routes to a flip flop.

In an age where clock networks can consume a great portion of total chip power, the use of trees for clock distribution can save power and routing resources compare to other clock network topologies [6]. In this dissertation, we concentrate upon skew minimized clock tree construction at design phase.

1.2 Problem statement and motivations

Despite their advantages, the available clock tree routing algorithms tend to achieve limited direct impact in practice, since practical considerations such as varying layer parasitic and via effects are often ignored. Also, in majority of clock routing algorithms, the routing paths are frequently abstracted as a single rectilinear plane ignoring the via cost, layer dependent routing cost and congestion impact in a particular direction. Contrary to these non realistic assumptions, most VLSI chips use between 4 to 14 layers of metal, and the global clock distribution typically uses most or all of them [7]. The upper metal layers are often thicker, so these layers are used for the longest wires. In almost every industrial chip, each wiring plane is used predominantly in either the X, or

Y direction which illustrates how most of the academic works are far from reality. In reality, routing a prescribed-skew clock distributions that minimize the process variation and vdd noise induced skew and jitter associated with critical timing paths on the chip, while using minimum wiring and power resources is an extremely challenging task. The clock routing difficulty is compounded with the fact that there is often a large amount of “blockages" on wiring levels and at the silicon levels. The network must of course be "optimally buffered" and the wires simulated with accurate transmission lines, which requires careful shielding and return-path design. Majority of clock tree algorithms rout an abstract clock tree and only consider the delay balancing without taking into account detail routing constraints. Lots of detours, undesirable vias and changes in routing path must be carried out by detail routers in order to layout the abstract clock tree in a practical manner. It is a difficult task to preserve the clock skew in its applicable budget during these changes. Therefore a more practical clock tree router is needed to take these practical issues into account in the early clock tree construction phase.

1.3 Overview of the dissertation

This research provides one of the first studies of the clock tree construction problem under via constraint. One of the major drawbacks associated with the available clock tree routers is the fact that they introduce too many vias. In deep sub-micron regime, vias not only affect the chip area, but also increase the resistance and reduce the circuit reliability due to electro-migration, signal echoing and process variations [8],[9]. In Chapter 2, we investigate the most recent trends in clock tree construction algorithms. Also we introduce some definitions and terms and describe well known algorithms and models which are used as bases for the dissertation contributions. In Chapter 3, a novel topological via minimization algorithm is illustrated. We propose an algorithm that minimizes the number of bends that is closely related to the number of vias. By performing simultaneous wire sizing and clock tree construction, the algorithm effectively reduces the number of bends. In Chapter 4, the impact of routing congestion in multi-layer clock routing is investigated. Among all practical routing issues in clock networks, clock tree congestion also plays a role in the final obtained clock tree specification. Having the congestion to be addressed later in the detail routing can affect both the total wire length and the clock tree skew which are the primary objectives for all clock routers. This is mainly because clock tree congestion and clock delay are often competing objectives. In order to avoid congestion, some wires must make detours, and the signal delay may consequently suffer. Besides skew and wire length, congestion has implicit impact on the number of bends for each wire segment [9]. A wire bend usually implies a switching of layers, which involves via resistance that adds to the delay and reduce reliability. In zero skew clock tree routing, congestion, control of the number of vias, and total wire length are rather inconsistent objectives, therefore a unified algorithm is needed to consider all of them simultaneously. A new unified algorithm is proposed in Chapter 4 to carry out clock routing in a non-abstract fashion which includes congestion information in different routing layer during clock tree construction. Chapter 5, includes a clock tree construction algorithm using skew compensation technique that reduces the total clock tree wire length and the total number of bends in the clock tree.

Chapter 2
Clock Distribution Network

The clock cycle time is mainly determined by two factors in synchronous circuits:1) signal delay in critical paths in the circuits. 2) Clock skew. The clock skew is the time between the maximum and minimum delay of clock signals from the clock source to all flip-flops (Usually between two adjacent Flip Flops) . In Figure 2.1 t_{skew}, clock skew between the two adjacent flip-flops, is computed by:

$$t_{skew} = t_i - t_j$$

Where t_i and t_j are the clock delays from the clock source to two adjacent flip-flops *i* and *j* respectively. In a synchronous system two flip-flops are called sequentially adjacent if data signal travels from flip flop i to j in one clock cycle as depicted in Figure 2.1. When data is launched by a late clock edge, or captured by an early clock, there is insufficient time for the data signal to arrive before the clock. In such a condition, skew would limit the maximum frequency of operation. This type of skew is usually called positive skew and causes long-path errors. Long path errors can be controlled by increasing the clock period. Also a serious race condition can occur when the launching clock is early relative to the capturing clock. Such a race condition is due to hold time violation in which the data arrives before the receiving flip-flop properly captures the data for the previous clock edge, causes the receiving flip flop to fall into a meta-stable state. This type of failure (short path failure) cannot be avoided by reducing the clock frequency and requires delay padding in combinational logic path [7].

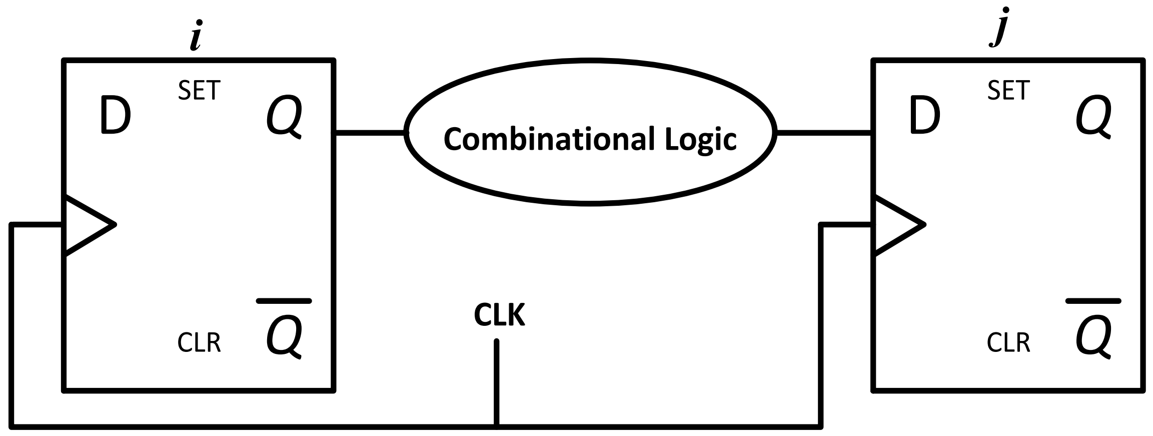


Figure 2-1 Sequentially adjacent flip-flops

Clock skew only affects sequentially adjacent flip-flops and it is likely to occur within a single macro or unit .Therefore many global clock distribution strategies seek to reduce local skew within subsets of the design, accepting larger skew between these subsets. Unfortunately, any sequential path traversing a boundary will be subject to much larger potential skew. An ideal global clock distribution achieves low local skew everywhere, not just within certain unit or macro boundaries [7].

2.1 Tree Topology Clock Network

Traditionally clock signal is distributed by a tree network which is characterized by unique paths that deliver the clock signal to every flip flop. Because a balanced clock tree is simpler to construct and analyze with mathematical models, tree is the only type of clock distribution supported by most commercial tools. Also unique source to sink clock path in tree topology enables skew to be intentionally used to improve performance [10]. With no redundancy in source-sink paths, clock tree introduces less wiring capacitance. Hence tree topology is a low power solution to distribute clock signal while easing dynamic clock gating. However, the tree topology sacrifices the clock network robustness. In general, clock trees are more vulnerable to jitter and process variation induced skew. It is well illustrated in [11] that even an exact zero skew clock tree network can have a great deal of skew caused by variability of interconnects and buffers along paths.

Simplest style of clock tree distribution is an H-tree. H-tree uses a recursive "H" routing pattern and inserts buffers at the regular intervals in the hierarchy. As shown in Figure 2.2, the path from clock source to all leaves in H-trees are perfectly balanced which provide zero skew clock deliverance. However skew can still be introduced by intra-die source of variation as studied in [12]. H-tree is a regular pattern tree which is suitable for uniform sink and capacitive load distributions. In practice, sink loads and locations are rarely uniform, hence H-trees are only typically useful for the top level of clock distribution hierarchy [7].

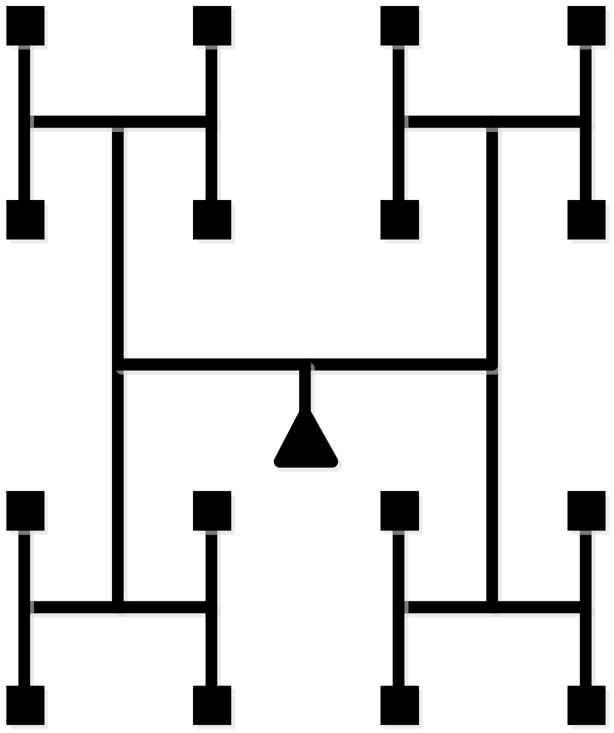


Figure 2-2 H-tree clock distribution network for 16 sinks

To accommodate non-uniformity in sink distribution and their loading capacitance, balanced merged clock trees offer equal delay to clock sinks as depicted in Figure 2.3. As such an arbitrary distributed clock sinks are connected by a tree where the clock source drives the root node and the clock sinks are the leaves of the tree. The clock tree is conceived by recursively merging a set of sub-trees in a bottom-up fashion. Merging is carried out in such a way to maintain balanced delay from the root of the newly created sub-tree to all leaves. The balancing relies on adopted delay models such as Elmore delay [13]. Although the simplified delay models are known to be only approximations of the actual delay, they still generate fairly low skew clock trees.

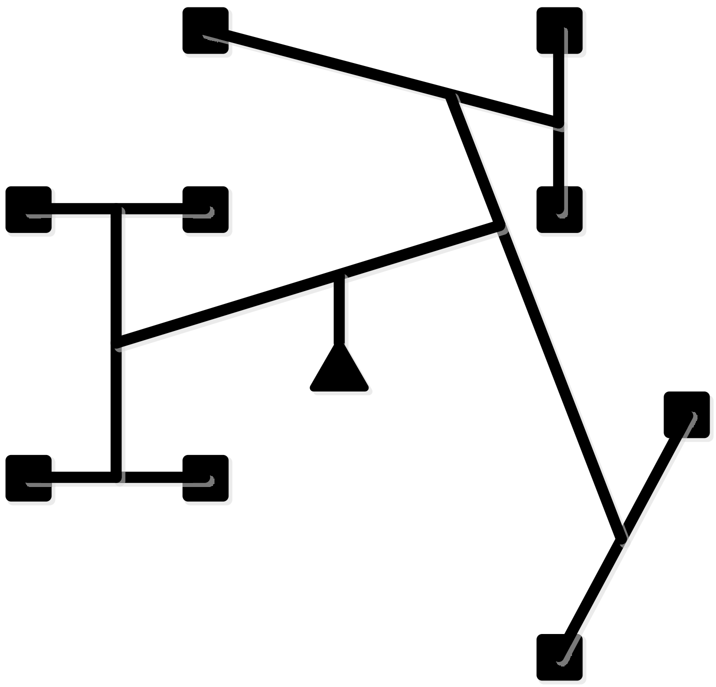


Figure 2-3 A non-regular binary clock tree

Many high performance designs use non-tree topology to improve performance in the presence of process variation. Non-tree distribution usually is used in a hybrid hierarchical structure, where the top level clock signal is still distributed with a tree structure, but the leaves of the tree is connected by low-resistance spines [17],[18] or a full mesh [14],[15],[16] which covers the chip area. By providing multiple driving paths to clock sinks, the mesh and spines compensate for the difference of delay between flip flops, hence skew is effectively reduced. However they consume significantly more power than traditional trees. Beside power, using a fine mesh to cover the entire chip area requires significant routing resources and seriously impacts the overall routability of a design. Also neither meshes nor spines allow skew to be intentionally used to improve performance (skew scheduling). Another variant of non-tree clock distribution in [18] uses a course global mesh distribution in the most top level and uses local trees to deliver the clock signal to all flip flops in the lower level. Using tree structure enables easy clock gating and skew scheduling.

2.2 Clock Tree Synthesis

The objective of clock tree synthesis is to create a buffered, routed tree such that the skew and total power is minimized. Let $S = \{s_1, s_2, ..., s_{1n}\} \subseteq R^2$ be the set of n clock sinks in a Manhattan plane. Each clock sink s_i is associated with a capacitance c_i and coordinates (x_i, y_i). The abstract topology of the set $S, T_t(S)$, is a rooted Steiner tree with the leaves corresponding to the clock sinks in S. The Internal nodes, referred as Steiner nodes, correspond to merging locations. As depicted in Figure 2.4, any node $k, k \in T_t$ 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_t; which is denoted as *Path*(w,k).

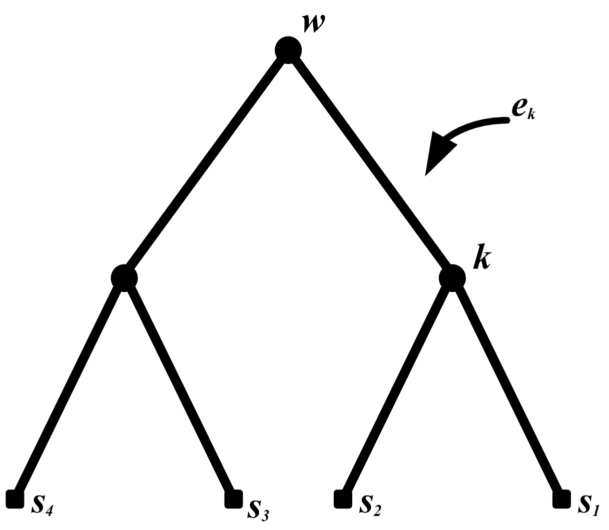


Figure 2.4 Connection topology for a binary clock distribution network

Clock tree synthesis is the process of mapping the internal nodes of the abstract tree into the coordinates in Manhattan plane to form a mapped topology, $T_m(S)$. $T_m(S)$ is the abstract embedded tree in which all clock tree edges are abstracted on a single plane. Diagonal edges must still be decomposed into vertical and horizontal segments and realized on multiple routing planes. Buffers are inserted before, during, or after routing to maintain acceptable slope at the sinks. Buffers can be inserted at internal tree nodes or along the clock tree edges. The buffers and wires can also be sized in a final tuning step.

2.3 Delay model

Clock tree construction algorithms extensively use approximation models to calculate signal delay through wires. Because inductive effects are negligible for wires inside a VLSI chip up to few Giga-hertz clock frequency, many approximation models estimate the delay of the interconnect wires using first moments of the delay response for an RC network. The Elmore delay [13] model which only uses the first moment of the delay response of the RC network could easily estimate latency-growing trends during the clock tree construction. However, due to the insufficient accuracy of the delay model, skew is difficult to analyze accurately during the clock tree construction. Therefore, embedding SPICE simulation in clock tree synthesis flow is becoming more frequent in recent publications [19],[20],[21]. However, the speed of simulation is the limiting factor in such methods.

In this dissertation, the π Elmore delay model is used for delay estimation in clock tree construction. A wire edge e_k can be modeled by a resistor r_k, and two capacitors $\frac{c_k}{2}$, where r_k and c_k are the resistance and capacitance of the wire edge e_k respectively. A clock tree can be modeled as an RC tree as depicted in Figure 2.4. The delay of the signal traveling from a node *w* to a node *k* is computed by [25]:

$$Dealy(w,k) = \sum_{i \in Path(w,k)} r_i(\frac{c_i}{2} + C_i)$$

Where C_i is the total lumped capacitance of the sub-tree rooted at node i.

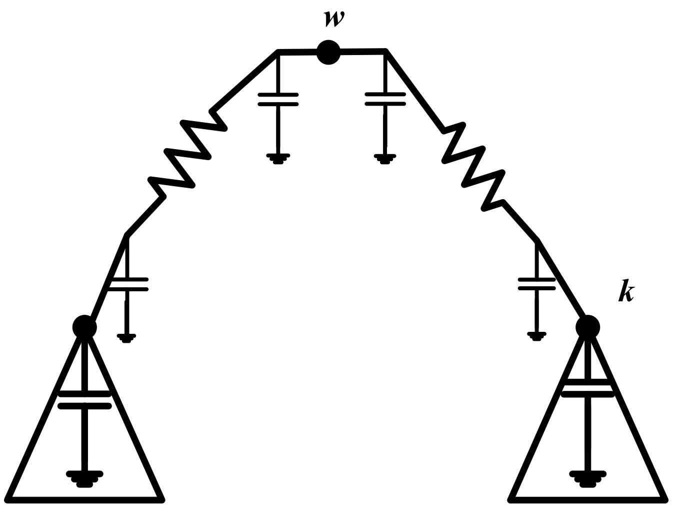


Figure 2-4 Elmore delay model for RC- network

2.4 Clock Tree routing and Topology generation

2.4.1 Method of Means and Medians (MMM)

Method of means and median [45] is a top-down method that recursively divide the region to two sub-regions of fairly equal size based on median sink locations in y and x dimension alternatively. Mean location of the original region and the two created sub-region is specified. The mean locations of the sub-regions are connected to the mean location of the original region. The process is repeated until each sub-region has maximum two sinks. In the method of means and median skew is only minimized heuristically by attempting to balance the clock tree at every level of hierarchy. This procedure is illustrated in Figure 2.5. MMM does not necessarily result in zero skew distribution, but the MMM constructed trees use very low wire length. The run time complexity of MMM algorithm in worst case is $O(nlogn)$, where n is the number of clock sinks.

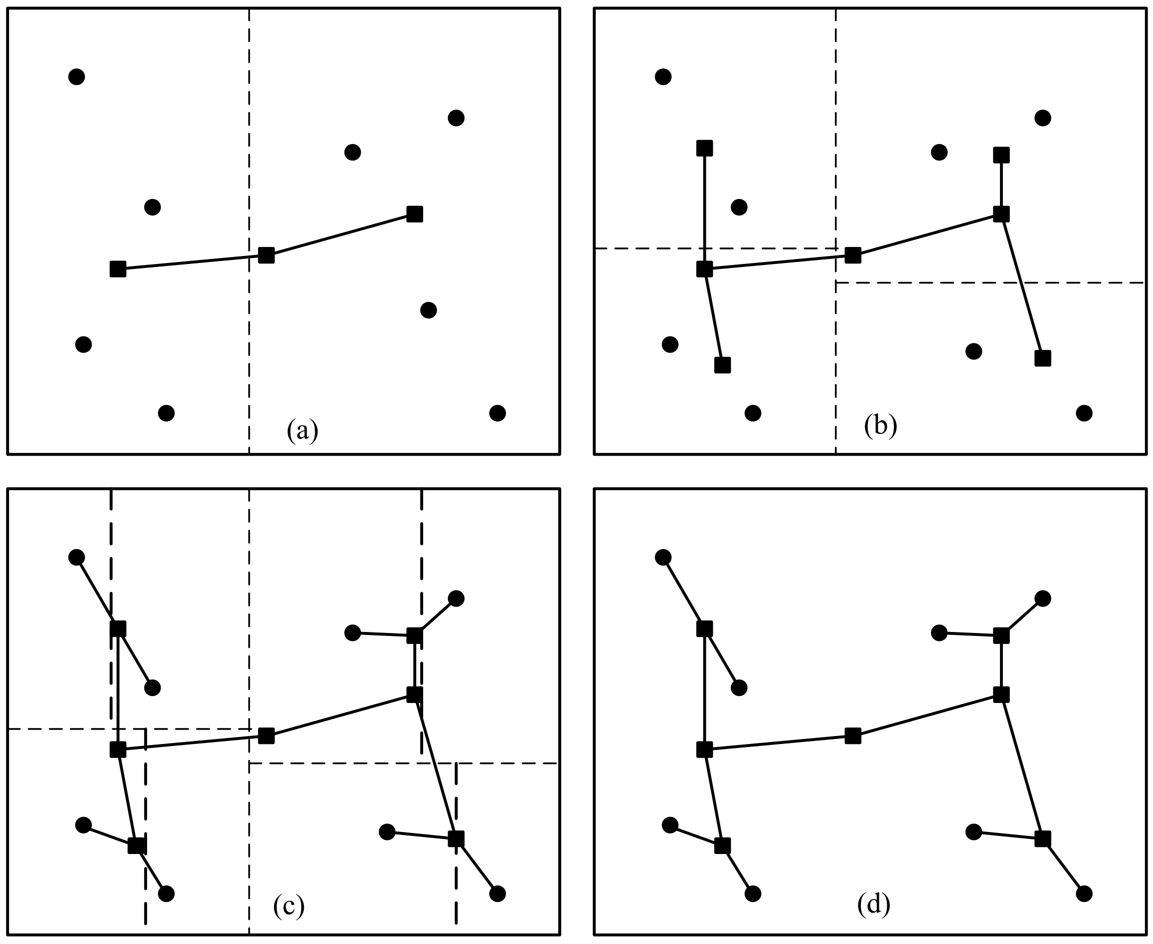


Figure 2.5 MMM Clock tree generation algorithm for 8 sinks

2.4.2 Geometric Matching clock tree construction

The MMM algorithm uses top-down partitioning and can potentially ignore local optimal matching in tree construction. Also it does not consider the delay balancing to determine the tapping points. The Geometric Matching Algorithm (GMA) [24], uses a recursive bottom-up approach to merge the best geometrically matching candidates in clock tree construction. Using any bipartite matching algorithm, GMA selects $\frac{n}{2}$ pairs of the n endpoints. As shown in Figure 2.6, the algorithm constructs a set of $\frac{n}{2}$ segments connecting the n endpoints pair-wise such that no two segments share an endpoint. Tapping points are determined on every constructed segment. A tapping point is determined such that the skew between the current pairs is minimized. The set of $\frac{n}{2}$ tapping points becomes new endpoints set for the next algorithm iteration. If wire elongation is allowed, the GMA can achieve zero skew. However the total wire length of the resulting tree is not as low as the MMM result. Also, GMA has a worst case run time complexity of $O(n^2 logn)$ for n clock sinks.

The MMM algorithm does not know about the sub-trees that have not been created yet which can result in unwanted skew. On the other hand, the geometric matching algorithm does not know how sub-trees will be merged further up in the hierarchy, which can result in extra wire overhead and power. Both of MMM and GMA methods emphasize on load balancing without evaluating actual delay. Tsay introduced an Elmore delay based layout embedding technique that can achieve exact zero skew for any given abstract tree [25].

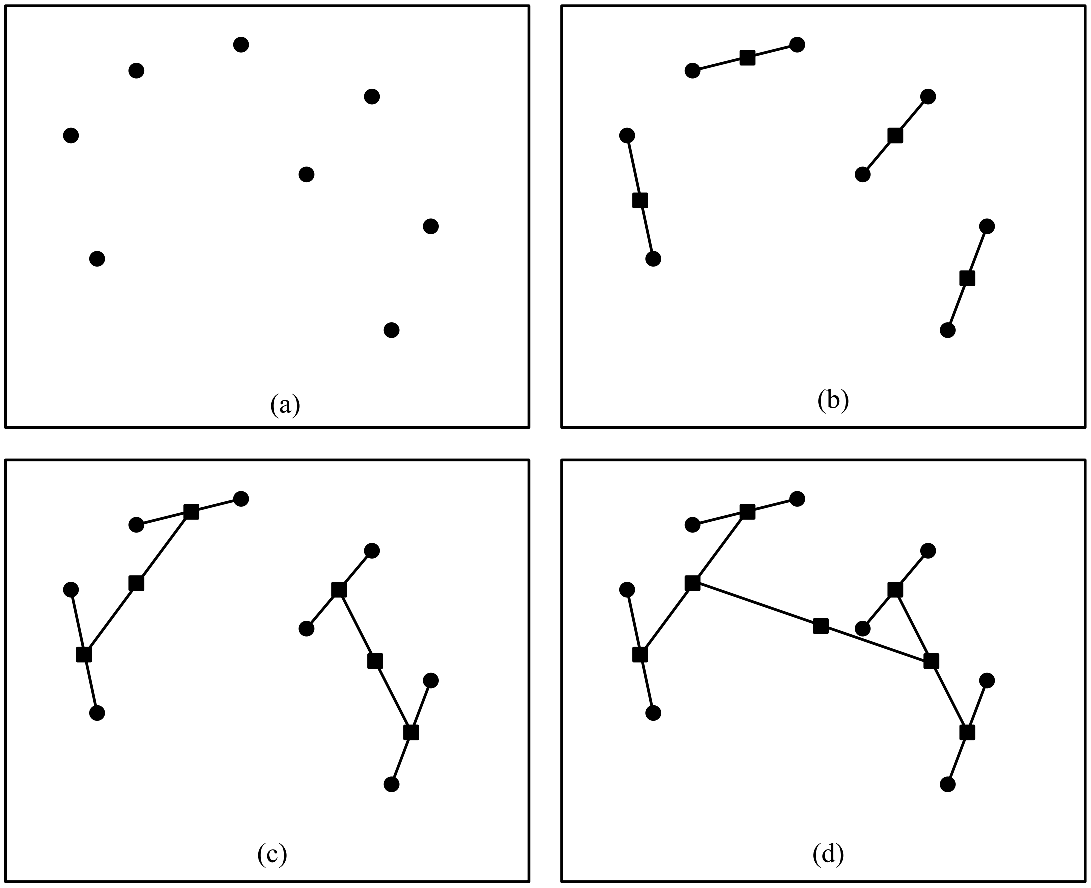


Figure 2-6 Geometric matching method for clock tree routing

2.4.3 Dynamic Programming based clock tree construction algorithms

In order to further reduce the wire length, the DME (Deferred Merge Embedding) algorithm was developed according to the observation that there are multiple locations for a merging node to satisfy skew specifications [26]. Instead of committing a merging node to particular location immediately, DME identifies and maintains the locus of the points suitable for each merging node in a bottom-up tree traversal. DME is a two phase clock tree construction algorithm in which a bottom-up pass performs merging of the sinks to find all potential zero skew merging locations then a top-down tree traversal is conducted to choose one location for every internal node such that the total wire-length is minimized. The same basic concept was later generalized to bounded skew clock tree construction [22],[28]. DME is a very mature layout embedding technique to obtain any skew specifications with minimal wire length and becomes a basis for many subsequent clock routing works.

A collection of points at the same Manhattan distance from a given point form a Manhattan circle. The shape of the Manhattan circle is a 45-degree tilted square. The locus of the points that define the area of the zero-skew merging of two points is called *merging segment* (*ms*). Given two points, s_1 and s_2 in the Manhattan plane, zero-skew merging points, or merging segment, are at the intersection of two Manhattan circles (Figure 2.7). The size of the Manhattan circles, d_1 and d_2, is calculated to balance the delay from the merging segment to the points s_1 and s_2. Both d_1 and d_2 are calculated using Tsay method based on Elmore delay model. In Figure 2.7, D= d_1 + d_2, where D is the Manhattan distance between the two points s_1 and s_2. Note that since routing is done in rectilinear directions, all distances are measured in Manhattan dimension. With rectilinear routing and equal resistance and capacitance on the Horizontal and vertical tracks, the intersection is always a line with slope of +1 or -1 denoted as Manhattan arc or simply the arc.

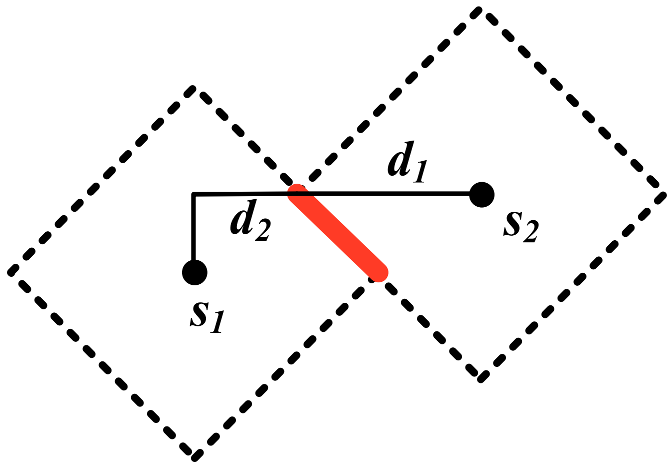


Figure 2.7 Zero skew merging of two points in Manhattan geometry

It has been proven in [26] that for two given merging segments (Manhattan arcs), the zero skew merging points also form a Manhattan arc. An example is given in Figure 2.8

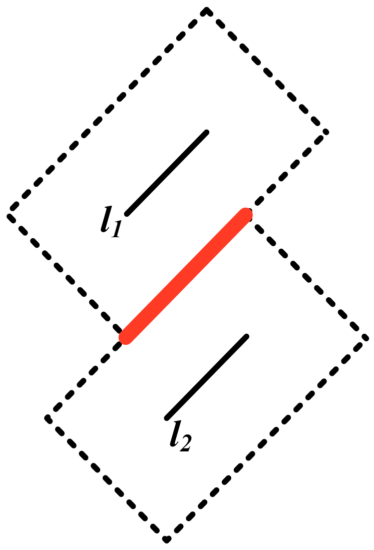


Figure 2.8 Zero skew merging of two arcs in Manhattan Geometry

In Figure 2.9, let ms_u and ms_v , be the two merging segments corresponding to the sub-trees, T_u and T_v , rooted at u and v. A Merging segment can be either a Manhattan arc or a single point. Zero-skew merging of the two sub-trees is obtained at the intersection of the two Manhattan circles. The radius of the two Manhattan circles can be find by dividing the total distance, D, between ms_u and ms_v. Each subtree, T_u and T_v ,has load capacitance, C_1 and C_2 and the phase delay of D_1 and D_2 respectively. If $d_1 = x$, then $d_2 = D - x$. Given a unit wire resistance, r, and capacitance, c, the resistance and capacitance of each wire segment is

$$R_1 = r \times d_1 = r \times x$$
$$c_1 = c \times d_1 = c \times x$$

and

$$R_2 = r \times d_2 = r \times (D - x)$$
$$c_2 = c \times d_2 = c \times (D - x),$$

respectively. To obtain zero skew based on Tsay method [25], delay of the two subtrees with wire segments to connect them must be equal.

$$D_1 + R_1\left(\frac{c_1}{2} + C_1\right) = D_2 + R_2\left(\frac{c_2}{2} + C_2\right)$$
$$D_1 + rx\left(\frac{cx}{2} + C_1\right) = D_2 + r(D - x)\left(\frac{c(D - x)}{2} + C_2\right)$$

If the equation is solved for x,

$$x = \frac{D_2 - D_1 + rD(C_2 + \frac{cD}{2})}{r(C_1 + C_2 + cD)}$$

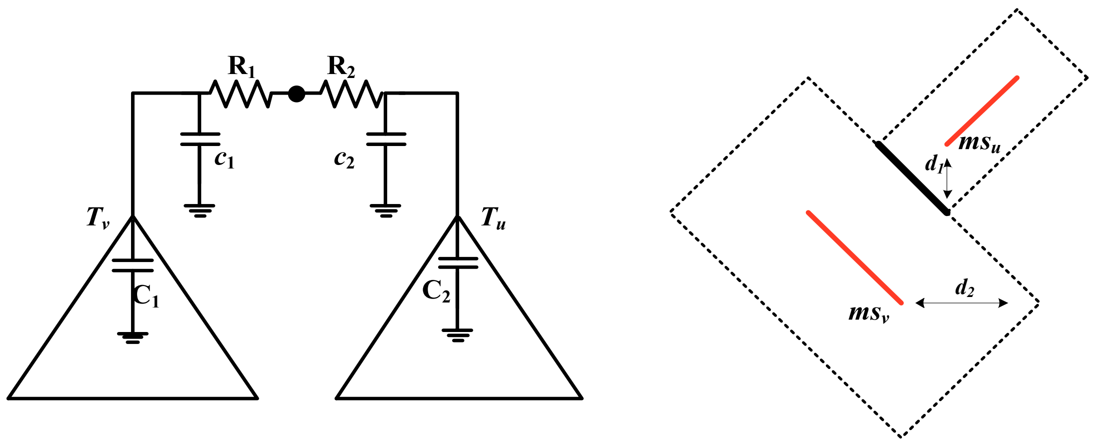


Figure 2-9 Balance merging point calculation based on Tsay method

If $x < 0$ or $x > D$, the two sub-trees cannot be merged within minimum distance separation, D. In such case, the parent merging segment is selected on ms_u or ms_v, which ever has a greater phase delay. Wire elongation will be introduced to equalize the delay between the two sub-trees. The DME algorithm requires an initial topology along with a set of the input sinks. The algorithm has linear time complexity with the given input topology. However, often greedy clustering and matching algorithm is used to cogenerate the topology at the same time [27].

In Figure 2.10, five clock sinks are to be connected by a clock tree with DME zero skew algorithm. Figure 2.10(a) shows the connection topology which is given to the DME algorithm along with the input set. During bottom-up merging, sinks s_2 and s_4 are merged into merging segment v_2 and sinks s_1 and s_3 are merged into the merging segment v_1. The merging segment v_3 is formed by merging the arc v_2 with the sink s_5. Finally, v_3 and v_1 are merged into the root merging segment r. After the bottom-up merging segment construction is completed, the root location is chosen on any desired place on the root merging segment. For every internal node, the closest point of the corresponding merging segment to the parent location is embedded. This procedure is repeated in breadth-first traversal manner until all sinks are reached (Figure 2.10(b)). The resulting clock tree has minimum total wire length on a fixed topology and zero skew under nominal process conditions [26].

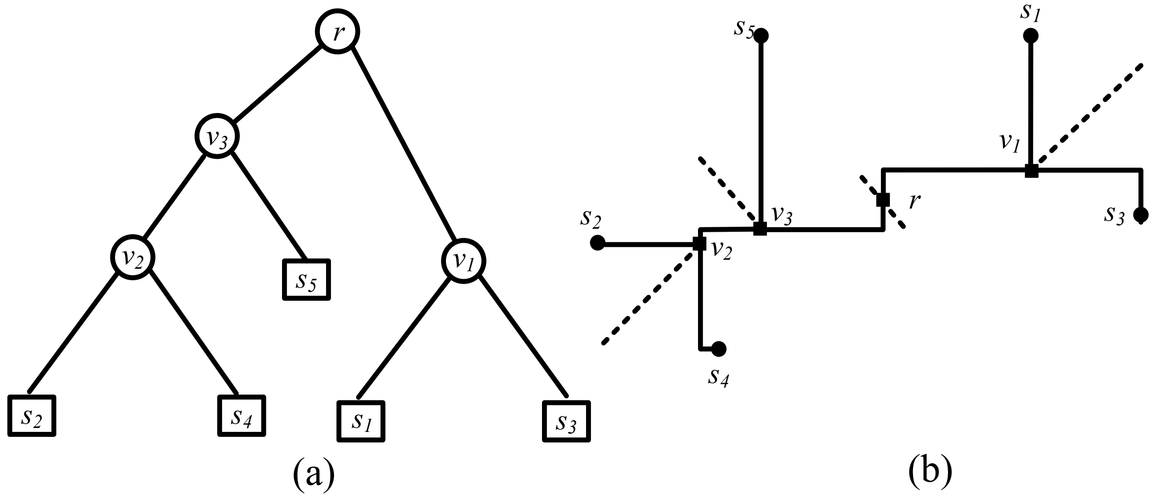


Figure 2-10 DME based clock tree construction on 5 sinks corresponding to the connection topology

Despite the theoretical advantages of DME based clock routers, they are not as influential as the traditional H-tree or mesh structures. In an effort to address some of the practical issues in DME, Kahng *et al* extended the traditional formulation so that the resulting algorithm can address obstacle avoidance and clock routing for varying layer parasitic with none–zero via resistance[28]. Assuming a particular routing pattern (HV) or (VH), they modified the original BST/DME[22] to consider the different parasitic parameters and via impedance in merging region determination phase. They assumed one time switching from horizontal to vertical layer for each edge (one bend). In their work each wire segment is assumed to be routed in L-shaped pattern, which means that their formulation fails if some wire segments are routed with Z-shape or with a direct connection (no-bend). Also their work is not guaranteed to achieve the optimum number of bends for a clock tree. In [28], 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 get the merging regions. However, the limitation of such a procedure is that it considers only parts of the merging regions. Consequently some possible Steiner points are eliminated, which could limit the capability of DME to reduce the wire length further. Haksu Kim *et al* suggested that the obstacle can be treated by devising a set of rules to go around it [29]. These rules are applied for a planar clock tree. However, the obstacle is not considered during merging segment determination. This implies that the impact of the obstacle on the total wire length has been neglected. More recently Haidar *et al* introduced the Shortest Path Polygon (SPP) to describe all shortest paths between two points in the presence of obstacles [30]. They incorporated SPP model in DME algorithm to handle the obstacles during the bottom-up merging segment. Some works have been proposed to deal with incremental engineering change in post clock distribution synthesis. Haidar *et al* suggested an incremental wire adjustment algorithm (AWA) to tune any given binary clock tree under any given skew bound [35]. AWA was developed based on the DME algorithm and converges quickly to the desired skew bound.

2.5 Buffer insertion clock network

Buffers are inserted in clock network to reduce the clock phase delay, and even more important, to satisfy the slew rate constraint. Delay of long wires, if un-buffered, can grow quickly and slew rates degrade rapidly. Nowadays clock distribution networks are huge capacitive interconnect networks which are almost impossible to operate at desired frequency without proper buffering. Clock tree buffering has many challenges that are different from general signal buffering. Instead of delay, clock skew is the primary objective. However, slew rates and power consumption are also important constraints. Clock networks are also typically much larger than signal nets, which makes the general buffering methods impractical to use in clock distribution network.

The fundamental dynamic programming paradigm used in general signal buffering is inadequate for clock trees. For clock trees, the optimal sub-structure property does not hold since an optimal solution does not necessarily have suboptimal solutions. Therefore many heuristic and greedy methods have been proposed for the clock tree buffer insertion problem. Zeng et al. proposed an algorithm that inserts a constant number of buffers on each clock path [31]. The buffers are moved and combined in a greedy fashion to improve the overall skew of the clock tree. In an approach presented by Xi and Dai, the iso-radius buffering was performed, where the radius is the wire distance of a buffer and the root [32]. Additional buffer levels are added until skew constraint is satisfied. Tellez and Sarrafzadeh proposed a greedy algorithm for zero skew buffer insertion and proved that the solution results in optimum number of buffer in a zero skew clock tree [33] . To control slew rate, a maximum capacitance limit is used to guide the insertion process. The algorithm traverse the clock tree in a bottom-up manner and uses two simple rules for node and edge buffer insertion:

1. **Edge Rule:** if the capacitance at the source of the edge is greater than the limit, split the edge and insert a node and buffer at the point on the edge nearest to the root such that the downstream capacitance does not violate the limit.
2. **Node Rule:** Add the necessary buffers to each sub-tree to equalize the number of buffer level in each sub-tree. If the capacitance at the node after these buffers are inserted is greater than the limit, drive each sub-tree with an additional buffer.

Tellez and Sarrafzadeh proved that their methodology inserts the minimum number of buffers such that the capacitance limit is not violated anywhere in the tree [33]. They also proposed a heuristic that allows an unequal, but limited, buffer skew. However, it was proven that their greedy bottom-up algorithm is not optimal for non-zero buffer skew bound [34].

Chapter 3

Via Aware Clock Trees

3.1 Vias in Clock distribution network

One of the major drawbacks associated with the available clock tree routers is the fact that they introduce too many vias. In deep sub-micron regime, vias not only affect the chip area, but also increase the resistance and reduce the circuit reliability due to electro-migration, signal echoing and process variations [39],[38],[8],[41]. Vias have a direct impact on interconnect routing because of their large physical footprint. Also, they implicitly aggravate blockages since vias may force interconnects to detour, which forces others to detour even further [40]. Via blockage is considered to be a pivotal factor that could limit multi-level metallization and chip miniaturization [40]. In multi-level interconnect technology, via failure due to electro-migration and variations in the metallization process are ever-increasing concerns, especially for high current density and wide-spanning nets such as clock networks. It has been shown that a via is generally more vulnerable to electro-migration, because the ampacity of a (tungsten) via is less than that of metal of the same width [41]. Hence, multiple vias are often used in parallel to improve reliability, but the more redundant the vias, the worse the area blockage problem. Excessive resistive vias and via failures can seriously degrade the system timing performance and they become more prevalent with the chip miniaturization. These via process variations manifest themselves in clock uncertainty which is totally undesirable for IC manufacturers.

The problem of via minimization in clock networks has not received considerable attention in academic research. Traditional via minimization algorithms are not suitable for reducing the number of vias in a clock network, since they modify the topology or layers of the nets and unbalance the clock tree [36]. Delay balancing is the main reason that such methods fail to work for clock networks. According to the author`s best knowledge, the only work that explicitly addresses the via minimization in clock routing was proposed by Chun-Hao et al [37]. They proposed *λ-Geometry* routing to improve wire length optimization. In their approach, metal layer ordering and layer assignment are used to minimize via cost in the *λ-geometry* DME algorithm. The algorithm does not work for Manhattan geometry preferred direction clock routing and fails to minimize via cost because it only considers the branches and realizes each clock edge with one bend. In fact they proposed only a bottom-up post-embedding phase to determine the best routing pattern for each segment without changing the embedding points.

3.2 Via-Minimization in Zero Skew clock Routing

Via minimization is done in the detail routing phase using layer assignment [36]. However, when the complexity of the circuit keeps increasing, it is more flexible and effective to minimize the number of vias as early as the global routing phase. Although planarity is desired for a large and critical net such as clock network; [42] due to circuit complexity and increasing demand for silicon area, no actual industrial product uses the planar approach for their clock network. Also, preferred direction layer routing is the most extensively used scheme for interconnect routing to reduce the impact of cross talk noise for adjacent layers [43]. This actually means even a simple clock distribution network should eventually be realized using different metal layers. In the preferred direction routing scheme, which is widely used in the industry, each wiring plane is used predominantly in either the X, or the Y direction. Corners and bends imply switching of layers, hence requiring the use of vias[9],[43],[44]. Therefore to reduce the number of vias, the number of bends for each wire needs to be controlled.

Note that the DME algorithm only determines the locations of the internal nodes, however the geometric layout of the edges are not specified. There are still certain degrees of freedom in top down embedding phase. The algorithm does not provide precise information of embedding an internal node when the algorithm finds a non-zero arc to embed the node. This freedom results in different routing quality and geometric pattern (i.e., number of bends). As shown in Figure 3.1, the point *p* is the embedding of the parent of the merging segment of the node corresponding to *v*, *ms(v)*. Embedding of point p has already been determined earlier in the top down pass. The DME is free to to choose embedding point of v to be at any point in *ms(v)* that is at the distance $| e_v |$ or less from the placement of v's parent P. Note that $|e_v|$ is the length of the edge that connects node v to its parent, P, and is calculated in bottom-up DME phase to preserve the zero skew merging property at each node. Three different scenarios are possible, such that each one results in different bends. This is shown in Figure3.1.

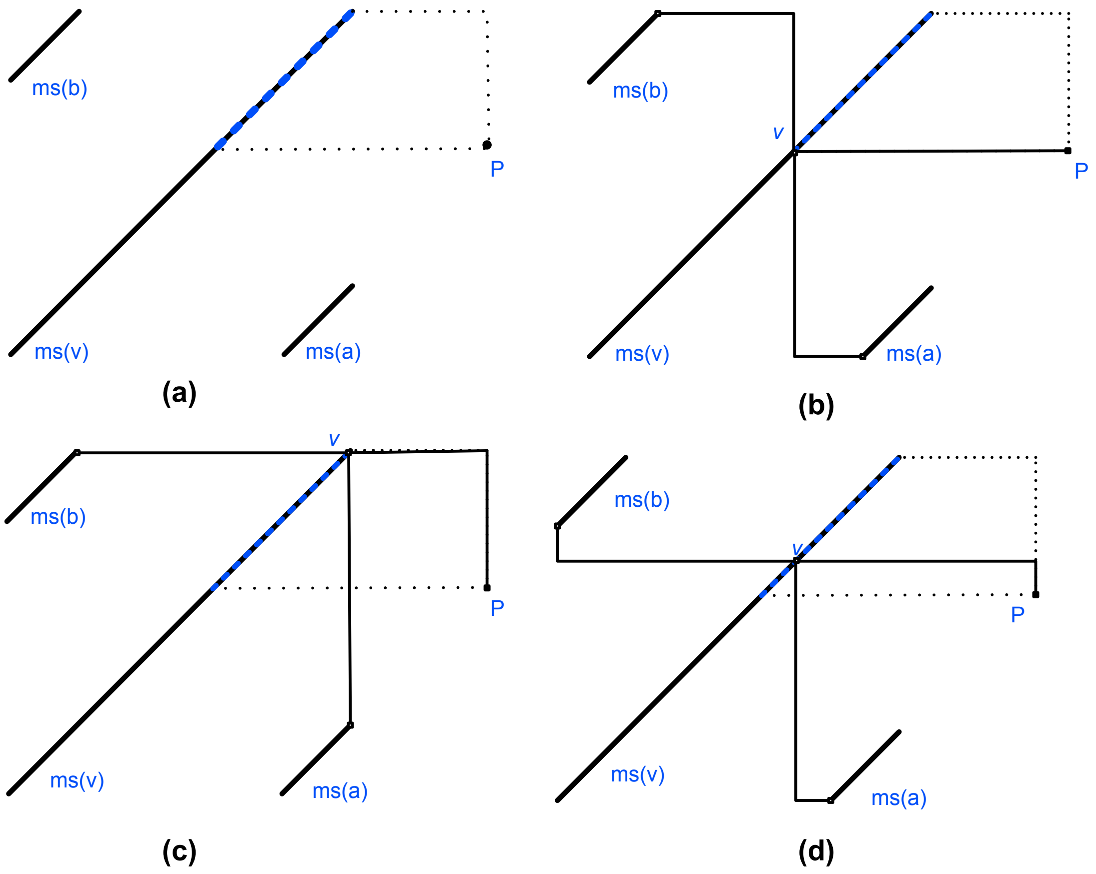


Figure 3-1 (a) A none zero arc which represents all possible loci of node v when its parent has already been determined, (b) Two-Bend embedding of node v, (c) one-Bend embedding of node v, (d) Three-Bend embedding of node v

3.2.1 Shortest Path Routing Region

A routing area that defines the potential routing region of two merging segments is called the *Shortest Path Routing Region*. The Shortest path routing region of two merging segments, l_1 and l_2, accommodates all minimum routing paths that connect the merging segments l_1 to the merging segment l_2. Shortest path routing region boundaries consist of either Manhattan arcs or rectilinear lines. This property is due to the fact that slope of merging segments can be only (-1) and (+1), therefore there are just four types of shortest path routing regions as illustrated in Figure 3.2. The zero skew merging segment, *ms*, corresponding to the arcs l_1 and l_2 is entirely located in shortest routing region of l_1 and l_2.

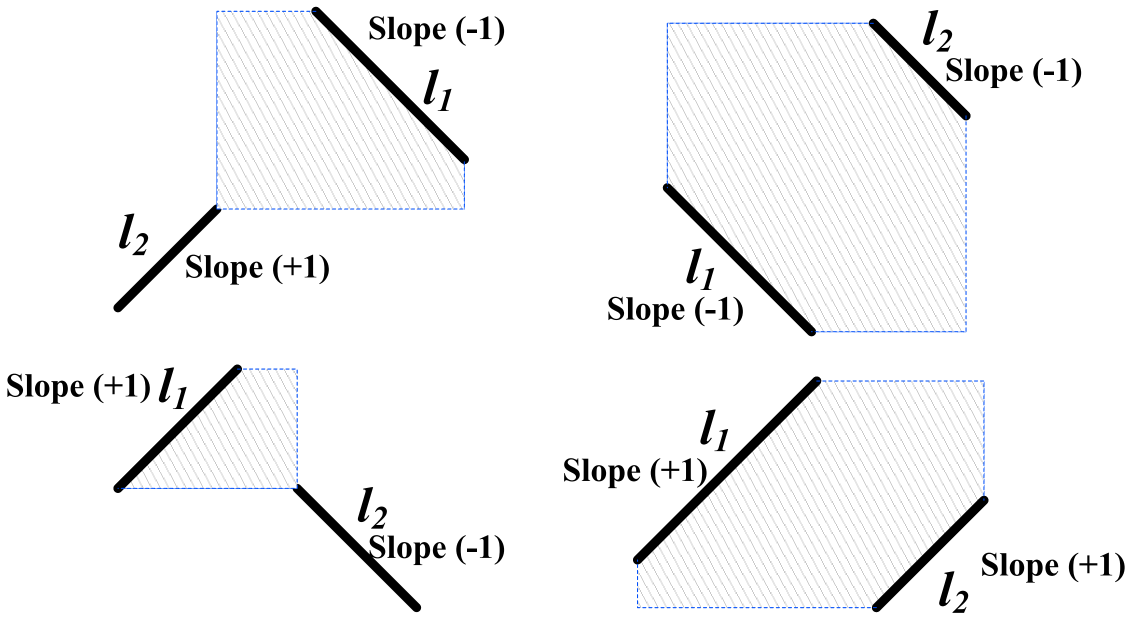
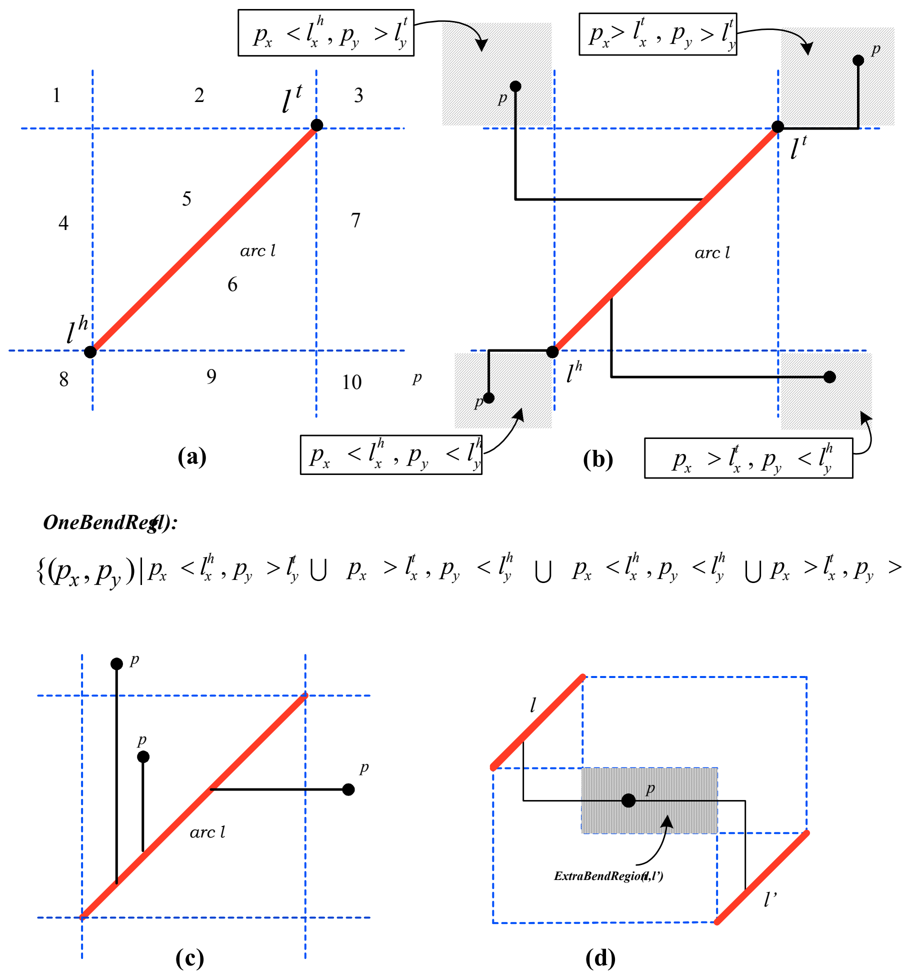


Figure 3-2 Four possible types of shortest path routing region due to four combinations of the slopes of the arcs, l1 and l2

Let *l* be a Manhattan arc with (-1) slope. The arc *l* can be uniquely identified by its two end points (head and tail). Assume l^h and l^t are the two points corresponding to head and tail points of the arc *l,* respectively. (l_x^h, l_y^h) and (l_x^t, l_y^t) are the coordinates of the head point, l^h, and the tail point, l^t. A point p defined by its x and y coordinates (p_x, p_y) is to be routed to the arc *l* with minimum wire length in Manhattan plane. In the Manhattan plane any wire can be routed using only rectilinear wire segments. The arc l divides the Manhattan plane into 10 regions as shown in Figure 3.3(a). Any point in the four shaded regions can only be routed to the arc *l* with at least one bend, as shown in Figure 3.3(b). The notation $\boldsymbol{OneBendReg(l)}$ refers to the union of all the four shaded regions corresponding to the arc *l*. On the other hand, any point that is not located in $OneBendReg(l)$ can simply be routed to the arc *l* without any bend as illustrated in Figure 3.3(c). Accordingly, the common area of $OneBendReg(l)$ and $OneBendReg(l')$, which will be called $\boldsymbol{ExtrabendRegion(l, l')}$, is the locus of the points that merge two arcs, $l\ and\ l'$, with minimum wire length using at least two bends. *ExtraBendRegion* for the arc *l* and l' is depicted by Figure 3.3(d).


$p_x < l_x^h, p_y > l_y^t$
$p_x > l_x^t, p_y > l_y^t$
1
2
3
4
5
6
7
8
9
10
l^t
l^h
arc l
p
$p_x < l_x^h, p_y < l_y^h$
$p_x > l_x^t, p_y < l_y^h$
(a)
(b)
(c)
(d)
l
l'
ExtraBendRegion(l,l')

OneBendReg(l):

$$\{(p_x, p_y) \mid p_x < l_x^h, p_y > l_y^t \cup p_x > l_x^t, p_y < l_y^h \cup p_x < l_x^h, p_y < l_y^h \cup p_x > l_x^t, p_y > l_y^t\}$$

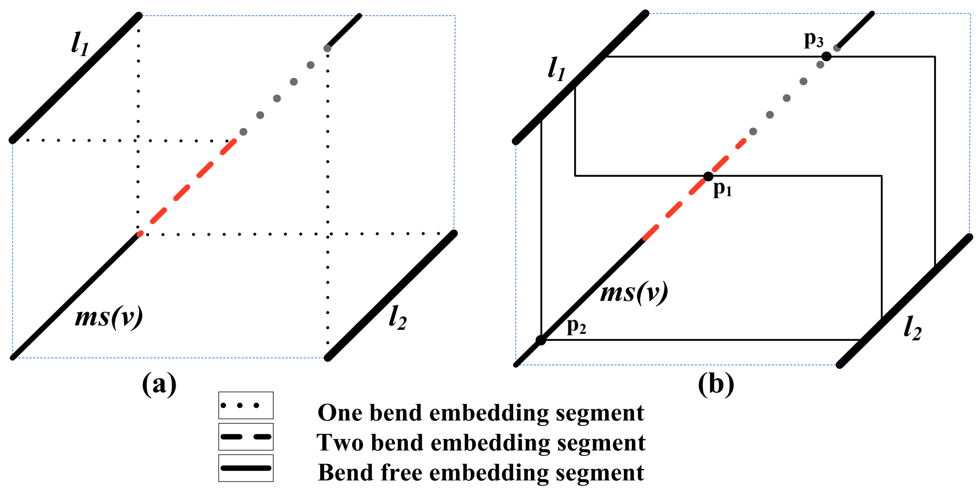
ExtraBendRegion(l,l')=OneBendReg(l) n OnebendReg (l')

Figure 3-3 One-Bend and Bend free routing scenarios for connecting a point p to a Manhattan arc with minimum Manhattan distance

3.2.2 Minimum Bend Zero Skew Tree

The proposed method for minimum bend zero skew clock trees is similar to the well-known DME approach; but during the bottom-up phase, the algorithm prunes away parts of the merging segments which may cause extra bends. In the top down phase, the algorithm is able to easily embed all the internal tapping points with minimum bend. During the bottom-up phase, the clock sinks are merged according to input topology. The merging determines the valid locations where the two sub-trees can join such that the new sub-tree has zero skew and minimum wire length. Every tapping point embedded on a merging segment has zero skew property. Therefore DME keeps the whole merging segment as the possible place for the tapping point. DME defers embedding the exact location of the tapping point until it completely constructs the tree of merging segments. In theory, keeping the whole merging segment leads to optimal wire length, but DME overlooks the difficulty that could arise later in the detail routing phase. Taking the whole merging segment as a locus of the tapping point is quite optimistic. Different parts in a merging segment may have different routing patterns and routing qualities (i.e, number of bends). In Figure 2-4(a), two arcs, l_1 and l_2, are merged into *ms(v)*, the zero skew merging segment. For any embedded point on *ms(v)*, three scenarios are possible. In each of them l_1 and l_2 are merged into a point on the arc, *ms(v)*, with minimum wire length, but each one has a different number of bends(Figure 3-4(b)).

Traditional DME does not distinguish between these embedding scenarios. To overcome this deficiency, we introduce *segment splitting scheme* to bind the embedding loci to be located in the regions which make fewer bends in the detail routing phase.



l_1
$ms(v)$
l_2
(a)
l_1
p_3
p_1
$ms(v)$
p_2
l_2
(b)
One bend embedding segment
Two bend embedding segment
Bend free embedding segment

Figure 3-4 One bend, two bend and bend free loci on a merging segment, ms(v).

3.2.3 Segment Splitting Subroutine

To embed all internal points with at most one-bend, the algorithm needs to construct the tree of merging segments that guarantee one-bend embedding in the top down phase. Segment Splitting subroutine serves to spilt a merging segment obtained in the ordinary DME into multiple segments. These new segments are entirely located in the regions that can accommodate one bend embedding in future steps. Instead of keeping the whole arc which results from the intersection of two merging segments, the algorithm splits the arc to smaller ones. The parts of the merging segment that make more than one bend are eliminated and the remaining parts are preserved for the next step.

Assume *l* and *l′* are two Manhattan arcs corresponding to two sibling nodes, a and b, in the clock tree topology. The algorithm computes the intersection of the arcs, *l* and *l′*, according to input topology similarly to the traditional DME. Assume *l* and *l′* are merged into *ms(v)*, where *v* is the parent of the nodes a and b. The algorithm determines shortest path routing region of the arcs l and l' and evaluates the *ExtrabendRegion(l,l')*. Based on the $ExtrabendRegion\ (l, l')$, the algorithm splits *ms(v)* into new merging segments and eliminates those ones that are located in the extra bend region. The remaining ones are inserted into a list corresponding to the node *v*. An example is illustrated in Figure 3.5. Unlike the traditional DME, in our approach each internal node is associated with a set of merging segments instead of one.

Let *MS (a)* and *MS (b)* be two sets of merging segments associated with the internal nodes a and b, respectively. The parent, say v, of a and b in clock tree topology has as many as $|MS(a)| \times |MS(b)|$ possible merging segments due to the merging of each segment $ms_i(a) \in MS(a)$ with each merging segment $ms_j(b) \in MS(b)$. The number of merging segments may grow exponentially during bottom-up construction of merging segments. To achieve an efficient implementation, we limit the number of merging segments of an internal node by a constant, say *k*. A simple *greedy* strategy for choosing the best *k* merging segments is to select *k* merging segments with the smallest total wire length. Taking advantage of the segment splitting scheme, the algorithm is capable of embedding all internal nodes with only one bend. Simple pseudo codes for the *Segment Splitting Scheme* are given in Figure 3.6 and 3.7.

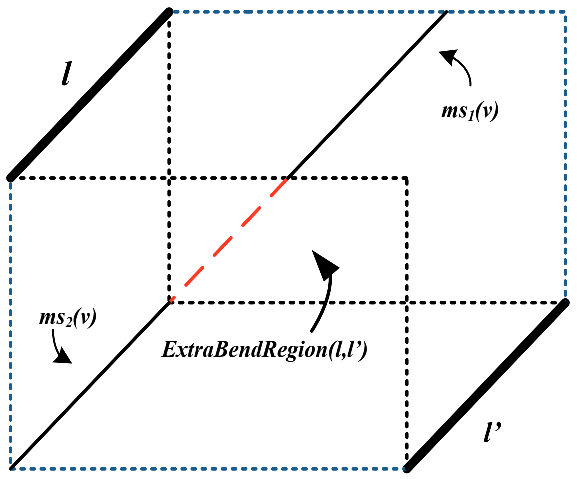


Figure 3-5 Bottom-up merging segment splitting according to ExtrabendRegion (l, l′)

```
Procedure: Modified_BottomUpTree_Construction ( A , B )

Input: Two Sets of Merging Segments A and B to be merged

Output : A Set of merging segment V

1. for each merging segment $a_i \in A$ and $b_j \in B$
      1.1 $v_r \leftarrow$ DME zero skew merging for $a_i$ and $b_j$
      1.2 $V \leftarrow$ MergingSplittingSegment ( $v_r$ , $a_i$ , $b_j$ )
2. Prune V such that it contains only *k* merging segments
   with smallest total wirelength.
```

Figure 3-6 Pseudo-algorithm for modified DME bottom-up merging segment construction

```
Procedure: MergingSegmentSplitting ( v , a , b )

Input: A Merging Segment v , its left and right child merging segment, a and b.

Output: A Set of merging segments, V , that contains all merging segments that can accommodate only one bend merging points.

1. ExtraBendRegin(a,b) ← OneBendReg(a) ∩ OneBendReg(b)
2. if v ∩ ExtraBendRegion (a , b) ≠ ∅ , then
2.1 Split the merging segment v into new segments $v_i$
2.2 For each $v_i$ do
        2.2.1 if $v_i \in$ ExtrabendRegion ( a , b ), eliminate $v_i$
              else V ← insert $v_i$
```

Figure 3-7 Pseudo-algorithm for Merging Segment Splitting subroutine

Lemma: Suppose that *a* and *b* are two sibling nodes in ZST (Zero Skew clock Tree) with the parent *v*. Let *ms(a)* and *ms(b)* be the merging segments corresponding to a and b. Also let *ms(v)* be the set of all placements which allow minimum merging cost within distance $|e_a|$ of *ms(a)* and $|e_b|$ of *ms(b)*. In the worst case, *Segment Splitting Scheme* is guaranteed to find a zero skew merging point, *v,* on *ms(v)* that can be routed rectilinearly to its child segments, *ms(a)* and *ms(b)*, with at most one bend.

Proof: without elongation d(ms(a), ms(b))= $|e_a| + |e_b|$, where d(ms(a), ms(b)) denotes the Manhattan distance between ms(a) and ms(b), therefore any wire that merge ms(a) and ms(b) with zero skew can be routed in the shortest path routing region of ms(a) and ms(b) which is always bound with either Manhattan arcs or rectilinear segments. A merging segment always crosses the rectilinear boundaries of the shortest path routing region at two points. In the worst case even if the merging segment totally located in two bend region of its corresponding child segments, embedding the parent merging segment on its end point results in only one bend merging.

Theorem 1*:* Given a set *S* containing n sinks and a binary tree connection topology G, if no elongation is needed, segment splitting algorithm produces a ZST *T* with at most n-1 bends.

Proof*:* Let G be a balanced binary tree topology and suppose that a and b are two sibling nodes in T with parent v. Due to Lemma 1, the segment splitting algorithm merges a and b into their parent v using at most one bend. The constructed tree of merging segments is always balanced since *ms(v)* is within distance $|e_a|$ of *ms(a)* and $|e_b|$ of *ms(b)* ($|e_a|$ and $|e_b|$ are calculated as in DME). Since T is a balanced binary tree with n leaves, therefore it has n-1 internal nodes and each internal node represents the merging of two siblings, therefore in the worst case clock tree has n-1 bends.

3.2.4 Simultaneous Wire Sizing

The proposed algorithm guarantees to merge two sibling nodes with at most one bend. In the clock tree generated by our method, every internal node is routed to its children with at most one L-Shaped pattern (one-bend route). The number of bends would be minimized further if a Steiner point in the clock tree could be connected to its both children with either a horizontal or a vertical wire segment. Hence, the embedding points must be taken from Steiner loci which can accommodate bend free merging points. A bend free merging point can be routed directly to both of its children with only one horizontal or one vertical segment. In a typical clock tree construction, frequently no

Steiner point is found to accommodate bend free merging. An example is given in Figure 3.8.

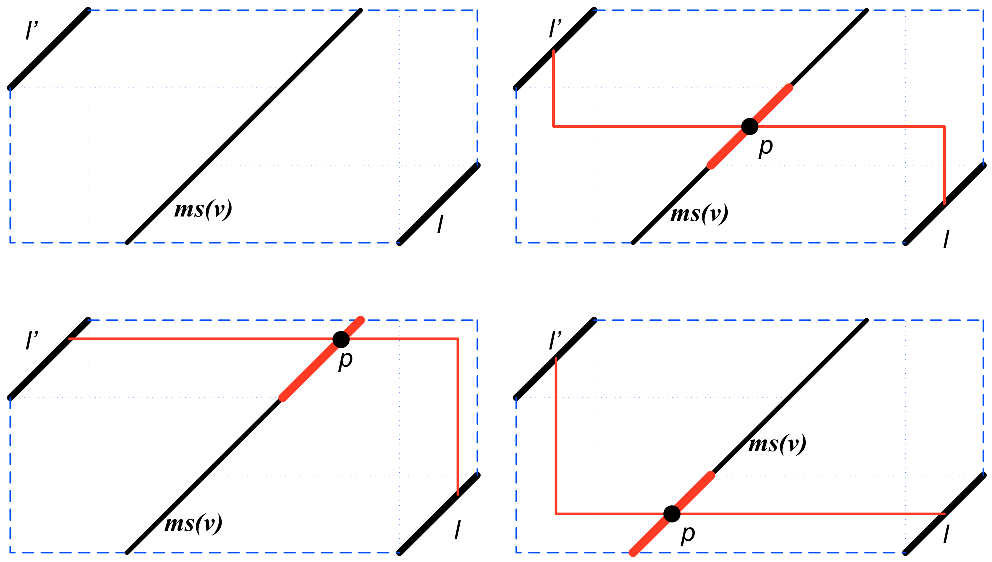


Figure 3-8 All possible embedding scenarios for a point, p, on a arc ms(v). All of them need at least one bend to be routed to both children arcs

The traditional DME algorithm does not consider the possibility of wire sizing (wire width) to obtain a better zero-skew clock tree. It has been shown in [23] that wire sizing can dramatically reduce the clock phase delay. When wire widths are considered as design variables, merging segments can be shifted by varying wire width. We take advantage of wire sizing to introduce a merging point tuning scheme to further reduce the number of bends.

Bend free merging region refers to a set of points in the shortest path routing region that can merge two arcs *l* and *l'* without any bends

Branch Tuning Subroutine: Let *ms(a)* and *ms(b)* be two merging segments corresponding to two internal nodes a and b in a clock tree topology. Assume *ms (a)* and *ms(b)* are merged into *ms (v)* where v is the parent of the node a and b. Note that *ms(v)* has already been determined using the Merging segment splitting scheme as explained in the previous section. Therefore *ms(v)* is the locus of the points that can merge two nodes, a and b, with zero skew using at most one-bend. To achieve bend free merging, the branch tuning subroutine determines the minimum shift required for *ms*(v) towards a bend free merging region. Accordingly $|l_a|$ and $|l_b|$ (edge length from the node v to a and to b) are updated and the algorithm will use the new edge's lengths to find the w_a and w_b (edge's wire widths for l_a and l_b) using π-model. Suppose that the branch tuning subroutine needs to shift the tapping point, *v,* towards one of the children of the node *v,* say "**a**" (left child). To maintain zero skew property at the branch point *v*, the right edge's width (w_b) need to be increased (Figure 3-9). The algorithm computes the required wire width for the right edge with the following equation:

$$w_b = \frac{l_b r_0 C(b)}{\left(t(a) - t(b)\right) + \frac{r_0 c_0}{2}\left(l_a{}^2 - l_b{}^2\right) + \frac{l_a r_0 C(a)}{w_a}}$$

Where l_a (l_b) and w_a (w_b) are the length and width of the wire from v to a (b), and r_0 and c_0 are the wire resistance and capacitance for an unit length and wire width. $t(a)$ and $t(b)$[1] are the clock delay from the nodes a and b to all leaves in the sub-tree rooted at a and b, respectively. Also, $C(a)$ denotes the downstream capacitance of the

[1] $t(a)$ and $t(b)$ are calculated using π-model similar to reference [25]

subtree rooted at a, and $C(b)$ denotes the downstream capacitance of the subtree rooted at b.

In order to reduce the phase delay, all the root-leaf paths need to be tapered. Hence, in the bottom up merging segment construction, the algorithm specifies a lower and upper bound for each edge's width. The wire width constraints for each clock tree edge at each level come from the successor levels (for the sake of proper wire tapering). Therefore, the algorithm attempts to tune the merging segment if the new edge wire widths comply with their feasible widths. We assume industry imposed restrictions for the minimum and maximum feasible wire width (W_{min} *and* $W_{\max}$). That is:

$$W_{min} \leq \text{ All wire widths in the clock Tree } \leq W_{max}$$

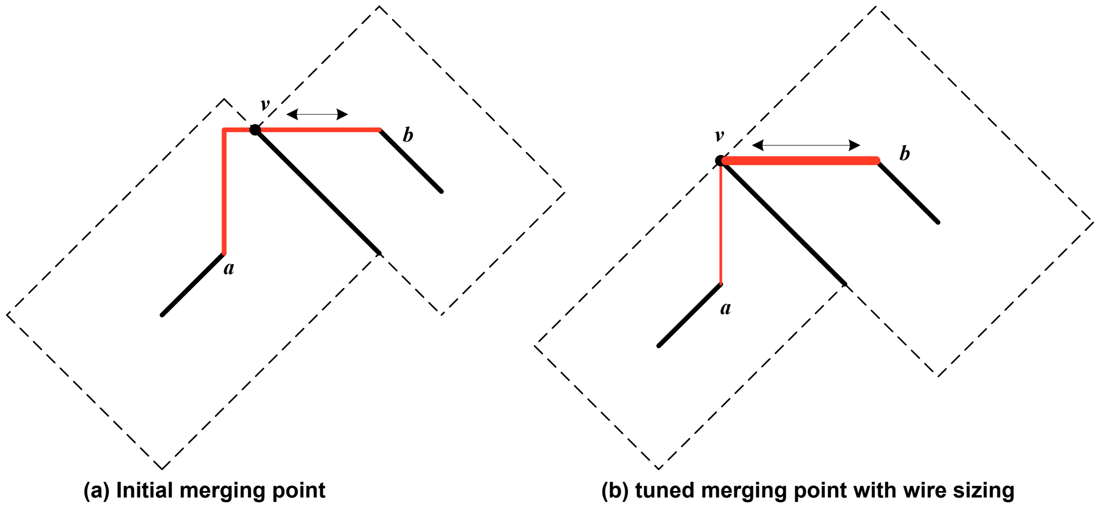

v
b
a
(a) Initial merging point
v
b
a
(b) tuned merging point with wire sizing

Figure 3-9 Branch point tuning to achieve a bend free merging

3.3 Experimental Results

We implemented the traditional DME and our modified DME on an initial topology obtained by the MMM algorithm. We quantified the number of bends in both the original and the modified DME. The benchmarks, r1 to r5, are taken from [25]. The per unit square wire resistance and capacitance used are 0.003 Ω and 0.02 fF respectively. The minimum and maximum feasible wire widths are 1μm and 4μm, respectively. For each benchmark we first quantified the total number of bends in the clock tree for the DME and our new approach. Total wire length and total capacitance are also measured for both algorithms. Tables 3.1, 3.2 and 3.3 show the results of the simulation for both DME and the modified algorithm. It can be seen that after applying the modifications all Steiner points merged to their two children with at most one-bend. The Results show almost 29 % reductions in "bend number" with only 3.4% increase in total capacitance which is practical since the clock tree is no longer routed using only minimum wire width from root to all leaves. Wire tapering reduces the root-leaf phase delay by almost 17.6% which can alleviate the impact of process variation on the clock skew. In this algorithm the tuned spilt merging segments no longer contain all possible loci of Steiner nodes therefore it does not guarantee to produce the optimal wire length in theory. Nevertheless, when combined with the greedy heuristic adopted, the modified DME algorithm has an increase of an average of 0.8 % wire length in all our experiments on the MMM initial topology while aggressively reducing the bend number.

Table 3-1 Simulation results for traditional DME

Benchmarks	r1	r2	r3	r4	r5
Number of clock pins	267	598	862	1903	3101
Total Wire Length ($\times 10^6$)	1.73	3.63	4.69	8.86	13.08
Total tree capacitance *(pF)*	34.41	69.96	89.86	175.16	263.97
Total bend number	309	694	1023	2293	3742
Two-Bend Branches	52	110	180	425	690
Clock Phase Delay (ns)	2.1	5.5	7.95	23.1	35.52

Table 3-2 Simulation results of the proposed algorithm

Benchmarks	r1	r2	r3	r4	r5
Number of clock pins	267	598	862	1903	3101
Total Wire Length($\times 10^6$)	1.744	3.66	4.76	8.91	13.11
Total tree capacitance *(pF)*	35.48	72.89	94.26	180.24	270.56
Total bend number	212	493	702	1607	2627
Two-BendBranches	0	0	0	0	0
Clock Phase Delay (ns)	1.71	4.66	7.21	18.17	27.06

Table 3-3 Percentage of changes in bend number, total capacitance and phase delay by using Modified DME

Benchmarks	r1	r2	r3	r4	r5
Number of Bend Reduction	27.1%	28.9%	31.3%	29.9%	29.7%
Phase delay Reduction	18.5%	15.2%	9.2%	21.3%	23.8%
Total Capacitance Increase	3.1%	4.2%	4.8%	2.9%	2.4%

Chapter 4

Pattern-Driven Zero Skew Clock Tree Router

The preferred direction routing scheme with two metal layers (horizontal and vertical) is assumed in the context of this chapter. Therefore reducing the number of bends directly minimizes the number of vias. To control the number of bends associated with wire segments, routes are considered to be of L-pattern. Pattern routing would also increase the predictability of the clock routing such that the final clock tree layout specification would not be seriously deviated from the expected one. A clock tree with 1000s of sinks in a local region could be very dense in some sections. Hence, routing of clock tree wiring with L-pattern significantly increases the routing constraint violation and the overflowed routing tracks.

Usually few metal layers are reserved for the clock net and clock routing is done before routing of data and control signals. Therefore clock routing would not be affected by the routing demands of other nets. In the context of this chapter, routing demand and congestion are referred to those of the clock tree itself.

We consider the pattern routing demand of the clock tree edges in three major clock tree construction steps i.e. topology generation, balanced delay merging (merging segment construction), and branch point embedding. A demand aware merging selection technique is proposed to distribute the routing demand to avoid overly congested routing areas. To do this we introduce a new merging cost function that includes a combination of the wire length and the expected routing demand.

Our methodology is able to:

1- Efficiently reduce the number of vias in clock trees.

2- Render more uniform distributions of the via throughout the clock trees.

3- Reduce the total clock tree wire length by avoiding unnecessary detours.

4- Minimize the skew variation after CT (clock tree) detailed routing.

Our algorithm is the first of its kind in closing the gap between clock tree synthesis, CTS, and actual clock layout which is usually ignored in most of academic clock routing algorithm.

4.1 Sampled Base Zero skew Merging Segment Construction

In traditional zero-skew merging, the whole merging segment is kept as possible places for the tapping point. To find the exact placement for tapping points, traditional zero skew clock tree algorithms defer the decision of choosing where to merge the two sub-trees until the parent merging point is decided. In theory, keeping the whole merging segment leads to optimal wire length, but it overlooks the difficulty of detail routing.

Different parts in a merging segment may have different routing patterns and routing qualities.

To capture the difference of parts in merging segment and propagate it to the upper level we exploit a sampled-base implementation [11] of traditional DME. Instead of computing the intersection of Manhattan arcs, collections of merging points (MP) are maintained that represent each merging segment. The set of merging points that represent a merging segment is referred to as merging point set (MPS). To calculate a MPS, the traditional merging segment is computed for all the closet pairs of merging points (MP) and then sampled. When there are more than two closest pairs of MPs, the union of all the resulting zero-skew MPs is taken and pruned. An example of sampled DME closest pair merging is shown in the Figure 4-1.

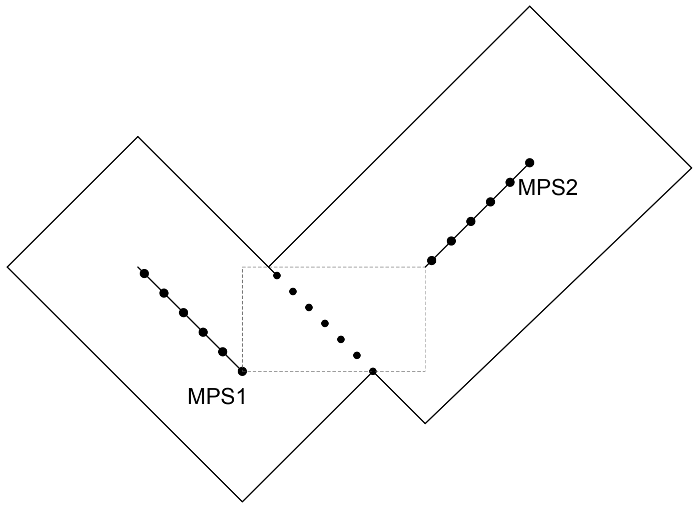


Figure 4-1 An example of sampled merging segment

The pruning divides the area into proximity grid of the user specified resolution and each MP is allocated to the appropriate grid location. Any single MP in a grid can replace all other MPs, therefore pruning only maintains a linear number of samples for each merging segment. It is important to note that since the MP calculation is pair wise, many of the resulting merging segments will overlap. As there are multiple closest pairs of MPs, there are potentially multiple equally viable minimum wire length zero skew solutions as illustrated in Figure 4-2. In the deterministic case these solutions are identical. But they might have different result on the other routing criteria (i.e. congestion).

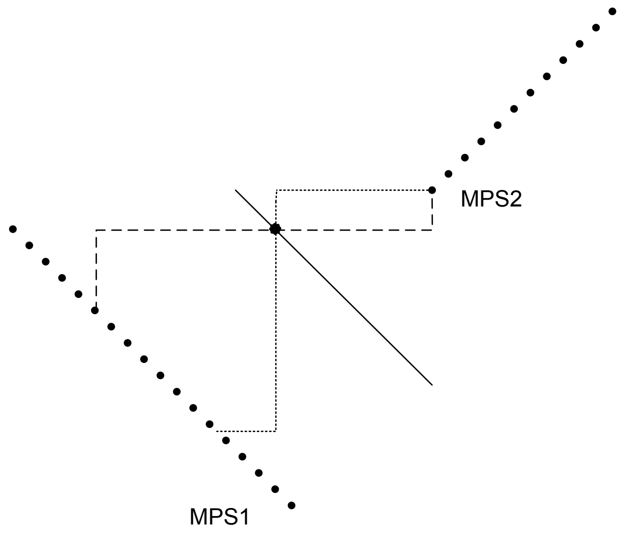


Figure 4-2 Two potential embedding scenarios in sampled DME

We intend to address the congestion driven clock routing with sampled- merging segment scheme. Unlike the traditional merging segment, in the sampled merging segment approach we can treat each sample separately and analyze the likelihood of congestion for each embedding solution.

4.2 Demand graph and probabilistic demand estimation

The entire routing region is tessellated into array of rectangular tiles (bins). The tiles are represented as a graph (demand graph) referred to as G(V_g,E_g), where V_g= {g_1 , g_2,...} corresponds to the set of grid cells, and edge $e_{ij} \in E_g$ corresponds to the boundary between two adjacent grid cells g_i, $g_j \in V_g$. The graph of Figure 4-3 shows a demand graph which serves to maintain routing demand information of a 3×4 tessellated routing plane.

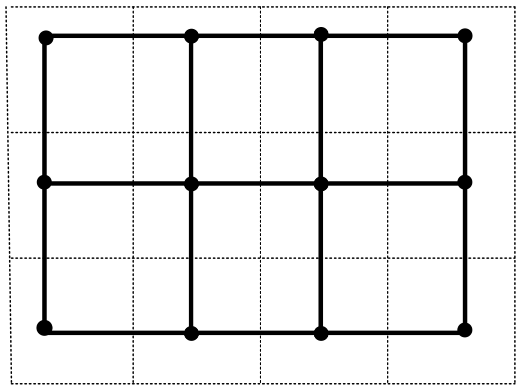


Figure 4-3 Tessellated routing plane and its graph representation

Since routing is performed in the Manhattan geometry, all wire segments must be either horizontal or vertical. For two points, *s* and *t,* in a Manhattan plane, all minimum distance wires that connect *s* to *t* are located in a rectangular *bounding box*, referred as $\overline{st}$, where *s* and *t* are the two opposite vertices of the bounding box. More generally a routing area that defines the potential routing region of two Manhattan arcs or a Manhattan arc and a point is also called the *Bounding box*. The bounding box of two Manhattan arcs l_1 and l_2, denoted as $\overline{l_1 l_2}$, accommodates all minimum routing paths that connect the arc l_1 to the arc l_2.Examples of the bounding box are illustrated in Figure 4-4(a) and 4-4(b).

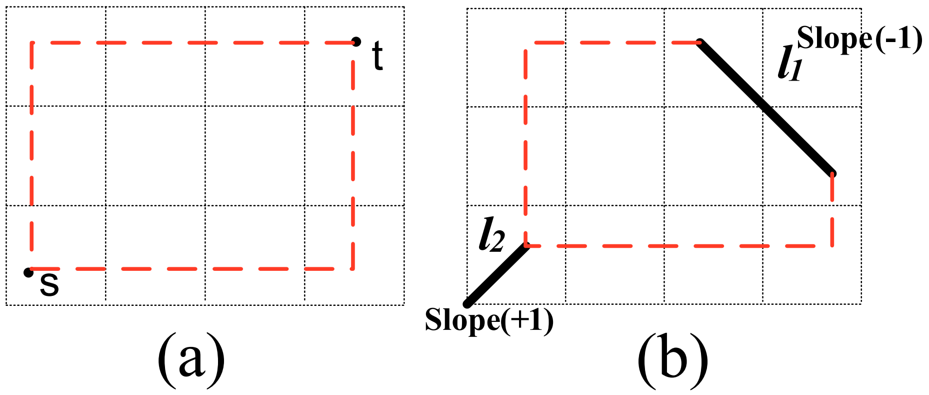


Figure 4-4 Routing bounding boxes in a tessellated plane

Definition: In the demand graph, the number of wires traveling along a graph edge b is designated as the demand, *d*(b).

The demand definition is based on deterministic routing path, which cannot help routers when the routing paths are not fixed. During the clock tree construction, the wiring paths are not determined. Therefore all paths can only be specified roughly with bounding boxes. We need to use a non-deterministic method to evaluate the demand in routing bins. For example, in our nondeterministic method, a path from *s* to *t* is replaced by its equivalent bounding box $\overline{st}$. Unlike the demand definition which is based on the deterministic wire routes, we will use the probabilistic demand to indicate the possibility of wires running along a graph edge. To control the number of bends (vias) associated with any wire segment, routes are considered to be of L-shape. In our probabilistic demand estimation, a uniform probability distribution is assumed for all routes in a bounding box. In other words, every route has the same likelihood to be chosen in later stages. Therefore we define the probabilistic demand for a L-edge $\overline{st}$ to a demand graph edge b to be the probability that the L-edge run along this edge, and is referred as $d_{prob}(b, \overline{st})$. For example in Figure 4-5, $d_{prob}(b, \overline{st})$for every edge b in the boundary of the bounding box will be 0.5 and all internal edges have demand equal to zero.

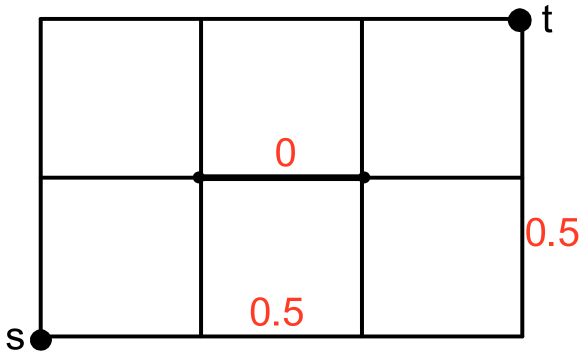


Figure 4-5 Demand graph of the (s,t) bounding box. Probabilistic demands for L-pattern routing are designated

4.3 Congestion-driven zero-skew clock tree routing

Given a set of clock sinks S= {s_1,s_2,…,s_n}, the congestion-driven zero skew clock router constructs a minimal wire length tree to connect the sinks according to a topology. The delay from the source to any sink in the tree is nominally zero according to the chosen delay model. The synthesized clock tree can be routed with a pattern router with minimum number of routing demand violation. The clock tree can be constructed incrementally in a bottom-up fashion.

Let K be a node set initialized by the leaf set S. At each iteration of the algorithm, two nodes v_1 and v_2 are taken from K, the parent node v is calculated using the zero-skew merge describe in chapter 3, and v is added to K. The algorithm repeatedly selects two leaves or merging segments and merges them. It is obvious that different zero skew clock trees are obtained if different strategy is taken for the selection.

4.3.1 Selection strategy

To our purpose an appropriate selection strategy is devised for minimal total wire length and expected routing demand of the tree edges. The merging cost of a pair of candidates, v_1 and v_2, comprises of the wire length increase cost and routing demand cost. If there is no detour, the total wire length increase is calculated by $d(v_1,v_2)$, the Manhattan distance between v_1 and v_2. The expected routing demand of prospective merging candidates is computed with the nondeterministic method.

4.3.1.1 Probabilistic demand calculation for a pair of candidates v_1 and v_2

The routing demand of the merging is evaluated by the bounding box $\overline{v_1 v_2}$, since the path that merges the two candidates is not determined.

A. *Merging two points*

In the tessellated routing plane in Figure 4-6(a), consider the L-pattern route from the point v_1 at to the other point v_2. Figure 4-6(b) shows the corresponding demand graph of the bounding box of the paths from v_1 to v_2. The routing in the bounding box now turns into finding shortest path from the node v_1 to the node v_2 in the equivalent demand graph. Assuming no obstacles between v_1 and v_2, there are two shortest routing paths with L pattern that can connect the point v_1 to the point v_2. The L-shape paths from v_1 to v_2 have equal chance to occur. Hence the probabilistic demand,d_{prob}, for any horizontal and vertical edge in the demand graph is derived as follow:

$$d_{prob}(b,\overline{v_1 v_2}) = \begin{cases} 0.5 & if\, b\ is\ boundary edge \\ 0 & otherwise \end{cases}$$

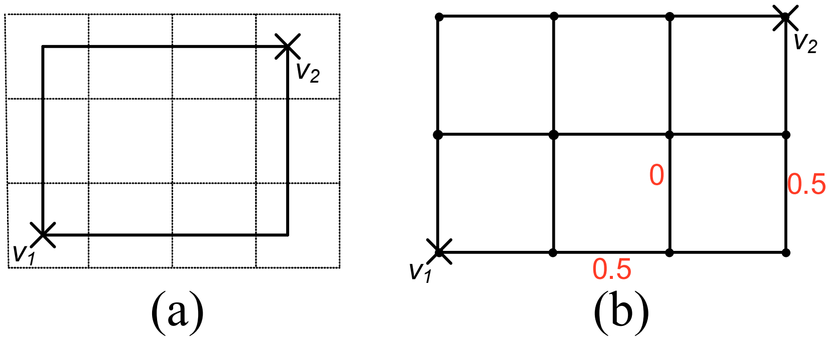


Figure 4-6 Demand calculation for two points

B. *Merging a point and a Manhattan arc*

Consider a L-pattern route from a point v_1 at bin (0,0) to an arc l, which has its head and tail points at the bins(h^x, h^y) and (t^x, t^y) respectively. In order to calculate the probabilistic demand for bin edges in the corresponding bounding box, first we sample the arc l in such a way that there is at least one sample in every bin that is encountered by the arc. All the sample points have the same chance to be connected to the point within the bounding box. The probability of choosing each sample is $\frac{1}{|t^x - h^x + 1|}$. The probabilistic demand for every bin edge b is equivalent to the following summation on all of the sample points on the arc l.

$$\sum_{\forall p \in l} \frac{1}{|t^x - h^x + 1|} \times d_{prob}(\, b, \overline{v_1 p}\,)$$

Where $d_{prob}(\, b, \overline{v_1 p}\,)$ is the probabilistic demand of the bin boundary b due to the bounding box $\overline{v_1 p}$ (bounding box of the route from v_l to p). Since $d_{prob}(\, b, \overline{sp}\,)$ is either 0 or 0.5 and can be computed in $O(1)$ for every bin boundary, the probabilistic demand for every bin boundary can be computed in $O(n)$, where n is the number of samples in the arc l. An example is illustrated in Figure 4-7. An arc l in a tessellated routing plane is given and h(5,1) and t(2,4) are the coordinates of the bins where the arc head and tail points are located. The arc is sampled as depicted in Figure 4-7(a). Each sample corresponds to a bin in the routing plane. The related demand graph is shown in Figure 4-7(b).The total number of possible L-shape routes from s to the arc l is equal to 8 and the probabilistic demands for the graph edges a and b are $\frac{1}{8}$ and $\frac{3}{8}$ respectively.

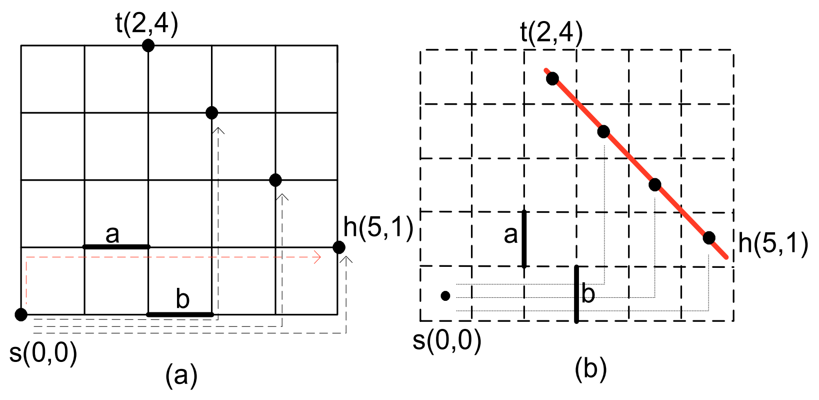


Figure 4-7 Probabilistic demands for a point-arc bounding box

C. *Merging two Manhattan arcs*

Assume l and l' are two Manhattan arcs in our tessellated routing plane. Let n and m be the number of samples in the arc l and l' respectively. The probabilistic demand of every bin edge b, in the corresponding bounding box $\overline{ll'}$ is computed with the following summation on all $\overline{p_1p_2}$, where p_1 and p_2 are samples on the arc l and l'.

$$\sum_{\overline{p_1p_2}} \frac{1}{n \times m} \times d_{prob}(b, \overline{p_1p_2})$$

4.3.2 Merging Selection Cost

The merging cost of a pair of candidates, v_1 and v_2, comprises of the wire length increase cost and routing demand cost. The average routing demand of v_1 and v_2, denoted by $ARD(v_1, v_2)$, refers to the average of the probabilistic demand of all edges involved in the bounding box of $\overline{v_1v_2}$.

We define the merging selection cost of v_1 and v_2 as $MC(v_1, v_2) = \alpha \times d(v_1,v_2) + \beta \times k \times ARD(v_1, v_2)$, where $d(v_1,v_2)$ is the Manhattan distance of v_1 and v_2, α and β are the user defined weights of the wire length and demand overhead respectively, and k is the normalization factor for ARD. We assume that during the merging selection process, each node can be merged to only one of its nearest neighbors. Merging far away nodes would likely increase the total tree wire length. We assume a constant N to denote the number of the nearest neighbors we consider for each node.

The nearest neighbor graph, NG, is constructed to maintain the nearest neighbor candidates. Each merging segment (or a point) corresponds to a node in the graph. An edge e_{ij} exists between two vertices v_i and v_j if v_i is among the three nearest neighbors of v_j or v_i is one of the three nearest neighbors of v_j. Every edge in the nearest neighbor graph implies a possible parent merging segment for two neighbors. Weight of an edge

e_{ij} in the nearest neighbor graph is equal to $MC(v_i, v_j)$. An example is shown in Figure 4-8.

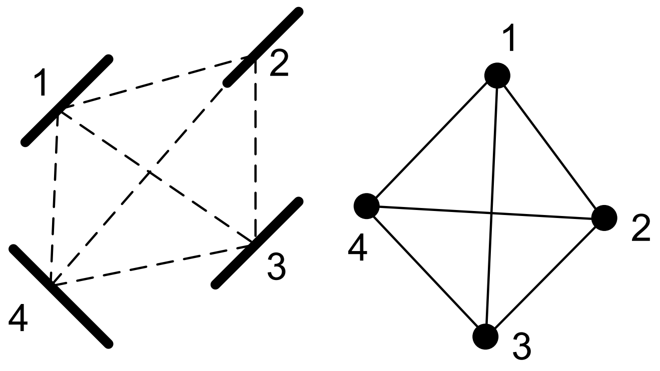


Figure 4-8 Nearest neighbour graph for N=3

4.4 The Algorithm

The pseudo-code for the pattern-driven zero skew clock routing is shown in Figure 4-9. The algorithm repeatedly selects two leaves or merging segments and merges them. At each iteration of the algorithm, two nodes v_i and v_j are taken from K. The selection is made using the nearest neighbor graph build from K and weighted with the new cost function. The parent node v is calculated using the zero-skew merge. The merging segment v is then sampled and added to K. Since v is the parent segment of v_i and v_j , two bounding boxes $\overline{v_i v}$ and $\overline{v v_j}$ can be determined. The demand graph edges involved in $\overline{v_i v}$ and $\overline{v v_j}$ will be updated. After n-1 iterations the algorithm reaches the root.

```
Algorithm  Pattern Driven Zero Skew Clock Tree Algorithm
Input: A set of sinks $S$={ $s_1$,$s_2$,...,$s_n$}
Output: A Zero skew tree that can be L-pattern routed with
          minimum overflowed tracks.
1: Tessellate the routing plane into N×N bins
2: Create the routing demand graph, DG($V_{DG}$,$E_{DG}$), where $V_{DG}$ corresponds to  bins in
   the tessellated routing region, $|V_{DG}|$=$N^2$,
3: Initialize G($V_{DG}$,$E_{DG}$) by assigning the sink positions  to nodes and edge weights to
   zero
4: K:=S
5: If |K|=1, Stop,  else do
6:      Create the nearest neighbor graph, NG($V_{NG}$, $E_{NG}$),  on K
         where  $|V_{NG}|$=|K| , $E_{NG}$={ e($v_i$, $v_j$) | $v_i$ is among the three
        nearest neighbors of $v_j$ or $v_i$ is one of the three
        nearest neighbors of $v_j$}
7:      Apply the weight function W: $E_{NG}$→R such that
        $w$(e($v_i$, $v_j$) )= α × $d(v_i, v_j)$ +$\beta \times ARD(v_i, v_j)$.
8:      Select the smallest  e($v_i$, $v_j$) in $E_{NG}$
9:      Calculate the parent segment, $v_p$, from $v_i$ and  $v_j$ using
        the zero skew merge
10:     sample the merging segment of $v_p$ according to our
        define grid resolution
11:     Update the all edges in demand graph which are
        involved in $\overline{v_i v}$  and $\overline{v v_j}$
12:     Remove $v_i$ and  $v_j$ from K and add $v$ to K
13:     Go to Step 5
```

Figure 4-9 Procedure of merging selection

The algorithm in Figure 4-9, defers embedding the branch points until the clock tree root is reached. During the bottom-up tree construction, every Steiner branch is represented by the *sampled* merging segment. A sampled merging segment maintains a collection of equally viable minimum wire length zero skew solutions. But these different solutions may have different impacts in routing demands. For example in Figure 4-10, embedding the parent branch point on the sample *s* or *v* in the merging segment would achieve minimum wire and zero skew tree. However, in Figure 4-10(a) the probabilistic demands of all bin edges involved in $\overline{av}$ and $\overline{vb}$ are $\frac{1}{2}$ but in Figure 4-10(b) the demands of bin edges involved in $\overline{sb}$ are equal to 1. This implies that the embedding solution of the parent branch point on *v* would be more flexible and reduces the chance of overflow.

The maximum probabilistic routing demand of bin edges involved in a bounding box $\overline{st}$ is called *peak demand of* the bounding box denoted as $pd(\overline{st})$. In Figure 4-10(b), $pd(\overline{as})$ and $pd(\overline{sb})$ are equal to $\frac{1}{2}$ and 1 respectively. We introduce a parameter called *accumulative peak demand*, Adp , for every sample point in the merging segments. Adp of a sample point in a merging segment is calculated incrementally from its children during the bottom-up tree construction. In Figure 2-10(b), $Adp(s)$ can be obtained by:

$$Adp(s) = Max\left\{(pd(\overline{as}) + Adp(a)), \left(pd\left(\overline{sb}\right) + Adp(b)\right)\right\}$$

The procedure illustrated in Figure 4-9 computes the parent merging segment for all the closest merging point pairs between two candidates. For each of these pairs, the traditional merging segment is computed and sampled. When there are more than one closest pairs, like s_1 and s_2 in Figures 4-10(c) and 4-10(d), some of the samples in the parent merging segment may overlap. In Figures 4-10(c) and 4-10(d) the merging point *v* is obtained from two pairs of points (s_1,b) and (s_2,b). The *accumulative peak demand* of the sample point *v* is calculated from s_1 or s_2 which ever minimizes the Adp(v).

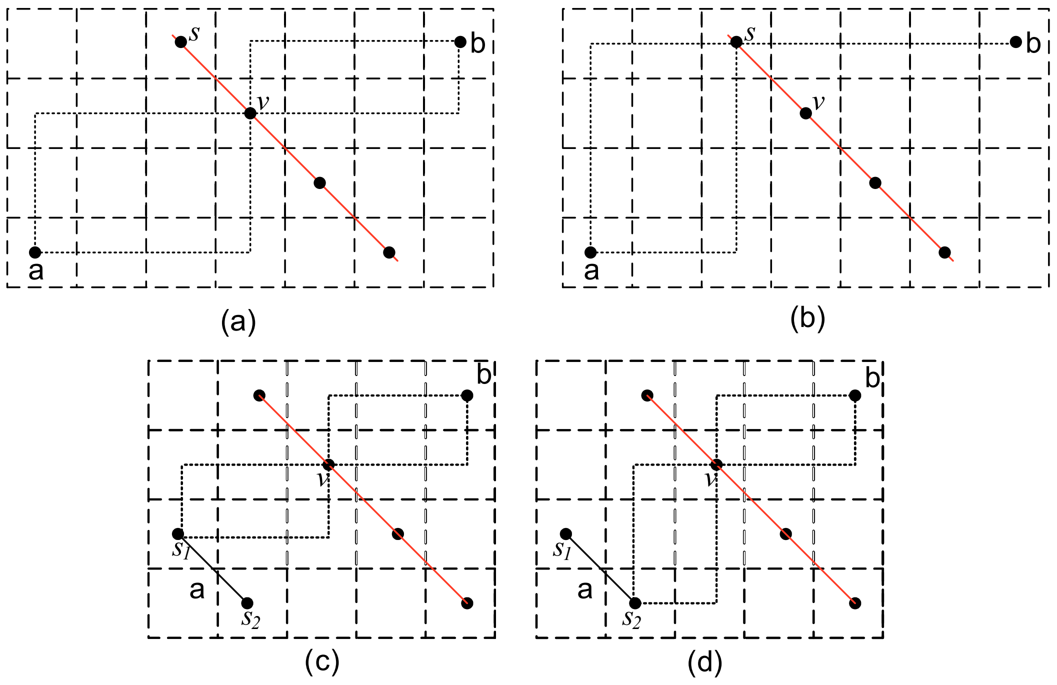


Figure 4-10 Choosing the sample point with minimum peak demand

In the bottom-up tree construction, *accumulative peak demand* values are calculated incrementally for every sample in the tree. Each sample point remembers from which points it was constructed; therefore it can be embedded in the top-down phase. When multiple embedding solutions are viable, the algorithm chooses a point that has a smaller *Adp*.

4.5 Experimental results

The pattern driven clock routing algorithm was implemented in GNU C++ and executed on a 2.00 Ghz Intel machine running Linux. We implemented the Greedy-DME[27] which does not require any input topology to have a fair comparison, since in our method the connection topology is also determined during the merging segment construction with regard to demand map.

In the benchmarks, r1 to r5, the per unit wire resistance and capacitance used are 0.003 Ω and 0.02 fF respectively. Our wire delay computation is based on Elmore model but it can be easily extended to a more accurate model. The die size is $4 \times 4\ cm^2$ and the grid resolution was chosen so that there are 500 bins in both horizontal and vertical direction.

We used two sets of parameter for the selection cost function as described in section 3.2, ($\alpha = 1, \beta = 0$) and ($\alpha = 0.7, \beta = 0.3$). The former set of parameters is chosen to investigate the sub-optimality caused by sampling. Table 4-1 illustrates the total clock tree wire length and run time for the synthesized abstract clock trees. Any sample point may not be in the optimal location. Repeated sub-optimality during the bottom-up phase can result in accumulated drift of a merging segment from its optimal location. But as the experimental results show, the total sub-optimality is not significant (around 0.2%). Changing the weight factors to favor the tree wiring congestion can slightly increase the average wire length.

The synthesized clock tree only includes the connection topology and the branches. We applied a two-pin maze-router to complete the clock tree layout. The maze router uses bends and detours to realize the clock tree layout. The clock tree specification changed significantly from its nominal values in Table 4-1.

Table 4-1 Wire length and run time comparison on zero skew abstract clock trees obtained from benchmarks r1-r5

		Greedy DME[19]		The new algorithm $\alpha = 1 , \beta = 0$		The new algorithm $\alpha = 0.7 , = 0.3$	
Bench marks	# Sinks	Wire length (µm) $\times 10^6$	Run time (s)	Wire length (µm) $\times 10^6$	Run time (s)	Wire length (µm) $\times 10^6$	Run time (s)
r1	267	1.43	0.9	1.43	12.1	1.51	61
r2	598	3.03	4.6	3.031	27.3	3.05	120
r3	862	4.052	9.72	4.052	41.5	4.1	223
r4	1903	7.23	48.5	7.24	107.3	7.26	556
r5	3101	12.8	127.6	12.93	289.3	13.21	1065

The total wire length, the number of vias (bends), the number of overflows are quantified for the maze-routed clock trees in Table 4-3. Both algorithms (Greedy-DME and our pattern driven clock router) synthesize nominal zero skew clock trees, but the skew specification could change dramatically by completing the clock tree layouts with the maze router. The post routing skew for the benchmarks r1-r5 is illustrated in Table 4-2. The proposed pattern-driven clock tree construction method is able to increase the predictability of the routing paths in a clock tree which reduces the unpredicted post routing skew (Table 4-2).

Table 4-2 Wire length and run time comparison on zero skew abstract clock trees obtained from benchmarks r1-r5

	r1	r2	r3	r4	r5
Greedy-DME	18(ps)	38(ps)	124(ps)	130(ps)	380(ps)
The new algorithm	24(ps)	43(ps)	55(ps)	21(ps)	117(ps)

The total wire length of the clock trees also increases compared to the nominal values before the maze routing. Table 4-3 compares the total wire length and the numbers of bends for two the two different algorithms after being routed by the maze router. The maze routing generally favors the total wire length but it can compromise the quality of the routing solution by increasing the number of vias (bends). This is clearly shown in Table 4-3. The proposed pattern driven clock construction technique reduces the number of bends (vias) by around 26% in the final clock tree layouts. The results indicate about 6% reduction in wire length.

For the second experiment, the results of our pattern driven clock routing algorithm were compared with the Greedy-DME generated solutions by feeding the synthesized abstract trees to a pattern router, Labyrinth [46]. Labyrinth, an academic detail router which is available at http://www.ece.ucsb.edu/~kastner/labyrinth/, was used because it is capable of performing the pattern routing on an input net list. Labyrinth splits the clock tree into two terminal nets, and sorts them from smallest to largest bonding box. We have the Labyrinth to pattern route(L-shape) 80% of the nets with the smallest bounding boxes while maze routing the rest. The pattern routed net remain unchanged during the rip up and re route iterations. As we expected, the Labyrinth (L-pattern routing) favors the number of bends when it routes the greedy-DME generated trees, but wire length and overflows are increased. On the other hand, in our proposed method, the routing demand impact of the L-pattern routing was taken in to account in the clock construction phase. The results in Table 4-4 show that the pattern driven clock tree layouts have almost 28% less bends (vias) than the greedy-DME when both are routed using Labyrinth. Our algorithm reduces the wire length by an average of 8% compare to greedy-DME. One of the major advantages of our algorithm is the fact that it produces a clock tree which is less complicated to route. The number of overflow is a rather negligible problem when maze routing the clock tree, as illustrated in Table 4-3. Applying the L-pattern routing (Labyrinth) increases the number of overflow to a level that cannot be ignored. Clock routing completion rate would reduce when the number of overflows increases, because it would exacerbate the burden of layout. Table 4-4 also shows that the number of overflows is reduced dramatically with labyrinth. The results in Table 4-4 also indicate that our pattern driven clock tree layouts have almost 74% less overflows than the greedy-DME when both are routed using the Labyrinth.

Table 4-3 Wire length and run time comparison on zero skew abstract clock trees obtained from benchmarks r1-r5

Benchmarks	# Sinks	Greedy-DME/ Maze route			Our new Algorithm / Maze route			Improvement		
		Wire length (µm) $\times 10^6$	# Bend vias	# Overflows	Wire length (µm) $\times 10^6$	# Bend vias	# Overflows	Wire length Cost (%)	# Bend (%)	#Overflows (%)
r1	267	1.51	591	0	1.55	511	0	-2.6	13.5	0
r2	598	3.17	1621	5	3.12	1202	0	1.5	25.8	100
r3	862	4.42	2914	11	4.1	1739	1	7.2	30.8	90.9
r4	1903	8.09	6383	10	7.3	4854	3	9.7	23.9	70
r5	3101	14.53	11925	15	12.6	7232	3	13.4	39.3	80

Table 4-4 Comparison of Greedy-DME and the New Algorithm by Labyrinth routing of the abstract tree (80% smallest nets L-route)

Benchmarks	# Sinks	Greedy-DME/ Labyrinth			Our new Algorithm / Labyrinth			Improvement		
		Wire length (µm) $\times 10^6$	# Bend vias	# Overflows	Wire length (µm) $\times 10^6$	# Bend vias	# Overflows	Wire length Cost (%)	# Bend (%)	#Overflows (%)
r1	267	1.63	379	8	1.58	301	0	3	20.5	100
r2	598	3.32	1584	7	3.22	1251	5	3.1	21	28.5
r3	862	4.48	3087	17	4.19	1934	2	6.4	37.3	88.3
r4	1903	8.39	4983	26	7.31	3802	8	12.8	23.1	69.2
r5	3101	15.91	9652	102	13.22	6351	11	16.9	34.2	89

Chapter 5

Reducing Wire length and Elongation using Skew Compensation Technique

5.1 Using Flip-Flops with different operating speeds as a skew compensation Technique to construct zero skew energy recovery clock network

In reference [47], a new skew compensation technique using flip-flops with different operating speeds was introduced. The new technique provides timing slacks that could be used in a clock distribution algorithm in order to reduce the total wire length and routing complexity. Traditionally, to manage the clock skew in a clock network, clock distribution algorithms attempt to balance the delay from the source to all sinks. This is accomplished mainly through wire length adjustment, wire width sizing, and buffer insertion. The clock distribution algorithm could also take advantage of the new proposed skew compensation technique along with other traditional balancing approaches to get the desired skew in a clock network with less total wire length. Consequently clock network power consumption will be decreased. Additional benefits of the proposed compensation technique are the reduction in the number of wire elongations and the added flexibility in the distribution network layout. The new compensation technique was incorporated into a zero skew clock tree router (ZST). A ZST is able to construct a clock tree that delivers the clock edges to all sinks with equal delay (nominal zero skew). The Differed Merge

Embedding algorithm (DME) was modified to accommodate the proposed skew compensation technique. In order to use the new technique in any ZST, two major issues should be considered:

1- Selecting which type of flip-flops to be used in every single location.
2- Taking advantage of the timing slacks provided by the new technique during the bottom up tree construction in order to reduce the total clock tree wire length.

Usually, a typical clock tree router is not aware of the underlying data-path and data flow dependency between the clock sinks. This assumption indicates that, at first there is no preference among the clock sinks to guide the algorithm in order to select between different types of flip-flops. In the proposed approach, the flip flops have three operating speeds: standard, slow and fast. Initially all sinks are chosen from the standard type. The best choice for different types of flip-flops at the sinks will be identified while the clock tree is being constructed. This new algorithm is developed based on the observation that a zero skew merging segment obtained by the traditional DME can be shifted towards one of its children by changing the flip-flop types in its left and right sub-trees. The tuning of a merging segment by changing the flip-flop type is illustrated in Fig. 5-1

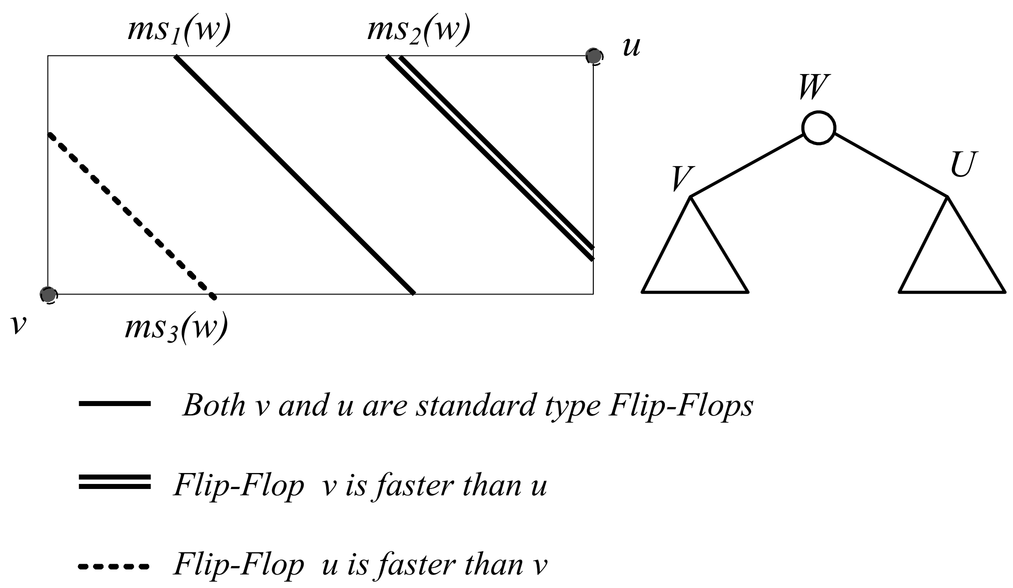

$ms_1(w)$
$ms_2(w)$
u
W
V
U
v
$ms_3(w)$
Both v and u are standard type Flip-Flops
Flip-Flop v is faster than u
Flip-Flop u is faster than v

Figure 5.1 Tuning a merging segment by changing the flip-flop type in left or right sub-tree

In Fig. 5-1, *U* and *V* are two sub-trees which their roots are embedded at locations *u* and *v*, respectively, and *U* and *V* are to be merged such that the new sub-tree *W* has zero skew and minimum wire length. The rectangle with *u* and *v* as opposite vertices encloses all the minimum distance, Manhattan connections between *u* and *v*. *ms*(*w*) is the locus of the points (merging segment) that can merge two points *u* and *v* with minimum wire length and zero skew. In Figure 5-1, $ms_1(w)$ is the merging segment that merges *v* and *u*, where sub-trees *V* and *U* both contain standard flip-flops.

As illustrated in the figure, by changing the flip-flops operating speed in either *U* or *V*, the merging segment shifts either towards *u* or *v*. For a pair of nodes (*u,v*), the algorithm considers up to seven different combinations of flip-flops operating speeds in *u* and *v*;

(*u*standard, *v*standard), (*u*standard, *vf*ast), (*u*standard, *v*slow), (*u*fast, *v*standard), (*u*fast, *v*slow), (*u*slow, *v*standard), and (*u*slow, *v*fast). There are two redundant combinations, (*u*slow, *v*slow) and (*u*fast, *v*fast). Since both of these combinations result in the same merging segment as in (*u*standard, *v*standard), the algorithm does not consider the two redundant cases to compute the merging segment. During the bottom-up phase, the algorithm computes the locus of the merging points (merging segment) where two sub-trees can join such that the new sub-tree has a zero skew. The new merging segment is computed for different combination of flip-flops in both sub-trees. Unlike the traditional DME, in the modified DME algorithm there is a set of merging segments corresponding to each node. Each merging segment is computed similarly to the DME, but the algorithm considers the proper matched delay for either left or right sub-tree. The three types of flip-flops enable the algorithm to use the matched delay values in order to compensate for the skew.

A greedy strategy was used to choose the types of flip flops. This means that if the types of flip-flops in a set of leaves in a sub-tree have already been determined, the algorithm will not change it in a later stage. For example in Fig. 5-1, if the algorithm specifies the slow flip-flop for the leaves in the sub-tree rooted at *v* and the fast flip-flops for the leaves in the sub-tree rooted at *u*, this implies that the decision for the types of flip-flops in the sub-tree *w* is already made. Indeed to achieve more optimum results, one can defer the decision making to the upper levels, but this will increase the timing complexity of the algorithm.

Let s_1, s_2, s_3 and s_4 be four nodes in a clock tree. The nodes are to be merged corresponding to the topology shown in Fig. 5-2(a), where s_1 and s_2 are the children of node *v* and node *u* is the parent of s_3 and s_4. In the upper level of the tree, *u* and *v* are to be merged into node *w*. Assume the flip-flop types in the sub-trees rooted at s_1, s_2, s_3 and s4 have not been specified by the algorithm.

The algorithm enumerates all the seven different choices for the flips-flops in (s_1, s_2) and (s_3, s_4). The merging segments for node *v* and *u* are calculated for all combinations. Let *MS(U)* and *MS(V)* refer to the set of the merging segments for all different flip-flop speed combinations for nodes *u* and *v*, respectively. Two newly determined sub-trees rooted at *v* and *u* need to be merged into *w*. To compute the merging point for node *w*, the algorithm selects one merging segment from *ms(u)* and *ms(v)* which results in minimum wire length. To reduce the total wire length, a sub-tree needs to be merged to another sub-tree that is not only nearby but also minimizes wire elongation. Therefore a merging cost function to include the distance and the wire elongation in a unified form is proposed. This merging cost is the same as the Manhattan distance between the roots of the two sub-trees if there was no elongation; otherwise the extra wire due to wire snaking is included in the merging cost.

The algorithm uses the unified wire length cost function to determine which merging segments should be selected from each one of its children. The best possible choice indicates the types of the flip-flops in s_1, s_2, s_3 and s_4 as shown in Fig.5-2(b). It should be noted that the Elmore delay that is used to model the delay in the square-wave based clock distribution networks algorithms is also valid for signals other than step signals and that the actual delay approaches the Elmore Delay as the input signal rise time increases [48]. This illustrates that the algorithms used to construct square-wave based clock distribution networks can be extended and applied to construct energy-recovery clock distribution networks with a sinusoidal clock signal.

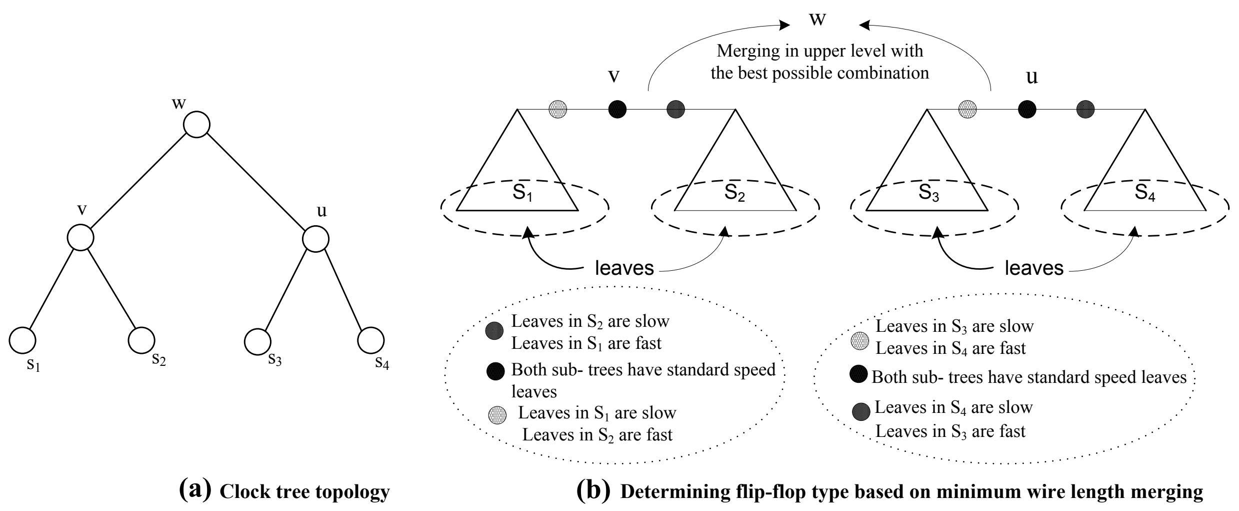


Figure 5-2 Modified DME (with multiple flip flop speeds)

5.2 Experimental Results

The traditional DME and the new modified DME algorithms were implemented in C++ to construct the clock tree. The initial clock tree topology in both cases was obtained by the method of means and medians (MMM). Both algorithms were run on a set of benchmarks (r1-r5) that contain from 267 up to 3101 clock sinks. The clock sink distribution in the benchmarks is the same as the one in [12]. The sinks are SCCER flip-flops with a loading capacitance of 74.98fF. The wire resistance and capacitance are 0.022Ω/µm and 0.083fF/µm. The unit resistance and capacitance are calculated for a 1um width metal seven in 90 nm process technology.

The simulation results obtained are shown in Table 5.1. By applying the proposed skew compensation technique, the total clock tree wire length has been reduced by an average of 11.5%. Reducing the total wire length leads to a reduction in the routing complexity as well as a reduction in the clock tree power consumption which is one of the most important issues in clock distribution networks design.

One of the major drawbacks associated with the DME based clock routers is the fact that they introduce many wire elongations to achieve a zero skew clock network. The elongation problem is exacerbated usually when the clock routers only consider the spatial proximity to find the best matching pairs. The results in Table 5.1 show a reduction of an average of 53.2% in the number of wire elongation. Wire elongation is a real burden in detail phase routing, because they introduce unnecessary bends and vias. The new algorithm is only a simple greedy heuristic that was developed to verify the advantages of using the new skew compensation technique. Indeed the algorithm is not guaranteed to get the best and optimal results. Nevertheless the results are encouraging. As was shown in Fig. 5-2, for an internal node w, the algorithm only computes the merging segment for the closet pair of the merging points of its children. Therefore, some potential Steiner points that might give a better result are pruned away. One could further explore other pruning schemes to get better results.

Table 5-1 Comparison of MMM-DME and the New Modified DME using the proposed skew compensation technique

Benchmarks	# Sinks	MMM-DME		New Modified DME		Improvement	
		Cost (μm)	# wire elongation	Cost (μm)	# wire elongation	Cost (%)	# wire elongation (%)
r1	267	2416227	10	2180193	7	9.7	30
r2	598	6435416	38	5610157	18	12.8	52
r3	862	8064415	71	7002777	24	13.1	66
r4	1903	2458224	136	22066543	59	10.2	56
r5	3101	3936935	324	34745355	123	11.7	62

Chapter 6

Conclusion and Future works

In this thesis we first proposed a via-aware clock tree construction scheme, which considers at most one-bend merging in all branches of clock trees. In preferred direction routing, bends imply switching between metal layers, therefore requiring use of vias. The proposed algorithm reduces via usage in clock trees by decreasing the number of bends. This reduction is accomplished by binding the merging segments to the areas that make fewer bends. The Algorithm uses wire sizing to tune the branch points in order to reduce the number of bends further. Simulation results show that the proposed scheme reduces the number of bends by an average of 29.6% with only 3.4% increase in total capacitance compared to the traditional DME algorithm.

We developed a method to calculate the probabilistic routing demand for clock tree edges during the bottom-up merging segment construction phase using their bounding boxes. We introduced a new demand driven cost function and used the cost function to construct a clock tree topology with regard to demand map of the clock net. A sampled base technique was adopted to store the incremental demand impact for each sample during the merging segment construction phase. We used the accumulated peak demand values to embed the branch points when multiple embedded solutions were available. The results confirm that routing a DME generated clock tree in practice will introduce extra wiring and bends that are hidden in all published DME based clock tree routers. Our algorithm is targeted towards practical clock tree generation with less number of vias. The total wire length was reduced by an average of 8%. Post-routing induced skew was decreased by an average of 31.8%. Also the total number of vias is reduced by 28%. Furthermore routing overflows are reduced considerably at the expense of more computation time.

We also developed a novel flip-flop placement algorithm for a set of fast, slow and normal speed flip-flops. This enabled us to use difference in flip-flop operating speed as a tool to compensate for skew. We used this in our algorithm and managed to reduce the total wire length and elongation.

6.1 Future works

6.1.1 Probabilistic congestion estimation in clock algorithm

As described in Chapter 4, a probabilistic method to estimate the likelihood of congestion during a clock tree embedding phase is developed based on the sampling method. As a result of this implementation, a clock tree router is able to lay out a more realistic clock tree and ease the burden of layout embedding in the detail routing phase. To have a clock tree with less hidden overhead and predictable wire length and wiring topology, several challenges must be addressed:

- Investigating a clustering based implementation for the congestion driven clock router to speed up the algorithm run time
- The grid-less model, SPP, introduced by Haydar et al. describes all the shortest paths between two points in the presence of obstacles. We believe that SPP can be incorporated in our methodology to reduce the complexity of algorithm as well as handling the congestion driven clock routing in the presence of routing blockages. We will investigate this further.
- Sample based merging segments are useful to consolidate most of the routability objectives in a unified optimization problem. To do that an investigation is needed for a unified weighted cost function to embrace the problems of via minimization, congestion awareness and reducing the total wire length.

6.1.2 Topology Generation with obstacle consideration

Obstacle consideration is one of the missing puzzles of a practical clock tree routing. As explained in the literature review, a widely accepted conclusion is that a sub-tree should be merged with its nearest neighbor. This assumption makes sense as long as there is no blockage in the routing region, since only spatial proximity is considered to identify the nearest neighbor. Indeed in the presence of obstacles, a formerly identified nearest pair could be a non-optimum pair to merge. The obstacle introduces detours and bends in the wiring paths that could degrade the resulting merging solution. To address this deficiency we propose the following

- To investigate the topology impact on the wire length and the number of vias. As a result we are expected to come up with a total wire length and number of bend estimation model to predict the lower bound for wire length and bend number prior to the clock routing. We notice that this model has not been developed and, we are expected to propose a more general prediction model for global routing problem besides the clock distribution. The model then can be used to guide a clock tree router in topology generation phase.
- The best reported clock tree router in terms of wire length is known to be the Greedy-DME introduced by Edahiro. In reality, Greedy DME suffers from serious deficiency such that it produces too many wire elongations mainly because it does not consider the sub-tree loading capacitance to find the best pair to merge. Therefore it is very likely for the algorithm to merge a heavily unbalanced pair of sub-trees just by taking the spatial proximity into account. With obstacles, this deficiency is compounded by the fact that the detours from the obstacles are completely overlooked.

6.1.3 Via Minimization in 3D ICs

The fabrication technology is rapidly moving toward stacking multiple die in vertical dimension referred as 3D integrated circuit. The vertical interconnections in 3D ICs are through TSVs (through silicon via). TSVs greatly reduce the circuit reliability and degrade the signal quality. 3D fabrication, usually consider an upper bound for number of TSVs in a 3D design. In 3D integration, the clock network connects a set of flip-flops that are located in different die in the vertical dimension. Via minimization in 3D clock routing algorithm is a new emerging concern that needs to be addressed in future.

References

[1] Xin-Wei Shih, Hsu-Chieh Lee, Kuan-Hsien Ho, Yao-Wen Chang, “High variation-tolerant obstacle-avoiding clock mesh synthesis should be givenwith symmetrical driving trees,” proceedings of International Conference in Computer Aided Design (ICCAD), Nov 2010, pp. 452-457

[2] C. Yeh, G. Wilke, H.Chen, S. Reddy, H. Nguyen, T. Miyoshi, W. Walker, R. Murgai, “Clock Distribution Architectures: A Comprehensive Study,” Proceedings of 7th International Symposium on Quality Electronic Design, 2006, pp. 85-91

[3] Haihua Su, Sachin S. Sapatnekar, “Hybrid structured clock network construction,” Proceedings of ACM/IEEE Computer Aided Design Conference(ICCAD), 2001, pp. 333-336

[4] G. Venkataraman, Zhou Feng, Jiang Hu, Peng L*i* ,“Combinatorial algorithms for fast clock mesh optimization,” Proceedings of ACM/IEEE Computer Aided Design Conference, ICCAD 2006, pp 563 – 567

[5] Anand Rajaram, David Z. Pan, Jiang Hu, “Improved Algorithms for Link-Based Non Tree Clock Networks for Skew Variability Reduction,” Proceedings of International Symposium on Physical Design (ISPD’05), 2005, pp. 55-62

[6] C.J Andeson, J. Petrovick, M. Keaty, J. Warnock, G. Nussbaum, J.M. Tendier, C. Carter, S. Chu, J. Clabes, J.Dilullo, P .Dudelly, P. Harvey, P.J. Restle, “Physical Design of a Fourth-Generation POWER GHz Microprocessor,” Proceedings of International Solid State circuit conference, 2001, pp. 232-233.

[7] Philiph J. Restle, Timothy G. McNamara, David A.Webber, Peter J.Camporese, K. F. Eng, K. A.Jenkins, D. H. Allen, M. P. Quaranta, D. W. Boerstler, C. J.Alpert, C. A. Carter, R. N. Bailey, J. G. Petrovick, B. L. Krauter, and B. D. McCredie, “A Clock Distribution for Microprocessors,” IEEE Journal of Solid-State Circuits, Vol. 36, No. 5, May 2001, pp. 792-799.

[8] Brad Smith, S. Blackley, *et al*, “A Comparison of Via Over-etch Variation between Conventional Al-W and Dual-Inlaid Copper Integrations,” IEEE international Conference on Interconnect Technology, 1999, pp. 106-108

[9] J. D. Cho, M. Sarrafzadeh, “Four-Bend Top-Down Global Routing,” IEEE Transactions on Computer-Aided Design, Vol. 17, No. 9, September 1998, pp. 793-802.

[10] Chuan Lin, Hai Zhou, “Clock Skew Scheduling with Delay Padding for Prescribed Skew Domains,” Proceedings of Asia-Pacific Design Automation Conference (ASP-DAC), 2007, pp. 541-546.

[11] Guthaus, M.R. Sylvester, D. Brown, R.B, “Process-induced skew reduction in nominal zero-skew clock trees,” Proceedings of Design Automation Asia and South Pacific Conference, ASP-DAC 2006, pp. 84-89

[12] M. Hashimoto, T. Yamamoto, and H. Onodera, "Statistical analysis of clock skew variation in H-tree structure," proceedings of international symposium on Quality electronic Design ISQAD, 2005, pp. 402-407.

[13] W.C. Elmore, "The transient response of damped linear networks," Journal of applied physics, Vol. 19, Jan 1948, pp. 55-63.

[14] Golden, M.; Arekapudi, S.; Dabney, G.; Haertel, M.; Hale, S.; Herlinger, L.; Kim, Y.; McGrath, K.; Palisetti, V.; Singh, M, "A 2.6GHz Dual-Core 64bx86 Microprocessor with DDR2 Memory Support," Proc Solid-State Circuits Conference ISSCC, 2006, pp. 325-336.

[15] Rupesh S. Shelar, "An algorithm for routing with capacitance/distance constraints for clock distribution in microprocessors," IEEE Transactions on Computer-Aided Design, Vol. 29, No. 2, 2010, pp. 245-249.

[16] Rupesh S. Shelar," Routing With Constraints for Post-Grid Clock Distribution in Microprocessors," ," Proceedings of International Symposium on Physical Design ISPD, 2009 pp. 141-148.

[17] S. Rusu, S. Tam, H. Muljono, D. Ayers, J. Chang, "A Dual-Core Multi-Threaded Xeon Processor with 16MB L3 Cache," Proceedings Solid-State Circuits Conference ISSCC, 2006, pp. 315-324.

[18] Rupesh S. Shelar, "An efficient clustering algorithm for low power clock tree synthesis," Proceedings of International Symposium on Physical Design ISPD, 2007 pp. 181-188

[19] Wen-Hao Liu, Yih-Lang Li, Hui-Chi Chen, "Minimizing clock latency range in robust clock tree synthesis," Proceeding of Design Automation Conference (ASP-DAC), 2010, pp. 389-394.

[20] Xin. W. Shih, Yao-Wen Chang, "Fast timing-model independent buffered clock-tree synthesis," proceedings of Design Automation Conference, 2010, pp. 80-85.

[21] Xin-Wei Shih, Chung-Chun Cheng, Yuan-Kai Ho, Yao-Wen Chang, "Blockage-avoiding buffered clock-tree synthesis for clock latency-range and skew minimization," Proceedings Design Automation Conference ASP-DAC 2010, pp. 395-400

[22] Jason Cong, Andrew B.Kahng, Cheng-kok Koh, and Albert Tsao, " Bounded Skew Clock and Steiner Routing," ACM Transaction on Design Automation of Electronic System, Vol 3, No 3, July 1998, pp. 341-388.

[23] Jeng-Liang Tsai, Tsung-Hao Chen, and Charlie Chung-Ping Chen, "Zero Skew Clock-Tree Optimization With Buffer Insertion/Sizing and Wire Sizing," IEEE Transaction on Integrated Circuit and Systems, Vol. 23, No. 4, 2004, pp 565-572.

[24] Cong, J.; Kahng, A.B.; Robins, G, "Matching-based methods for high-performance clock routing," IEEE Transactions on Computer-Aided Design, Volume 12, Issue 8, Aug 1993 pp. 1157 – 1169

[25] R. S. Tsay, "An Exact Zero Skew Clock Routing," IEEE Transaction on Computer-Aided Design, Vol . 12, No. 2, 1991, pp. 242-249.

[26] Ting-Hai Chao, Yu-Chin Hsu, Jan-Ming Ho, Kenneth D. Boese , “Zero Skew Clock Routing with Minimum Wirelength,” IEEE Transaction on Circuits and Systems, Vol 93, No 11, November 1992, pp.799-814

[27] Masato Edahiro, “An efficient zero-skew routing algorithm,” Proceedings of ACM/IEEE Design Automation Conference, 1994, pp. 375-380.

[28] A.Kahng and C.Tsao, “More Practical Bounded-Skew Clock routing,” Proceedings of 34th ACM/IEEE Design Automation Conference, June 1997, pp. 594-599.

[29] Haksu Kim Dian Zhou, “Efficient implementation of a planar clock routing with the treatment of obstacles,” IEEE Transactions on Computer-Aided Design, Vol. 9, Issue. 10, October 2000, pp. 1220-1225

[30] H.Saaied, D.Al-Khalili, A.J Al-Khalili, “Clock Tree Tuning using Shortest Path Polygon,” Proceedings of IEEE International Conference of SOC, 2004, pp. 59-62.

[31] X. Zeng, D. Zhou, and W. Li, “Buffer insertion for clock delay and skew minimization,” proceedings of International Symposium on Physical Design ISPD, 1999, pp.36-41

[32] J.G. Xi, and W.W. Dai, “Buffer insertion and sizing under process variation for low power clock distribution,” Proceedings of Design Automation Conference, 1995, pp. 383-388.

[33] G. E. Tellez, and M. Sarrafzadeh,“ Minimal buffer insertion in clock trees with skew and slew rate constraints,” IEEE Transactions on Computer Aided Design of Integrated Circuits, Vol 16, No 4, pp. 1545-1552

[34] Alpert C, Devgan A , “Wire Segmenting For Improved Buffer Insertion,” In Design Automation Conference proceedings, 1997, pp.588-593

[35] H. Saaied, D.Al-Khalili, A.J.Al-Khalili and M.Nekili "Simultaneous Adaptive Wire Adjustment and Local Topology Modification for Tuning a Bounded Skew Clock Tree,” IEEE Trans on Computer Aided Design of Integrated Systems, Vol 24, No. 10, Oct. 2005, pp. 1637-1643

[36] Chin-Chih Chang, Jason Cong, “An Efficient Approach to Multilayer Layer Assignment with an Application to Via Minimization,” IEEE Trans. On Computer-Aided Design, Vol. 18 No. 5, May 1999, pp. 608-620

[37] Chun-Hao Wang, and Wai-kei Mak, “λ-Geometry Clock Tree Construction with Wire length and Via Minimization,” Proceedings of International Symposium on VLSI Desing and Test, 2007, pp.1-4

[38] Duane Boning, Sani Nassif, “Models of Process Variations in Device and Interconnect,” Design of High performance µp Circuits, Chapter 06, pp. 98-11.

[39] Qinglun Chen, Jin Zhao, “Via and Return Path Discontinuity Impact on High Speed Digital Signal Quality,” IEEE Conference on Electrical Performance of Electronic Packaging, 2005, pp. 215-217.

[40] Qiang Chen, Jeffry A. Davis, Payman Zarkeshha and James D. Meindl, “A Compact Physical via Blockage Model,” IEEE Transaction on VLSI, Vol. 8, NO. 6, December 2000, pp. 689-692

[41] Lieniq J, Jerke G, “Electro-migration Aware Physical Design of Integrated Circuits,” Proceedings of IEEE International Conference on VLSI Design, 2005, pp77-82.

[42] Andrew B.Kahng, Albert Tsao, “Planar-DME: A Single-Layer Zero Clock Tree Router,” IEEE Trans on Computer-Aided- Design, Vol. 15, No 1, January 1996, pp.8-1996.

[43] Mehmat.C. Yildiz, Patrick H. Madden, “Preferred Direction Steiner Tree,” IEEE Trans on Computer-Aided-Design, Vol. 21, No 11. November 2002, pp.1968-1372

[44] Yang Yang, Tong Jing, Xianlong Hong, Yu Hu, Qi Zhu, Xiaodong Hu, Guiying Yan, “ Via-Aware Global Routing for Good VLSI Manufacturability and High Yield,” proceedings of 16th ASAP IEEE conference, July 2005, pp. 198-203

[45] M. A. Jackson, A. Srinivasan and E. S. Kuh, “Clock routing for high performance IC’s,” Proceedings of ACM/IEEE Design Automation Conference, pp. 573-579..June 1990

[46] R. Kastner, E. Bozorgzadeh, M.Sarrafzadeh, “Pattern Routing: Use and theory for Increasing Predictability and Avoiding Coupling,” IEEE Transactions on Computer-Aided Design of Integrated Circuit and Systems, Vol. 21, No. 7, July 2007, pp. 777-790

[47] S. E. Esmaei, A. M. Farhangi, A. J. Al-Khalili, G. E. R. Cowan, “A Novel Approach for Skew Compensation in Energy Recovery Clock Distribution Networks,” Submitted to IEEE Transaction on Computer Aided Design.

[48] R. Gupta; B. Tutuianu; L. T. Pileggi, “The Elmore delay as a bound for RC trees with generalized input signals”, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 16, issue. 1, Jan. 1997, pp. 95-104